(SentiMix Hindi-English)

M. Shehroz Wali Khan
FAST School of Computing
NUCES, Islamabad
Islamabad, Pakistam
i170308@nu.edu.pk

***Abstract*—**Mixing languages, also known as code- mixing, is a norm in multilingual societies. Multilingual people, who are non-native English speakers, tend to code-mix using English-based phonetic typing and the insertion of anglicisms in their main language. In addition to mixing languages at the sentence level, it is fairly common to find the code-mixing behavior at the word level. This linguistic phenomenon poses a great challenge to conventional NLP systems, which currently rely on monolingual resources to handle the combination of multiple languages. The objective of this proposal is to bring the attention of the research community towards the task of sentiment analysis in code-mixed social media text. Specifically, we focus on the combination of English with Spanish (Spanglish) and Hindi (Hinglish), which are the 3rd and 4th most spoken languages in the world respectively.

Introduction

The task was to predict the sentiment of a given code-mixed tweet. The sentiment labels are positive, negative, or neutral, and the code-mixed languages will be English-Hindi and English-Spanish. Besides the sentiment labels, we will also provide the language labels at the word level. The word-level language tags are en (English), spa (Spanish), hi (Hindi), mixed, and univ (e.g., symbols, @ mentions, hashtags). Efficiency will be measured in terms of Precision, Recall, and F-measure.

Propose Approach

Here are the following steps were involved in doing the project:-

1. *Importing required libraries*

I imported all the required libraries and some extras in case I'd need them in the future.

```
import numpy as np
import pandas as pd
import re
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import chi2_contingency

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
from sklearn.feature_extraction.text import CountVectorizer

from subprocess import check_output
```

2. *Mounting Google Drive to Colab*

I uploaded all the data sets in the google drive. Before loading the datasets into the colab, the first thing that is needed is to mount the google drive in the colab. When you write the code written below and run it, colab provides a url and adds a field to write an authorization code. You have to put the authorization code which you will be getting by visiting the url and allowing some permission through your google account. In the end it provides you a code and you just have to paste it into this field and press enter.
After that this text will appear and this will confirm that the drive is now mounted.

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

3. *Defining Reading Functions*

I have defined two functions. First for train sets and second for test sets.
readandSplit():

The main purpose of this function is to get open and read the datasets and use some splitting functions. Read each tweet and append it into the tweets list and read label i.e. sentiments (positive, negative, neutral) and append it into label list and return both lists.

readInList():

The main purpose of this function is to get open and read the test datasets and use some splitting functions. Read each tweet and append it into the tweets list and return this list.

For Train Datasets

```
def readandSplit(filename):
  with open(filename) as f:
        data = f.read().split('\n')
  tweets = []
  label = []
  tweet = ""


  for d in data:
      if(len(d)==0):
          continue
      elif(len(d.split())==3):
          label.append(d.split()[2])
          if(tweet):
              tweets.append(tweet)
              tweet = ""
      elif(len(d.split())==2):
              tweet += d.split()[0]+' '

  tweets.append(tweet)

  return label,tweets
```

For Test Datasets

```
def readInList(filename):
  with open(filename) as f:
        data = f.read().split('\n')
  tweets = []
  label = []
  tweet = ""


  for d in data:
    if(len(d)==0):
      continue
    elif(d.split()[0]=='meta'):
      if(tweet):
        tweets.append(tweet)
        tweet = ""

    elif(len(d.split())==2):
              tweet += d.split()[0]+' '

  tweets.append(tweet)

  return tweets
```

4. *Loading Datasets & Data Preprocessing*

Loading Datasets
We will be passing the train and test datasets into the readandSplit functions and assign it into lists of label and trainData (for Train Datasets)
And assign it into a list of vallebl, valData (Validation Datasets).
And to get the maximum training datasets, we are concatenating trainData with valData and label with vallebl.

```
label,trainData= readandSplit("/content/drive/MyDrive/Dataset/Train/train_14k_split_conll.txt")
vallebl,valData= readandSplit("/content/drive/MyDrive/Dataset/Validation/dev_3k_split_conll.txt")

trainData+=valData
label+=vallebl
```

Vectorize the TrainData
I used a method from CountVectorizer from sklearn.feature_extraction.text
I.e fit_transform() on trainData which learn the vocabulary dictionary and return a document-term matrix and then convert it into an array using toarray() and assign it into variable X.

```
vectorizer = CountVectorizer(min_df=1)
X = vectorizer.fit_transform(trainData).toarray()
```

5. *Model*

Note: *I also used LSTM Algorithm for it, but the accuracy was low around 58 and it took so much time to train so I just stopped using it and shift it towards Random Forest Classifiers*

Final Model Implemented: ***Random Forest Classifier***

- *A. Putting Data into the Random Forest Classifier*
- *B. Fitting the Training Data*
- *C. Predicting the Test Data*

Brief Details:-
I called the RandomForestClassifier function and then used fit() to fit the train Data. It took around 5-10 minutes in training.

```
rfc1=RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=42,
                       verbose=0, warm_start=False)
rfc1.fit(X, label)
```

After fitting the test data, I used the predict() of RandForestClassifier to predict the testData after putting it into the list of list1 and then vectorizing it using vectorizer.transform(list1) and then convert into toarray() and assign it into p. After that we converted p into the list.

```
list1=[]
for i in range(len(testData)):
  testData[i]
  list1.append(testData[i])
  list1

p=rfc1.predict(vectorizer.transform(list1).toarray())
```

Results

1. *Calculating Accuracy*

We have the predicted labels. To get the accuracy, we need the original test label so we will open and read it and then add all the sentiment columns entries into the list and in the end we have the original Labels in the testLabels list.

```
#ORIGINAL TEST LABELS

with open("/content/drive/MyDrive/Dataset/Test/test_labels_hinglish.txt") as f:
      data = f.read().split('\n')

intermediateLabels= []
tweet = ""

i=0

for i in range(len(data)):

  for j in range(len(data[i])):
    if(data[i][j]==',' ):
      intermediateLabels.append(data[i][j+1:])

testLabels=[]

for i in range(len(intermediateLabels)-1):
  testLabels.append(intermediateLabels[i+1])
```

```
from sklearn.metrics import accuracy_score

accuracy_score(testLabels, p)
```

```
0.6456666666666667
```

Note: First I simply called out the RandomForestClassifier() and I got a 63.5 score but when I used the following parameters and put the random_state=42, the accuracy increased to 64.8%.

shehroz_308_c 2 12/09/20 0.648 (101)

2. *Getting the uids from testData*

As we needed answer.txt which has two columns.

1. Uids
To get the uids, I loaded the test dataset and appended all the uids from the file into the list and that's how I got the list of uids.
2. Predicted labels/sentiment.
We already got the list i.e. p.

```
p=list(p)
```

```
#Taking out Uids
with open("/content/drive/MyDrive/Dataset/Test/Hindi_test_unalbelled_conll_updated.txt") as f:
      data = f.read().split('\n')

uids = []
tweet = ""


for d in data:
  if(len(d)==0):
    continue
  elif(d.split()[0]=='meta'):
    uids.append(d.split()[1])
```

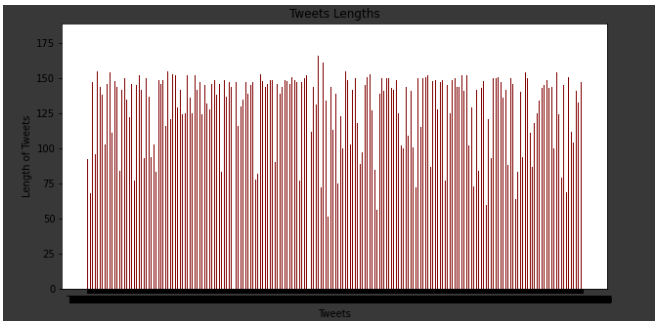
3. *Saving the results in answer.txt*

Save the sentiments along with uids in the answer.txt file

```
print(len(p))
count=0
store1 = []
for i in range(0, len(uids)):
    save = ""
    res = p[i]

    if (res == "positive"):
        save += uids[i]
        save += ",positive\n"
        count+=1

    elif (res == "neutral"):
        save +=uids[i]
        save += ",neutral\n"
        count+=1

    elif (res == "negative"):
        save += uids[i]
        save += ",negative\n"
        count+=1
    else:
        print(uids[i])

    store1.append(save)


with open('answer.txt','w',encoding='utf-8') as file:
    i=0
    while (i<len(store1)):
        file.write(store1[i])
        i+=1
```

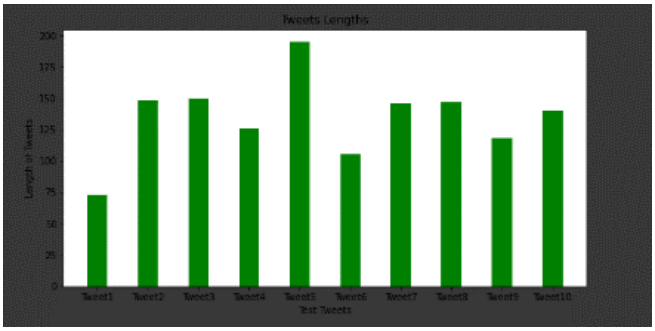
Graphs

1. *Length of tweets of Train Data*




```
sizeTweets=[]
tweetNumber=[]
for i in range(len(trainData)):
  sizeTweets.append(len(trainData[i]))
  tweetNumber.append("Tweet"+str(i+1))

fig = plt.figure(figsize = (10, 5))

# creating the bar plot
plt.bar(tweetNumber, sizeTweets, color ='maroon',
        width = 0.4)

plt.xlabel("Tweets")
plt.ylabel("Length of Tweets")
plt.title("Tweets Lengths")
plt.show()
```

2. *Length of first 10 tweets of Test Data*




```
sizeTweets=[]
tweetNumber=[]
for i in range(len(testData[:10])):
  sizeTweets.append(len(testData[i]))
  tweetNumber.append("Tweet"+str(i+1))

fig = plt.figure(figsize = (10, 5))

# creating the bar plot
plt.bar(tweetNumber, sizeTweets, color ='green',
        width = 0.4)

plt.xlabel("Test Tweets")
plt.ylabel("Length of Tweets")
plt.title("Tweets Lengths")
plt.show()
```


ACKNOWLEDGMENT

I would like to express my special thanks of gratitude to my Artificial Intelligence Teacher *Mr. Umair Arshad* for his able guidance and support in completing my Project.

REFERENCES