CS564 : Foundations of Machine Learning
Assignment 3

By Priyanka Sachan (1901CS43)

Problem Statement

The assignment targets to implement Hidden Markov Model (HMM) to perform Named Entity Recognition (NER) task.

Installation

Install the following dependencies either using pip or through conda in a Python 3.5+ environment:

```
python3 -m pip install pandas
```

Running the program

Use the following command to run the program :

```
Python3 hmm_ner.py
```

Implementation

Code added in zip file or check Notebook.

Note

We have used BIO tagging here

```
'B'   : Beginning of named entity
'I'   : Inside of named entity
'O'   : Outside of named entity
```

Import libraries and packages

```
import numpy as np # linear algebra
import pandas as pd # data processing

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

Import data

```
dataset=[]
with open('NER-Dataset-Train.txt') as f:
   lines = f.readlines()
   sentence = []
   for line in lines:
     if line == '\n':
       if sentence:
         dataset.append(sentence)
         sentence = []
     else:
       sentence.append(tuple(line.split()))
```

HMM algorithm

Emission Probabilities

```
# Calculate emission probabilities
def emission_probabilities(data_train):
 emp={}
 count_words={'O':0,'I':0,'B':0}
 for i in range(len(data_train)):
   for j in range(len(data_train[i])):
     if data_train[i][j][0] not in emp.keys():
       emp[data_train[i][j][0]]={'O':0,'I':0,'B':0}
     emp[data_train[i][j][0]][data_train[i][j][1]]+=1
     count_words[data_train[i][j][1]]+=1
 for i in emp.keys():
   emp[i]['B']/=count_words['B']
   emp[i]['I']/=count_words['I']
   emp[i]['O']/=count_words['O']
 return emp
```

```
Emision Probabilities
    'Rickey    Smiley       had        me  laughing       all       day  \
O  0.000000  0.000000  0.001599  0.003655  0.000076  0.003579  0.005330
I  0.000000  0.003215  0.000000  0.000000  0.000000  0.000000  0.003215
B  0.002146  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000

      today         .         "  ...  @wethekings   Stetson  November  \
O  0.005102  0.034341  0.004112  ...     0.000076  0.000000  0.000076
I  0.000000  0.006431  0.000000  ...     0.000000  0.000000  0.000000
B  0.000000  0.000000  0.000000  ...     0.000000  0.002146  0.000000

     pissed      @TMZ    tested  positive   cocaine   sources  http://bi
O  0.000076  0.000076  0.000076  0.000076  0.000076  0.000076   0.000076
I  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000   0.000000
B  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000   0.000000

[3 rows x 4396 columns]
```

Start Probabilities

```python
# Calculate start probabilities
def start_probabilities(data_train):
 sp={'O':0,'I':0,'B':0}
 for i in range(len(data_train)):
   sp[data_train[i][0][1]]+=1
 sp['B']/=len(data_train)
 sp['I']/=len(data_train)
 sp['O']/=len(data_train)
 return sp
```

```
Start Probabilities
{'O': 0.9236111111111112, 'I': 0.0, 'B': 0.0763888888888889}
```

Transmission Probabilities

```python
# Calculate transmission probabilities
def transmission_probabilities(data_train):
 tmp={'B':{'B':0,'I':0,'O':0},'I':{'B':0,'I':0,'O':0},'O':{'B':0,'I':0,'O':0}}
 tout_count={'B':0,'I':0,'O':0}
 for i in range(len(data_train)):
   for j in range(len(data_train[i])-1):
     tmp[data_train[i][j][1]][data_train[i][j+1][1]]+=1
     tout_count[data_train[i][j][1]]+=1
 for i in ['B','I','O']:
   tmp[i]['B']/=tout_count[i]
   tmp[i]['I']/=tout_count[i]
   tmp[i]['O']/=tout_count[i]
 return tmp
```

```
Transmission Probabilities
          B         I        O
B  0.000000  0.009804  0.03284
I  0.471739  0.307190  0.00000
O  0.528261  0.683007  0.96716
```

Training model
& Validating model using 5 fold cross validation

```
sentences=len(dataset)//5
for idx in range(5):
  print('ROUND -',idx+1,'/ 5')
 # Split dataset in 4:1 ratio
 data_train= dataset[: sentences*(idx)]+dataset[sentences*(idx+1):]
 print('Length of train data: ',len(data_train))
 data_test=dataset[sentences*idx: sentences*(idx+1)]
 print('Length of test data: ',len(data_test))

 # Calculate emission probabilities
 emp=emission_probabilities(data_train)
 print('\nEmisision Probabilities')
 print(pd.DataFrame.from_dict(emp))

 # Calculate start probabilities
 sp=start_probabilities(data_train)
 print('\nStart Probabilities')
 print(sp)

 # Calculate transmission probabilities
 tmp=transmission_probabilities(data_train)
 print('\nTransmission Probabilities')
 print(pd.DataFrame.from_dict(tmp))

 # Validate on test set
 labels=[]
 preds=[]
 for i in range(len(data_test)):
   label=[]
   pred=[]
   for j in range(len(data_test[i])):
     label.append(data_test[i][j][1])
     if data_test[i][j][0] not in emp.keys():
         pred.append('UNK')
         continue
     if j==0:
       p={'B':emp[data_test[i][j][0]]['B']*sp['B'],
          'I':emp[data_test[i][j][0]]['I']*sp['I'],
```

```
          'O':emp[data_test[i][j][0]]['O']*sp['O']}
    elif pred[j-1]=='UNK':
      p={'B':emp[data_test[i][j][0]]['B'],
          'I':emp[data_test[i][j][0]]['I'],
          'O':emp[data_test[i][j][0]]['O']}
    else:
      p={'B':emp[data_test[i][j][0]]['B']*tmp[pred[j-1]]['B'],
          'I':emp[data_test[i][j][0]]['I']*tmp[pred[j-1]]['I'],
          'O':emp[data_test[i][j][0]]['O']*tmp[pred[j-1]]['O']}
    pred.append(max(zip(p.values(), p.keys()))[1])
  labels+=label
  preds+=pred
 print('\nMeasures')
accuracy=accuracy_score(labels, preds)
print('Accuracy Score: ',accuracy)
recall=recall_score(labels, preds,average='weighted')
print('Recall Score: ',recall)
precision=precision_score(labels, preds,average='weighted')
print('Precison Score: ',precision)
f1=f1_score(labels, preds,average='weighted')
print('F1 Score: ',f1)

print('----------------------------------------------------------------------')
```

```
Measures
Accuracy Score:  0.7235345581802275
Recall Score:  0.7235345581802275
Precison Score:  0.9537699603990373
F1 Score:  0.8193976940103095
```

Testing model

```
# Validate on test set
preds=[]
for i in range(len(data_test)):
 pred=[]
 for j in range(len(data_test[i])):
   if data_test[i][j][0] not in emp.keys():
       pred.append('UNK')
       continue
   if j==0:
     p={'B':emp[data_test[i][j][0]]['B']*sp['B'],
         'I':emp[data_test[i][j][0]]['I']*sp['I'],
         'O':emp[data_test[i][j][0]]['O']*sp['O']}
   elif pred[j-1]=='UNK':
     p={'B':emp[data_test[i][j][0]]['B'],
         'I':emp[data_test[i][j][0]]['I'],
         'O':emp[data_test[i][j][0]]['O']}
   else:
     p={'B':emp[data_test[i][j][0]]['B']*tmp[pred[j-1]]['B'],
         'I':emp[data_test[i][j][0]]['I']*tmp[pred[j-1]]['I'],
         'O':emp[data_test[i][j][0]]['O']*tmp[pred[j-1]]['O']}
   pred.append(max(zip(p.values(), p.keys()))[1])
 preds.append(pred)
 print('\nTest-',i)
 print(data_test[i])
 print(pred)
```

```
Test- 22
["'Come", 'to', '"', 'MRB', '@', '61', 'Roadhouse', 'BBQ', '"', 'Sunday', ',',
'October', '1.', 'from', '12:..', 'pm', 'to', '4:..', 'pm', '.', 'We', 'are',
'playing', 'for', 'the', '61', 'Roadhouse', '...', "http://fb.me/G1FbJGw1'"]
['O', 'O', 'O', 'B', 'O', 'O', 'UNK', 'I', 'O', 'O', 'O', 'O', 'O', 'O', 'O',
'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'UNK', 'O', 'UNK']
```