**SOURCE CODE**

from tkinter import messagebox

from tkinter import \*

from tkinter import simpledialog

import tkinter

import matplotlib.pyplot as plt

import numpy as np

from tkinter import filedialog

from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

from string import punctuation

from nltk.corpus import stopwords

import pandas as pd

from emoji import UNICODE\_EMOJI

main = tkinter.Tk()

main.title("NLP-based Extended Lexicon Model for Sarcasm Detection with Tweets and Emojis") #designing main screen

main.geometry("1300x1200")

sid = SentimentIntensityAnalyzer()

global filename

global dataset

global process

global sarcastic

global sentiment

def checkSarcasm(sentence):

pos = []

neg = []

neu = []

arr = sentence.split(' ')

for i in range(len(arr)):

word = arr[i].strip()

if word == 'smilingfacewithhearteyes':

word = 'excellent'

if word == 'loudlycryingface':

word = 'bad'

if word == 'winkingfacewithtongue':

word = 'happy'

if (sid.polarity\_scores(word)['compound']) >= 0.1:

pos.append(word)

elif (sid.polarity\_scores(word)['compound']) <= -0.1:

neg.append(word)

else:

neu.append(word)

return pos,neg,neu

def clean\_doc(doc):

tokens = doc.split()

table = str.maketrans('', '', punctuation)

tokens = [w.translate(table) for w in tokens]

tokens = [word for word in tokens if word.isalpha()]

stop\_words = set(stopwords.words('english'))

tokens = [w for w in tokens if not w in stop\_words]

tokens = [word for word in tokens if len(word) > 1]

tokens = ' '.join(tokens) #here upto for word based

return tokens

def upload():

global filename

global dataset

dataset = []

filename = filedialog.askopenfilename(initialdir="dataset")

text.delete('1.0', END)

text.insert(END,filename+" loaded\n");

train = pd.read\_csv(filename,encoding='utf8',sep='\t')

count = 0

for i in range(len(train)):

tweet = train.get\_value(i,0,takeable = True)

print(tweet)

if str(tweet) != 'nan':

tweet = tweet.lower()

icon = train.get\_value(i,1,takeable = True)

if str(icon) != 'nan':

icon = UNICODE\_EMOJI[icon.strip()]

icon = ''.join(re.sub('[^A-Za-z\s]+', '', icon))

icon = icon.lower()

else:

icon = ''

msg = ''

if str(tweet) != 'nan':

arr = tweet.split(" ")

for k in range(len(arr)):

word = arr[k].strip()

if len(word) > 2:

msg+=word+" "

textdata = msg.strip()+" "+icon

#print(textdata)

dataset.append(textdata)

text.insert(END,'Total tweets found in dataset is : '+str(len(dataset)))

def Preprocessing():

text.delete('1.0', END)

global process

process = []

text.insert(END,'Messages after preprocessing and removing stopwords\n')

text.insert(END,'====================================================================================\n')

for i in range(len(dataset)):

sentence = dataset[i]

sentence = sentence.lower()

sentence = clean\_doc(sentence)

text.insert(END,sentence+'\n')

process.append(sentence)

def firstAlgorithm():

text.delete('1.0', END)

global sarcastic

sarcastic = []

for i in range(len(process)):

sentence = process[i]

if sentence == 'smilingfacewithhearteyes':

sentence = 'excellent'

if sentence == 'loudlycryingface':

sentence = 'bad'

if sentence == 'winkingfacewithtongue':

sentence = 'happy'

sentiment\_dict = sid.polarity\_scores(sentence)

negative\_polarity = sentiment\_dict['neg']

positive\_polarity = sentiment\_dict['pos']

neutral\_polarity = sentiment\_dict['neu']

compound = sentiment\_dict['compound']

result = ''

if compound >= 0.1 :

result = 'Positive'

elif compound <= -0.1:

result = 'Negative'

else :

result = 'Neutral'

if result =='Positive' or result == 'Neutral':

pos,neg,neu = checkSarcasm(sentence)

if len(neg) > 0:

sarcastic.append("Sarcastic")

else:

sarcastic.append("Non Sarcastic")

else:

sarcastic.append("Non Sarcastic")

text.insert(END,'Tweets : '+dataset[i]+"\n")

text.insert(END,'Positive Polarity : '+str(positive\_polarity)+"\n")

text.insert(END,'Negative Polarity : '+str(negative\_polarity)+"\n")

text.insert(END,'Neutral Polarity : '+str(neutral\_polarity)+"\n")

text.insert(END,'Result : '+sarcastic[i]+"\n")

text.insert(END,'====================================================================================\n')

def secondAlgorithm():

global sentiment

sentiment = []

text.delete('1.0', END)

for i in range(len(process)):

sentence = process[i]

if sentence == 'smilingfacewithhearteyes':

sentence = 'excellent'

if sentence == 'loudlycryingface':

sentence = 'bad'

if sentence == 'winkingfacewithtongue':

sentence = 'happy'

sentiment\_dict = sid.polarity\_scores(sentence)

negative\_polarity = sentiment\_dict['neg']

positive\_polarity = sentiment\_dict['pos']

neutral\_polarity = sentiment\_dict['neu']

compound = sentiment\_dict['compound']

result = ''

if compound >= 0.1 :

result = 'Positive'

sentiment.append(result)

elif compound <= -0.1:

result = 'Negative'

sentiment.append(result)

else :

result = 'Neutral'

sentiment.append(result)

sar = ''

if result =='Positive' or result == 'Neutral':

pos,neg,neu = checkSarcasm(sentence)

if len(neg) > 0:

sar = "Sarcastic"

else:

sar = "Non Sarcastic"

else:

sar = "Non Sarcastic"

text.insert(END,'Tweets : '+dataset[i]+"\n")

text.insert(END,'Positive Polarity : '+str(positive\_polarity)+"\n")

text.insert(END,'Negative Polarity : '+str(negative\_polarity)+"\n")

text.insert(END,'Neutral Polarity : '+str(neutral\_polarity)+"\n")

text.insert(END,'Result : '+sar+"\n")

text.insert(END,'Sentiment Prediction : '+result+'\n')

text.insert(END,'====================================================================================\n')

def sarcasticGraph():

sar = 0

non\_sar = 0

for i in range(len(sarcastic)):

if sarcastic[i] == "Sarcastic":

sar = sar + 1

if sarcastic[i] == "Non Sarcastic":

non\_sar = non\_sar + 1

height = [sar,non\_sar]

bars = ('Sarcastic','Non Sarcastic')

y\_pos = np.arange(len(bars))

plt.bar(y\_pos, height)

plt.xticks(y\_pos, bars)

plt.show()

def sentimentGraph():

label\_X = []

category\_X = []

pos = 0

neg = 0

neu = 0

for i in range(len(sentiment)):

if sentiment[i] == 'Positive':

pos = pos + 1

if sentiment[i] == 'Negative':

neg = neg + 1

if sentiment[i] == 'Neutral':

neu = neu + 1

label\_X.append('Positive')

label\_X.append('Negative')

label\_X.append('Neutral')

category\_X.append(pos)

category\_X.append(neg)

category\_X.append(neu)

plt.pie(category\_X,labels=label\_X,autopct='%1.1f%%')

plt.title('Sentiment Graph')

plt.axis('equal')

plt.show()

font = ('times', 16, 'bold')

title = Label(main, text='NLP-based Extended Lexicon Model for Sarcasm Detection with Tweets and Emojis')

title.config(bg='LightGoldenrod1', fg='medium orchid')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=0,y=5)

font1 = ('times', 12, 'bold')

text=Text(main,height=30,width=100)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=400,y=100)

text.config(font=font1)

font1 = ('times', 12, 'bold')

uploadButton = Button(main, text="Upload Dataset", command=upload)

uploadButton.place(x=50,y=100)

uploadButton.config(font=font1)

preButton = Button(main, text="Preprocess Dataset", command=Preprocessing)

preButton.place(x=50,y=150)

preButton.config(font=font1)

firstButton = Button(main, text="First System Lexicon + Polarity Computation", command=firstAlgorithm)

firstButton.place(x=50,y=200)

firstButton.config(font=font1)

secondButton = Button(main, text="Second System Lexicon + Sentiment Prediction", command=secondAlgorithm)

secondButton.place(x=50,y=250)

secondButton.config(font=font1)

graphButton = Button(main, text="Sentiments Graph", command=sentimentGraph)

graphButton.place(x=50,y=300)

graphButton.config(font=font1)

gButton = Button(main, text="Sarcastic Graph", command=sarcasticGraph)

gButton.place(x=50,y=350)

gButton.config(font=font1)

main.config(bg='OliveDrab2')

main.mainloop()

**RESULTS AND DISCUSSION**

Figure 1 illustrates the graphical user interface (GUI) of the sarcasm detection application. It shows the buttons, text fields, and other elements that users interact with. Users can perform tasks such as uploading datasets, initiating preprocessing, running algorithms, and visualizing results through this interface. In Figure 2, the user interface is shown after the user has uploaded a dataset. The uploaded dataset contains tweets and associated data. The interface displays a confirmation message to inform the user that the dataset has been successfully uploaded along with the count of tweets in given dataset. Figure 3 presents the user interface displaying preprocessed data after applying the Natural Language Toolkit (NLTK) library for text processing. The interface shows cleaned and tokenized text, and potentially processed emojis. This step aims to prepare the data for further analysis. Figure 4 showcases the results obtained after applying lexicon-based sentiment analysis with polarity computation on the preprocessed data. The user interface displayed each tweet's polarity scores, indicating their positivity, negativity, and neutrality. The interface also indicates whether the tweet is categorized as positive, negative, or neutral.

A screenshot of a computer

Description automatically generated

Figure 1: User interface application of proposed NLP-based sarcasm detection model.

A screenshot of a computer

Description automatically generated

Figure 2: Illustration of user interface application after uploading the dataset.

A screenshot of a computer

Description automatically generated

Figure 3: Proposed UI application with preprocessed data using NLTK.

In Figure 5, the user interface presents the results of sentiment prediction using lexicon-based sentiment analysis. Alongside polarity scores, the UI also indicates whether a tweet is predicted as sarcastic or not based on sentiment and certain words. Figure 6 displays a pie chart that visualizes the distribution of sentiments in the dataset. The chart is divided into segments representing positive, negative, and neutral sentiments. The size of each segment indicates the proportion of tweets falling into each sentiment category. Figure 7 shows a graph that visualizes the sarcasm predictions made by the proposed NLP-based extended lexicon model. The graph likely has two bars: one representing the number of sarcastic tweets and the other representing the number of non-sarcastic tweets. This visualization provides insights into the model's ability to detect sarcasm.

A screenshot of a computer

Description automatically generated

Figure 4: Obtained results of proposed UI application with lexicon + polarity computation.

A screenshot of a computer

Description automatically generated

Figure 5: Results of lexicon + sentiment prediction for proposed sarcasm detection model.

A pie chart with different colored circles

Description automatically generated

Figure 6: Pie chart of sentiment prediction on tweets dataset with different classes (positive, negative, and neutral).

A bar graph with a white background

Description automatically generated with medium confidence

Figure 7: Sarcasm prediction graph using proposed NLP-based extended lexicon model.