CS6109: Compiler Design

Sentiment Analysis of Bilingual Tweets

Dhivyashri Ramesh - 2019103015
Kavya Sridhar - 2019103027

Department of Computer Science and Engineering,
College of Engineering Guindy Campus,
Anna University,
Chennai-25

Table of Contents

INTRODUCTION

Meaning and sentiment (emotion) conveyed through text are extremely valuable to analyze for various purposes. To gauge the user market, to understand public sentiment on social platforms like Twitter, and for many more such purposes.

Sentiment analysis helps data analysts within large enterprises gauge public opinion, conduct nuanced market research, monitor brand and product reputation, and understand customer experiences. In addition, data analytics companies often integrate third-party sentiment analysis APIs into their own customer experience management, social media monitoring, or workforce analytics platform, in order to deliver useful insights to their own customers. Therefore, having accurate methods to present accurate analyses of such texts is important.

Lexicon-based analysis, as previous research in this domain has shown, is useful and stable. However, to analyze sentiment from text with contraction, informal textual content and to be specific in our paper, bilingual tweets, there needs to be a further extension of methods of analysis.

Taking the Twitter dataset, this project hopes to implement methods to analyze bilingual tweets and classify them into positive and negative polarities. Here in bilingual, we will only be dealing with the languages of Tamil and English.

The challenges in taking a lexical approach to find out the polarities of the tweets would be to deal with the following issues:

- Obtaining the translation of bilingual words in tweets
- Emoticons present in the tweets representing emotion
- Contractions present in the tweets to be converted into meaningful labels representing the right meaning
- Removing unnecessary noise in the tweets

Once the preprocessing has been done, analyzing the emotion can be done after the process of feature extraction. This forms the sentiment analysis of the project. To aid this is the use of machine learning concepts and implementation of classifiers. This project makes use of the Naive Bayes Classifier and Logistic Regression Classifier.

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e., every pair of features being classified is independent of each other. Logistic regression predicts the output of a categorical dependent variable. Therefore, the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

With the above-mentioned method and references to previous attempts at the same, we hope to accomplish the following goals:

- We aim to implement a Twitter sentiment analysis model that helps to overcome the challenges of identifying the sentiments of bilingual tweets (Tamil and English).
- Classify the tweets present in the dataset into positive and negative.
- Implement the model for different classifiers and evaluate their performance based on accuracy and F1 scores.

RELATED WORKS

SNO	TITLE , AUTHOR ,YEAR	PROPOSED METHOD	MERITS	IDEAS FOR ADOPTION
1	Lexicon-Based Approach to Sentiment Analysis of Tweets Using R Language: Second International Conference,Nitika Nigam, Divakar Yadav, ICACDS 2018, Dehradun, India, April 20-21, 2018,	To classify the given set of tweets into two classes: Positive and Negative by extracting the semantics from the tweets and calculating polarity score.	Result is conclusive enough for the dataset . .	Leveraging the score calculation mechanism used here. Also adopting same style of lexicon based dictionary creation.
2	Tools and Techniques for Lexicon Driven Sentiment Analysis: A Review, Munir Ahmad Shabib Aftab, Syed Shah Muhammad Usman waheed Waheed	Different tools and methods were analysed for lexicon based analysis and their accuracy rates compared.	Different methods explored paving way for understanding the various nuances of analysis. Analysed for 3 datasets, hence giving varying pictures.	Adopting the proposed methods for positive, negation, blind negation and split words.
3	DepecheMood++: a Bilingual Emotion Lexicon Built Through Simple Yet Powerful Techniques Oscar Araque Lorenzo Gatti2 , Jacopo Staiano3 , Marco Guerini4,5 1Grupo de Sistemas Inteligentes, 2015	From Rappler and Corriere and (Guerini and Staiano)	The different methods mentioned have their own effect on accuracy for eg adding a word frequency cutoff parameter leads to a benefit in the performance of the generated lexicon; in this they find an optimal value of 10.	The methods mentioned to remove random subsets, of decreasing size, from the original lexicon vocabulary
4	Predicting Tamil Movies Sentimental Reviews Using Tamil Tweets- Vallikannu Ramanathan, T. Meyyappan and S.M. Thamarai- 2019	Petta Movie review.	It deals with the negation problem and Tamil SentiWordNet with adjectives to classify the sentiment.	To analyze the semantic meaning between the words, contextual semantic sentiment analysis is applied.

5	Expressively vulgar: The socio-dynamics of vulgarity and its effects on sentiment analysis in social media Isabel Cachola*‡ Eric Holgate*† Daniel Preoţiuc-Pietro♦ Junyi Jessy Li† 2018	This study performs a large-scale, data-driven empirical analysis of vulgar words using social media data	Introduced a new data set of 6.8K vulgar tweets labeled for sentiment on a five-point scale by nine annotators Token insertion and concatenation improves the prediction of negative tweets, and the prediction of non-negative tweets remain stable.	Masking, Token Insertion and concatenation for analysing vulgar tweets.

Data Dictionary:

Natural Language processing: Natural language processing (NLP) refers to the branch of computer science—and more specifically, the branch of artificial intelligence or AI—concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.

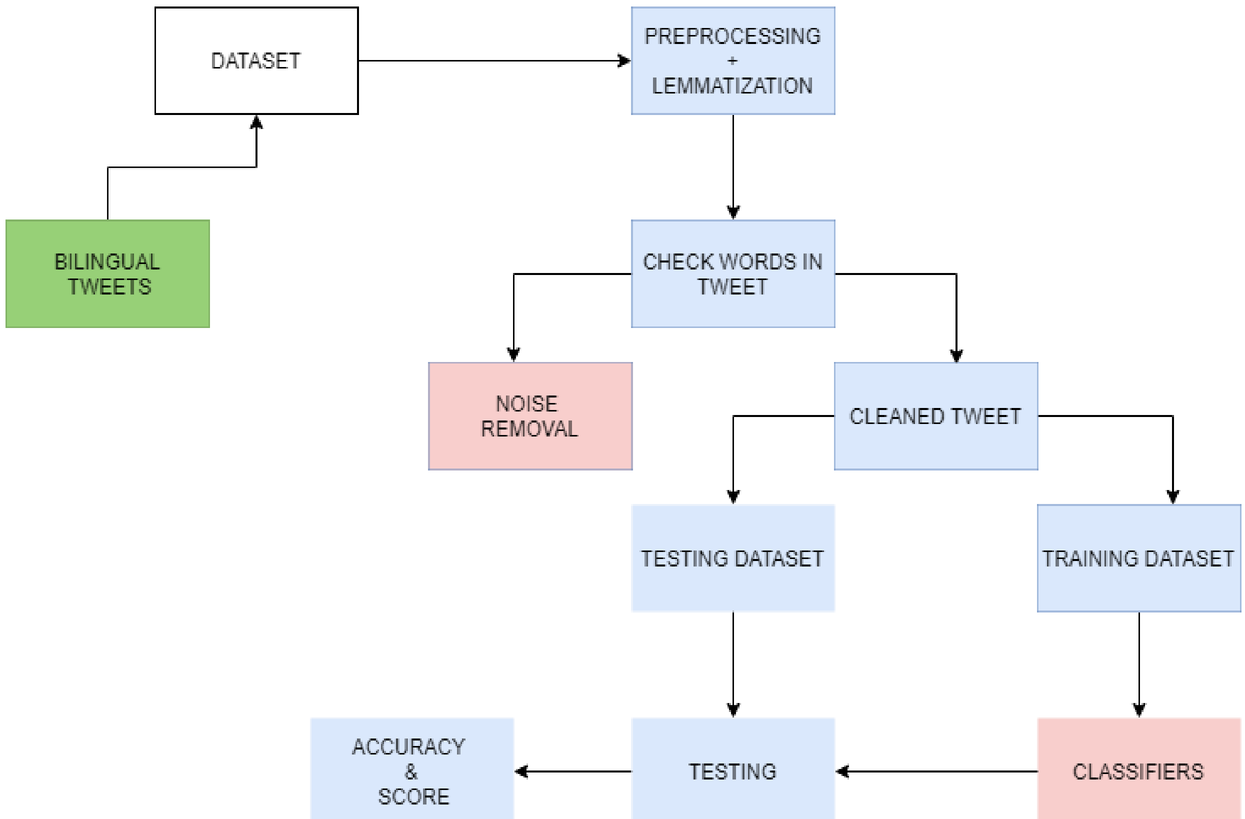
Lemmatization: Lemmatisation (or lemmatization) in linguistics is the process of grouping together the inflected forms of a word so they can be analysed as a single item, identified by the word's lemma, or dictionary form.

Naive Bayes Classifier: Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

Logistic Regression Classifier: Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

Empirical Analysis: Empirical analysis is a type of research dedicated to the discovery of concrete, verifiable evidence. Guided by the scientific method, empirical analysis allows researchers to remove personal bias and instead use concrete, accurate and repeatable real-world evidence to draw conclusions.

SYSTEM ARCHITECTURE



Module 1: Dataset

The dataset that was used was obtained from “Kaggle” called the Sentiment140 dataset. It contains 16 Million tweets extracted using the twitter API. The tweets have been annotated (0 = Negative, 1 = Positive) and they can be used to detect sentiment. The two columns that we will be needing are Label and Tweet.

	Label	number	date	no_query	name	Tweet
0	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
1	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
2	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
3	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....
4	0	1467811372	Mon Apr 06 22:20:00 PDT 2009	NO_QUERY	joy_wolf	@Kwesidei not the whole crew
...	...	...	...	...	...	...
1048570	4	1960186342	Fri May 29 07:33:44 PDT 2009	NO_QUERY	Madelineduggan x	My GrandMa is making Dinenr with my Mum
1048571	4	1960186409	Fri May 29 07:33:43 PDT 2009	NO_QUERY	OffRoad_Dude	Mid-morning snack time... A bowl of cheese noo...
1048572	4	1960186429	Fri May 29 07:33:44 PDT 2009	NO_QUERY	Falchion	@ShaDeLa same here say it like from the Termi...
1048573	4	1960186445	Fri May 29 07:33:44 PDT 2009	NO_QUERY	jonasobsessedx	@DestinyHope92 im great thaanks wbuu?
1048574	4	1960186607	Fri May 29 07:33:45 PDT 2009	NO_QUERY	sugababez	cant wait til her date this weekend

1048575 rows × 6 columns

Here the date, no_query, name, and number columns are irrelevant; hence we will not be dealing with them.

Module 2: Data Preprocessing

Data Preprocessing involves data cleaning, changing labels, dropping unwanted columns, removing Emoticons, replacing contraction, lemmatization and removal of noise.

After this, we must add cleaned tweets to the dataset and add cleaned tweets to a file. Then this must be added to the dataset and displayed.

The various steps are implemented as below:

1. Changing labels

Involves changing labels from 0 to negative and 1 to positive.

```
#0 to negative and 4 to positive
l=[]
for i in data["Label"]:
  if(i==0):
    l.append("negative")
  else:
    l.append("positive")
data['Comment']=l

data
```

2. Dropping unwanted columns

```
data=data.drop(columns=['number', 'date','name','no_query','Label'])
data
```

	Tweet	Comment
0	is upset that he can't update his Facebook by ...	negative
1	@Kenichan I dived many times for the ball. Man...	negative
2	my whole body feels itchy and like its on fire	negative
3	@nationwideclass no, it's not behaving at all....	negative
4	@Kwesidei not the whole crew	negative
...	...	...
4995	@radcs when are you putting a photo up?	positive
4996	oh wait, thunderstorms tomorrow?! ohSHIT, then...	positive
4997	@gwane and I'd go with either a nose ring or a...	positive
4998	what that - dissertation script is finished an...	positive
4999	@SITSGirls it does indeed. hope you are well t...	positive

5000 rows × 2 columns

3. *Removing Emoticons*

Involves mapping emoticons to emotions and replacing them with equivalent in dictionary

```
[19] def deEmojify(inputString):
       return inputString.encode('ascii', 'ignore').decode('ascii')
```

```
def emoticons():
    return {
        ":)":"smiley",
        ":-)":"smiley",
        ":-]":"smiley",
        ":-3":"smiley",
        ":->":"smiley",
        "8-)":"smiley",
        ":-}":"smiley",
        ":)":"smiley",
        ":]":"smiley",
        ":3":"smiley",
        ":>":"smiley",
        "8)":"smiley",
        ":}":"smiley",
        ":o)":"smiley",
        ":c)":"smiley",
        ":^)":"smiley",
        "=]":"smiley",
```

4. *Replace contraction*

The respective contractions are replaced via the contractions dictionary to display the complete word

```
def contractions():
    return {
        "ain't":"is not",
        "amn't":"am not",
        "aren't":"are not",
        "can't":"cannot",
        "'cause":"because",
        "couldn't":"could not",
        "couldn't've":"could not have",
        "could've":"could have",
        "daren't":"dare not",
        "daresn't":"dare not",
        "dasn't":"dare not",
        "didn't":"did not",
        "doesn't":"does not",
        "don't":"do not",
        "e'er":"ever",
        "em":"them",
        "everyone's":"everyone is",
        "finna":"fixing to",
        "gimme":"give me",
        "gonna":"going to",
        "gon't":"go not",
        "gotta":"got to",
        "hadn't":"had not",
        "hasn't":"has not",
```

5. Lemmatization

```
#Lemmatization : Root form of words in tweet

def lemmatization(sent):
  lemmatize=WordNetLemmatizer()
  sentence_after_lemmatization=[]
  for word,tag in pos_tag(word_tokenize(sent)):
    if(tag[0:2]=="NN"):
      pos='n'
    elif(tag[0:2]=="VB"):
      pos='v'
    else:
      pos='a'
    lem=lemmatize.lemmatize(word,pos)
    sentence_after_lemmatization.append(lem)

  st=""

  for i in sentence_after_lemmatization:
    if(i!="be" and i!="is" and len(i)!=1):
      st=st+" "+i

  c=0
  list_text=st.split()
  flag=0
  new_st=""
  for i in list_text:
    temp=i
    if(flag==1):
      flag=0
      continue
    if(i=="not" and (c+1)<len(list_text)):
      for syn in wordnet.synsets(list_text[c+1]):
        antonyms=[]
        for l in syn.lemmas():
          if l.antonyms():
            antonyms.append(l.antonyms()[0].name())
            temp=antonyms[0]
            flag=1
            break
        if(flag==1):
          break
    new_st=new_st+" "+temp
    c+=1
  return new_st
```

6. *Removal of noise*

This involves the removal of unwanted and irrelevant characters in the tweet.

```python
def removal_of_noise(sent):
    clean_sent=[]
    temp_st=""
    list_sent=sent.split(" ")
    c=0
    d=contractions()
    emoji=emoticons()
    for word in list_sent:
        #removal of url
        word = re.sub(r"http\S+", "", word)
        word = re.sub(r"[www.][a-zA-Z0-9_]+[.com]", "", word)
        #removal of account handles '@'
        word = re.sub("(@[A-Za-z0-9_]+)","", word)

        #replacing emoticons with their respective words
        if(word in emoji.keys()):
            word=emoji[word]
        #replacing short form words with their full form
        if(word.lower() in d.keys()):
            word=d[word.lower()]
        if(c==0):
            temp_st=word
        else:
            temp_st=temp_st+" "+word
        c=c+1
    sent=temp_st
    stop_words = set(stopwords.words('english'))
    stop_words.add('is')
    stop_words.remove('not')
    for word in word_tokenize(sent):
        if(word.lower() not in stop_words and word.lower() not in string.punctuation and word!="'" and word!="''" ):
            #print(word)
            word=spell.correction(word.lower())
            word=re.sub("[0-9]+","",word)
            word=re.sub("[.]+"," ",word)
            word=re.sub("[-]+"," ",word)
            word=re.sub("[_]+"," ",word)
            word = re.sub("~"," ", word)
            if(len(word)!=1):
                clean_sent.append(word.lower())
    cleaned_st=""
    for i in clean_sent:
        cleaned_st=cleaned_st+" "+i
    #print(cleaned_st)
    return lemmatization(cleaned_st)
```

7. *Preprocessing done as a whole by implementing the preproc() function*

```python
def preproc(text):
    #HTML tags are removed below
    text =BeautifulSoup(text).get_text()
    text =text.replace("’","'")
    new_text=sent_tokenize(text)
    result=0
    new_str=""
    #Emoticons removal + Noise Removal
    for i in new_text:
        j=deEmojify(i)
        res=removal_of_noise(j)
        new_str=new_str+" "+res
    return new_str
```

8. Displaying cleaned tweets

```
clean_list=[]
for i in data["Tweet"]:
  print()
  print(i)
  x=preproc(i)
  clean_list.append(x)
  print()
  print(x)
  print("------------------------")
```

```
 enjoy nice weather
------------------------

@trendhunter Very nice pics! Thanks for sharing it

 nice pic  thanks share
------------------------

@Hollywood_Trey


------------------------

is so anxious for thursday! I can't wait to see mike its been 3 and a half months!

 anxious thursday  not wait see mike half month
------------------------

@FinancegradTH Some days I have too much to say. Not a bad thing to be at a loss for words, you use your brain for much greater things.

 day much say  good thing loss word use brain much great thing
------------------------
```

9. Writing Cleaned Tweets to a file

```
[31] with open('cleaned_tweet.txt', 'w') as f:
       for item in clean_list:
         f.write("%s\n" % item)
```

10. Adding Cleaned Tweets as a column to data

```
[91] #reading from file cleaned tweets and storing in a cleaned tweets column in the dataframe
     filename = "cleaned_tweet.txt"
     with open(filename) as f:
       lines = f.read().splitlines()
     lines
     data["cleaned_tweets"]=lines
```

11. Preprocessed Data

```
data
```

	Label	number	date	no_query	name	Tweet	Comment	cleaned_tweets
0	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...	negative	upset not
1	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...	negative	
2	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire	negative	itchy
3	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....	negative	mad not
4	0	1467811372	Mon Apr 06 22:20:00 PDT 2009	NO_QUERY	joy_wolf	@Kwesidei not the whole crew	negative	not
...	...	...	...	...	...	...	...	...
4995	4	1468239135	Tue Apr 07 00:28:57 PDT 2009	NO_QUERY	Radt	@radcs when are you putting a photo up?	positive	
4996	4	1468239145	Tue Apr 07 00:28:57 PDT 2009	NO_QUERY	Disaster08	oh wait, thunderstorms tomorrow?! ohSHIT, then...	positive	love love love
4997	4	1468239164	Tue Apr 07 00:28:58 PDT 2009	NO_QUERY	lilac_dreamer	@gwane and I'd go with either a nose ring or a...	positive	
4998	4	1468239307	Tue Apr 07 00:29:01 PDT 2009	NO_QUERY	TButt1983	what that - dissertation script is finished an...	positive	
4999	4	1468239336	Tue Apr 07 00:29:01 PDT 2009	NO_QUERY	designmama	@SITSGirls it does indeed. hope you are well t...	positive	well

5000 rows × 8 columns

Module 3: Feature Extraction

Feature extraction generally refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set. It yields better results than applying machine learning directly to the raw data. Here we perform reading of the adjective file then extract adjectives from the tweets and create frequency adjective distribution. This will yield us cleaned tweets with features and we can display data with their features.

1. Reading adjective File

```
[93]  filename = "drive/MyDrive/CD PROJECT/english_adjectives.txt"
      with open(filename) as f:
        lines = f.read().splitlines()
      lines
      adjectives=lines
```

2. Extracting adjectives from file

```
all_words=[]
negative=["not"]
for i in data["cleaned_tweets"]:
  for word in word_tokenize(i):
    if(word in adjectives or word in negative):
     all_words.append(word)
```

3. Creating frequency distribution

```
[95]  import nltk
      BagOfWords = nltk.FreqDist(all_words)
      #BagOfWords
      #len(BagOfWords)
      word_features = list(BagOfWords.keys())[:5000]
      #len(word_features)
      #word_features
```

4. Replacing clean tweet with feature

```
new_list=[]
for i in data["cleaned_tweets"]:
  st=""
  for j in i.split():
    if(j in word_features):
      st=st+" "+j
  new_list.append(st)

data["cleaned_tweets"]=new_list
```

5. Displaying data and feature

```
data
```

	Tweet	Comment	cleaned_tweets
0	is upset that he can't update his Facebook by ...	negative	upset not
1	@Kenichan I dived many times for the ball. Man...	negative	
2	my whole body feels itchy and like its on fire	negative	itchy
3	@nationwideclass no, it's not behaving at all....	negative	mad not
4	@Kwesidei not the whole crew	negative	not
...	...	...	...
4995	@radcs when are you putting a photo up?	positive	
4996	oh wait, thunderstorms tomorrow?! ohSHIT, then...	positive	love love love
4997	@gwane and I'd go with either a nose ring or a...	positive	
4998	what that - dissertation script is finished an...	positive	
4999	@SITSGirls it does indeed. hope you are well t...	positive	well

5000 rows × 3 columns

Module 4: Training and Testing

This step involves splitting the dataset into training and testing samples, and then proceeding with training and testing the data.

Training:

```
[ ]  training_set=[]
     count=0
     for i in (X_train["cleaned_tweets"]):
       training_set.append((i.split(),Y_train[count]))
       count+=1

     def list_to_dict(words_list):
       return dict([(word, True) for word in words_list])

     training_set_formatted = [(list_to_dict(element[0]), element[1]) for element in training_set]
     #training_set_formatted
```

Testing:

```
test_set=[]
count=0
for i in (X_test["cleaned_tweets"]):
  test_set.append((i.split(),Y_test[count]))
  count+=1

def list_to_dict(words_list):
  return dict([(word, True) for word in words_list])

test_set_formatted= [(list_to_dict(element[0]), element[1]) for element in test_set]
```

Logistic Regression Classifier

```
print("LOGISTIC REGRESSION\n")
LogReg_clf = SklearnClassifier(LogisticRegression())
LogReg_clf.train(training_set_formatted)
print("Acuracy Percentage = ", (nltk.classify.accuracy(LogReg_clf, test_set_formatted))*100)
accuracy.append([(nltk.classify.accuracy(LogReg_clf, test_set_formatted))*100,"LogReg"])
classifiers.append([LogReg_clf,"LogisticRegression"])
target_names = [ 'positive','negative']
print("\nClassifiaction Report\n")
print(classification_report(Y_test, preds, target_names=target_names))
```

```
LOGISTIC REGRESSION

Acuracy Percentage =  60.8

Classifiaction Report

              precision    recall  f1-score   support

    positive       0.56      0.79      0.65       357
    negative       0.69      0.44      0.54       393

    accuracy                           0.60       750
   macro avg       0.63      0.61      0.60       750
weighted avg       0.63      0.60      0.59       750
```

Naïve Bayes Classifier

```
print("NAIVE BAYES\n")
classifier = nltk.NaiveBayesClassifier.train(training_set_formatted)
print("Accuracy Percentage = ", (nltk.classify.accuracy(classifier, test_set_formatted))*100)
classifiers.append([classifier,"NaiveBayes"])
accuracy.append([(nltk.classify.accuracy(classifier, test_set_formatted))*100,"NB"])
target_names = [ 'positive','negative']
print("\nClassifiaction Report\n")
print(classification_report(Y_test, preds, target_names=target_names))
```

```
NAIVE BAYES

Accuracy Percentage =  60.4

Classifiaction Report

              precision    recall  f1-score   support

    positive       0.56      0.79      0.65       357
    negative       0.69      0.44      0.54       393

    accuracy                           0.60       750
   macro avg       0.63      0.61      0.60       750
weighted avg       0.63      0.60      0.59       750
```

Module 5: Performance Evaluations

The classifiers used have been explained under the data dictionary section of this document and the same have been used to determine performance evaluations. Here the F1 Score and accuracy percentage have been presented with the confusion matrix.

Naive Bayes:

```
NAIVE BAYES

Accuracy Percentage =  59.46666666666667

Classifiaction Report

              precision    recall  f1-score   support

    positive       0.69      0.43      0.53       399
    negative       0.55      0.78      0.64       351

    accuracy                           0.59       750
   macro avg       0.62      0.61      0.59       750
weighted avg       0.62      0.59      0.58       750
```

Logistic Regression:

```
LOGISTIC REGRESSION

Acuracy Percentage =  59.333333333333336

Classifiaction Report

              precision    recall  f1-score   support

    positive       0.69      0.44      0.53       399
    negative       0.55      0.77      0.64       351

    accuracy                           0.59       750
   macro avg       0.62      0.60      0.59       750
weighted avg       0.62      0.59      0.58       750
```

Module 6: Bilingual Tweets

The final module now deals with splitting the text into features and calling the preprocessing functions. Once the feature extraction is done, each classifier is evoked.

1. Import libraries

```
from googletrans import Translator
translator = Translator(service_urls=['translate.googleapis.com'])
```

2. Function to create List of Features

```
[69] def features(text):
       new_list=[]
       for i in text.split():
         if(i in adjectives):
           new_list.append(i)
       return new_list
```

3. Classify Function
    1. PreProcessing
    2. Extract Features
    3. Test Data using Classifiers
    4. Print Result

```
[114] def text_classify(text):
        cleaned_text=preproc(text)
        temp=features(cleaned_text)
        test_data=list_to_dict(temp)
        print(temp)
        print("Tweet given by user : ",text)
        for i in classifiers:
          print(i[1])
          determined_label=i[0].classify(test_data)
          print("This Tweet is ",determined_label)
          print("\n\n")
        c=0
```

4. Translating users input to english

```
[66]  #input from the user which will be used to classify
      def tanglish(input_text):
        translator = Translator(service_urls=['translate.google.co.in'])
        x=translator.translate(input_text,src="ta",dest="en")
        text_classify(x.text)
```

```
[67]  #input from the user which will be used to classify
      from textblob import TextBlob
      def tanglish2(input_text):
        l=input_text.split()
        st=""
        for i in l:
          word=TextBlob(i)
          if(word.detect_language()=="ta"):
            translator = Translator(service_urls=['translate.google.co.in'])
            x=translator.translate(i,src="ta",dest="en")
            st=st+" "+x.text
          else:
            st=st+" "+i
        text_classify(st)
```

Calling the function for processing:

```
def func(input_text):
  l=input_text.split()
  flag=0
  for i in l:
    k=len(i)
    if(k<3):
      flag=1
      tanglish(input_text)
  if(not(flag)):
    tanglish2(input_text)
```

Result and Discussion

The output:

```
func("appa is angry")
```

```
['angry']
Tweet given by user :  Dad Is Annry
LogisticRegression
This Tweet is  negative
------------------------------
NaiveBayes
This Tweet is  negative
------------------------------
```

```
func("it is a big veedu")
```

```
['big']
Tweet given by user :  It's A Big House
LogisticRegression
This Tweet is  positive
------------------------------
NaiveBayes
This Tweet is  positive
------------------------------
```

```
func("saapadu is bad")
```

```
['bad']
Tweet given by user :  Meals Is Bad
LogisticRegression
This Tweet is  negative
------------------------------
NaiveBayes
This Tweet is  negative
------------------------------
```

```
func("Today is a gud day")
```

```
['good']
Tweet given by user :  Today Is a Good Day
LogisticRegression
This Tweet is  positive
-------------------------------
NaiveBayes
This Tweet is  positive
-------------------------------
```

Conclusion and Final Summary

Observing the output obtained, the classification of text takes place to provide the sentiment of the text. The code successfully presents the underlying emotions through preprocessing, feature extraction and sentiment analysis. The data set taken has been cleaned, the unnecessary columns dropped, the redundant words were removed, emoticons were converted to the corresponding words in the dictionary, contractions were converted to their complete format and feature extraction was proceeded with. Then the Google translate module was used to account for the Tamil words.

In conclusion, we've created a method by which tweets containing words typed in Tamil can be also analyzed by sentiment analysis, making sentiment analysis inclusive of regional languages.

References

1. K. Ravi and V. Ravi, "Sentiment classification of Hinglish text," *2016 3rd International Conference on Recent Advances in Information Technology (RAIT)*, 2016, pp. 641-645, doi: 10.1109/RAIT.2016.7507974.
2. Nigam, Nitika & Yadav, Divakar. (2018). Lexicon-Based Approach to Sentiment Analysis of Tweets Using R Language: Second International Conference, ICACDS 2018, Dehradun, India, April 20-21, 2018, Revised Selected Papers, Part I. 10.1007/978-981-13-1810-8_16.
3. Expressively vulgar: The socio-dynamics of vulgarity and its effects on sentiment analysis in social media Isabel Cachola*‡ Eric Holgate*† Daniel Preoţiuc-Pietro♦ Junyi Jessy Li†‡Department of Mathematics, †Department of Linguistics,The University of Texas at Austin
4. Lexicon-Based Approach to Sentiment Analysis of Tweets Using R Language: Second International Conference, ICACDS 2018, Dehradun, India, April 20-21, 2018, Revised Selected Papers, Part I