4222 – SURYA GROUP OF INSTITUTIONS

VIKRAVANDI

NAANMUDHALVAN PROJECT

Subject Code - SB3001

Course Name - Experience Based Practical Learning

SENTIMENT ANALYSIS FOR MARKETING

Prepared by,

S.DHANUSH

REG. NO: 42221106018

ECE 3rd year

Problem Definition and Design Thinking

OBJECTIVE:

Sentiment analysis can do wonders for any marketer. By understanding what your target audience is thinking on a scale that only sentiment analysis can achieve, you can tweak a product, campaign, and more, to meet their needs and let your customers know you're listening.

Sentiment analysis is an artificial intelligence technique that uses machine learning and natural language processing (NLP) to analyse text for polarity of opinion (positive to negative).

DATA COLLECTION:

Depending on your research goal and scope, you can choose from different data sources and methods for sentiment analysis. Some common data sources are online reviews, social media platforms, blogs, forums, news articles, surveys, and emails. Some common methods are web scraping, APIs, surveys, and online tools.

DATA PREPROCESSING:

- ✓ The code begins by importing the necessary libraries, including pandas for data handling, matplotlib and seaborn for visualization, and scikit-learn for machine learning.
- ✓ The airline tweet dataset is loaded from a CSV file.

For cleaning the data, we will do the following:

Combine both test and training set so we can preprocess both together

Remove redundant characters- numerics, special characters (not

hashtags), short words, usernames(@user)
Tokenise the processed tweet
Stemming- strip suffixes to get the root word.

Introduction for sentiment analysis

Sentiment analysis is used to analyse raw text to drive objective quantitative results using natural language processing, machine learning, and other data analytics techniques. It is used to detect positive or negative sentiment in text, and often businesses use it to gauge branded reputation among their customers.
There are various types of sentiment analysis where the models focus on feelings and emotions, urgency, even intentions, and polarity. The most popular types of sentiment analysis are:

A. Fine-grained sentiment analysis

B. Emotion detection

C. Aspect based sentiment analysis

D. Multilingual sentiment analysis

Sentiment analysis is critical because it helps businesses to understand the emotion and sentiments of their customers. Companies analyze customers' sentiment through social media conversations and reviews so they can make better-informed decisions. The Global Sentiment Analysis Software Market is projected to reach US$4.3 billion by the year 2027. Between 2017 and 2023, the global sentiment analysis market will increase by a CAGR of 14%.

FEATURE EXTRACTION:

In sentiment analysis, we detect tweets that have negative sentiment, i.e, racist, sexist or general hate speech. Here, tweets with a label '1' denote a negative tweet, while '0' denotes the absence of hate speech in the tweet.
We extract features using the following:

1. Bag of Words Features
2. TF-IDF features
3. Word Embedding's

VISUALIZATION:

- The code creates a histogram to visualize the distribution of airline sentiments.
- It also creates a pie chart to visualize the sentiment distribution using percentages.

We will analyse the text of the tweet and its relation to the sentiment with the following:
Wordcloud: Most used words (have bigger fonts), for positive and negative tweets. Reference.
Hashtags: Analyse the effect of hashtags on the tweet sentiment.

```
all_words=' '.join([text for text in combine['tidy_tweet']])
from wordcloud import WordCloud
word_cloud=WordCloud(width=800, height=500,
random_state=21,max_font_size=110,colormap=' generate(all_words)
plt.figure(figsize=(10, 7))
plt.imshow(word_cloud)
plt.axis('off')
plt.show()
```

INSIGHTS GENERATION:

Sentiment Analysis is a natural language processing (NLP) technique used to determine the sentiment of data, i.e. whether the data is positive, negative or neutral. NLP is a branch of artificial intelligence concerning linguistics, more specifically how a computer understands, processes, and analyses large amount of natural language data. It has a vast amount of other use cases such as text classification, speech recognition, chat-bots and more, however the main focus for this project will be Sentiment Analysis. Customer Insights One of the main applications of Sentiment Analysis is for Customer Insights. A customer insight is an interpretation used by businesses to gain a deeper understanding of how their audience feels towards their product or business, allowing them to better understand their consumers needs and improve their product/service accordingly.

```
# Import Libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score, confusion_matrix
```

from sklearn.model_selection import train_test_split

visualize:

 # Visualize the distribution of airline sentiments using a pie chart
sentiment_counts = df['airline_sentiment'].value_counts()
plt.figure(figsize=(8, 8))
plt.pie(sentiment_counts, labels=sentiment_counts.index, autopct='%1.1f%%', startangle=140)
plt.title('Distribution of Airline Sentiments')
plt.axis('equal') *# Equal aspect ratio ensures that pie is drawn as a circle.*

# function to collect hashtags
def hashtag_extract(x):
hashtags = []
for i in x:
ht = re.findall(r"#(\w+)", i)
hashtags.append(ht)
return hashtags
In [22]:
linkcode
#hashtag list for non negative tweets
HT_non_negative = hashtag_extract(combine['tidy_tweet'][combine['label'] == 0])
#hashtag list for negative tweets
HT_negative = hashtag_extract(combine['tidy_tweet'][combine['label'] == 1])
#unnest list
HT_non_negative = sum(HT_non_negative,[])

HT_negative = sum(HT_negative,[])

INNOVATION

MACHINE LEARNING

Sentiment analysis is a type of machine learning tool. Machine learning works with natural language processing to make up the core building blocks of the sentiment analysis process. Surprisingly, one model that performs particularly well on sentiment analysis tasks is the convolutional neural network, which is more commonly used in computer vision models.

Machine learning approach

Machine learning involves showing a large volume of data to a machine so that it can learn and make predictions, find patterns, or classify data. The three machine learning types are supervised, unsupervised, and reinforcement learning.

- *Lexicon-based Methods.*
- *Automated/Machine Learning Methods.*
- *Hybrid approache*

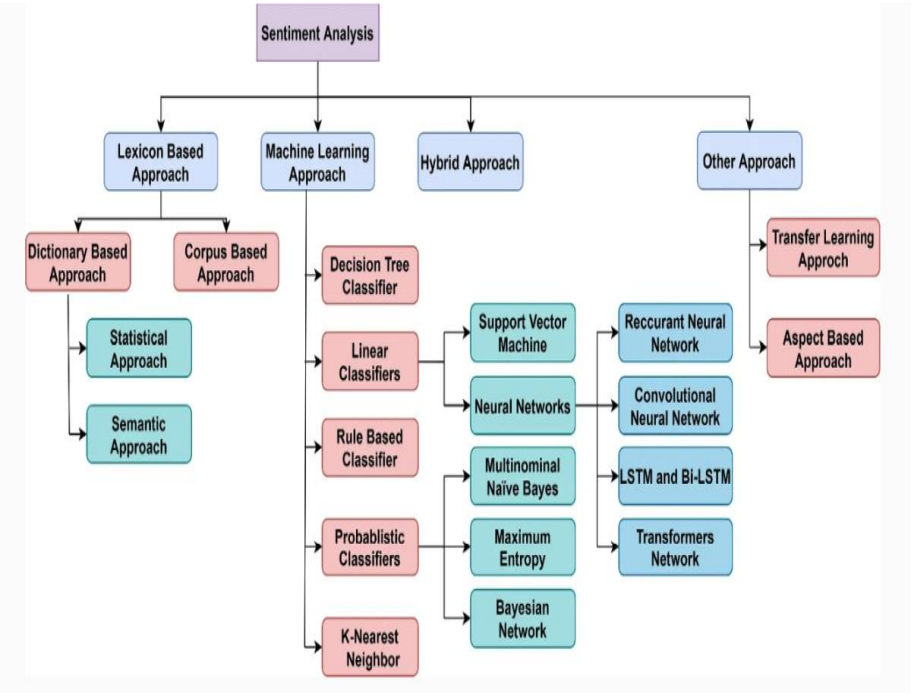


Fig:1 Sentiment Analysis

Machine learning algorithm

- *Supervised*
- *semi-supervised*
- *unsupervised*
- *reinforcement.*

Comparison to Human Prediction:

One might ask what is the difficulty of our two tasks and what level of accuracy would be considered successful. To answer the question of how hard the two tasks are, we can compare our system's performance against that of humans. We conducted an scaled-down version of the experiment where we had humans attempt the same two classification task as our models. Performance at the human level is often considered the target goal in sentiment analysis.

Best dataset for sentiment analysis

- *. Amazon Review Data.*
- *Stanford Sentiment Treebank.*
- *Financial Phrasebank.*
- *Webis-CLS-10 Dataset.*
- *CMU Multimodal Opinion Sentiment and Emotion Intensity.*
- *Yelp Polarity Reviews.*
- *WordStat Sentiment Dictionary.*
- *Sentiment Lexicons For 81 Languages*

If you have ever looked up sentiment analysis online, chances are you've come across a project analyzing Twitter feeds. This is an excellent project because there are millions of public tweets on Twitter every day, as well as being housed by various APIs that work to collect content. You can use a Twitter crawler or an API source to build a dataset of a portion of these tweets and analyze them.

A great project is to build a tweet dataset on a specific topic or hashtag, and categorize each tweet's sentiment as positive or negative, with the ultimate goal of forming an aggregated sentiment on the topic as a whole.

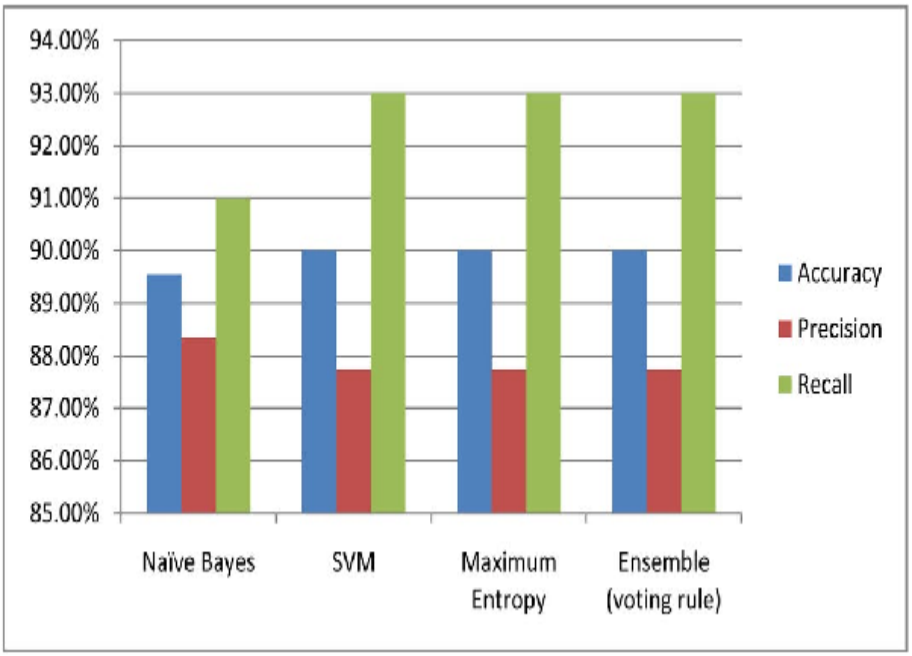


Fig. 2. Performance of Different classifiers in Twitter Sentiment Analysis

Book Sentiment Analysis

As a book lover, I always look for ways to leverage what I already love to learn new things. So, if you like books and novels, you can build a sentiment analyzer for your favorite book and learn all the basics of sentiment analysis as you do so. You can do that by downloading your favorite book as a pdf and then processing and manipulating the text. You can find a similar project using R here.

Fig:3 It is represent the book sentiment analysis

Data Cleaning

Data cleaning in sentiment analysis is the process of removing redundant and incorrect values in data that is meant for analysis. This is a necessary step in the sentiment analysis process, whatever the business requirement may be - whether customer experience analysis, employee satisfaction analytics, or brand experience insights. Removing all the unnecessary data items that do not belong in your dataset is an essential part of sentiment analysis data preparation, without which the insights you receive will be inaccurate and cannot be relied on.

fig: 4 Data cleaning process

Random Forest Algorithm

We upload the dataset (.csv file) by using the pandas API in python and we apply the TfidfVectorizer. After that we train the dataset and save the model file in the '.sav' file format. We store the fit model in .sav format for future testing

purposes. Random Forest algorithm is a supervised algorithm for both classification and regression algorithm. It computes the decision based on the highest scores of

multiple decision trees. In the following figure, we can see the implementation code of the training the dataset using Random Forest algorithm.

```
plt.scatter(x, y, color='blue') # Visualising the Random Forest Regression results

# arrange for creating a range of values
# from min value of x to max
# value of x with a difference of 0.01
# between two consecutive values
X_grid = np.arrange(min(x), max(x), 0.01)

# reshape for reshaping the data
# into a len(X_grid)*1 array,
# i.e. to make a column out of the X_grid value
X_grid = X_grid.reshape((len(X_grid), 1))

# Scatter plot for original data


# plot predicted data
plt.plot(X_grid, regressor.predict(X_grid),
         color='green')
plt.title('Random Forest Regression')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

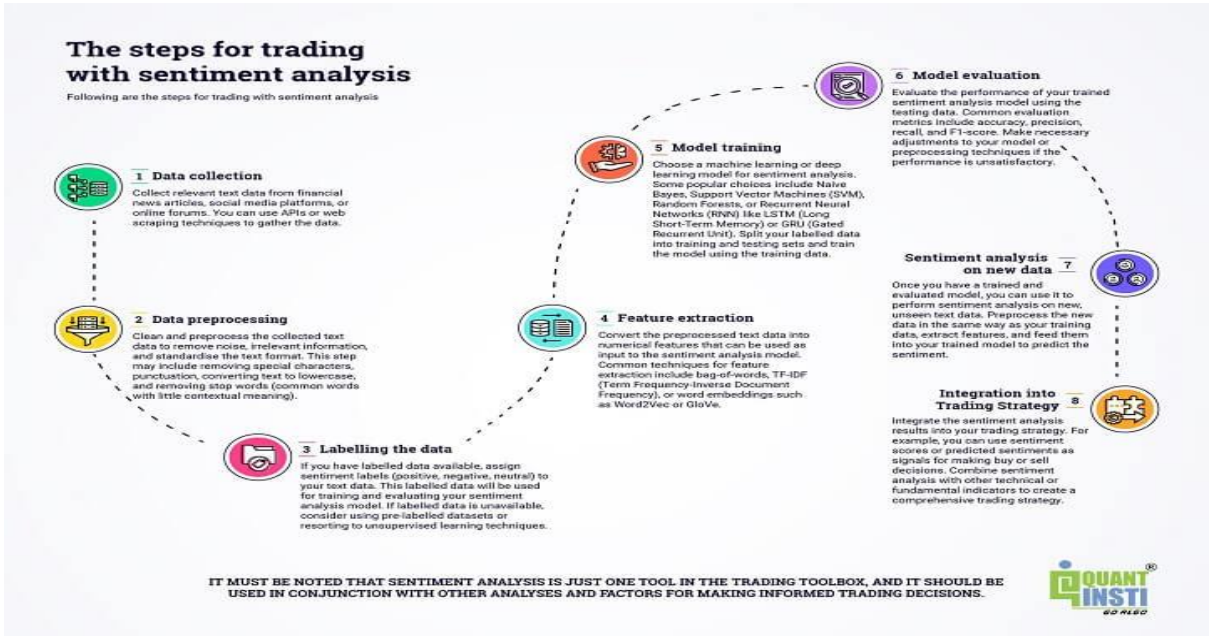


Fig:5 Steps for trading with sentiment analysis

Other Attempt:

In addition to what we used in our final model, we had other work that taught us more about extracting emotion from EP. 3 For the max label task, due to the unbalanced distribution of categories we used a balanced human testing set instead of a random subset of the original testing set. Note that this is a harder problem for our SVM classifier since it was trained on an unbalanced training set. As a result the numbers reported here are lower than the ones reported in Results.

```
print("==== Embedding Layer ====")
for p in params[0:5]:
print(".format(p[0], str(tuple(p[1].size()))))
print("==== First Transformers ====")
for p in params[5:21]:
print("{:<60} {:>12}".format(p[0], str(tuple(p[1].size()))))
print("==== Output Layer ====")
for p in params[-4:]:
print("{:<60} {:>12}".format(p[0], str(tuple(p[1].size()))))
Data Cleaning
params = list(model.named_parameters())
print("The BERT model has {:} different named parameters.".format(len(params)))
```

Developing part 1: Preprocessing

preprocessing is the process of cleaning our data set. There might be missing values oroutliers in the dataset.

PREPROCESSING THE GIVEN DATASET:

```
df = pd.read_csv('../input/covid-19-nlp-text-classification/Corona_NLP_train.csv',encoding='ISO- 8859-1')

df_test = pd.read_csv('../input/covid-19-nlp-text-classification/Corona_NLP_test.csv')
```

	UserName	ScreenName	Location	TweetAt	OriginalTweet	Sentiment
0	3799	48751	London	16-03-2020	@MeNyrbie @Phil_Gahan @Chrisitv https://t.co/i...	Neutral
1	3800	48752	UK	16-03-2020	advice Talk to your neighbours family to excha...	Positive
2	3801	48753	Vagabonds	16-03-2020	Coronavirus Australia: Woolworths to give elde...	Positive
3	3802	48754	NaN	16-03-2020	My food stock is not the only one which is emp...	Positive
4	3803	48755	NaN	16-03-2020	Me, ready to go at supermarket during the #COV...	Extremely Negative

DATA INFO:

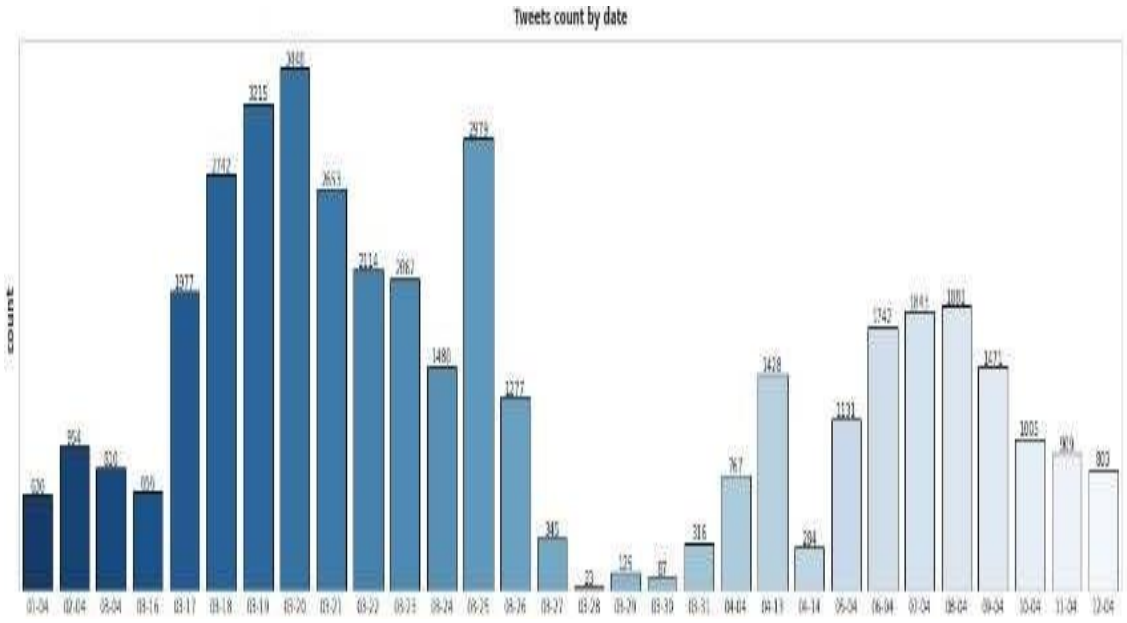
```
<class
'pandas.core.frame.DataFrame'>
RangeIndex: 41157 entries, 0 to
41156 Datacolumns (total 6
columns):
 #   Column    Non-Null Count  Dtype
--  -------        ----------------- ------

0  UserName  41157 non-null  int64
1  ScreenName     41157 non-null  int64
2  Location   32567 non-null  object
3  TweetAt    41157 non-null  object
4  OriginalTweet  41157 non-null  object  5  Sentiment    41157 non-null
   object dtypes: int64(2),object(4) memory usage: 1.9+ MB
```

DUPLICATE TWEET:

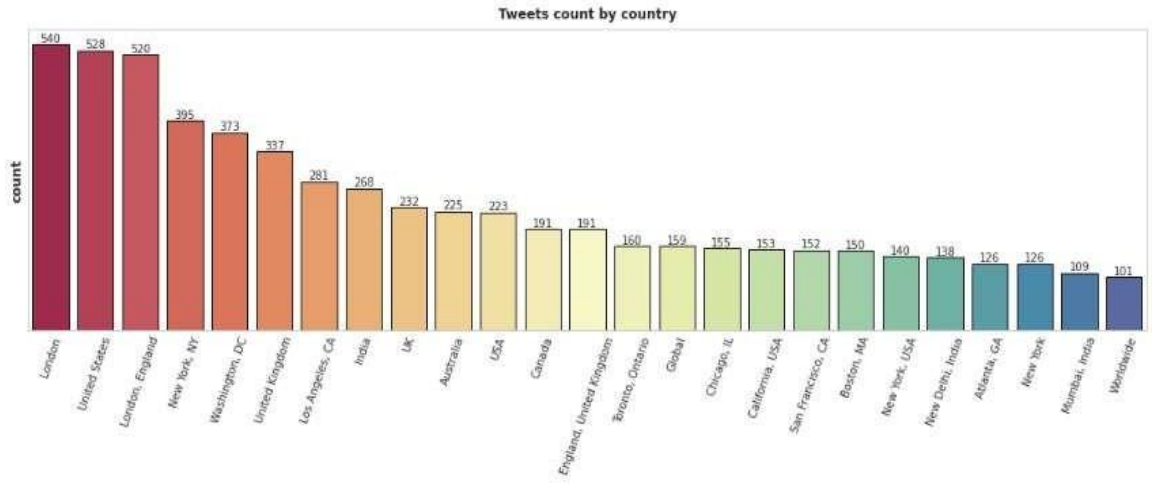
```
df.drop_duplicates(subset='OriginalTweet',inplace=True)
```

```
In [93]: df.info()
<class
'pandas.core.frame.DataFrame'>
Int64Index: 41157 entries, 0 to
41156 Datacolumns (total 6
columns):
 #   Column    Non-Null Count  Dtype
--  -------        ----------------  ------

 0  UserName  41157 non-null  int64
 1  ScreenName     41157 non-null  int64
 2  Location       32567 non-null  object
 3  TweetAt        41157 non-null  datetime64[ns]
 4  OriginalTweet  41157 non-null  object
 5  5   Sentiment41157 non-null  object


dtypes: datetime64[ns](1), int64(2), object(3)
```

TWEETS COUNT BY DATA



TWEETS PER COUNTRY AND CITY



TWEETS DEEP CLEANING

df =

df[['OriginalTweet','Sentime

nt']]

df_test = df_test[['OriginalTweet','Sentiment']]

Then we define custom functions to clean the text of the tweets.

In [100]: linkcode
##CUSTOM DEFINED FUNCTIONS TO CLEAN THE TWEETS

#Clean emojis from
text def
strip_emoji(text):

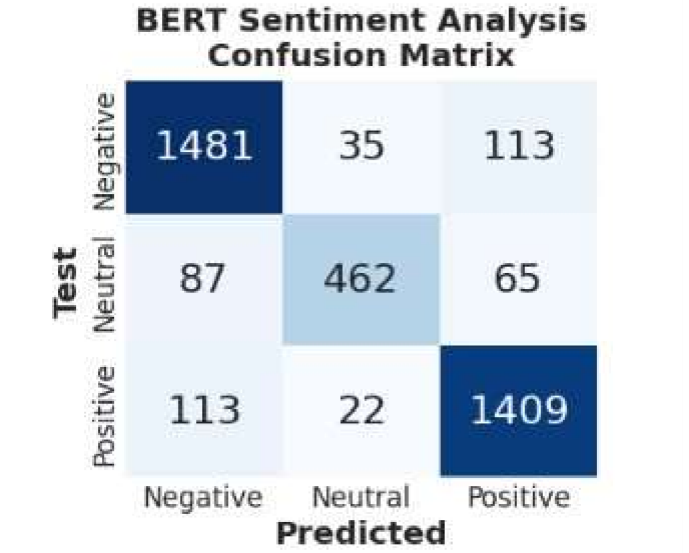
return re.sub(emoji.get_emoji_regexp(), r"", text) *#remove emoji*

#Remove punctuations, links, mentions and \r\n new line characters def
strip_all_entities(text): text = text.replace('\r', '').replace('\n', ' ').replace('\n', '
').lower() *#remove \n and \r and lowercase*text =
re.sub(r"(?:\@|https?\://)\S+", "", text) *#remove links and mentions*

text = re.sub(r'[^\x00-\x7f]',r'', text) *#remove non utf8/ascii characters such as*
'\x9a\x91\x97\x9a\x9

7'

banned_list= string.punctuation +
'Ã'+'±'+'ã'+'¼'+'â'+'»'+'§'table =
str.maketrans('', '', banned_list)
text = text.translate(table) return text



#clean hashtags at the end of the sentence, and keep those in the middle of the sentence by removing just the # symbol def clean_hashtags(tweet):

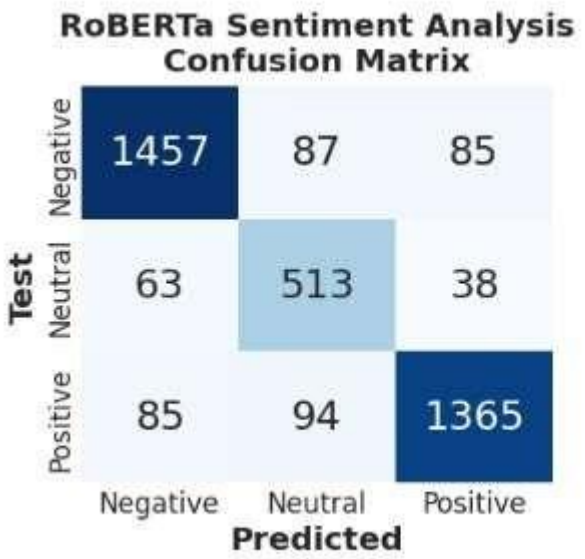
new_tweet = " ".join(word.strip() for word in re.split('#(?!(?:hashtag)\b)[\w-
]+(?=(?:\s+#[\w-]+)*\s
*$)', tweet)) *#remove last hashtags*

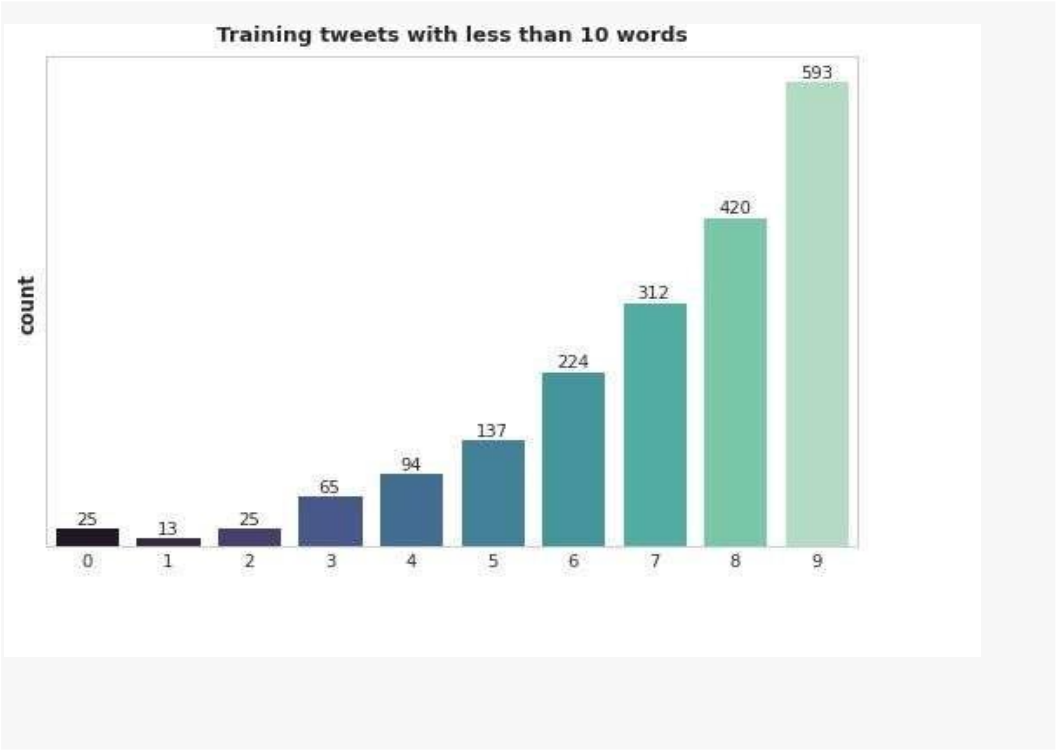
new_tweet2 = " ".join(word.strip() for word in re.split('#|_', new_tweet))
#remove hashtags symbolfrom words in the middle of the sentence return
new_tweet2

TWEETS ROBERT CONFUSION:

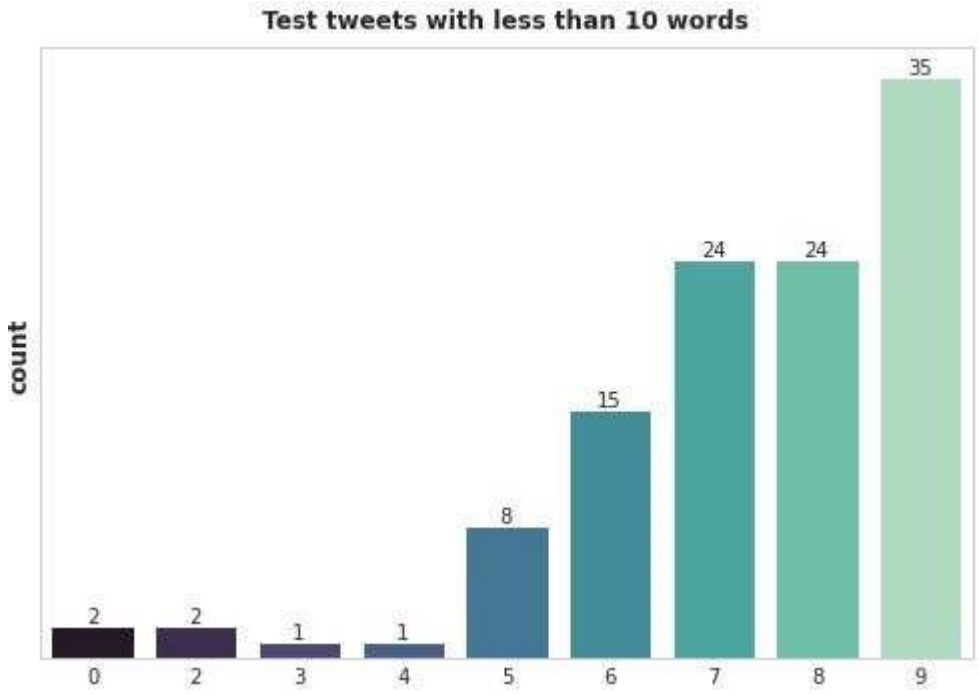
```
#Filter special characters such as & and $ present in some words
def filter_chars(a):
    sent = [] for word in
a.split(''):  if ('$' in word)
| ('&' in word):
        sent.append('')
else
:       sent.append(word)
return ' '.join(sent)

def remove_mult_spaces(text): # remove multiple spaces
return re.sub("\s\s+" , " ", text)
```

```
text_len_test = []
for text in df_test.text_clean:
    tweet_len = len(text.split())
    text_len_test.append(tweet_le
    n)
```



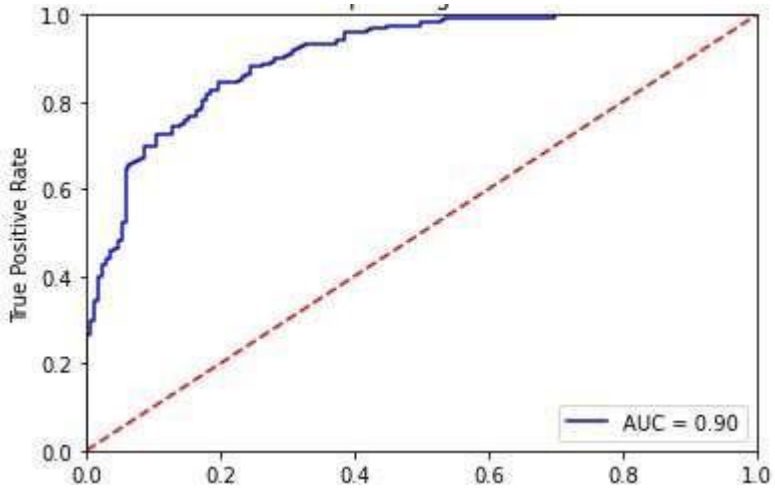



```
plt.figure(figsize=(7,5))
ax = sns.countplot(x='text_len', data=df_test[df_test['text_len']<10],
palette='mako') plt.title('Testtweets with less than 10 words')
plt.yticks([])
ax.bar_label(ax.contain
ers[0])
plt.ylabel('count')
plt.xlabel('') plt.show()
```



SENTIMENT COLUMN ANAYSIS

Positive 11381
Negative 9889
Neutral 7560
Extremely Positive 6618
Extremely Negative 5475
Name: Sentiment, dtype: int64



Development part 2

What is NLP for sentiment analysis:

Sentiment analysis (or opinion mining) is a natural language processing (NLP) technique used to determine whether data is positive, negative or neutral. Sentiment analysis is often performed on textual data to help businesses monitor brand and product sentiment in customer feedback, and understand customer needs.

What is generating insights in sentiment analysis:

Sentiment analysis solutions apply consistent criteria to generate more accurate insights. For example, a machine learning model can be trained to recognise that there are two aspects with two different sentiments. It would average the overall sentiment as neutral, but also keep track of the details.

Which algorithm is used for sentiment analysis in NLP

Overall, Sentiment analysis may involve the following types of classification algorithms: Linear Regression. Naive Bayes. Support Vector Machines.

How to use NLP to do sentiment analysis?

In sentiment analysis, Natural Language Processing (NLP) is essential. NLP uses computational methods to interpret and comprehend human language. It includes several operations, including sentiment analysis, named entity recognition, part-of-speech tagging, and tokenization.

NLP methods for sentiment analysis

1. using a dictionary of manually defined keywords,
2. creating a 'bag of words',
3. using the TF-IDF strategy.

Gaining Insights and Making Decisions with Sentiment Analysis

Well, by now I guess we are somewhat accustomed to what sentiment analysis is. But what is its significance and how do organizations benefit from it? Let us try and explore the same with an example. Assume you start a company that sells perfumes on an online platform. You put up a wide range of fragrances out there and soon customers start flooding in. After some time you decide to change the pricing strategy of perfumes—you plan to increase the prices of the popular fragrances and at the same time offer discounts on unpopular ones. Now, in order to determine which fragrances are popular, you start going through customer reviews of all the fragrances. But you're stuck! They are just so many that you cannot go through them all in one lifetime. This is where sentiment analysis can rope you out of the pit.

You simply gather all the reviews in one place and apply sentiment analysis to it. The following is a schematic representation of sentiment analysis on the reviews of three fragrances of perfumes—Lavender, Rose, and Lemon. (Please note that these reviews might have incorrect spellings, grammar, and punctuations as it is in the real-world scenarios)

From these results, we can clearly see that:

- Fragrance-1 (Lavender) has highly positive reviews by the customers which indicates your company can escalate its prices given its popularity.
- Fragrance-2 (Rose) happens to have a neutral outlook amongst the customer which means your company should not change its pricing.
- Fragrance-3 (Lemon) has an overall negative sentiment associated with it—thus, your company should consider offering a discount on it to balance the scales.

This was just a simple example of how sentiment analysis can help you gain insights into your products/services and help your organization make decisions.

Why is natural language processing important:

Businesses use massive quantities of unstructured, text-heavy data and need a way to efficiently process it. A lot of the information created online and stored in databases is natural human language, and until recently, businesses could not effectively analyze this data. This is where natural language processing is useful.

Features

The advantage of natural language processing can be seen when considering the following two statements: "Cloud computing insurance should be part of every service-level agreement," and, "A good SLA ensures an easier night's sleep -- even

in the cloud." If a user relies on natural language processing for search, the program will recognize that *cloud computing* is an entity, that *cloud* is an abbreviated form of cloud computing and that *SLA* is an industry acronym for service-level agreement.

Step 1 — Installing NLTK and Downloading the Data

You will use the NLTK package in Python for all NLP tasks in this tutorial. In this step you will install NLTK and download the sample tweets that you will use to train and test your model.

First, install the NLTK package with the pip package manager:

```
1. pip install nltk==3.3
```

This tutorial will use sample tweets that are part of the NLTK package. First, start a Python interactive session by running the following command:

```
1. pytho3
2.
```

Then, import the nltk module in the python interpreter.

```
1. import nltk
```

Download the sample tweets from the NLTK package:

```
1. nltk.download('twitter_samples')
```

Running this command from the Python interpreter downloads and stores the tweets locally. Once the samples are downloaded, they are available for your use.

You will use the negative and positive tweets to train your model on sentiment analysis later in the tutorial. The tweets with no sentiments will be used to test your model.

If you would like to use your own dataset, you can gather tweets from a specific time period, user, or hashtag by using the Twitter API.

Now that you've imported NLTK and downloaded the sample tweets, exit the interactive session by entering in exit(). You are ready to import the tweets and begin processing the data.

Step 2 — Tokenizing the Data

Language in its original form cannot be accurately processed by a machine, so you need to process the language to make it easier for the machine to understand. The

first part of making sense of the data is through a process called *tokenization*, or splitting strings into smaller parts called *tokens*.

A token is a sequence of characters in text that serves as a unit. Based on how you create the tokens, they may consist of words, emoticons, hashtags, links, or even individual characters. A basic way of breaking language into tokens is by splitting the text based on whitespace and punctuation.

To get started, create a new .py file to hold your script. This tutorial will use nlp_test.py:

```
1. nano nlp_test.py
2. In this file, you will first import the twitter_samples so you can work with that data:
```

nlp_test.py

```
from nltk.corpus import twitter_samples
```

This will import three datasets from NLTK that contain various tweets to train and test the model:

- negative_tweets.json: 5000 tweets with negative sentiments
- positive_tweets.json: 5000 tweets with positive sentiments
- tweets.20150430-223406.json: 20000 tweets with no sentiments

Next, create variables for positive_tweets, negative_tweets, and text:

nlp_test.py

```
from nltk.corpus import twitter_samples
positive_tweets = twitter_samples.strings('positive_tweets.json')
negative_tweets = twitter_samples.strings('negative_tweets.json')
text = twitter_samples.strings('tweets.20150430-223406.json')
```

The strings() method of twitter_samples will print all of the tweets within a dataset as strings. Setting the different tweet collections as a variable will make processing and testing easier.

Before using a tokenizer in NLTK, you need to download an additional resource, punkt. The punkt module is a pre-trained model that helps you tokenize words and sentences. For instance, this model knows that a name may contain a period (like “S. Daityari”) and the presence of this period in a sentence does not necessarily end it. First, start a Python interactive session:

```
1. python3
```

Run the following commands in the session to download the punkt resource:

```
1. import nltk
2. nltk.download('punkt')
```

Once the download is complete, you are ready to use NLTK's tokenizers. NLTK provides a default tokenizer for tweets with the .tokenized() method. Add a line to create an object that tokenizes the positive_tweets.json dataset:

nlp_test.py

```
from nltk.corpus import twitter_samples
positive_tweets = twitter_samples.strings('positive_tweets.json')
negative_tweets = twitter_samples.strings('negative_tweets.json')
text = twitter_samples.strings('tweets.20150430-223406.json')
tweet_tokens = twitter_samples.tokenized('positive_tweets.json')
```

If you'd like to test the script to see the .tokenized method in action, add the highlighted content to your nlp_test.py script. This will tokenize a single tweet from the positive_tweets.json dataset:

nlp_test.py

```
from nltk.corpus import twitter_samples
positive_tweets = twitter_samples.strings('positive_tweets.json')
negative_tweets = twitter_samples.strings('negative_tweets.json')
text = twitter_samples.strings('tweets.20150430-223406.json')
tweet_tokens = twitter_samples.tokenized('positive_tweets.json')[0]
print(tweet_tokens[0])
```

Save and close the file, and run the script:

```
1. python3 nlp_test.py
```

The process of tokenization takes some time because it's not a simple split on white space. After a few moments of processing, you'll see the following:

Output

```
['#FollowFriday',
 '@France_Inte',
 '@PKuchly57',
 '@Milipol_Paris',
 'for',
 'being',
 'top',
 'engaged',
 'members',
 'in',
 'my',
 'community',
 'this',
 'week',
 ':)']
```

Here, the .tokenized() method returns special characters such as @ and _. These characters will be removed through regular expressions later in this tutorial.
Now that you've seen how the .tokenized() method works, make sure to comment out or remove the last line to print the tokenized tweet from the script by adding a # to the start of the line:

nlp_test.py

```
from nltk.corpus import twitter_samples
positive_tweets = twitter_samples.strings('positive_tweets.json')
negative_tweets = twitter_samples.strings('negative_tweets.json')
text = twitter_samples.strings('tweets.20150430-223406.json')
tweet_tokens = twitter_samples.tokenized('positive_tweets.json')[0]
#print(tweet_tokens[0])
```

Your script is now configured to tokenize data. In the next step you will update the script to normalize the data.

Step 3 — Normalizing the Data

Words have different forms—for instance, "ran", "runs", and "running" are various forms of the same verb, "run". Depending on the requirement of your analysis, all of these versions may need to be converted to the same form, "run". *Normalization* in NLP is the process of converting a word to its canonical form.

Normalization helps group together words with the same meaning but different forms. Without normalization, "ran", "runs", and "running" would be treated as different words, even though you may want them to be treated as the same word. In this section, you explore *stemming* and *lemmatization*, which are two popular techniques of normalization.

Stemming is a process of removing affixes from a word. Stemming, working with only simple verb forms, is a heuristic process that removes the ends of words.

In this tutorial you will use the process of lemmatization, which normalizes a word with the context of vocabulary and morphological analysis of words in text. The lemmatization algorithm analyzes the structure of the word and its context to convert it to a normalized form. Therefore, it comes at a cost of speed. A comparison of stemming and lemmatization ultimately comes down to a trade off between speed and accuracy.

Before you proceed to use lemmatization, download the necessary resources by entering the following in to a Python interactive session:

```
1. python3
```

Run the following commands in the session to download the resources:

```
1. import nltk
2. nltk.download('wordnet')
3. nltk.download('averaged_perceptron_tagger')
```

Copy

wordnet is a lexical database for the English language that helps the script determine the base word. You need the averaged_perceptron_tagger resource to determine the context of a word in a sentence.

Once downloaded, you are almost ready to use the lemmatizer. Before running a lemmatizer, you need to determine the context for each word in your text. This is achieved by a tagging algorithm, which assesses the relative position of a word in a sentence. In a Python session, Import the pos_tag function, and provide a list of tokens as an argument to get the tags. Let us try this out in Python:

```
1. from nltk.tag import pos_tag
2. from nltk.corpus import twitter_samples
3. tweet_tokens = twitter_samples.tokenized('positive_tweets.json')
4. print(pos_tag(tweet_tokens[0]))
```

/

Here is the output of the pos_tag function.

```
Output
[('#FollowFriday', 'JJ'),
 ('@France_Inte', 'NNP'),
 ('@PKuchly57', 'NNP'),
 ('@Milipol_Paris', 'NNP'),
 ('for', 'IN'),
 ('being', 'VBG'),
 ('top', 'JJ'),
 ('engaged', 'VBN'),
 ('members', 'NNS'),
 ('in', 'IN'),
 ('my', 'PRP$'),
 ('community', 'NN'),
 ('this', 'DT'),
 ('week', 'NN'),
 (':)', 'NN')]
```

From the list of tags, here is the list of the most common items and their meaning:

- NNP: Noun, proper, singular
- NN: Noun, common, singular or mass
- IN: Preposition or conjunction, subordinating
- VBG: Verb, gerund or present participle

- VBN: Verb, past participle

Here is a full list of the dataset.
In general, if a tag starts with NN, the word is a noun and if it stars with VB, the word is a verb. After reviewing the tags, exit the Python session by entering exit().

To incorporate this into a function that normalizes a sentence, you should first generate the tags for each token in the text, and then lemmatize each word using the tag.

Update the nlp_test.py file with the following function that lemmatizes a sentence:

nlp_test.py

```
...

from nltk.tag import pos_tag
from nltk.stem.wordnet import WordNetLemmatizer

def lemmatize_sentence(tokens):
    lemmatizer = WordNetLemmatizer()
    lemmatized_sentence = []
    for word, tag in pos_tag(tokens):
        if tag.startswith('NN'):
            pos = 'n'
        elif tag.startswith('VB'):
            pos = 'v'
        else:
            pos = 'a'
        lemmatized_sentence.append(lemmatizer.lemmatize(word, pos))
    return lemmatized_sentence

print(lemmatize_sentence(tweet_tokens[0]))
```

This code imports the WordNetLemmatizer class and initializes it to a variable, lemmatizer.
The function lemmatize_sentence first gets the position tag of each token of a tweet. Within the if statement, if the tag starts with NN, the token is assigned as a noun. Similarly, if the tag starts with VB, the token is assigned as a verb.

Save and close the file, and run the script:

```
1. python3 nlp_test.py
```

Here is the output:

```
Output
['#FollowFriday',
 '@France_Inte',
 '@PKuchly57',
 '@Milipol_Paris',
 'for',
 'be',
 'top',
 'engage',
 'member',
 'in',
 'my',
 'community',
 'this',
 'week',
 ':)']
```

You will notice that the verb being changes to its root form, be, and the noun members changes to member. Before you proceed, comment out the last line that prints the sample tweet from the script.

Now that you have successfully created a function to normalize words, you are ready to move on to remove noise.

Step 4 — Removing Noise from the Data

In this step, you will remove noise from the dataset. *Noise* is any part of the text that does not add meaning or information to data.

Noise is specific to each project, so what constitutes noise in one project may not be in a different project. For instance, the most common words in a language are called *stop words*. Some examples of stop words are “is”, “the”, and “a”. They are generally irrelevant when processing language, unless a specific use case warrants their inclusion.

In this tutorial, you will use regular expressions in Python to search for and remove these items:

- Hyperlinks - All hyperlinks in Twitter are converted to the URL shortener t.co. Therefore, keeping them in the text processing would not add any value to the analysis.
- Twitter handles in replies - These Twitter usernames are preceded by a @ symbol, which does not convey any meaning.

- Punctuation and special characters - While these often provide context to textual data, this context is often difficult to process. For simplicity, you will remove all punctuation and special characters from tweets.

To remove hyperlinks, you need to first search for a substring that matches a URL starting with http:// or https://, followed by letters, numbers, or special characters. Once a pattern is matched, the .sub() method replaces it with an empty string. Since we will normalize word forms within the remove_nois

Conclusion

Sentiment analysis is a technique used to understand the emotional tone of the text. It can be used to identify positive, negative, and neutral sentiments in a piece of writing

Sentiment analysis focuses on the polarity of a text (positive, negative, neutral), but it also goes beyond polarity to detect specific feelings and emotions (angry, happy, sad, etc.), urgency (urgent, not urgent), and even intentions (interested vs. not interested).

What is the goal of sentiment analysis:

Sentiment analysis is a technique used to understand the emotional tone of the text. It can be used to identify positive, negative, and neutral sentiments in a piece of writing

Sentiment analysis focuses on the polarity of a text (positive, negative, neutral), but it also goes beyond polarity to detect specific feelings and emotions (angry, happy, sad, etc.), urgency (urgent, not urgent), and even intentions (interested vs. not interested).

What is future scope of sentiment analysis:

However, sentiment analysis will delve deeper in the future, beyond the concept of positive, negative, or neutral, to reach and comprehend the significance of understanding conversations and what they reveal about consumers.