**Machine Learning**

**LAB 03**



**DEPARTMENT OF COMPUTER SCIENCE**

**FACULTY OF ENGINEERING & COMPUTER SCIENCES**

**NATIONAL UNIVERSITY OF MODERN LANGUAGES**

**H-9, ISLAMABAD**

**PAKISTAN**

**LAB 3: TEXT PREPROCESSING**

**Objectives:**

* To understand and implement the text preprocessing techniques
* **Time Required** : 3 hrs

**Programming Language** : Python

**Software Required** : Anaconda

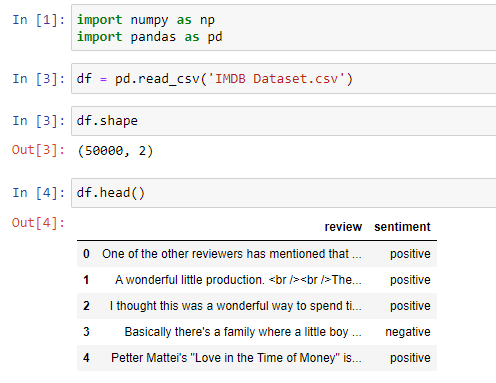
**Introduction**

To prepare the text data for the model building we perform text preprocessing. Apart from numerical data, Text data is available to a great extent which is used to analyze and solve business problems. But before using the data for analysis or prediction, processing the data is important.

Some of the preprocessing steps are:

* Removing punctuations
* Removing URLs
* Removing Stop words
* Lower casing
* Tokenization
* Stemming
* Lemmatization

Let’s start by importing the pandas library and reading the data.

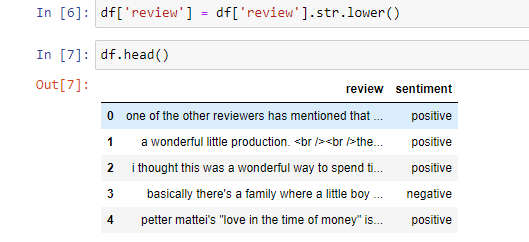


1. **Lowering the Text**

It is one of the most common text preprocessing Python steps where the text is converted into the same case preferably lower case. But it is not necessary to do this step every time you are working on an NLP problem as for some problems lower casing can lead to loss of information.

For example, if in any project we are dealing with the emotions of a person, then the words written in upper cases can be a sign of frustration or excitement.





1. **Removing HTML Tags**

Most of the times when you want to process a tonne of html files in your corpus, you would have to think about cleaning the HTML as a pre-processing step.

**Using Regex**

Regular expressions are the most popular and powerful method for any of the complex string extraction process you want to carry out. Widely used in data mining and string matching algorithms, regex can be easily employed in searching for string patters between HTML tags.

import re

def remove\_html\_tags(text):

pattern = re.compile('<.\*?>')

return pattern.sub(r'', text)

text = "<html><body><p> Movie 1</p><p> Actor - Aamir Khan</p><p> Click here to <a href='http://google.com'>download</a></p></body></html>"

remove\_html\_tags(text)

df['review'] = df['review'].apply(remove\_html\_tags)

df['review']

1. **Removing URLs**

def remove\_url(text):

pattern = re.compile(r'https?://\S+|www\.\S+')

return pattern.sub(r'', text)’

text1 = 'Check out my notebook https://www.kaggle.com/campusx/notebook8223fc1abb'

text2 = 'Check out my notebook http://www.kaggle.com/campusx/notebook8223fc1abb'

text3 = 'Google search here www.google.com'

text4 = 'For notebook click https://www.kaggle.com/campusx/notebook8223fc1abb to search check [www.google.com](http://www.google.com)'

remove\_url(text1)

1. **Removing Punctuation**

def remove\_punc1(text):

return text.translate(str.maketrans('', '', exclude))

remove\_punc1(df['review'][5])

1. **Chat Words Treatment**

chat\_words\_str = """

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

B4N=Bye For Now

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

"""

chat\_words\_list = []

chat\_words\_map\_dict = {}

for line in chat\_words\_str.split("\n"):

if line != "":

cw = line.split("=")[0]

cw\_expanded = line.split("=")[1]

chat\_words\_list.append(cw)

chat\_words\_map\_dict[cw] = cw\_expanded

chat\_words\_list = set(chat\_words\_list)

def chat\_words\_conversion(text):

new\_text = []

for w in text.split():

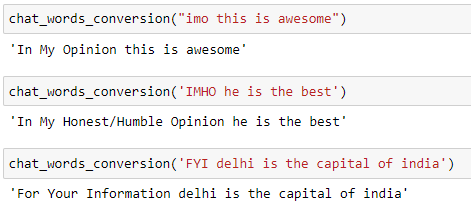
if w.upper() in chat\_words\_list:

new\_text.append(chat\_words\_map\_dict[w.upper()])

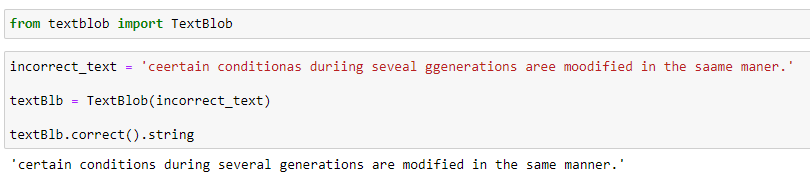
else:

new\_text.append(w)

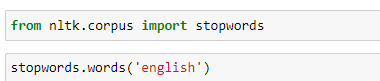
return " ".join(new\_text)

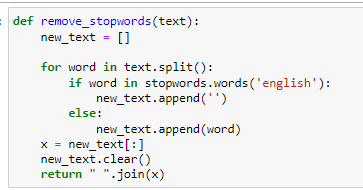


1. **Spelling Correction**



1. **Remove Stopwords**





remove\_stopwords('probably my all-time favorite movie, a story of selflessness, sacrifice and dedication 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')



1. **Removing Emoji**

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

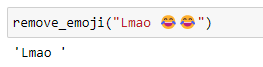
u"\U0001F1E0-\U0001F1FF" # flags (iOS)

u"\U00002702-\U000027B0"

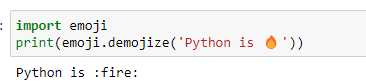
u"\U000024C2-\U0001F251"

"]+", flags=re.UNICODE)

return emoji\_pattern.sub(r'', text)



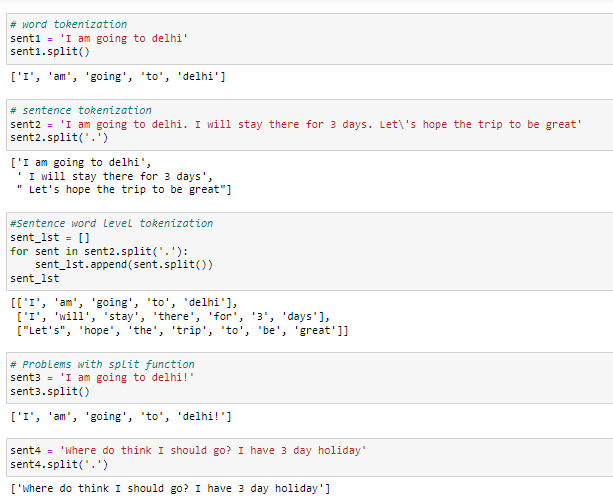
**Replacing Emoji**

****

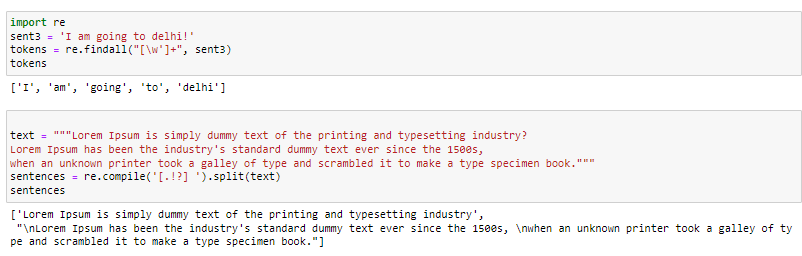
1. **Tokenization**

Tokenization is the process of breaking down the given text in natural language processing into the smallest unit in a sentence called a token. Punctuation marks, words, and numbers can be considered tokens. So why do we need Tokenization? We may want to find the frequencies of the words in the entire text by dividing the given text into tokens. Then, models can be made on these frequencies. Or we may want to tag tokens by word type.

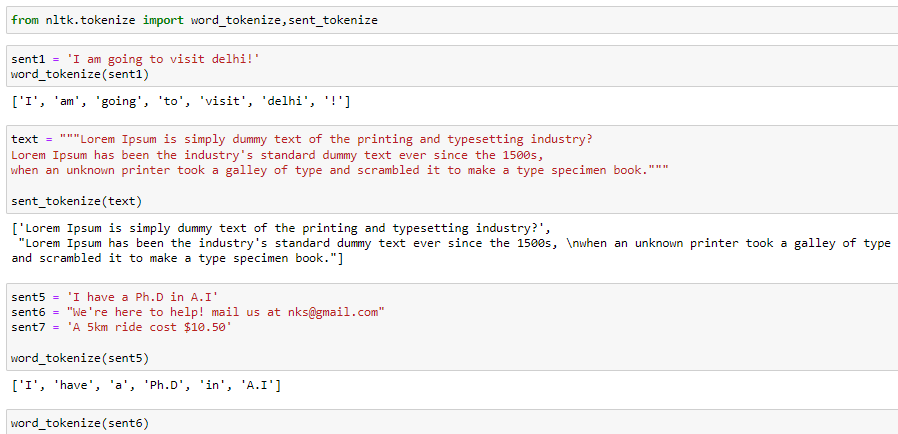
* 1. **Using the split function**

****

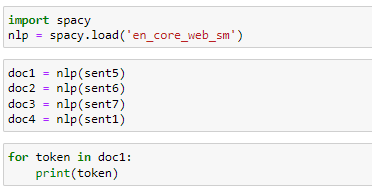
* 1. **Using Regular Expression**

****

* 1. **Using NLTK**

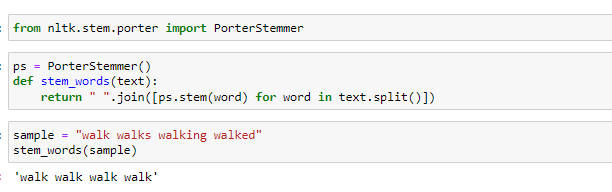


* 1. **Using Spacy**

****

1. **Stemming**

Stemming is the process of finding the root of words. words are reduced to their word stems. A word stem need not be the same root as a dictionary-based morphological root, it just is an equal to or smaller form of the word. It can occasionally lead to unmeaningful common base roots.

****

text = '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 having seen it some 15 or more times in the last 25 years paul lukas performance 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 dressedup midgets than children but that 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 thumbs theyd all be up for this movie'

print(text)

stem\_words(text)

1. **Lemmatizer**

It is the process of finding the form of the related word in the dictionary. It is different from Stemming. It involves longer processes to calculate than Stemming

import nltk

from nltk.stem import WordNetLemmatizer

wordnet\_lemmatizer = WordNetLemmatizer()

sentence = "He was running and eating at same time. He has bad habit of swimming after playing long hours in the Sun."

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')))

**LAB TASK**

You will load the Movies.csv file and load it as data frame in Python. Now analyze the dataset and carry out all the text pre-processing steps on the Overview. Carry out maximum data cleaning tasks and in the end carry out Tokenization, stemming and Lemmatization