**Sentiment Analysis of COVID-19 Tweets – Visualization Dashboard**

**Introduction :**

This is the process of determining whether the users’ tweets on COVID-19 using some hashtags for e.g. #coronavirus or #covid are positive, negative or neutral. Also known as “Opinion Mining”, Sentiment Analysis refers to the use of Natural Language Processing (NLP) to determine the attitude, opinions and emotions of a speaker, writer, or other subject within an online mention. For classification, SVM algorithm is being used instead of generally used Naïve Bayes as SVM serves better performance. Then certain python libraries such as NumPy, Pandas and many other libraries for text processing and classification such as NLTK, Text Blob, SPACY are applied on the model created. After training and preparation of validation state, the covid-19 tweets are tested using model to give appropriate results explaining sentiment of particular tweet. Coding part is done on Anaconda IDE & IBM Watson studio.

Then this ML model is deployed to IBM cloud and web using web framework Flask. So in general, this predictive analysis recognizes the sentiments and views of the masses on important news like extension of lockdown or easiness and impacts on various sectors.

**Literature Survey :**

Twitter sentiment analysis is developed to analyze

customers perspectives toward the critical to success in the

marketplace. The program is using a machine-based learning

approach which is more accurate for analyzing a sentiment;

together with natural language processing techniques will be

used.

As a result, program will be categorized sentiment into

positive and negative, which is represented in a pie chart and

html page Although, the program has been planned to be

developed as a web application, due to limitation of Django

which can only work on Linux server or LAMP. Thus, it

cannot be realized. Therefore, further enhancement of this

element is recommended in future study.

Twitter sentiment analysis is developed to analyze

customers perspectives toward the critical to success in the

marketplace. The program is using a machine-based learning

approach which is more accurate for analyzing a sentiment;

together with natural language processing techniques will be

used.

As a result, program will be categorized sentiment into

positive and negative, which is represented in a pie chart and

html page Although, the program has been planned to be

developed as a web application, due to limitation of Django

which can only work on Linux server or LAMP. Thus, it

cannot be realized. Therefore, further enhancement of this

element is recommended in future study.

The objectives of the study are first, to study the sentiment

analysis in microblogging which in view to analyze feedback

from a customer of an organization’s product; and second, is

to develop a program for customers’ review on a product

which allows an organization or individual to sentiment and

analyzes a vast amount of tweets into a useful format

**[1] Problem :** The objectives of the study are first, to study the sentiment analysis in microblogging which in view to analyse feedback from a customer of an organization’s product; and second, is to develop a program for customers’ review on a product which allows an organization or individual to sentiment and analyses a vast amount of tweets into a useful format.

**Solution :** Twitter sentiment analysis is developed to analyse customers perspectives toward the critical to success in the marketplace. The program is using a machine-based learning approach which is more accurate for analysing a sentiment; together with natural language processing techniques will be used. As a result, program will be categorized sentiment into positive and negative, which is represented in a pie chart and html page Although, the program has been planned to be developed as a web application, due to limitation of Django which can only work on Linux server or LAMP. Thus, it cannot be realized. Therefore, further enhancement of this element is recommended in future study.

**[2] Problem :** Companies manufacturing such products have started to poll these microblogs to get a sense of general sentiment for their product. Many times these companies study user reactions and reply to users on microblogs. One challenge is to build technology to detect and summarize an overall sentiment.

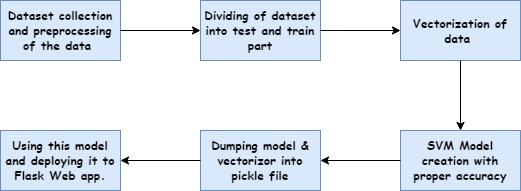
**Solution :** We presented results for sentiment analysis on Twit-ter. We use previously proposed state-of-the-art unigram model as our baseline and report an overall gain of over 4% for two classiﬁcation tasks: a binary, positive versus negative and a 3-way positive versus negative versus neutral. We presented a comprehensive set of experiments for both these tasks on manually annotated data that is a random sample of stream of tweets. We investigated two kinds of models : tree kernel and feature based models and demonstrate that both these models outperform the unigram baseline. For our feature-based approach, we do feature analysis which reveals that the most important features are those that combine the prior polarity of words and their parts-of-speech tags. We tentatively conclude that sentiment analysis for Twitter data is not that different from sentiment analysis for other genres.

**[3] Problem :** The existing sentiment analysis techniques are useful in various applications such as disaster relief and humanitarian assistance, marketing and trade predictions, checking political polls, advertising market, scientific surveys, checking customer loyalty, finding job opportunities, population health care and understanding students’ learning experiences.

**Solution :** In this paper, we have firstly presented the detailed procedure to carryout sentiment analysis process to classify highly unstructured data of Twitter into positive or negative categories. Secondly, we have discussed various techniques to carryout sentiment analysis on Twitter data including knowledge based technique and machine learning techniques. Moreover, we presented the parametric comparison of the discussed supervised machine learning techniques based on our identified parameters. It has been found that various techniques applied for sentiment analysis are domain specific and language specific. Hence, the future opportunities in the domain of sentiment analysis include developing a technique to perform sentiment classification that can be applicable to any data regardless of domain.

**Theoretical Analysis :**

**I] Block Diagram :**

****

**fig. Block Diagram**

**II] Software Designing :**

This ML model is used for sentiment analysis. Basically it classifies the tweets into negative or positive. For classification purposes here the algorithm we are using is SVM. It is better than other ML algorithms like Naïve Bayes for our model of the sentiment analysis purposes. Other libraries like NumPy, pandas are definitely necessary. There are many other libraries for text processing and classification such as NLTK, Text Blob, SPACY and many more. For model building we have used csv file containing various tweets from twitter. From this dataset we need to dropout duplicate rows or any unnecessary columns that don’t play any role for classification. Then the next step is cleaning that is removing unnecessary symbols or text punctuations etc. We further used tokenizer that is used for classifications of that tweet. Then comes the part of SVM model and this is how we can classify those tweets. We have also used one more library that is matplotlib to plot the graphs of various results which is then passed to dashboard section of website.

For prediction or the classification of live tweets, we have used the tweets through web scrapping. Libraries used are twitter scrapper and many more. This tweet is again pre-processed and then it is transferred to the model for prediction or the classification purpose. We use here multiple proxy so that server don’t block our request of fetching the tweets. Then we deploy this model to web using Flask framework which uses pickle files created and various functions declared for specific functionalities on website.

As a part of difference or uniqueness we have added a part in this that is conversion of text into speech. For this conversion we used libraries like gtts, playsound & many more.

**Experimental Investigations :**

SVM is a supervised machine learning model that uses classification algorithms; here we have used it for classifying tweets into positive or negative. Kernel used is linear in SVM algorithm. But before going on to the model data pre-processing is done that is all the stop words are removed. Text is tokenized so that the large data is converted into words. Then lemmatization is done so that the root word is extracted from the given word. Then at last a Tfidf vectorizer is used to convert the data into vectors and then it is passed on to the model for training and testing purpose. Here accuracy we have observed is 96 % . At last the model is dumped into a pickle file and then is used for prediction purpose. Model is deployed using Flask web app. There are some cool features like web scrapping and text to speech features. If some people are unable to read the scrapped tweet he can get it by hearing. At last there are visualization which are updated daily and hence we can get what people are thinking now adays and what further steps can be taken by the authority.

**Flowchart :**

**fig. Flowchart**

**Result :**

The final product is the model deployed on the web. It contains three sections. First section checks sentiment of tweet entered by the user and display the result rendering the html page. Second section displays the live tweets by users using web scraping and also predicts the sentiment of the tweet. Here text to audio feature is also used which converts the tweet scraped into audio form. The third section is for visualization dashboard. It contains graphs which updates daily. This Flask Website is styled using various CSS styles in html page. The Final UI of the website is :



**fig. UI**

**Advantages and Disadvantages :**

**Advantages -**

1. The model provides with high accuracy sentiment analysis and also have undergone special training for covid19 tweets.
2. The accuracy is around 96% which makes the model usable by everyone.
3. A special feature of text to speech is added which will also help visually impaired ones.
4. A beautiful UI along with contently graphs are also present which gives live sentiment data of a week.
5. Easy scraping feature is also added in case user wants to scrape live tweet.

**Disadvantages -**

1. The accuracy of model is around 96% which means user will not get correct data every time (but will get most of the time).
2. While scraping tweets, there can be network problems which will lead to slow down of the scraping process.
3. Network issues can also cause problems in plotting of graphs and lead to slowdown of the site.

**Applications :**

1. The predictive analysis provided by our model can help recognize the sentiments and views of the masses on important news like extension of lockdown or easiness and sectors now allowed to function and these sentiments then further help the manufacturers/service providers predict about the demand/supply chain as per the positivity/negativity provided by the people and hence work on the same.
2. If the people are unhappy about any decisions taken by the authorities then they would react negatively of the same and our model will bring the same on to notice and then authorities could think about the matter once again.
3. Prediction of unhappy business sectors of the country can be predicted and special schemes for them can be provided by the government.
4. We can see the high frequency and negativity of the tweets and then can inform the authorities about the same and ensure peace and safety in the country.

**Conclusion :**

This report focuses on sentiment analysis of covid-19 tweets and displaying results on visualization dashboard. Python libraries such as pandas, NumPy, nltk, sklearn, pickle, twitterscraper, matplotlib and many more are used for the implementation of the concept. SVM model gives better performance than Naïve Bayes. The final project is a proper website with flask in backend using pickle files of vectorizer and model designed with the help of HTML and CSS. It predicts sentiment of statement entered by user, displays live tweets and its sentiment and also contains visualization dashboard containing graphs which updates daily. This website can further be improved and implemented as an application which serves user better.

**Future Scope :**

1. We can provide user with a choice to scrape and predict tweet from desired date.
2. We can add geolocation feature which will scrape tweets and predict sentiments from a specified location and graph of same will be plotted.
3. We can make an app of the same for mobile devices.
4. Extensions for browsers can also be made which will predict and show sentiments next to tweets on twitter page.

**Bibliography :**

**Appendix -**

**Source Codes :-**

**model.py**

import pandas as pd

import numpy as np

import re

import pickle

from nltk.tokenize import word\_tokenize

from nltk import pos\_tag

from nltk.corpus import stopwords

from nltk.stem import WordNetLemmatizer

from collections import defaultdict

from nltk.corpus import wordnet as wn

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn import model\_selection,svm

from sklearn.metrics import accuracy\_score

dataset = pd.read\_csv("tweets2.csv",encoding='latin-1',low\_memory=False)

#dataset.head()

dataset['sentiment\_text'] = dataset['sentiment\_text'].str.lower()

def remove\_pattern(input\_txt, pattern):

r = re.findall(pattern, input\_txt)

for i in r:

input\_txt = re.sub(i, '', input\_txt)

return input\_txt

dataset['sentiment\_text'] = np.vectorize(remove\_pattern)(dataset['sentiment\_text'], "@[\w]\*")

dataset['sentiment\_text'] = dataset['sentiment\_text'].str.replace("[^a-zA-Z#]", " ")

dataset['sentiment\_text'] = dataset['sentiment\_text'].apply(lambda x: ' '.join([w for w in x.split() if len(w)>3]))

print("Done")

dataset['sentiment\_text']=[word\_tokenize(entry) for entry in dataset['sentiment\_text']]

print("Done")

tag\_map = defaultdict(lambda : wn.NOUN)

tag\_map['J'] = wn.ADJ

tag\_map['V'] = wn.VERB

tag\_map['R'] = wn.ADV

i=0

for index,entry in enumerate(dataset['sentiment\_text']):

# Declaring Empty List to store the words that follow the rules for this step

Final\_words = []

# Initializing WordNetLemmatizer()

word\_Lemmatized = WordNetLemmatizer()

# pos\_tag function below will provide the 'tag' i.e if the word is Noun(N) or Verb(V) or something else.

for word, tag in pos\_tag(entry):

# Below condition is to check for Stop words and consider only alphabets

if word not in stopwords.words('english') and word.isalpha():

word\_Final = word\_Lemmatized.lemmatize(word,tag\_map[tag[0]])

Final\_words.append(word\_Final)

# The final processed set of words for each iteration will be stored in 'text\_final'

dataset.loc[index,'text\_final'] = str(Final\_words)

i=i+1

print(i)

#dataset['sentiment\_text']=LancasterStemmer.stem(dataset['sentiment\_text'])

print(dataset['text\_final'])

Train\_X, Test\_X, Train\_Y, Test\_Y = model\_selection.train\_test\_split(dataset['text\_final'],dataset['sentiment'],test\_size=0.3)

Tfidf\_vect = TfidfVectorizer(max\_features=100000)

Tfidf\_vect.fit(dataset['text\_final'])

Train\_X\_Tfidf = Tfidf\_vect.transform(Train\_X)

Test\_X\_Tfidf = Tfidf\_vect.transform(Test\_X)

pickle.dump(Tfidf\_vect, open('VECT.sav','wb'))

print("Done")

print(Tfidf\_vect.vocabulary\_)

print(Train\_X\_Tfidf)

SVM = svm.SVC(C=1.0, kernel='linear', degree=3, gamma = 'auto')

SVM.fit(Train\_X\_Tfidf,Train\_Y)

pickle.dump(SVM, open('SVM.sav','wb'))

print("Done")

# predict the labels on validation dataset

predictions\_SVM = SVM.predict(Test\_X\_Tfidf)

# Use accuracy\_score function to get the accuracy

print("SVM Accuracy Score -> ",accuracy\_score(predictions\_SVM, Test\_Y)\*100)

**MLWeb.py**

import pickle

from flask import Flask, request, render\_template, send\_file

from twitterscraper import query\_tweets

import datetime as dt

import pandas as pd

import random

from gtts import gTTS

import os

from playsound import playsound

import matplotlib.pyplot as plt

import numpy as np

app = Flask(\_\_name\_\_)

app.config['JSON\_AS\_ASCII'] = False

vectorize = pickle.load(open('VECT.sav', 'rb'))

classifier = pickle.load(open('SVM.sav', 'rb'))

@app.route("/", methods=['GET','POST'])

def return\_sentiment():

if request.method == 'GET':

return render\_template('index.html')

if request.method == 'POST':

sentence = request.form['sentence']

if(sentence):

sentence\_vector = vectorize.transform([sentence])

sentiment = classifier.predict(sentence\_vector).tolist()

return render\_template('index.html', original\_input= {'sentence':sentence}, result = sentiment[0])

@app.route("/scrape/",methods=['GET','POST'])

def return\_data():

if request.method == 'POST':

today = dt.date.today()

yesterday = today - dt.timedelta(days = 1)

begin\_date=yesterday

end\_date = today

limit = 100

lang='en'

tweets = query\_tweets("corona",begindate = begin\_date,enddate = end\_date ,limit=limit ,lang=lang)

df = pd.DataFrame(t.\_\_dict\_\_ for t in tweets)

i=df.text.str.strip()

j=df.timestamp

l=random.randint(1,len(i))

tweet=i[l]

tweet\_vector = vectorize.transform([tweet])

sent = classifier.predict(tweet\_vector).tolist()

mytext = tweet

language = 'en'

myobj = gTTS(text = mytext, lang = language, slow = False)

myobj.save("audio.mp3")

return render\_template('index.html', tw = tweet, time=j[l], sent = sent[0])

else:

return render\_template('index.html')

@app.route("/scrape/audio/",methods=['GET'])

def return\_audio():

playsound("audio.mp3")

os.remove("audio.mp3")

return render\_template('index.html')

@app.route("/dashboard/",methods=['GET'])

def return\_plots():

today = dt.date.today()

yesterday = today - dt.timedelta(days = 1)

x=[0 for t in range(7)]

y=[0 for t in range(7)]

for m in range(7):

begin\_date = yesterday - dt.timedelta(days = m)

end\_date = today - dt.timedelta(days = m)

lang='en'

limit=500

tweets = query\_tweets("COVID",begindate = begin\_date, enddate = end\_date ,limit = limit ,lang=lang)

df = pd.DataFrame(t.\_\_dict\_\_ for t in tweets)

i=df.text

j=df.timestamp

n = random.sample(range(0,len(i)),100)

for k in n:

date = j[k]

tweet= i[k]

tweet\_vector = vectorize.transform([tweet])

sent = classifier.predict(tweet\_vector).tolist()

if(sent[0]==1):

x[m]=x[m]+1

else:

y[m]=y[m]+1

dates = [today-dt.timedelta(days = z) for z in range (7)]

x=np.array(x)

y=np.array(y)

plt.bar(dates, x, width=0.8, label='p', color='gold', bottom=y)

plt.bar(dates, y, width=0.8, label='n', color='silver')

plt.ylim(0,120)

plt.ylabel("No. of Sentiments")

plt.xlabel("Date")

plt.xticks(rotation=25)

plt.legend(loc='centre', bbox\_to\_anchor=(1,0.60), ncol = 1)

plt.title("Sentiment Analysis")

plt.savefig(‘…/static/img/plot2.png')

plt.show()

colors = ['lightskyblue', 'lightcoral']

labels = ['Positive', 'Negative',]

sizes=[sum(x),sum(y)]

explode = (0.2, 0)

patches, texts = plt.pie(sizes, explode=explode, colors=colors, shadow=True, startangle=45)

plt.legend(patches, labels, loc="best")

plt.axis('equal')

plt.tight\_layout()

plt.savefig('… /static/img/plot3.png')

plt.show()

return render\_template('index.html')

if \_\_name\_\_ == "\_\_main\_\_":

app.run(host="localhost", port=8000, debug=True)