Text Preprocessing for NLP (Natural Language Processing),Beginners to Master

Ujjawal Verma · Follow
Published in Analytics Vidhya
10 min read · Feb 15, 2020

Listen Share More

cc:https://www.flickr.com/photos/stevensnodgrass/6274372541/

In this blog we will talking about the text preprocessing for Natural Language Processing (NLP) problems. Basically, *NLP is an art to extract some information from the text*. Now a days many organization deal with huge amount of text data like customers review, tweets,news letters,emails, etc. and get much more information from text by using NLP & Machine Learning.

The first step of NLP is text preprocessing, that we are going to discuss. Here I am using Amazon Reviews: Unlocked Mobile Phones dataset for text preprocessing.

So, Before starting we all need to know why text preprocessing is required?

Why text preprocessing ?

As we know Machine Learning needs data in the numeric form. We basically used encoding technique (BagOfWord, Bi-gram,n-gram, TF-IDF, Word2Vec) to encode text into numeric vector. But before encoding we first need to clean the text data and *this process to prepare(or clean) text data before encoding is called text preprocessing,* this is the very first step to solve the NLP problems.

I am doing text preprocessing step by step on sentiment analysis of *Amazon Reviews: Unlocked Mobile Phones* dataset, Let's play with the data... 😊 😎

Content

1. *Import the dataset & Libraries.*
2. *Dealing with Missing Values.*
3. *Labeling the Dataset.*
4. *Data Cleaning and text preprocessing.*

1. Import the dataset & Libraries

First step is usually importing the libraries that will be needed in the program. A library is essentially a collection of modules that can be called and used.

```
# Importing the Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


# Loading the dataset
dataset = pd.read_csv("Amazon_Unlocked_Mobile_Data.csv")
```

let's look at the dataset we got, its look like as shown below, Here we can see there is 6 features *'Product Name', 'Brand Name', 'Price', 'Rating', 'Reviews' and 'Review Votes'.*

```
dataset.head(10)
```

	Product Name	Brand Name	Price	Rating	Reviews	Review Votes
0	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	5	I feel so LUCKY to have found this used (phone...	1.0
1	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	4	nice phone, nice up grade from my pantach revu...	0.0
2	LG G4 H815 5.5-Inch Factory Unlocked Smartphon...	LG	285.99	1	Hello e-v-e-r-y-o-n-e!!!@@@@!!!!!! 😕 @DONT BUY...	1.0
3	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	5	Very pleased	0.0
4	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	4	It works good but it goes slow sometimes but i...	0.0
5	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	4	Great phone to replace my lost phone. The only...	0.0
6	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	1	I already had a phone with problems... I know ...	1.0
7	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	2	The charging port was loose. I got that solder...	0.0
8	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	2	Phone looks good but wouldn't stay charged, ha...	0.0
9	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	5	I originally was using the Samsung S2 Galaxy f...	0.0

2. Dealing With Missing Values

In this step we will check the null values in our dataset and replace or drop as per the dataset.

```
dataset.isna().sum()
```

```
Product Name        0
Brand Name      65171
Price            5933
Rating              0
Reviews            62
Review Votes    12296
dtype: int64
```

We are doing sentiment analysis on this dataset. So we required basically two features '*Rating*' and '*Review*'. As above, Reviews having only 62 null values. Now, we will first trim our dataset with only two features and then remove these all 62 records with the help of below code.

```
dataset = dataset[['Rating','Reviews']]
dataset.dropna(inplace=True)
```

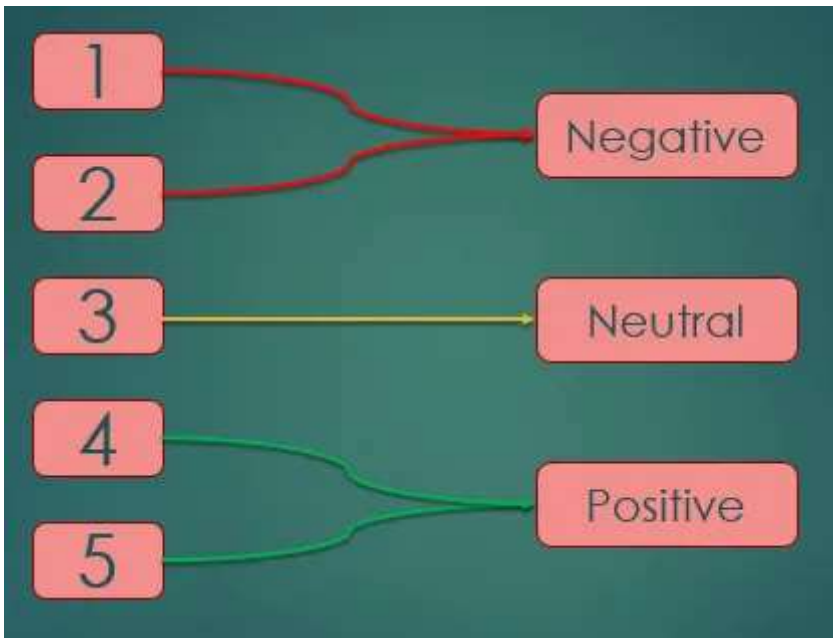
```
dataset.isna().sum()
```

```
Rating     0
Reviews    0
dtype: int64
```

As we can see all null values has been removed from our dataset. Let's Create labels according to the rating given by customers.

3. Labeling The Dataset

As per our dataset there is rating from 1 to 5. So, According to rating we will create there labels, *Positive*(for 1 & 2 Rating), *Neutral*(for 3 Rating) and *Negative* (for 4,& 5 Rating).



Labeling

```
def LableFunc(rating):
    if rating >= 4:
        return 'Positive'
    elif rating <= 2:
        return 'Negative'
    else:
        return 'Neutral'

dataset['Label'] = dataset['Rating'].apply(LableFunc)
```

So, As above code we *create the label as per the rating. Let's look at the dataset.*

```
dataset.head(10)
```

	Rating	Reviews	Label
0	5	I feel so LUCKY to have found this used (phone...	Positive
1	4	nice phone, nice up grade from my pantach revu...	Positive
2	1	Hello e-v-e-r-y-o-n-e!!!@@@@!!!!!! 😊 @DONT BUY...	Negative
3	5	Very pleased	Positive
4	4	It works good but it goes slow sometimes but i...	Positive
5	4	Great phone to replace my lost phone. The only...	Positive
6	1	I already had a phone with problems... I know ...	Negative
7	2	The charging port was loose. I got that solder...	Negative
8	2	Phone looks good but wouldn't stay charged, ha...	Negative
9	5	I originally was using the Samsung S2 Galaxy f...	Positive

4. Data Cleaning And Text Preprocessing.

We are only considering the '*Reviews*' feature from the dataset for text preprocessing. I will do few steps here to clean the text data, Generally it's depends on the text data or problem requirement. Here i am explaining this process step-by-step.

Preprocessing the raw text:

This involves the following:

I. Removing URL.

II. Removing all irrelevant characters (Numbers and Punctuation).

III. Convert all characters into lowercase.

IV. Tokenization

V. Removing Stopwords

VI. Stemming and Lemmatization

VII. Remove the words having length <= 2

VIII. Convert the list of tokens into back to the string

> *For better understanding we take a review from the dataset and see how it will change after each steps.*

```
dataset['Reviews'].iloc[2]
```

```
"Hello e-v-e-r-y-o-n-e!!!@@@@!!!!!! 😊 @DONT BUY THIS PHONE at all-first of al
l that say the phone in new , i took it to the lab after 6 month the phone is d
ead dead ,you can save itthay open the phone in the lab and say!!!!the phone is
renew ,and its cheepe commponents.I payed 400$ for only 6 month ,now i need to
buy new one this LG G4 is dead .not nice 'people say to me dont buy from http
s://www.amazon.com/gp/aw/d/B01GYUDMFY/ref=ya_aw_od_pi?ie=UTF8&psc=1 at al!!!"
```

Example

I. Removing URL —

As we can see there is an url and we don't want it to be a part of our corpus. Let's remove it by using below line of code.

```
# Importing the library 're' to remove non url, Numbers and punctuation
import re

def clean_url(review_text):
    return re.sub(r'http\S+', '', review_text)

dataset['CleanReview'] = dataset['Reviews'].apply(clean_url)
```

> *Result: As we can see the url (highlighted by green color) has been removed.*

```
"Hello e-v-e-r-y-o-n-e!!!@@@@!!!!!! ?? @DONT BUY THIS PHONE at all-first o
f all that says the phone in new , i took it to the lab after 6 month the
phone is dead dead ,you can save it,they open the phone in the lab and say
s!!!! the phone is renew ,and its cheapest commponents.I payed 400$ for on
ly 6 month ,now i need to buy new one this LG G4 is dead .not a best thing
'people  are saying to me dont buy from https://www.amazon.com/gp/aw/d/B01
GYUDMFY/ref=ya_aw_od_pi?ie=UTF8&psc=1 at all, it's troubling!!!"
```

Removing url

```
"Hello e-v-e-r-y-o-n-e!!!@@@@!!!!!! ?? @DONT BUY THIS PHONE at all-first o
f all that says the phone in new , i took it to the lab after 6 month the
phone is dead dead ,you can save it,they open the phone in the lab and say
s!!!! the phone is renew ,and its cheapest commponents.I payed 400$ for on
ly 6 month ,now i need to buy new one this LG G4 is dead .not a best thing
'people  are saying to me dont buy from  at all, it's troubling!!!"
```

Removing URL

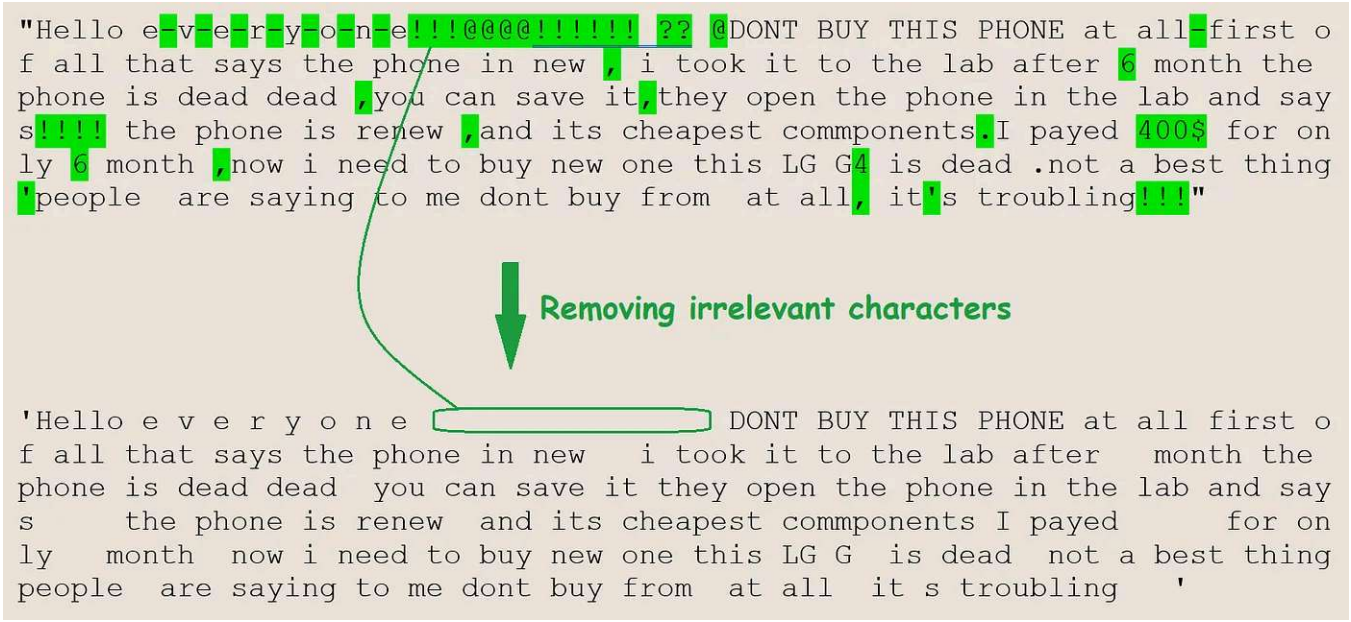
II. Removing all irrelevant characters (Numbers and Punctuation) —

Remove numbers if they are not relevant to your analyses (0–9). And punctuation also will be remove. Punctuation is basically the set of symbols [!”#$%&’()*+,-./:;<=>?@[\]^_`{|}~]:

```
def clean_non_alphanumeric(review_text):
    return re.sub('[^a-zA-Z]',' ',review_text)

dataset['CleanReview'] = dataset['CleanReview'].apply(clean_non_alphanumeric)
```

> *Result: All numeric and punctuation has been replaced with space ’ ‘ .*



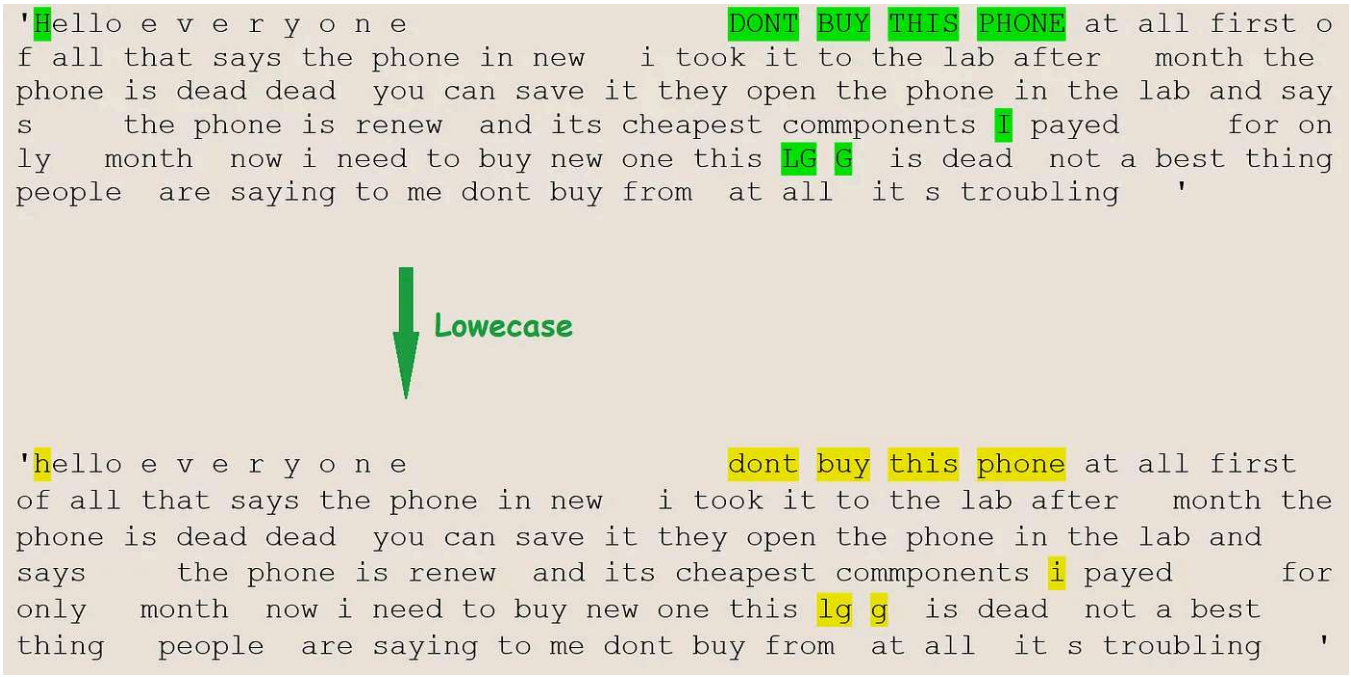
Removing all irrelevant characters

III. Convert all characters into lowercase —

All words changes into lower case or uppercase to avoid the duplication. Because “Phone” and “phone” will be considered as 2 separate words if this step is not done.

```
def clean_lowercase(review_text):
    return str(review_text).lower()
dataset['CleanReview'] = dataset['CleanReview'].apply(clean_lowercase)
```

> *Result: All upper case that are highlighted by green color has been replaced with lower case (highlighted by yellow color).*



Convert all characters into lowercase

IV. Tokenization —

Tokenization is the process of splitting the given text into smaller pieces called tokens. Words, numbers, punctuation marks, and others can be considered as tokens. We will use Natural language tool kit (nltk) library for tokenization.

> *Note: If we have data in the form of paragraphs, and we want to convert the paragraph into sentences, then we will use nltk.sent_tokenize(paragraph).*

Here we will use below line of code to perform tokenization.

```
# Using nltk (Natural language tool kit) library for tokenization
import nltk
from nltk.tokenize import word_tokenize


def clean_tokenization(review_text):
    return word_tokenize(review_text)

dataset['CleanReview'] = dataset['CleanReview'].apply(clean_tokenization)
```

> *Result: As we can see the string has been changed into tokens, that has been stored in the form of 'list of string' .*

```
'hello e v e r y o n e                    dont buy this phone at all first o
f all that says the phone in new   i took it to the lab after   month the
phone is dead dead  you can save it they open the phone in the lab and say
s     the phone is renew  and its cheapest commponents i payed      for on
ly   month  now i need to buy new one this lg g  is dead  not a best thing
people  are saying to me dont buy from  at all  it s troubling   '
```

Tokenization

```
['hello', 'e', 'v', 'e', 'r', 'y', 'o', 'n', 'e', 'dont', 'buy', 'this',
'phone', 'at', 'all', 'first', 'of', 'all', 'that', 'says', 'the',
'phone', 'in', 'new', 'i', 'took', 'it', 'to', 'the', 'lab', 'after',
'month', 'the', 'phone', 'is', 'dead', 'dead', 'you', 'can', 'save', 'it',
```


Tokenization

Now we get list of string of each record (or row). Let's look to the dataset.

```
dataset.head(5)
```

	Rating	Reviews	Label	CleanReview
0	5	I feel so LUCKY to have found this used (phone...	Positive	[i, feel, so, lucky, to, have, found, this, us...
1	4	nice phone, nice up grade from my pantach revu...	Positive	[nice, phone, nice, up, grade, from, my, panta...
2	1	Hello e-v-e-r-y-o-n-e!!!@@@@!!!!!! 😊 @DONT BUY...	Negative	[hello, e, v, e, r, y, o, n, e, dont, buy, thi...
3	5	Very pleased	Positive	[very, pleased]
4	4	It works good but it goes slow sometimes but i...	Positive	[it, works, good, but, it, goes, slow, sometim...

V. Removing Stopwords —

"Stopwords" are the most common words in a language like "the", "a", "me", "is", "to", "all",. These words do not carry important meaning and are usually removed from texts. It is possible to remove stopwords using Natural Language Toolkit (nltk). You also may check the list of stopwords by using following code.

```
# importing the libraries required to remove the stopwords and punctuation
from nltk.corpus import stopwords
# let's look at the stopwords in english
stopwords.words('english')
```

```
['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
 "you've",
 "you'll",
 "you'd",
 'your',
 'yours',
 'yourself',
 'yourselves',
 'he',
 'him',
 'his',
```

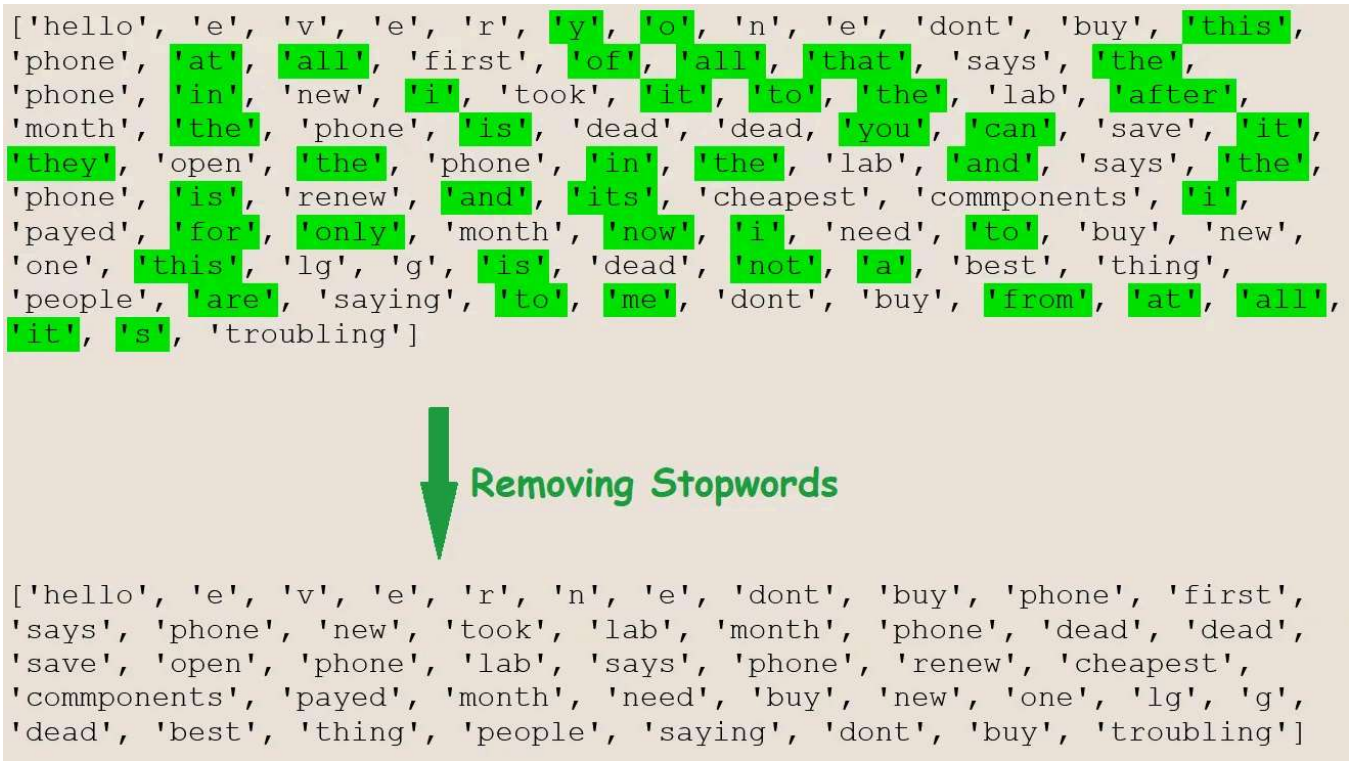
List of Stopwords

So, these are the stopwords which we need to remove, Let's remove the stopwords.

```
stop_words = set(stopwords.words('english'))
def clean_stopwords(token):
    return [item for item in token if item not in stop_words]

dataset['CleanReview'] = dataset['CleanReview'].apply(clean_stopwords)
```

> *Result: Now here we can see that all highlighted tokens has been removed from the corpus after apply the function clean_stopwords().*



Removing Stopwords

VI. Stemming and Lemmatization —

The aim of both processes is the same: reducing the inflectional forms of each word into a common base or root. Both process are different, let's see what is stemming and lemmatization.

Stemming usually refers to a crude process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational units (the obtained element is known as the stem).

On the other hand, *lemmatization* consists in doing things properly with the use of a vocabulary and morphological analysis of words, to return the base or dictionary form of a word, which is known as the lemma.

If we stem the sentence "I saw an amazing thing "we would obtain 's' instead of 'saw', but if we lemmatize it we would obtain 'see', which is the lemma.

$$\text{I saw an amazing thing} \xrightarrow{\text{stem}} \text{I s an amazing thing}$$

$$\text{I saw an amazing thing} \xrightarrow{\text{lemma}} \text{I see an amazing thing}$$

Both techniques could remove important information but also help us to normalize our corpus (although lemmatization is the one that is usually applied). Actually

stemming create some words, that may not have any meaning, so we usually use lemmatization.

I will show you the difference between both with the help of code and result.

let's look at stemming first.

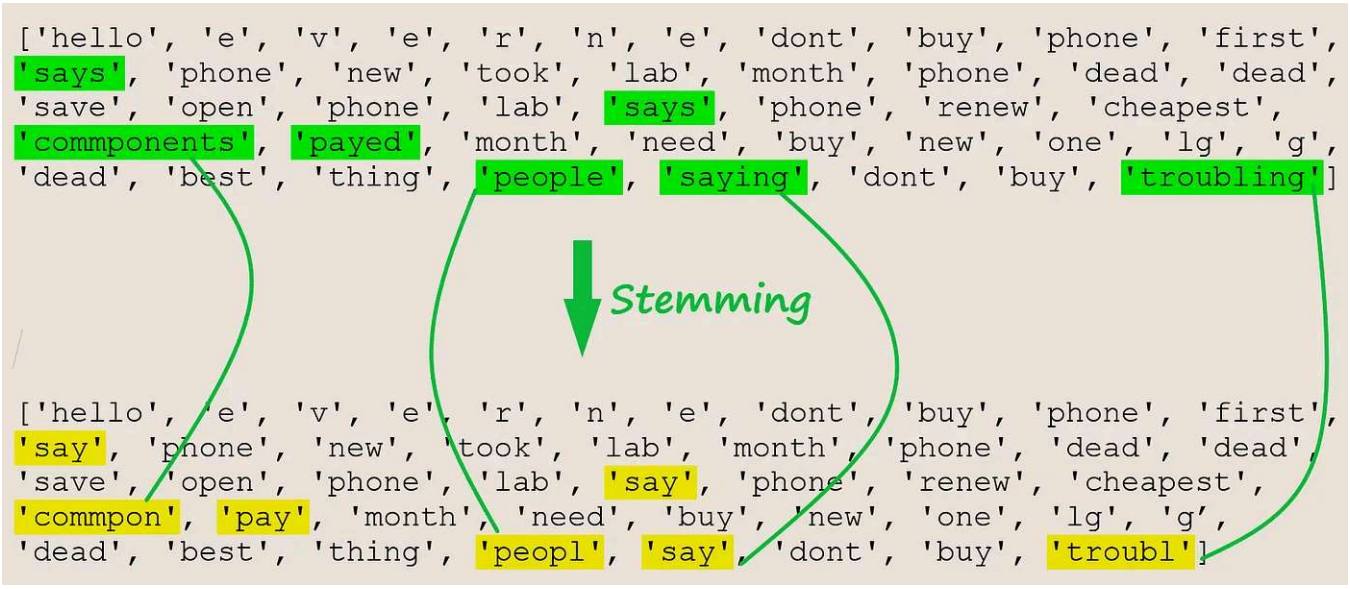
- *Stemming:*

```
from  nltk.stem import PorterStemmer

stemmer = PorterStemmer()
def clean_stem(token):
    return [stemmer.stem(i) for i in token]

dataset['CleanReview'] = dataset['CleanReview'].apply(clean_stem)
```

> *Result: As observe the output, there is some words has been stem like 'commponents' to ' 'commpon', 'says' to 'say', 'people' to 'peopl' and 'troubling' to 'troubl'.*



Stemming

Now we can see here, there is some words changed those have no meaning, and this is the challenge to use stemming. Let's go forward for lemmatization and see the difference in the output.

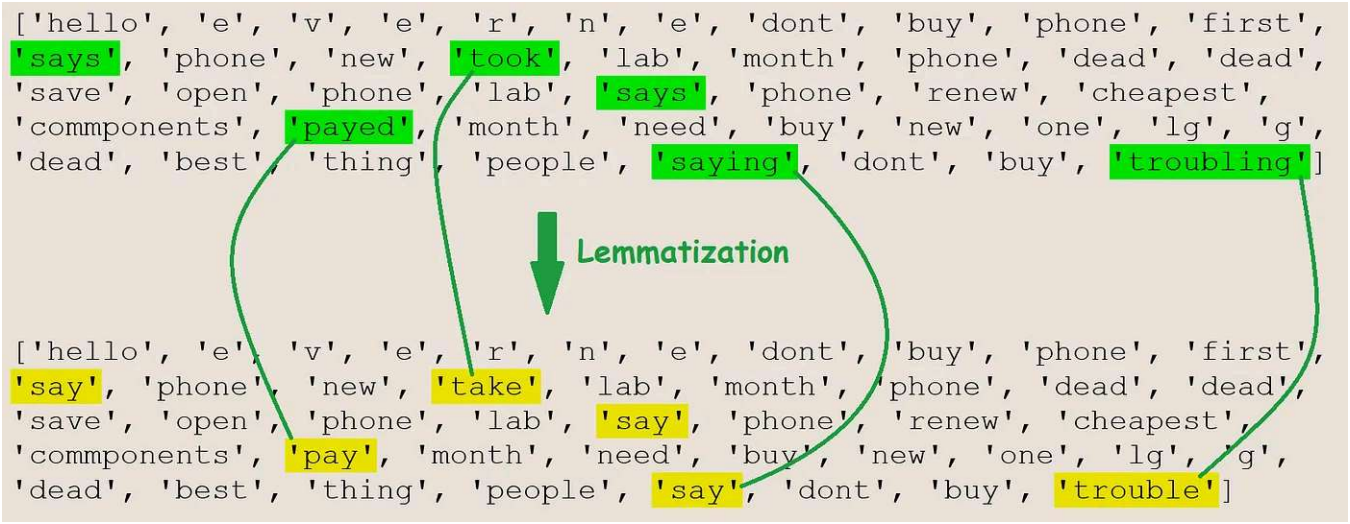
- *Lemmatization:*

```
from nltk.stem import WordNetLemmatizer
lemma=WordNetLemmatizer()

def clean_lemmatization(token):
    return [lemma.lemmatize(word=w,pos='v') for w in token]

dataset['CleanReview'] = dataset['CleanReview'].apply(clean_lemmatization)
```

> *Result: Now here we can see it finds the root word, like 'troubling' to 'trouble', 'took' to 'take' and 'payed' to 'pay' . So, As opposed to stemming, lemmatization does not simply chop off inflections. Instead it uses lexical knowledge bases to get the correct base forms of words.*



Lemmatization

So, here we can see the difference between stemming and lemmatization. Generally we are using lemmatization.

Let's consider lemmatization output for further process.

VII. Remove the words having length <= 2 —

Basically, after performing all required process in text processing there is some kind of noise is present in our corpus, so like that i am removing the words which have very short length.

```
def Clean_length(token):
    return [i for i in token if len(i) > 2]

dataset['CleanReview'] = dataset['CleanReview'].apply(Clean_length)
```

> *Result:Removing the words those have length less than or equal to 2.*

```
['hello', 'e', 'v', 'e', 'r', 'n', 'e', 'dont', 'buy', 'phone', 'first',
'say', 'phone', 'new', 'take', 'lab', 'month', 'phone', 'dead', 'dead',
'save', 'open', 'phone', 'lab', 'say', 'phone', 'renew', 'cheapest',
'commponents', 'pay', 'month', 'need', 'buy', 'new', 'one', 'lg', 'g',
'dead', 'best', 'thing', 'people', 'say', 'dont', 'buy', 'trouble']
```

Removing word having less lenth

```
['hello', 'dont', 'buy', 'phone', 'first', 'say', 'phone', 'new', 'take',
'lab', 'month', 'phone', 'dead', 'dead', 'save', 'open', 'phone', 'lab',
'say', 'phone', 'renew', 'cheapest', 'commponents', 'pay', 'month', 'need'
'buy', 'new', 'one', 'dead', 'best', 'thing', 'people', 'say', 'dont',
'buy', 'trouble']
```

Remove the words having length <= 2

Now we get the required corpus after text preprocessing. now we will convert back this list to string. for encoding the text.

VII. Convert the list of tokens into back to the string —

```
def convert_to_string(listReview):
    return ' '.join(listReview)

dataset['CleanReview'] = dataset['CleanReview'].apply(convert_to_string)
```

> *Result: After preforming the above code, getting a string of input list.*

```
['hello', 'dont', 'buy', 'phone', 'first', 'say', 'phone', 'new', 'take',
'lab', 'month', 'phone', 'dead', 'dead', 'save', 'open', 'phone', 'lab',
'say', 'phone', 'renew', 'cheapest', 'commponents', 'pay', 'month', 'need'
'buy', 'new', 'one', 'dead', 'best', 'thing', 'people', 'say', 'dont',
'buy', 'trouble']
```

Converting into string

```
'hello dont buy phone first say phone new take lab month phone dead dead s
ave open phone lab say phone renew cheapest commponents pay month need buy
new one dead best thing people say dont buy trouble'
```

Converting tokens into string

Let's look at our dataset after performing text preprocessing.

```
dataset.head()
```

	Rating	Reviews	Label	CleanReview
0	5	I feel so LUCKY to have found this used (phone...	Positive	feel lucky find use phone use hard phone line ...
1	4	nice phone, nice up grade from my pantach revu...	Positive	nice phone nice grade pantach revue clean set ...
2	1	Hello e-v-e-r-y-o-n-e!!!@@@@!!!!!! ?? @DONT BU...	Negative	hello dont buy phone first say phone new take ...
3	5	Very pleased	Positive	please
4	4	It works good but it goes slow sometimes but i...	Positive	work good slow sometimes good phone love

Final data after text preprocessing

Now we get the dataset which we required for encoding the text.

Same thing can be achieved by using single function which I want to share with you.

```
import re
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

stop_words = set(stopwords.words('english'))
lemma=WordNetLemmatizer()

def clean_Review(review_text):
    review_text = re.sub(r'http\S+', '', review_text)                    #Removing the url
    review_text = re.sub('[^a-zA-Z]',' ',review_text)                    #Removing Numbers and Punctuation
    review_text = str(review_text).lower()                               #Convert all characters into lowercase
    review_text = word_tokenize(review_text)                             #Tokenization
    review_text = [item for item in review_text if item not in stop_words] #Removing Stop Words
    review_text = [lemma.lemmatize(word=w,pos='v') for w in review_text]  #Lemmatization
    review_text = [i for i in review_text if len(i) > 2]                  #Remove the words having length <= 2
    review_text = ' '.join(review_text)                                  #Converting list to string
    return review_text


dataset['CleanReview'] = dataset['Reviews'].apply(clean_Review)
```

Function for Text Preprocessing

So, these are the steps using for text preprocessing for NLP problems. You don't have need to follow all process, some times you have need to cover less steps. Actually It's depends on your dataset and as well as your problem.

Let's See one more interesting thing about the text preprocessing, which is text visualization.

Text Visualization

After text preprocessing, let's look at our corpus and observe is this ready for encode to numeric vector or not?

Here I will visualize out tokens in corpus label wise, means we have to split our dataset corresponding to labels. which can be done by following code.

```
# split out dataset with respect to labels
PositiveReview = dataset[dataset.Label == 'Positive']['CleanReview']
NeutralReview = dataset[dataset.Label == 'Neutral']['CleanReview']
NegativeReview = dataset[dataset.Label == 'Negative']['CleanReview']
```

Split data with respect to labels

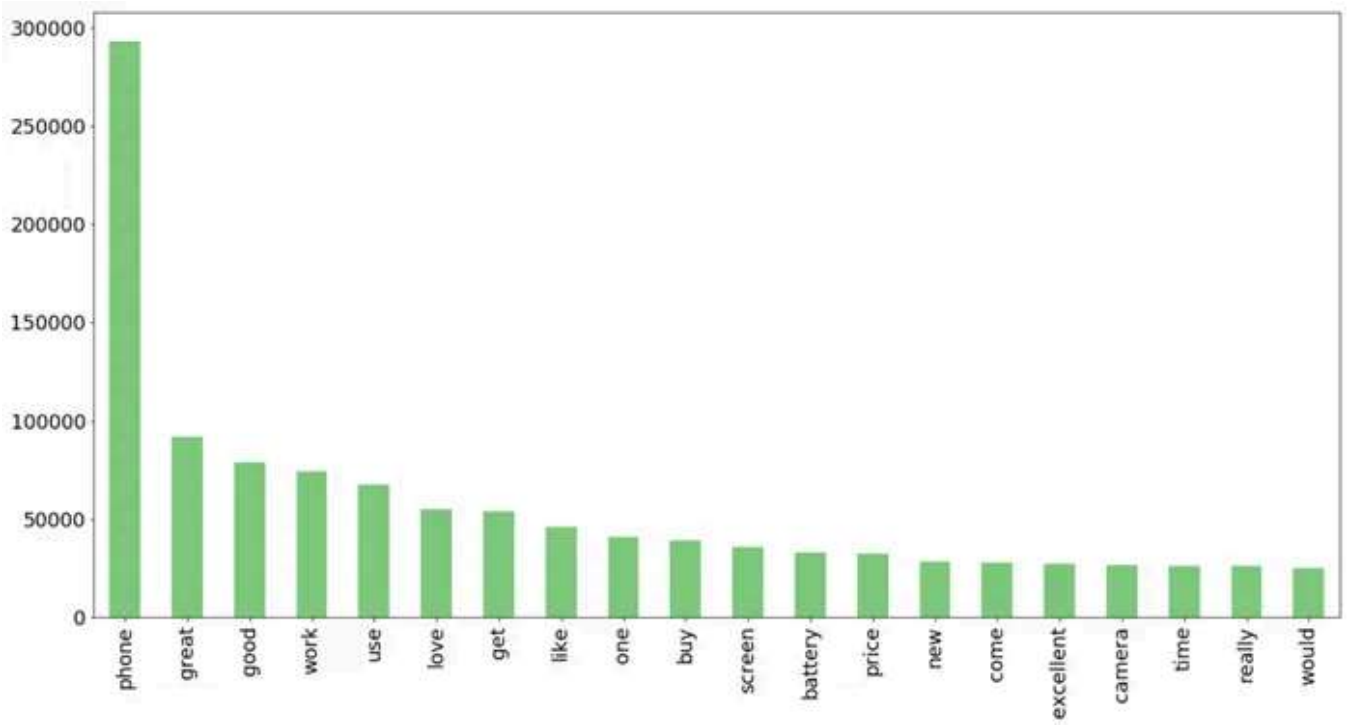
Let's visualize our corpus corresponding to the labels, Here i am taking top 20 words and see their frequency in our corpus with the help of below line of code.

```
color = ['Accent','Paired', 'Pastel1']
splitedData = [PositiveReview,NeutralReview,NegativeReview]

for item in range(3):
    plt.figure(figsize=(20,10))
    pd.Series(' '.join([i for i in splitedData[item]]).split()).value_counts().head(20).plot(kind='bar',colormap= color[item])
    plt.show()
```

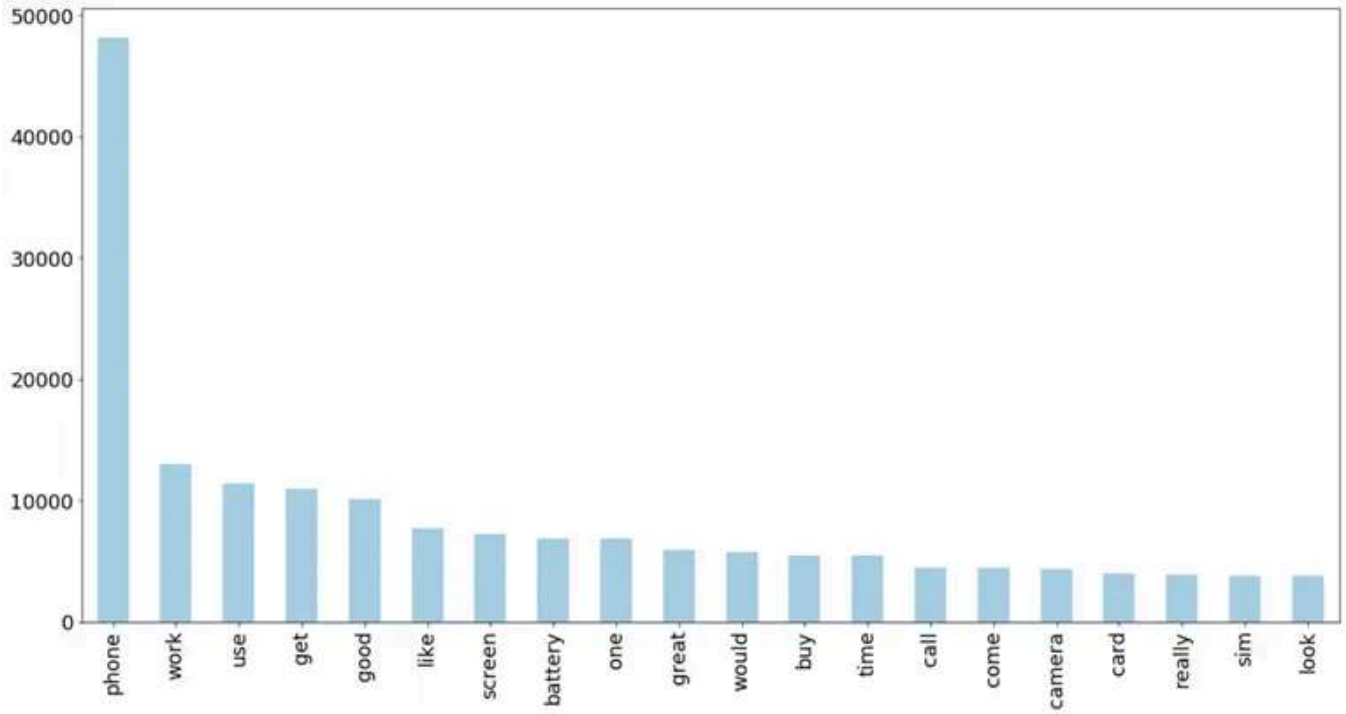
Will get the result as below for Positive, Neutral and Negative review.

- *For Positive corpus*



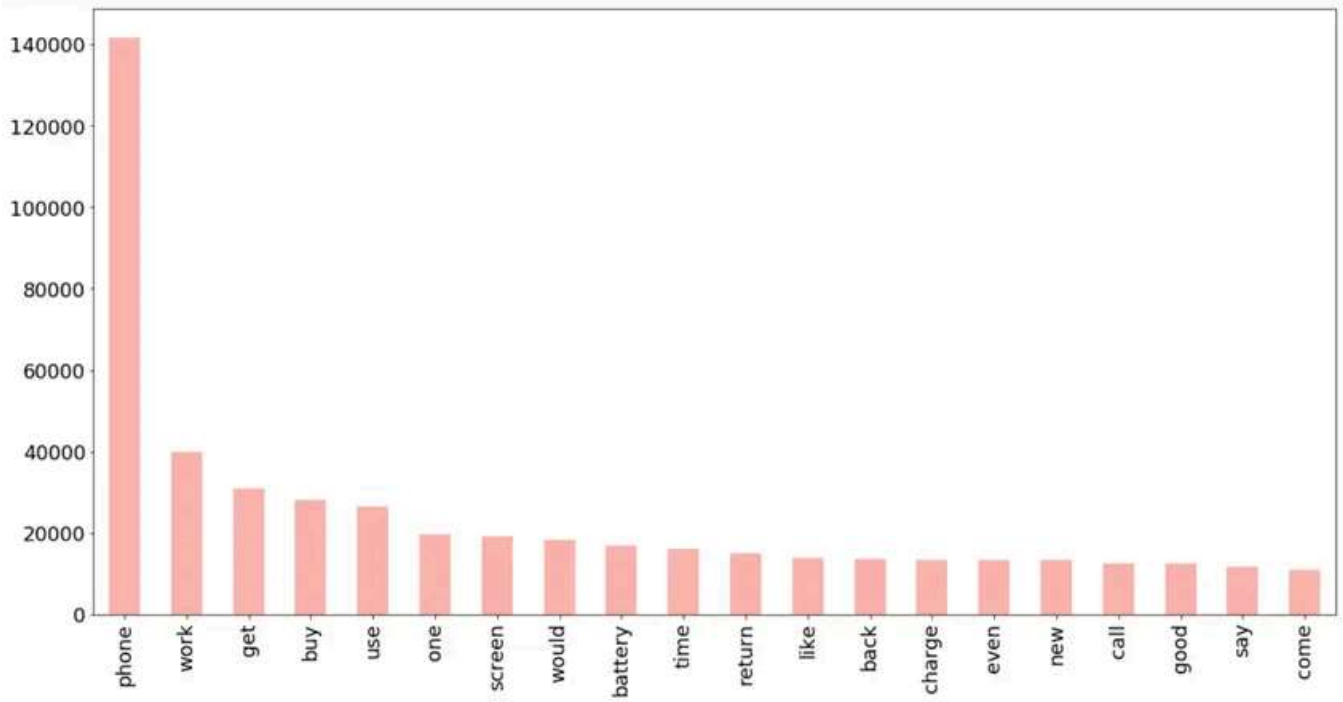
Top 20 words in Positive corpus

- *For Neutral Corpus*



Top 20 words in Neutral corpus

- *For Negative Corpus*



Top 20 words in Negative corpus

Observation: Here we can clearly observe that there is a word '*phone*', which is common in all the labels and having highest frequency. So, it will be a good step if we remove the 'phone' word from our corpus for a better performance.

> *Note: You also can remove some other common words if you want like, 'would', 'get', those are also not carry important deal with respect to your problem.*

So, here I am only removing 'phone' word and this will be our final step in text preprocessing. Let's go for it.

Removing 'phone' from corpus: —

```
# Now we will create a method to remove the phone word from our corpus
def Phone_Word_Remove(review):
    return ' '.join([i for i in review.split() if i != 'phone'])

PositiveReview = PositiveReview.apply(Phone_Word_Remove)
NeutralReview = NeutralReview.apply(Phone_Word_Remove)
NegativeReview = NegativeReview.apply(Phone_Word_Remove)
```

So, if you visualize your corpus again then you will see there is no 'phone' word present there, and it is our final corpus which is ready for encoding.

Generally we used *BagOfWord, Bi-gram,n-gram, TF-IDF* & *Word2Vec* technique to encode text into numeric vector. and after that applied Machine Learning for sentiment analysis. I will cover all encoding techniques in my upcoming blogs.

So This is the ending of this blog, hope you enjoyed and got something from here. Please let me know if I missed something in text preprocessing.

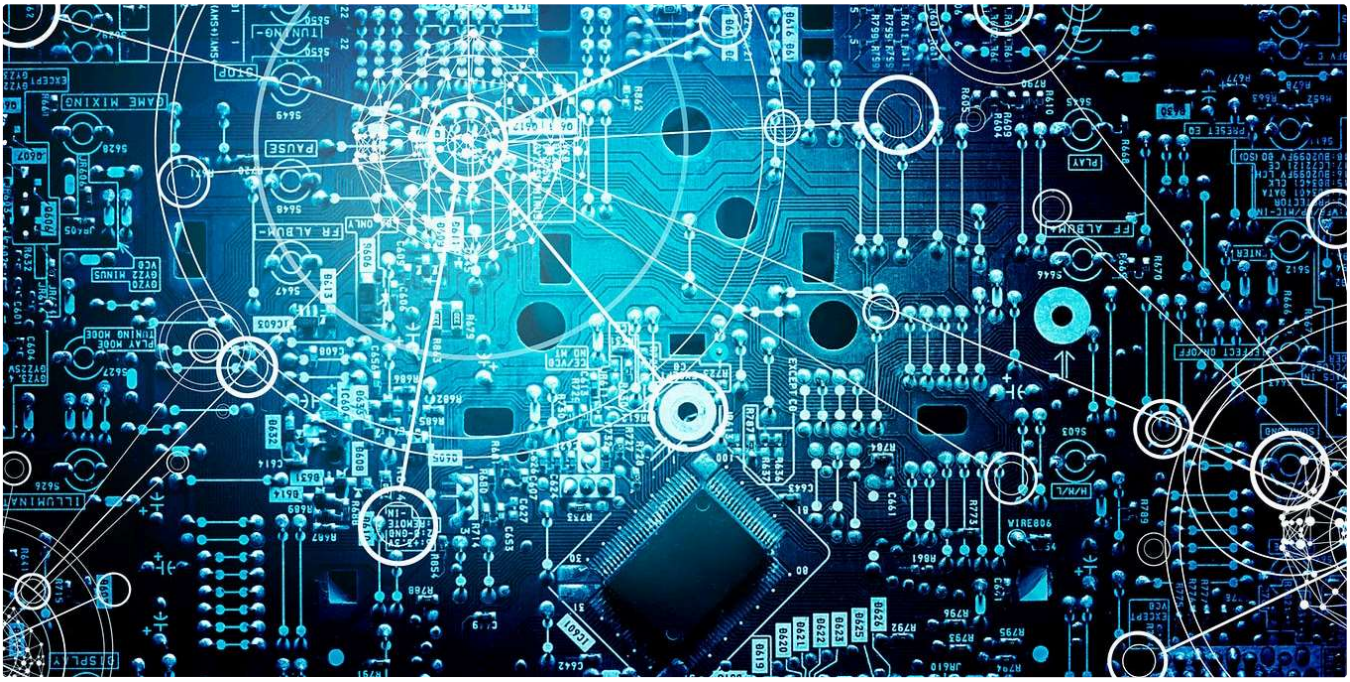
Happy Learning, Keep growing... 😊 😊

Follow

Written by Ujjawal Verma

70 Followers · Writer for Analytics Vidhya

More from Ujjawal Verma and Analytics Vidhya



Ujjawal Verma in Analytics Vidhya

Data Cleaning and Preprocessing

Data preprocessing involves the transformation of the raw dataset into an understandable format. Preprocessing data is a fundamental stage...

10 min read · Nov 19, 2019

153 3

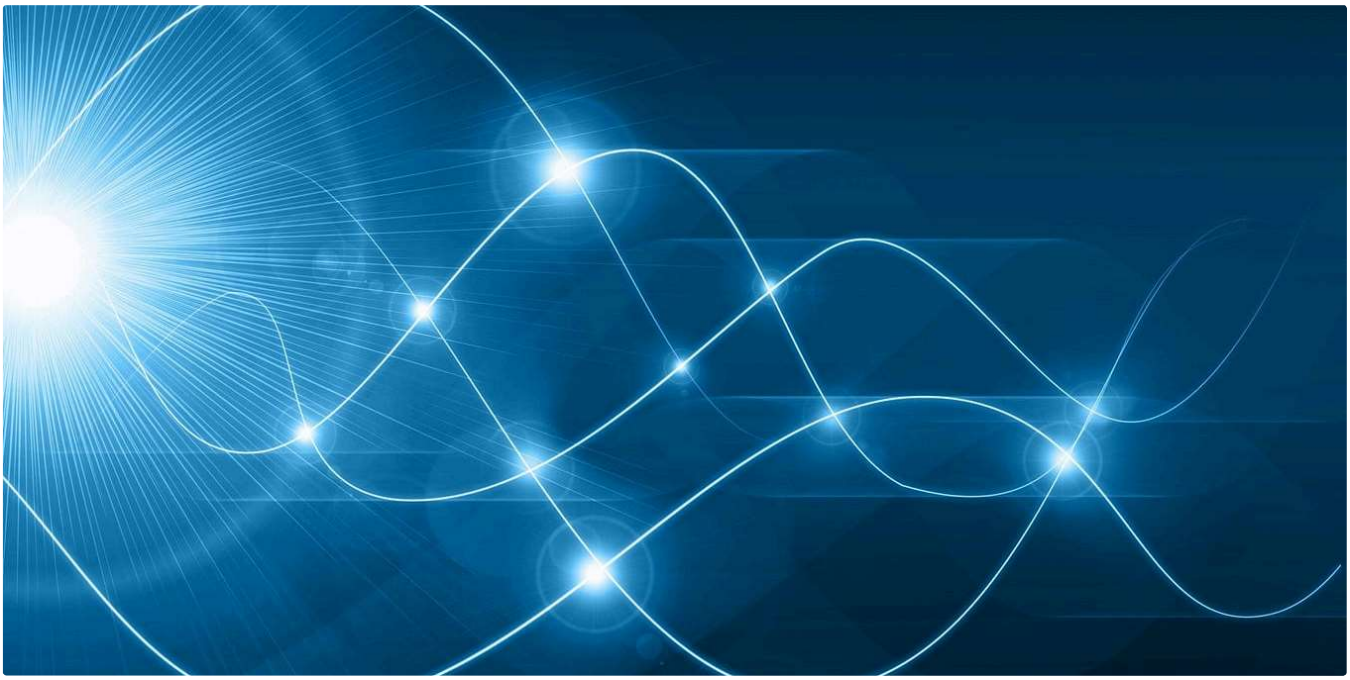
Ampofo Amoh - Gyebi in Analytics Vidhya

How to build your first Desktop Application in Python

Search the entire internet for uses of the Python programming language and they list them with Desktop Applications marked as not very...

12 min read · Dec 12, 2020

1.4K 20



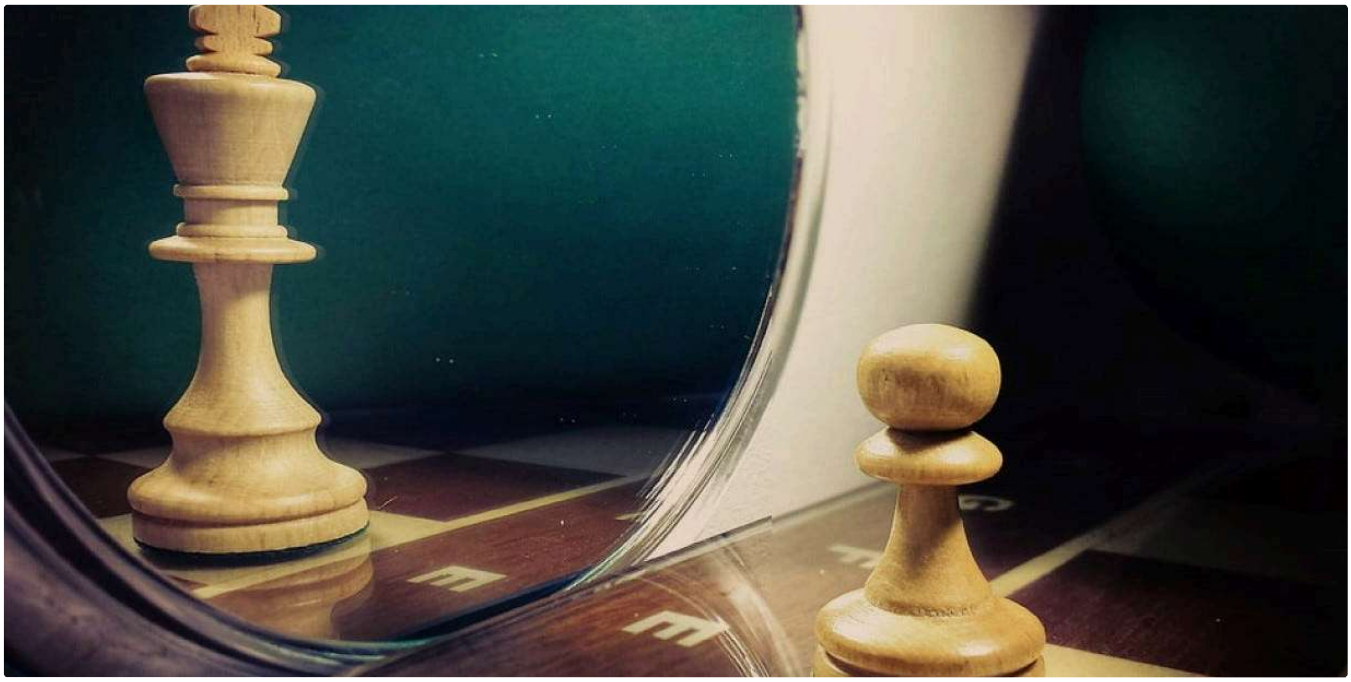
Leland Roberts in Analytics Vidhya

Understanding the Mel Spectrogram

(and Other Topics in Signal Processing)

6 min read · Mar 6, 2020

2.1K 25



Ujjawal Verma in Analytics Vidhya

R-Square(R^2) and Adjusted R-Square

Hi everyone, today we will talking about the R-Square and Adjusted R-Square, so to get more knowledge about the goodness of your linear...

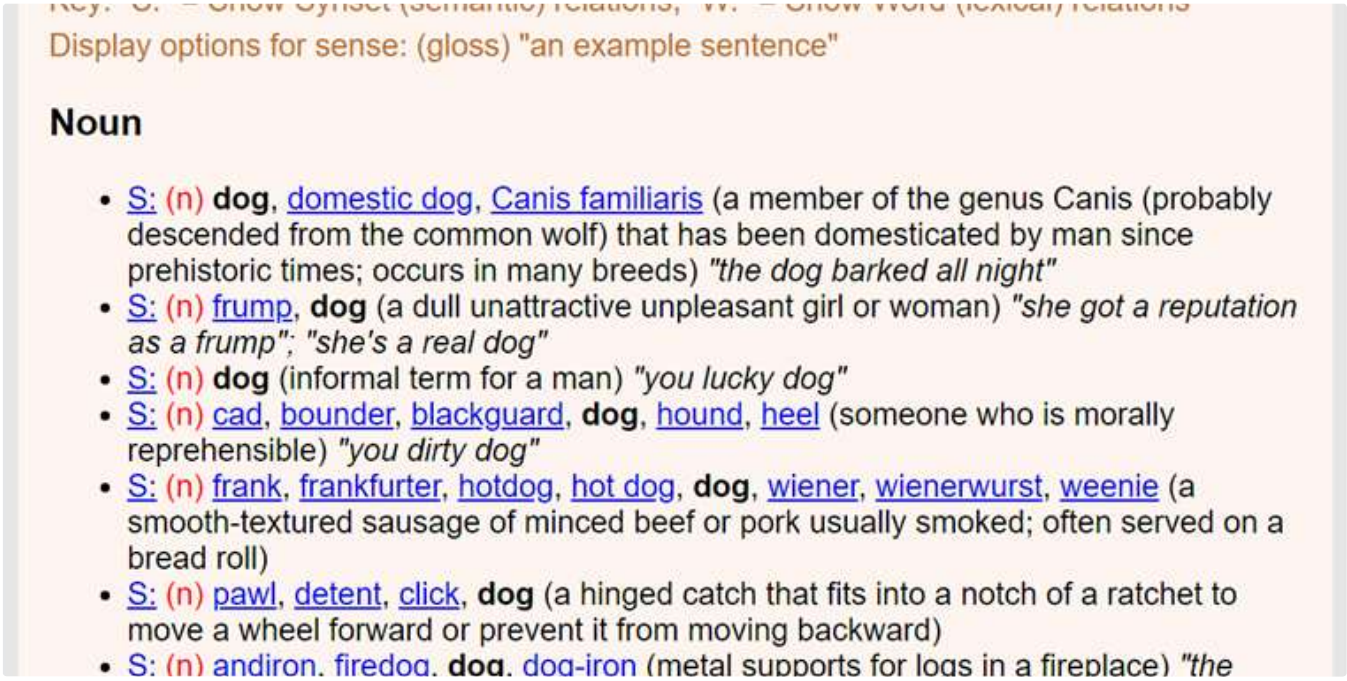
6 min read · Jan 2, 2020

138

See all from Ujjawal Verma

See all from Analytics Vidhya

Recommended from Medium



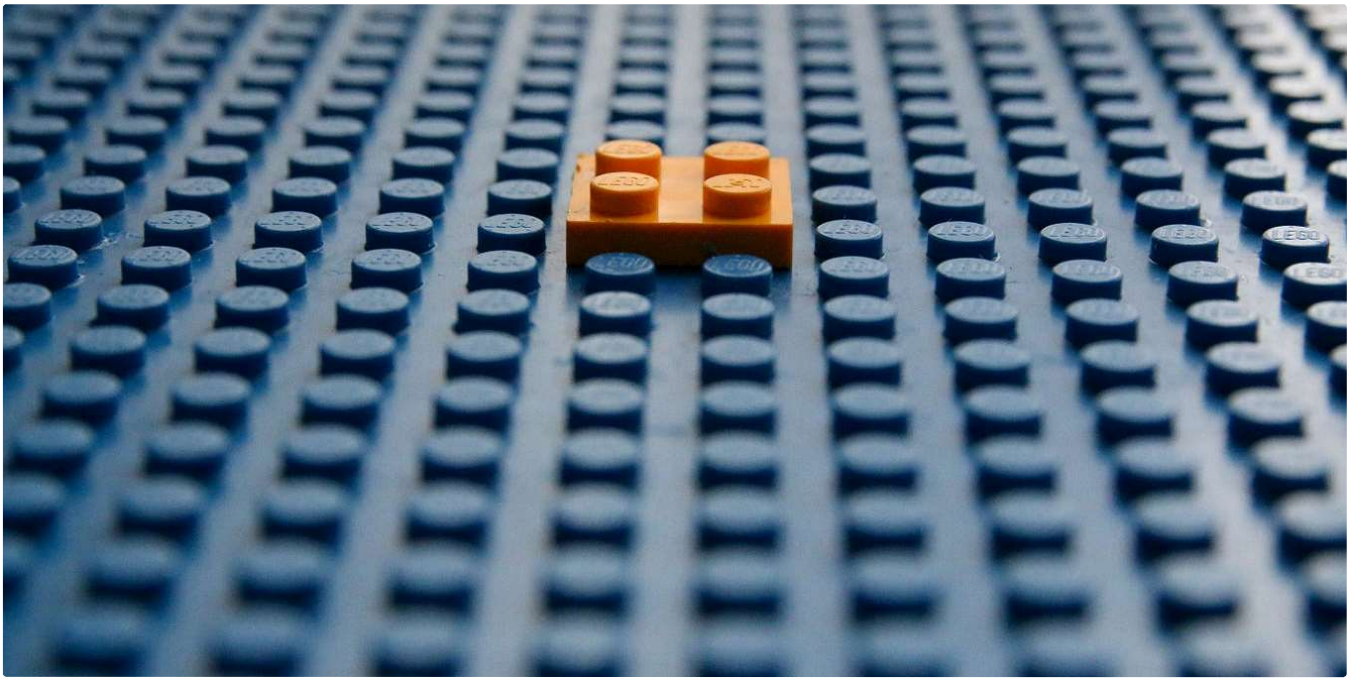
om pramod

WordNet in Action: Enhancing NLP Capabilities Through NLTK

Words might seem distinct at first glance but often share similarities within certain contexts. Consider the words “big” and “large.” At a...

2 min read · Dec 14, 2023

609



Awaldeep Singh

Understanding the Essentials: NLP Text Preprocessing Steps!

Introduction

8 min read · Dec 31, 2023

Furkan Ayık

Mastering Text Classification with BERT: A Comprehensive Guide

Understand how to build advanced classifiers with fine-tuning BERT and its variants.

10 min read · Dec 17, 2023

62

Reyhaneh Esmaielbeiki

BERT, GPT and BART: a short comparison

A visual guide for easy comparison of BERT, GPT and BART

2 min read · Nov 27, 2023

61 2

Asjad Ali

Understanding the NLP Pipeline: A Comprehensive Guide

Natural Language Processing (NLP) has emerged as a critical domain in modern technology, enabling machines to understand, interpret, and...

9 min read · Jan 1, 2024

10