Twitter Sentiment Analysis

Minor Project-1

ABSTRACT

With the advancement of web technology and its growth, there is a huge volume of data present in the web for internet users and a lot of data is generated too. The Internet has become a platform for online learning, exchanging ideas and sharing opinions. Social networking sites like Twitter, Facebook, Google+ are rapidly gaining popularity as they allow people to share and express their views about topics, have discussions with different communities, or post messages across the world. There has been a lot of work in the field of sentiment analysis of twitter data. This project focuses mainly on sentiment analysis of twitter data which is helpful to analyze the information in the tweets where opinions are highly unstructured, heterogeneous and are either positive or negative, or neutral in some cases. In this project, we aim to tackle the problem of analysing the sentiment of tweets tweeted by twitter users.

Dataset used in this study is *1.6 million* tweets from kaggle.We are going to categorize the tweets as positive or negative sentiments. At last, we also give insight into our future work on sentiment analysis.

LIST OF ABBREVIATIONS

API	Application Programming Interface
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
ORM	Object-Relational Mapping
POS	Parts-Of-Speech
SA	Sentiment Analysis
SQL	Structured Query Language
SVM	Support Vector Machines
GUI	Graphical User Interface

LIST OF FIGURES

Table of Contents

INTRODUCTION

Nowadays, the age of the Internet has changed the way people express their views, opinions. It is now mainly done through blog posts, online forums, product review websites, social media ,etc. Nowadays, millions of people are using social network sites like Facebook, Twitter, Google Plus, etc. to express their emotions, opinions and share views about their daily lives. Through the online communities, we get an interactive media where consumers inform and influence others through forums. Social media is generating a large volume of sentiment rich data in the form of tweets, status updates, blog posts, comments, reviews, etc. Moreover, social media provides an opportunity for businesses by giving a platform to connect with their customers for advertising. People mostly depend upon user generated content over online to a great extent for decision making. For e.g. if someone wants to buy a product or wants to use any service, then they firstly look up its reviews online, discuss it on social media before making a decision. The amount of content generated by users is too vast for a normal user to analyze. So there is a need to automate this, various sentiment analysis techniques are widely used.

Sentiment analysis (SA) tells users whether the information about the product is satisfactory or not before they buy it. Marketers and firms use this analysis data to understand about their products or services in such a way that it can be offered as per the user‟s requirements. Textual Information retrieval techniques mainly focus on processing, searching or analyzing the factual data present. Facts have an objective component but there are some other textual contents which express subjective characteristics. These contents are mainly opinions, sentiments, appraisals, attitudes, and emotions, which form the core of Sentiment Analysis (SA). It offers many challenging opportunities to develop new applications, mainly due to the huge growth of available information on online sources like blogs and social networks. For example, recommendations of items proposed by a recommendation system can be predicted by taking into account considerations such as positive or negative opinions about those items by making use of SA

BACKGROUND STUDY

In recent years a lot of work has been done in the field of ‘’**Sentiment Analysis on Twitter** ‘’ by a number of researchers. In its early stage it was intended for binary classification which assigns opinions or reviews to bipolar classes such as positive or negative only.

Pak and Paroubek (2010) [1] proposed a model to classify the tweets as objective, positive and negative. They created a twitter corpus by collecting tweets using Twitter API and automatically annotating those tweets using emoticons. Using that corpus, they developed a sentiment classifier based on the multinomial Naive Bayes method that uses features like Ngram and POS-tags. The training set they used was less efficient since it contains only tweets having emoticons.

[2]Parikh and Movassate (2009) [2] implemented two models, a Naive Bayes bigram model and a Maximum Entropy model to classify tweets. They found that the Naive Bayes classifiers worked much better than the Maximum Entropy model.

[3]Go and L.Huang (2009) [3] proposed a solution for sentiment analysis for twitter data by using distant supervision, in which their training data consisted of tweets with emoticons which served as noisy labels. They build models using Naive Bayes, MaxEnt and Support Vector Machines (SVM). Their feature space consisted of unigrams, bigrams and POS. They concluded that SVM outperformed other models and that unigram was more effective as features.

[4]Barbosa et al. (2010) [4] designed a two phase automatic sentiment analysis method for classifying tweets. They classified tweets as objective or subjective and then in the second phase, the subjective tweets were classified as positive or negative. The feature space used included retweets, hashtags, link, punctuation and exclamation marks in conjunction with features like prior polarity of words and POS.

[5]Bifet and Frank (2010) [5] used Twitter streaming data provided by Firehouse API , which gave all messages from every user which are publicly available in real-time. They experimented with multinomial naive Bayes, stochastic gradient descent, and the Hoeffding tree. They arrived at a conclusion that the SGD-based model, when used with an appropriate learning rate was better than the rest used.

[6]Agarwal et al. (2011) [6] developed a 3-way model for classifying sentiment into positive, negative and neutral classes. They experimented with models such as: unigram model, a feature based model and a tree kernel based model. For the tree kernel based model they represented tweets as a tree.The feature based model uses 100 features and the unigram model uses over 10,000 features. They arrived at a conclusion that features which combine prior polarity of words with their parts-of-speech(pos) tags are most important and play a major role in the classification task. The tree kernel based model outperformed the other two models.

[7]Davidov et al.(2010) [7] proposed an approach to utilize Twitter user-defined hashtags in tweets as a classification of sentiment type using punctuation, single words, n-grams and patterns as different feature types, which are then combined into a single feature vector for sentiment classification. They made use of the K-Nearest Neighbor strategy to assign sentiment labels by constructing a feature vector for each example in the training and test set.

[8]Po-Wei Liang et.al.(2014) [8] used Twitter API to collect twitter data. Their training data falls in three different categories (camera, movie , mobile). The data is labeled as positive, negative and non-opinions. Tweets containing opinions were filtered. The Unigram Naive Bayes model was implemented and the Naive Bayes simplifying independence assumption was employed. They also eliminated useless features by using the Mutual Information and Chi square feature extraction method. Finally , the orientation of a tweet is predicted. i.e. positive or negative.

[9]Pablo et. al. [9] presented variations of Naive Bayes classifiers for detecting polarity of English tweets. Two different variants of Naive Bayes classifiers were built namely Baseline (trained to classify tweets as positive, negative and neutral), and Binary (makes use of a polarity lexicon and classifies as positive and negative. Neutral tweets neglected). The features considered by classifiers were Lemmas (nouns, verbs, adjectives and adverbs), Polarity Lexicons, and Multiword from different sources and Valence Shifters.

[11]Turney et al [10] used the bag-of-words method for sentiment analysis in which the relationships between words was not at all considered and a document is represented as just a collection of words. To determine the sentiment for the whole document, sentiments of every word were determined and those values are united with some aggregation functions.

[12]Kamps et al. [11] used the lexical database WordNet to determine the emotional content of a word along different dimensions. They developed a distance metric on WordNet and determined the semantic polarity of adjectives.

[13]Xia et al. [12] used an ensemble framework for Sentiment Classification which is obtained by combining various feature sets and classification techniques. In their work, they used two types of feature sets (Part-of-speech information and World Relations) and three base classifiers (Naive Bayes, Maximum Entropy and Support Vector Machines) . They applied ensemble approaches like fixed combination, weighted combination and Meta-classifier combination for sentiment classification and obtained better accuracy.

[14]Louet. al. [13] highlighted the challenges and an efficient technique to mine opinions from Twitter tweets. Spam and wildly varying language makes opinion retrieval within Twitter a challenging task.

REQUIREMENT ANALYSIS

3.1. Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis.

3.2. NLTK

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

3.3. Pickle

Python pickle module is used for serializing and de-serializing a Python object structure . Any object in Python can be pickled so that it can be saved on disk. What pickle does is that it "serializes" the object first before writing it to file. Pickling is a way to convert a python object (list, dict, etc.) into a character stream.

3.4. Flask

Flask is an API of Python that allows us to build up web-applications. It was developed by Armin Ronacher. Flask's framework is more explicit than Django's framework and is also easier to learn because it has less base code to implement a simple web-Application.

3.5. Flask-SQLalchemy

Flask SQLAlchemy is an ORM tool which establishes the relationship between the objects and the tables of the relational databases.

The mapping between the both is important because the python is capable of storing the data in the form of objects whereas the database stores the data in the form of relational tables, i.e. the collection of rows and columns.

DETAILED DESIGN AND IMPLEMENTATION

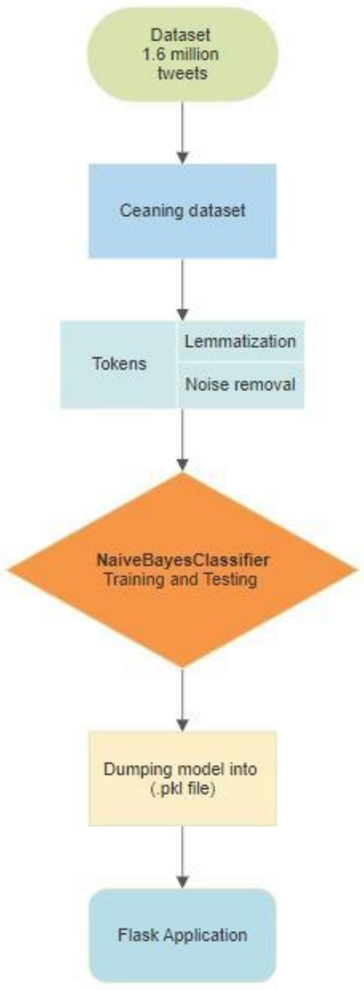
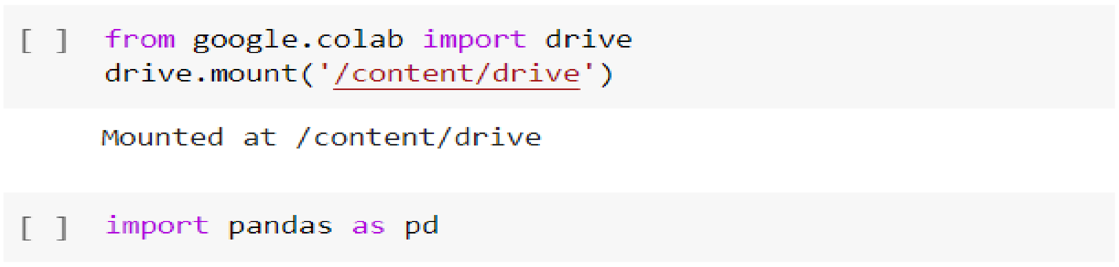

Dataset 1.6 million tweets
Ceaning dataset
Tokens
Lemmatization
Noise removal
NaiveBayesClassifier Training and Testing
Dumping model into (.pkl file)
Flask Application

Fig 1- Schematic diagram of project design

First the dataset of 1.6 million tweets is manipulated using the python library (Pandas).After specifying column names the unwanted features are dropped and then every tweet is divided into tokens after that lemmatization technique is applied and stop words,hyperlinks,punctuations are removed then the model is trained using naive bayes classifier with 70% training and 30% testing data.After that the model is serialized into a pkl file using pickle library.
After that flask is introduced along with sqlite3 as a database for deployment of the model.

4.1. Organizing Dataset

Uploading dataset from drive in google colaboratory
Dataset specification
0 for negative and 4 for positive.
A total of 1.6 million tweets with 799k negative tweets and 800k positive tweets

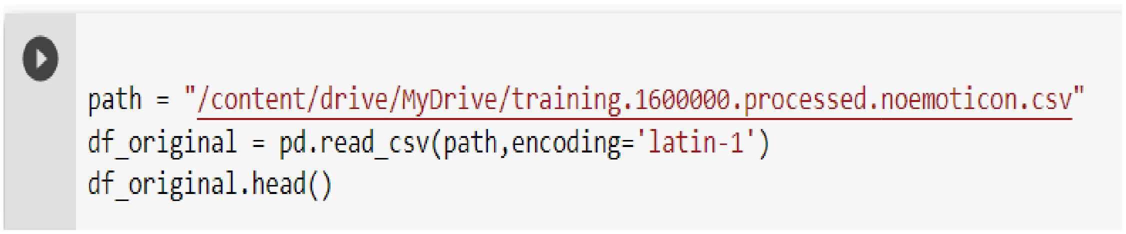


```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

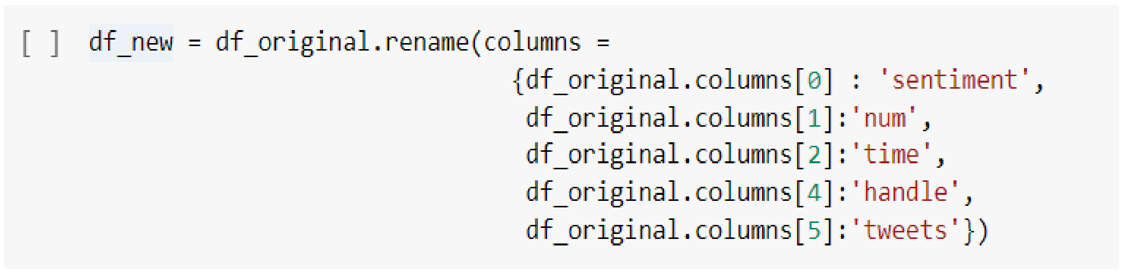
```
import pandas as pd
```

Fig 2- Importing pandas for dataset manipulation



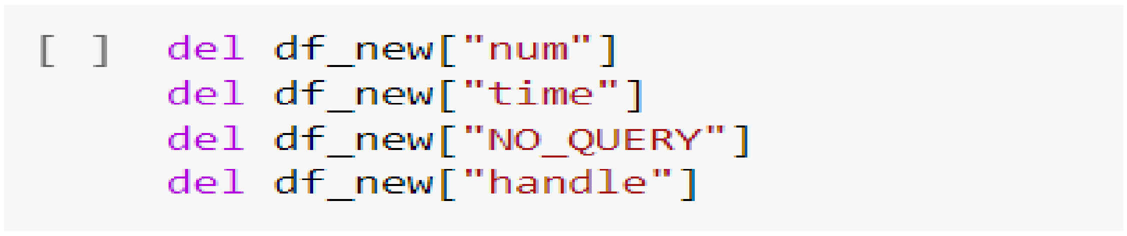
```
path = "/content/drive/MyDrive/training.1600000.processed.noemoticon.csv"
df_original = pd.read_csv(path,encoding='latin-1')
df_original.head()
```

Fig 3- Providing the path of the dataset from drive



```
df_new = df_original.rename(columns =
                               {df_original.columns[0] : 'sentiment',
                                df_original.columns[1]:'num',
                                df_original.columns[2]:'time',
                                df_original.columns[4]:'handle',
                                df_original.columns[5]:'tweets'})
```

Fig 4- Specifying names to the columns in the dataset

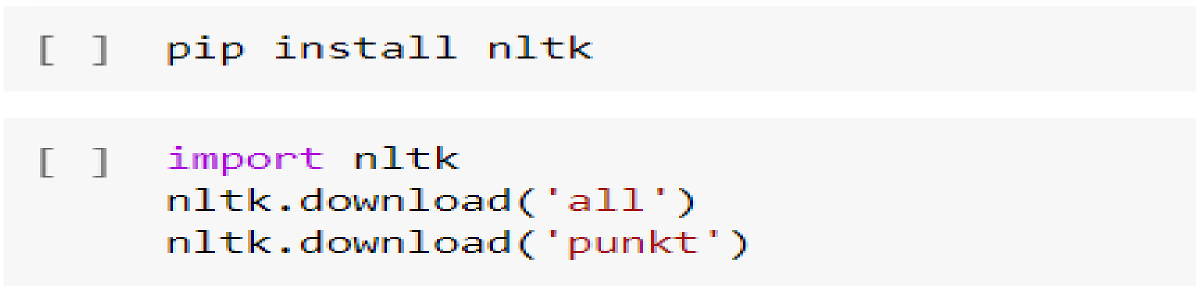


```
del df_new["num"]
del df_new["time"]
del df_new["NO_QUERY"]
del df_new["handle"]
```

Fig 5- Dropping unnecessary columns from the dataset

4.2. Installing NLTK

Using the NLTK package in Python for all NLP tasks for sentiment analysis.

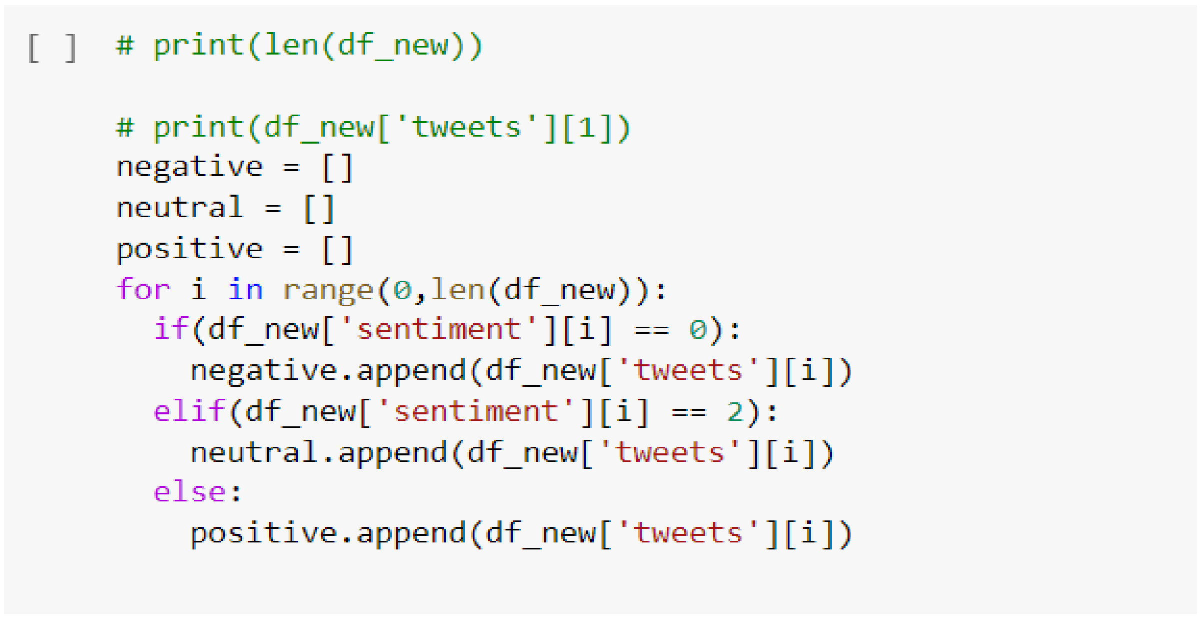


```
[ ]  pip install nltk
```

```
[ ]  import nltk
     nltk.download('all')
     nltk.download('punkt')
```

Fig 6 - Installing nltk

Now we are separating positive and negative tweets according to the sentiment column specification

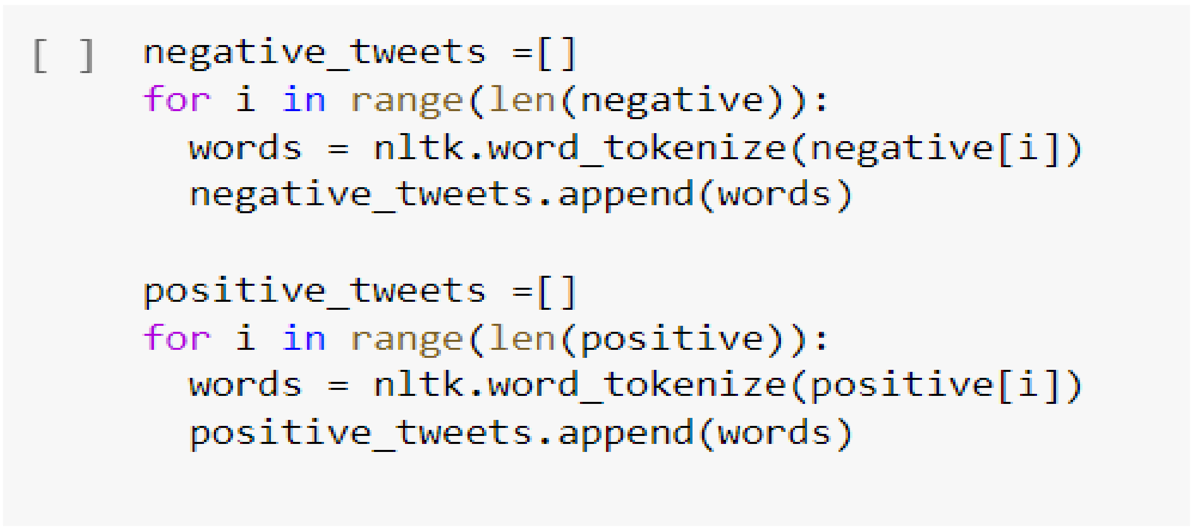


```
[ ]  # print(len(df_new))

     # print(df_new['tweets'][1])
     negative = []
     neutral = []
     positive = []
     for i in range(0,len(df_new)):
       if(df_new['sentiment'][i] == 0):
         negative.append(df_new['tweets'][i])
       elif(df_new['sentiment'][i] == 2):
         neutral.append(df_new['tweets'][i])
       else:
         positive.append(df_new['tweets'][i])
```

Fig 7- Separating positive and negative tweets

4.3. <u>Tokenizing the data</u>

Language in its original form cannot be accurately processed by a machine, so we 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.



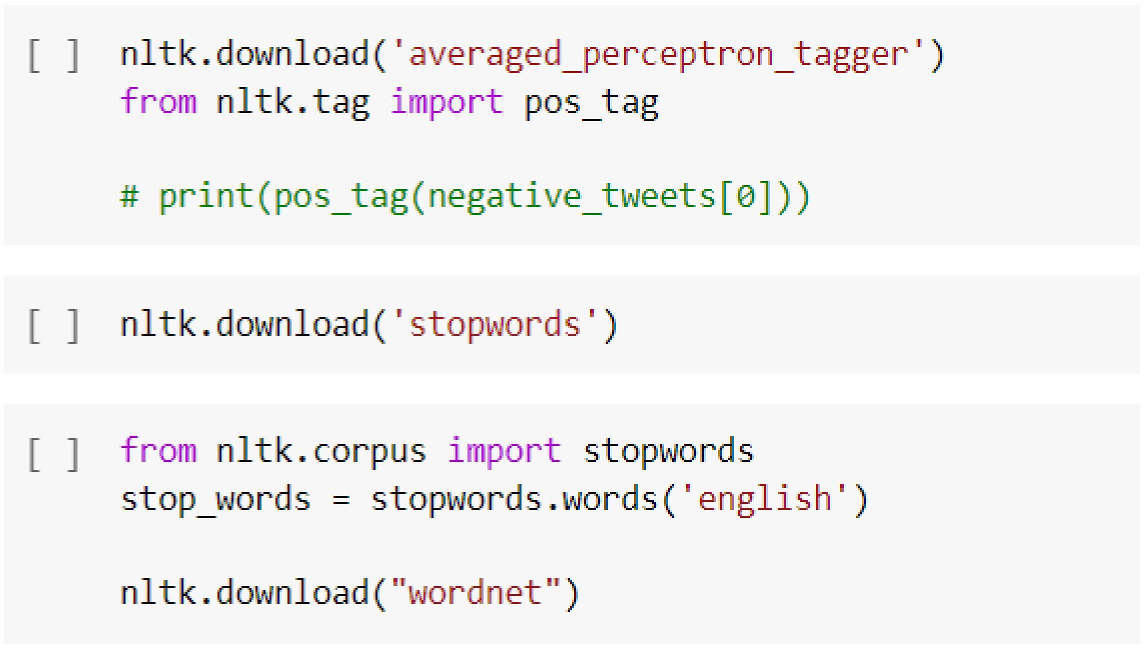
```
[ ]  negative_tweets =[]
     for i in range(len(negative)):
       words = nltk.word_tokenize(negative[i])
       negative_tweets.append(words)

     positive_tweets =[]
     for i in range(len(positive)):
       words = nltk.word_tokenize(positive[i])
       positive_tweets.append(words)
```

Fig 8- Tokenizing the tweets

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.

4.4. 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 analysis ; It might be required to convert all of these into the same form i.e. “run”. Normalization in NLP is the process of converting a word to its canonical form.



```
[ ]  nltk.download('averaged_perceptron_tagger')
     from nltk.tag import pos_tag

     # print(pos_tag(negative_tweets[0]))
```

```
[ ]  nltk.download('stopwords')
```

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

     nltk.download("wordnet")
```

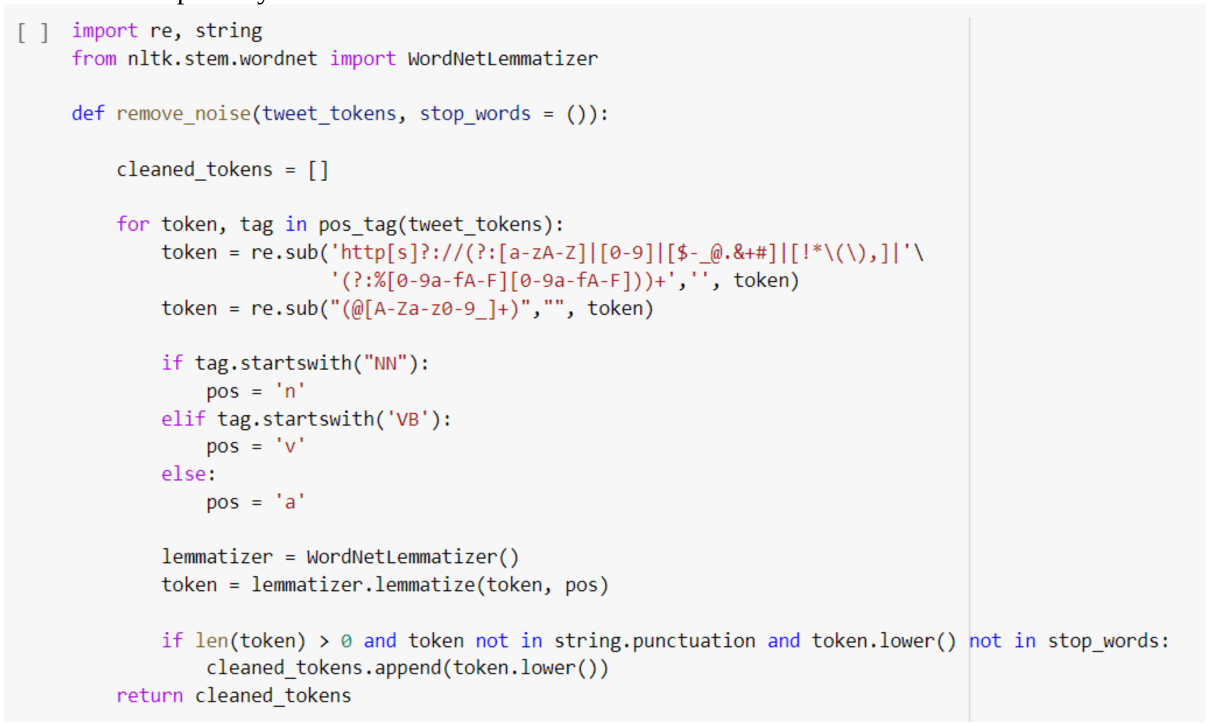
Fig 9- Normalising the tweets

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
VBM: Verb, past participle Function which will lemmatize a sentence:

4.5. Removing noise from data

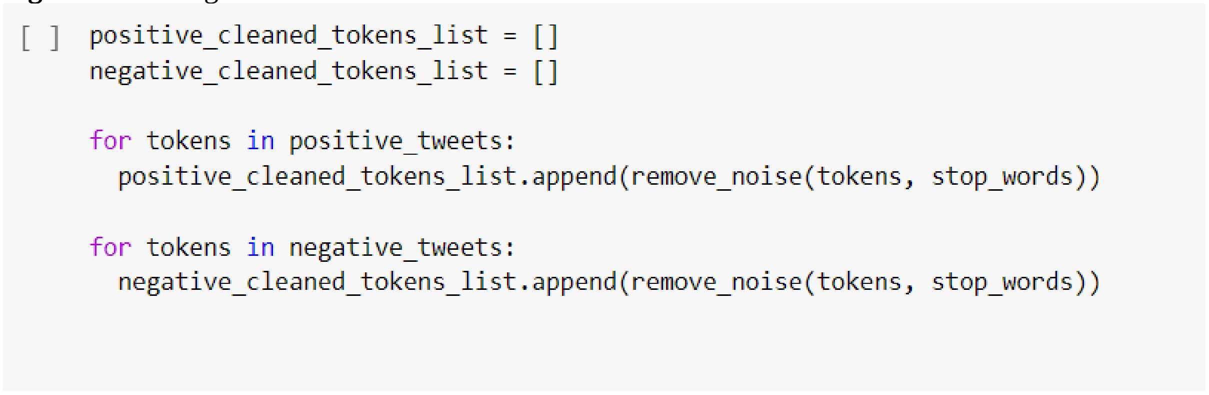
Removing noise from the dataset. Noise is any part of the text that does not add meaning or information to data.

In addition to this, Removal stop words using a built-in set of stop words in NLTK, which needs to be downloaded separately.



```
[ ]  import re, string
     from nltk.stem.wordnet import WordNetLemmatizer

     def remove_noise(tweet_tokens, stop_words = ()):

         cleaned_tokens = []

         for token, tag in pos_tag(tweet_tokens):
             token = re.sub('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+#]|[!*\(\),]|'\
                            '(?:%[0-9a-fA-F][0-9a-fA-F]))+','', token)
             token = re.sub("(@[A-Za-z0-9_]+)","", token)

             if tag.startswith("NN"):
                 pos = 'n'
             elif tag.startswith('VB'):
                 pos = 'v'
             else:
                 pos = 'a'

             lemmatizer = WordNetLemmatizer()
             token = lemmatizer.lemmatize(token, pos)

             if len(token) > 0 and token not in string.punctuation and token.lower() not in stop_words:
                 cleaned_tokens.append(token.lower())
         return cleaned_tokens
```

Fig 10 -Removing Noise



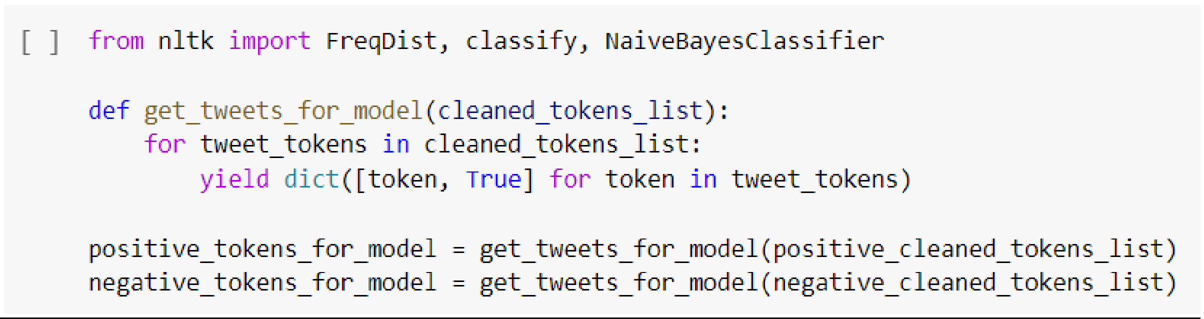
```
[ ]  positive_cleaned_tokens_list = []
     negative_cleaned_tokens_list = []

     for tokens in positive_tweets:
       positive_cleaned_tokens_list.append(remove_noise(tokens, stop_words))

     for tokens in negative_tweets:
       negative_cleaned_tokens_list.append(remove_noise(tokens, stop_words))
```

Fig 11- Cleaned list created

4.6. Preparing data for model

Sentiment analysis is a process of identifying an attitude of the author on a topic that is being written about. We will create a training data set to train a model. It is a supervised learning machine learning process, which requires to associate each dataset with a "sentiment" for training. Here, our model will use the "positive" and "negative" sentiments. We will split the dataset into two parts. The purpose of the first part is to build the model(Training), whereas the next part tests the performance of the model(Testing). In the data preparation step, we will prepare the data for sentiment analysis by converting tokens to the dictionary form and then split the data for training and testing purposes.

Converting Tokens to a Dictionary First, we will prepare the data to be fed into the model. We will use the Naive Bayes classifier in NLTK to perform the modeling exercise.The model will require a list of words in a tweet and a Python dictionary with words as keys and True as values.We will add the following code to convert the tweets from a list of cleaned tokens to dictionaries with keys as the tokens and True as values. The corresponding dictionaries are stored in positive_tokens_for_model and negative_tokens_for_model.



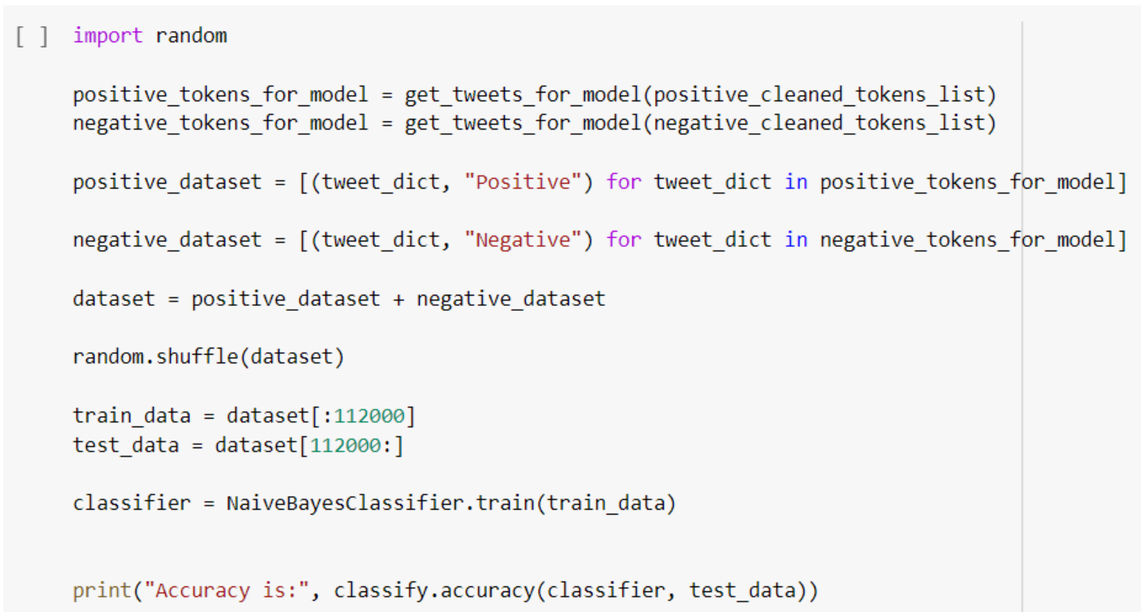
```
[ ]  from nltk import FreqDist, classify, NaiveBayesClassifier

     def get_tweets_for_model(cleaned_tokens_list):
         for tweet_tokens in cleaned_tokens_list:
             yield dict([token, True] for token in tweet_tokens)

     positive_tokens_for_model = get_tweets_for_model(positive_cleaned_tokens_list)
     negative_tokens_for_model = get_tweets_for_model(negative_cleaned_tokens_list)
```

Fig 12-Converting tokens to dictionary

4.7. <u>Building and testing the model</u>

The Naïve Bayesian classifier works as follows: Suppose that there exist a set of training data, D, in which each tuple is represented by an n−dimensional feature vector, X=x 1,x 2,..,x n, indicating n measurements made on the tuple from n attributes or features. Assume that there are m classes, C 1,C 2,...,C m. Given a tuple X, the classifier will predict that X belongs to C i if and only if: P(C i|X)>P(C j|X), where i,j∈[1,m]andi≠j. P(C i|X) is computed as:

$$P(C_i|X) = \prod_{k=1}^{n} P(x_k|C_i)$$

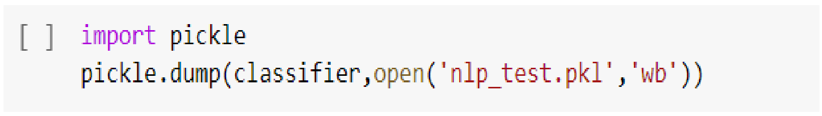


```
[ ]  import random

     positive_tokens_for_model = get_tweets_for_model(positive_cleaned_tokens_list)
     negative_tokens_for_model = get_tweets_for_model(negative_cleaned_tokens_list)

     positive_dataset = [(tweet_dict, "Positive") for tweet_dict in positive_tokens_for_model]

     negative_dataset = [(tweet_dict, "Negative") for tweet_dict in negative_tokens_for_model]

     dataset = positive_dataset + negative_dataset

     random.shuffle(dataset)

     train_data = dataset[:112000]
     test_data = dataset[112000:]

     classifier = NaiveBayesClassifier.train(train_data)


     print("Accuracy is:", classify.accuracy(classifier, test_data))
```

Fig 13- Applying Naive Bayes Classifier

4.8. <u>Importing pickle</u>

We open the file in "wb" mode instead of "w" as all the operations are done using bytes in the current working directory. A new file named "nlp_test.pkl" is created, which converts the mylist data in the byte stream.

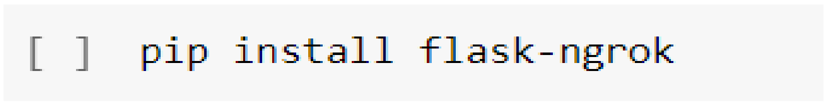


```
[ ]  import pickle
     pickle.dump(classifier,open('nlp_test.pkl','wb'))
```

Fig 14- Import Pickle

4.9. <u>Introducing flask</u>

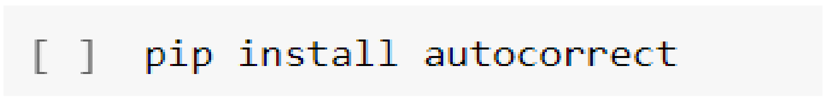
To use flask in google colaboratory install flask-ngrok library



```
[ ]  pip install flask-ngrok
```

Fig 15- Install Flask

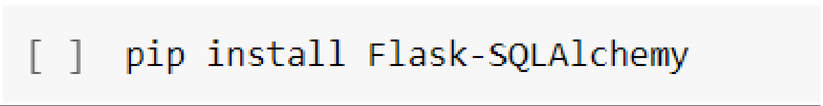
Now , we are Importing autocorrect module from python to check the spelling of tweet tweeted by user



```
[ ]  pip install autocorrect
```

Fig 16- Importing autocorrect module

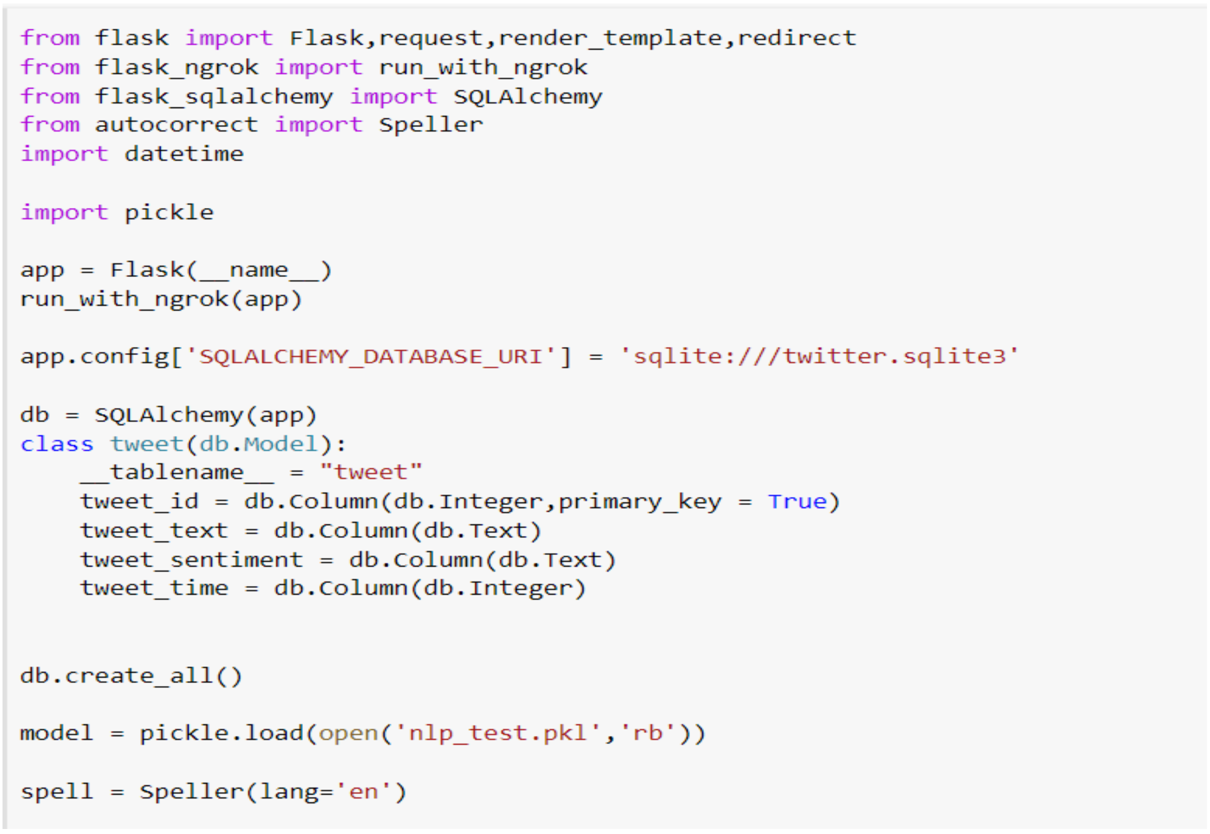
4.9.1. <u>Flask-SQLalchemy</u>



```
[ ]  pip install Flask-SQLAlchemy
```

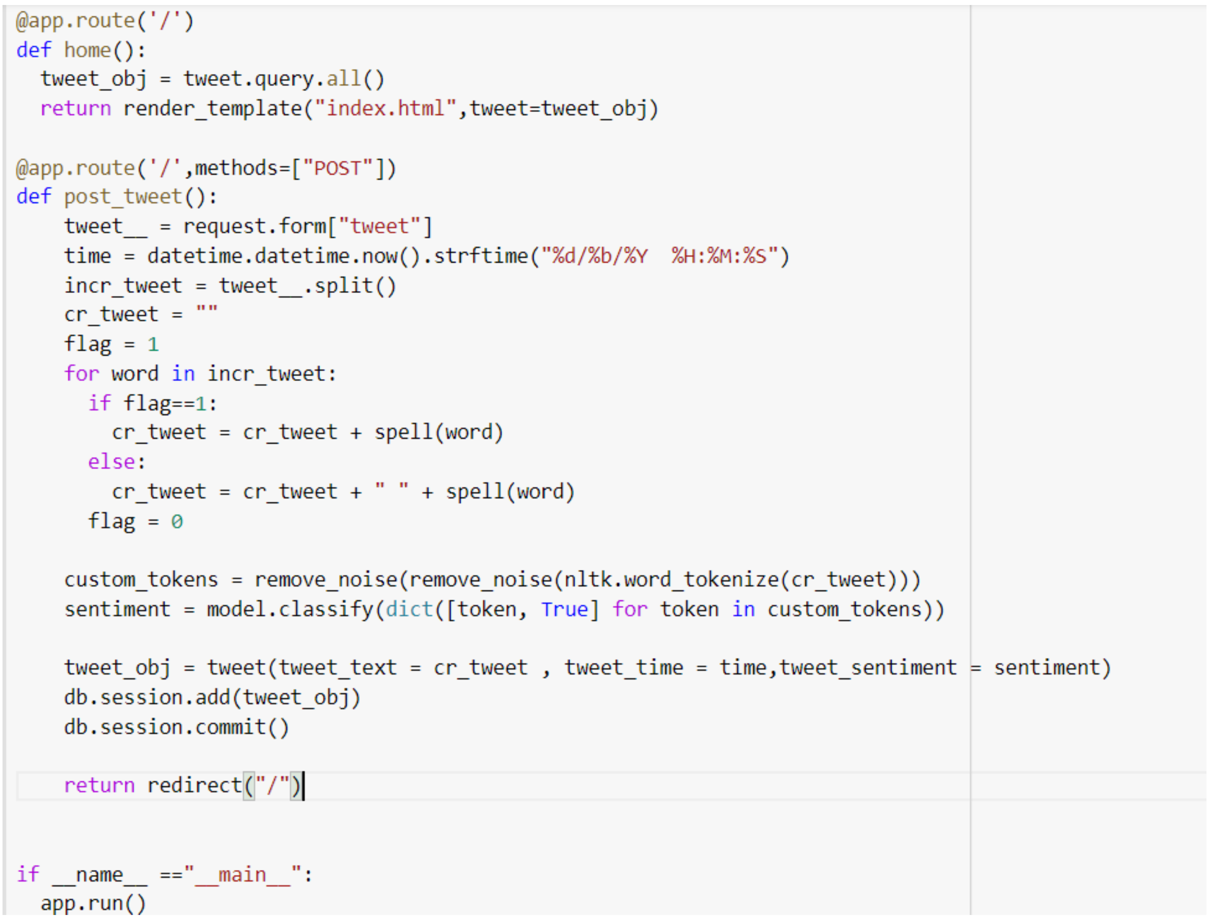
Fig 17 -Install Flask-SQLAlchemy in flask application

Creating a Flask application object and set the URI for the database to use.Then use the application object as a parameter to create an object of class SQLAlchemy.The object contains an auxiliary function for the ORM operation.It also provides a parent Model class that uses it to declare a user-defined model.



```python
from flask import Flask,request,render_template,redirect
from flask_ngrok import run_with_ngrok
from flask_sqlalchemy import SQLAlchemy
from autocorrect import Speller
import datetime

import pickle

app = Flask(__name__)
run_with_ngrok(app)

app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///twitter.sqlite3'

db = SQLAlchemy(app)
class tweet(db.Model):
    __tablename__ = "tweet"
    tweet_id = db.Column(db.Integer,primary_key = True)
    tweet_text = db.Column(db.Text)
    tweet_sentiment = db.Column(db.Text)
    tweet_time = db.Column(db.Integer)


db.create_all()

model = pickle.load(open('nlp_test.pkl','rb'))

spell = Speller(lang='en')
```

Fig 18- Making model to store data of user in database



```python
@app.route('/')
def home():
  tweet_obj = tweet.query.all()
  return render_template("index.html",tweet=tweet_obj)

@app.route('/',methods=["POST"])
def post_tweet():
    tweet__ = request.form["tweet"]
    time = datetime.datetime.now().strftime("%d/%b/%Y  %H:%M:%S")
    incr_tweet = tweet__.split()
    cr_tweet = ""
    flag = 1
    for word in incr_tweet:
      if flag==1:
        cr_tweet = cr_tweet + spell(word)
      else:
        cr_tweet = cr_tweet + " " + spell(word)
      flag = 0

    custom_tokens = remove_noise(remove_noise(nltk.word_tokenize(cr_tweet)))
    sentiment = model.classify(dict([token, True] for token in custom_tokens))

    tweet_obj = tweet(tweet_text = cr_tweet , tweet_time = time,tweet_sentiment = sentiment)
    db.session.add(tweet_obj)
    db.session.commit()

    return redirect("/")


if __name__ =="__main__":
  app.run()
```

Fig 19-Building routes

4.9.2. HTML
Hypertext Markup Language, a standardized system for tagging text files to achieve font, color, graphic, and hyperlink effects on World Wide Web pages.

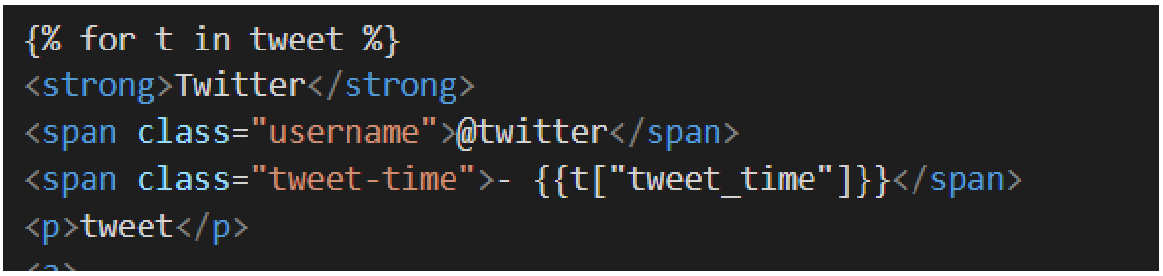
index.html is used

4.9.3. CSS
CSS stands for Cascading Style Sheets. CSS describes how HTML elements are to be displayed on screen,paper,or in other media. CSS saves a lot of work. It can control the layout of multiple web pages all at once. External stylesheets are stored in CSS files.

styles.css is used

4.9.4. Jinja Templating

To show dynamic content in browser from database jinja templating is used



```
{% for t in tweet %}
<strong>Twitter</strong>
<span class="username">@twitter</span>
<span class="tweet-time">- {{t["tweet_time"]}}</span>
<p>tweet</p>
```

Fig 20-Jinja templating to render dynamic content from database

EXPERIMENTATION AND RESULTS

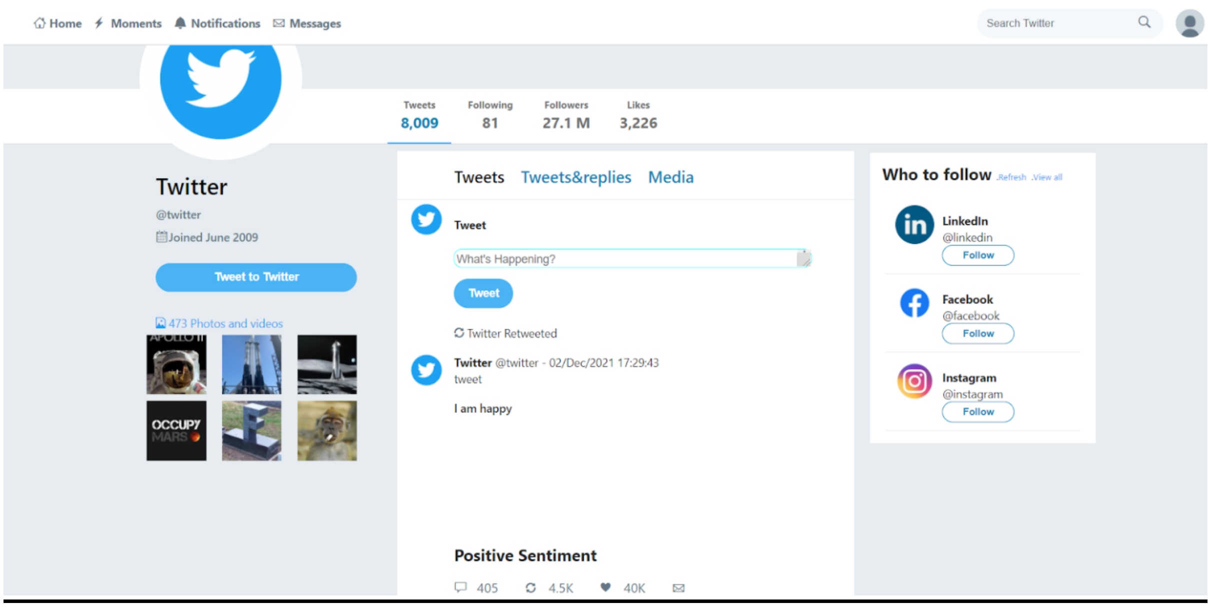
Accuracy of Model

```
Accuracy is: 0.7448230811983072
```

Fig 21- Accuracy of model

User Interface-

This is the user interface of the Flask application.Here user will enter tweet in provided textarea and and after clicking on the button the post request will be triggered and and the tweet will be fetched in flask route (‘ / ’ post request) and it’ll be spell check and then will be added to the database along with the sentiment predicted by the model.Time and date of tweet will also be added to the database and and will be rendered with the help of jinja templating on the browser.



Twitter Retweeted

Twitter @twitter - 02/Dec/2021 17:29:43

tweet

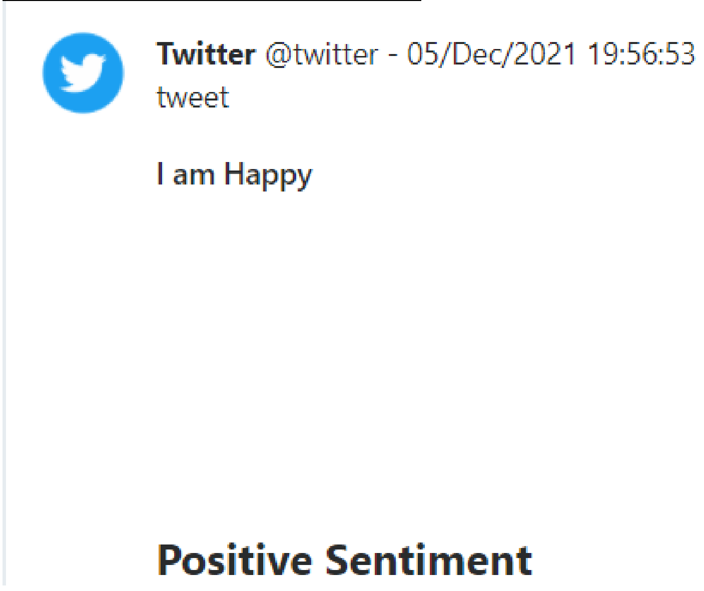
I am happy

Positive Sentiment

Fig 22- Output screenshot of Flask Application

Here are some outputs as per the user input

Positive sentiment Experiment



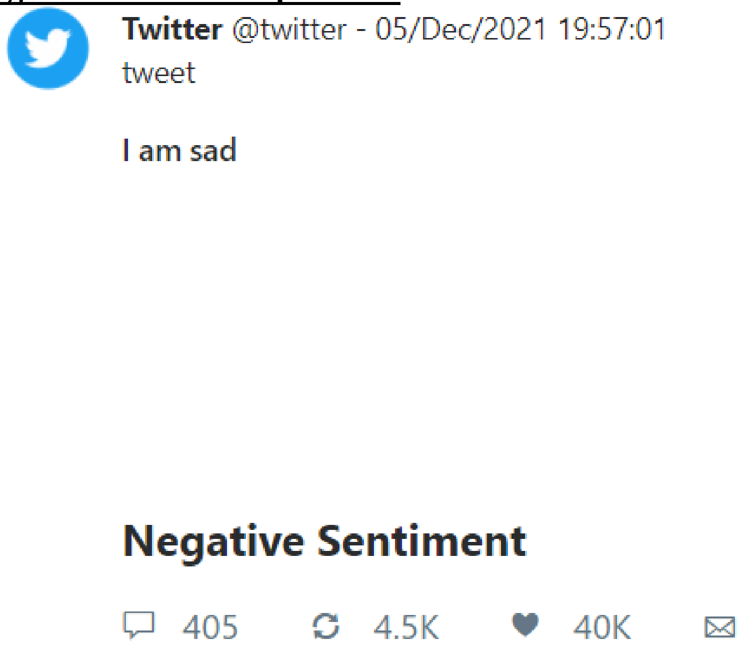
Twitter @twitter - 05/Dec/2021 19:56:53
tweet

I am Happy

Positive Sentiment

Fig 23- Output screenshot when user input is 'I am Happy' and it's corresponding output predicted by the model

Negative sentiment Experiment



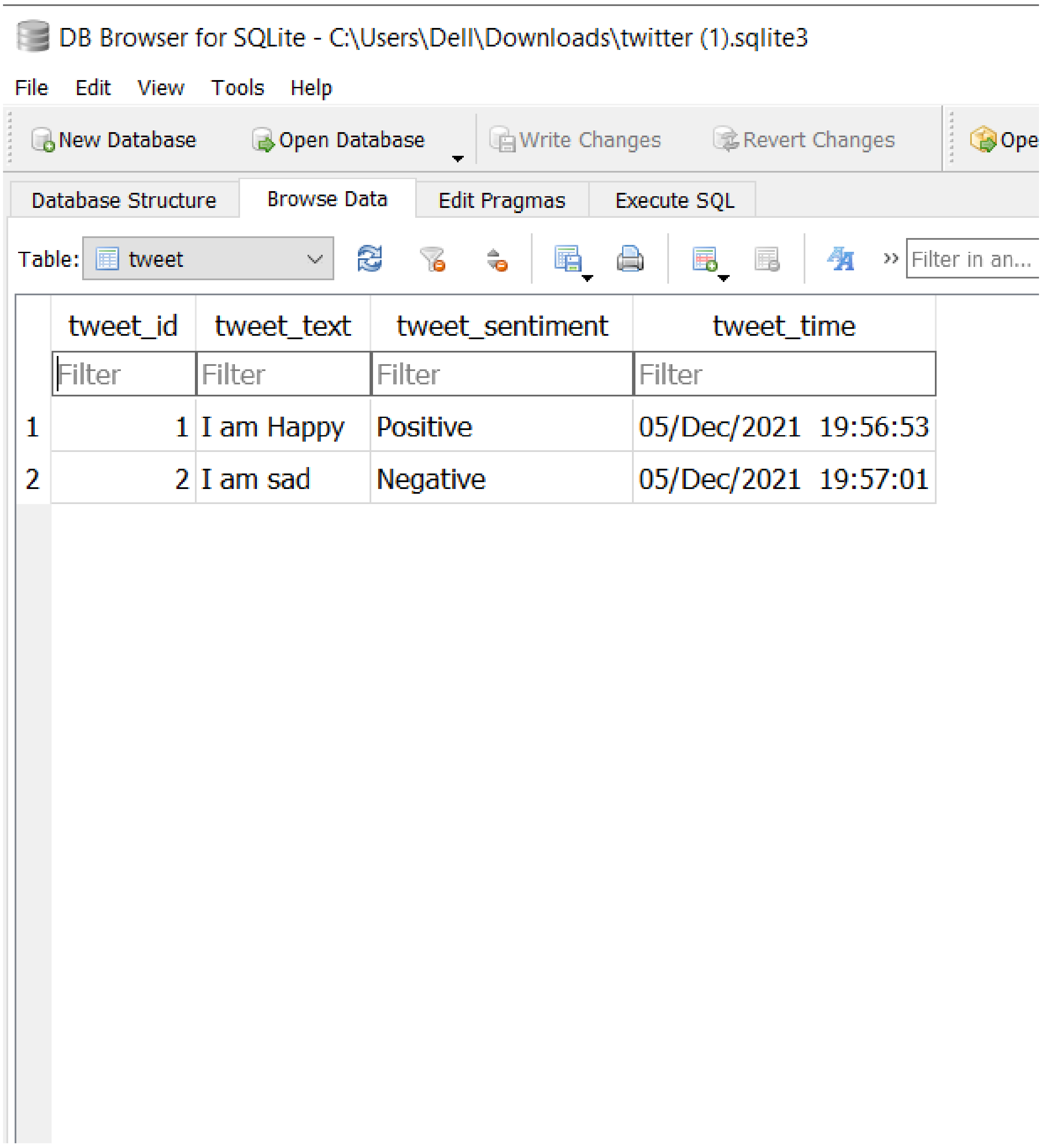
Twitter @twitter - 05/Dec/2021 19:57:01
tweet

I am sad

Negative Sentiment

Fig 24- Output screenshot when user input is' I am sad' and it's corresponding output predicted by the model

Data stored in the sqlite3 database


DB Browser for SQLite - C:\Users\Dell\Downloads\twitter (1).sqlite3

	tweet_id	tweet_text	tweet_sentiment	tweet_time
1	1	I am Happy	Positive	05/Dec/2021 19:56:53
2	2	I am sad	Negative	05/Dec/2021 19:57:01

Fig 25- DB Browser(sqlite) GUI showing data stored via form

Conclusion of the Report and Future Scope

Twitter sentiment analysis comes under the category of text and opinion mining. It focuses on analyzing the sentiments of the tweets and feeding the data to a machine learning model to train it and then check its accuracy, so that we can use this model for future use according to the results. It comprises steps like data collection, text preprocessing, sentiment detection, sentiment classification, training and testing the model. This research topic has evolved during the last decade with models reaching the efficiency of almost 85%-90%. But it still lacks the dimension of diversity in the data. Along with this it has a lot of application issues with the slang used and the short forms of words. Many analyzers don’t perform well when the number of classes are increased. Also, it’s still not tested that how accurate the model will be for topics other than the one in consideration. Hence sentiment analysis has a very bright scope of development in future

Some of the *future scopes* that can be included in our research work are:Use of parser can be embedded into the system to improve results.We can improve our system that can deal with sentences of multiple meanings.We can also increase the classification categories so that we can get better results.We can start work on multi languages like Hindi, Tamil, Marathi to provide sentiment analysis to more locals.

REFERENCES-

[1] A.Pak and P. Paroubek. „Twitter as a Corpus for Sentiment Analysis and Opinion Mining". In Proceedings of the Seventh Conference on International Language Resources and Evaluation, 2010, pp.1320-1326.

[2] R. Parikh and M. Movassate, "Sentiment Analysis of User- GeneratedTwitter Updates using Various Classi_cation Techniques",CS224N Final Report, 2009

[3] Go, R. Bhayani, L.Huang. "Twitter Sentiment ClassificationUsing Distant Supervision". Stanford University, Technical Paper,2009

[4] L. Barbosa, J. Feng. "Robust Sentiment Detection on Twitter From Biased and Noisy Data". COLING 2010: Poster Volume,pp. 36-44.

[5] Bifet and E. Frank, "Sentiment Knowledge Discovery inTwitter Streaming Data", In Proceedings of the 13th InternationalConference on Discovery Science, Berlin, Germany: Springer,2010, pp. 1-15.

[6] Agarwal, B. Xie, I. Vovsha, O. Rambow, R. Passonneau, "Sentiment Analysis of Twitter Data", In Proceedings of the ACL 2011Workshop on Languages in Social Media,2011 , pp. 30-38

[7] Dmitry Davidov, Ari Rappoport." Enhanced Sentiment Learning Using Twitter Hashtags and Smileys". Coling 2010: Poster Volumepages 241{249, Beijing, August 2010

[8] Po-Wei Liang, Bi-Ru Dai, "Opinion Mining on Social MediaData", IEEE 14th International Conference on Mobile Data Management,Milan, Italy, June 3 - 6, 2013, pp 91-96, ISBN: 978-1-494673-6068-5, http://doi.ieeecomputersociety.org/10.1109/MDM.2013.

[9] Pablo Gamallo, Marcos Garcia, "Citius: A Naive-Bayes Strategyfor Sentiment Analysis on English Tweets", 8th InternationalWorkshop on Semantic Evaluation (SemEval 2014), Dublin, Ireland,Aug 23-24 2014, pp 171-175.

[10] P. D. Turney, "Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews," in Proceedings of the 40th annual meeting on association for computational linguistics, pp. 417–424, Association for Computational Linguistics, 2002.

[11] J. Kamps, M. Marx, R. J. Mokken, and M. De Rijke, "Using wordnet to measure semantic orientations of adjectives," 2004..

[12] R. Xia, C. Zong, and S. Li, "Ensemble of feature sets and classification algorithms for sentiment classification," Information Sciences: an International Journal, vol. 181, no. 6, pp. 1138–1152, 2011

[13] ZhunchenLuo, Miles Osborne, TingWang, An effective approach to tweets opinion retrieval", Springer Journal onWorldWideWeb,Dec 2013, DOI: 10.1007/s11280-013- 0268-7