Project Progress Report 2

My task in the project is to build a classifier to classify the text snippets extracted from the books.

Since I had not received the exact snippets extracted from the books, I had to extract some sample snippets on my own for the experiments. Given few text files of the books I wrote the python code to extract the snippets (a piece of 42 words where “Chicago School” occurs in the book, where 20 words from either side of “Chicago School” are extracted).

```
import os.path, io
from nt import listdir
folderName = "G:\STUDY\MS in CS\Spring 2017\CS522 ADM\Project\snippets jstor\snippets jstor\\"
file_list = [f for f in listdir(folderName)]
outputfolder="G:\STUDY\MS in CS\Spring 2017\CS522 ADM\Project\Snippets"

for txtfile in file_list:
    with io.open(folderName + txtfile,'r',encoding='utf-8') as tfile:
        counter = 0,prevLine = ’’, nextLine = '', temp = ’’, prevLineContained = 0
        try:
            lines = tfile.readlines()
            for i,line in enumerate(lines):
                if "Chicago school" in line or "Chicago School" in line:
                    for c in line:
                        c.encode('utf-8')
                    counter = counter + 1
                    if "Chicago school" in line:
                        thisline = line.split("Chicago school")
                    elif "Chicago School" in line:
                        thisline = line.split("Chicago School")
                    else:
                        continue

                    prevWords = []
                    #finding previous 20 characters
                    for k in reversed(thisline[0].split(" ")):
                        if len(prevWords) < 20:
                            prevWords.append(k)
                        else:
                            break
                    j = i - 1
                    while j > 0 and len(prevWords) < 20:
                        words = reversed(lines[j].split(" "))
                        j = j - 1
                        for word in words:
                            for c in word:
                                c.encode('utf-8')
                            prevWords.append(word)
                            if len(prevWords) >= 20:
                                break

                    nextWords = []
                    #finding previous 20 characters
                    for k in thisline[1].split(" "):
                        if len(nextWords) < 20:
                            nextWords.append(k)
                        else:
                            break
                    j = i + 1
                    while j < len(lines) and len(nextWords) < 20:
                        words = lines[j].split(" ")
                        j = j + 1
                        for word in words:
                            for c in word:
```

```
                c.encode('utf-8')
            nextWords.append(word)
            if len(nextWords) >= 20:
                break
        fileParts = txtfile.split(".")
        snippet = ""
        snippet = ' '.join(reversed(prevWords)) + " Chicago School" + ' '.join(nextWords)
        #snippet file name
        fileName = outputfolder + \
                "\\" + \
                fileParts[0] + \
                "__" + \
                str(counter) + \
                "." + fileParts[1]

    with io.open(fileName,'w+',encoding='utf-8') as writeFile:
        writeFile.write(snippet + '\n')
except Exception as e:
    print("Caught an exception")
    continue
```

Approaches:

I decided to try below classifiers for classification with tfidf using bi-grams model:

1. Logistic Regression
2. Support Vector Machines
3. Neural Networks

I am using above classifiers as my data i.e., snippets are smaller texts and are coherent on a smaller topic as they span within couple of lines in the book. And the above classifiers will learn the parameters for the model based on the relation of the words in the snippet.

The bi-gram model helps to find the content in which each word is used and should perform better than just the word counts.

Implementation and Evaluation:

I am using the scikit implementation of above mentioned classifiers.
I have also used scikit implementation to find the tfidf of the text snippets.

For evaluating the model performances, I am using the cross validation Accuracy score of the classifiers. The higher the accuracy, the better is the model, but I want to make sure that the model doesn't over fit the training data.

Experiment and steps:

Below are my approaches and steps in mu experiments conducted as of now:

1. Read the data from the files.
2. Preprocess the data i.e. remove English stop words, remove punctuations.
3. Find the tfidf representation of the bi-grams in the data with occurrences between (6,80)
4. Build the model
5. Find the quality if the model using the cross validation score of the model on the data with different cv values ranging from 3 to 10.

Below is the code snippet I wrote to load the data, preprocess it and find tfidf matrix for bi-grams:

```
import os.path, string
from os import listdir
snippetFolder = "/home/abc/ADM/Train_Data"
testSnippetFolder = "/home/abc/ADM/Test_Data"
folder_list = [f for f in listdir(snippetFolder)]
puncs = string.punctuation

X = [], Y = []
for folder in folder_list:
   files_list = [f for f in listdir(snippetFolder + "/" + folder)]
   for file in files_list:
      filePath = snippetFolder + "/" + folder + "/" + file
      f = open(filePath,'r',encoding='utf-8')
      data = f.readlines()
      textSnippet = ''.join(data)
      textSnippet = textSnippet.replace("'\n","")

      for p in puncs:
        if p in textSnippet:
          textSnippet = textSnippet.replace(p,"")

      X.append(textSnippet)
      Y.append(folder)

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import cross_val_score

tfidf = TfidfVectorizer(input='content',encoding='utf-
8',decode_error='ignore',strip_accents='ascii',ngram_range=(2,2),stop_words='english',max_df=80,min_df=6,max_features=None,norm='l1')
train = tfidf.fit_transform(X,Y)
```

Below are the classifiers leant and their average accuracy scores:

```
cv=5
#Logistic Regression Classifier
from sklearn.linear_model import LogisticRegression
clf_lr = LogisticRegression()
acc = cross_val_score(clf_lr,train,Y,scoring="accuracy",cv=cv)
print(acc)
print(np.mean(acc))
```

```
[ 0.54166667  0.60869565  0.69565217  0.60869565  0.57142857]
0.605227743271
```

```
#Support Vector Machine Classifier
from sklearn.svm import SVC, LinearSVC
clf_svc = SVC(C=0.1,random_state=123)
acc = cross_val_score(clf_svc,train,Y,scoring="accuracy",cv=cv)
print(acc)
print(np.mean(acc))

clf_svc = LinearSVC(C=0.1,random_state=123)
acc = cross_val_score(clf_svc,train,Y,scoring="accuracy",cv=cv)
print(acc)
print(np.mean(acc))
```

```
[ 0.375       0.39130435  0.39130435  0.39130435  0.38095238]
0.385973084886
[ 0.58333333  0.60869565  0.69565217  0.60869565  0.57142857]
0.613561076605
```

```
#Neural Network Classifier
from sklearn.neural_network import MLPClassifier
clf_NN =
MLPClassifier(hidden_layer_sizes=(5,10),activation="logistic",solver="lbfgs",alpha=0.01,learning_rate="adaptive",max_iter=100,shuffle=True,ea
rly_stopping=True,random_state=123)
acc = cross_val_score(clf_NN,train,Y,scoring="accuracy",cv=cv)
print(acc)
print(np.mean(acc))
```

```
[ 0.58333333  0.60869565  0.69565217  0.60869565  0.61904762]
0.623084886128
```

Summary and Results:

Classifier	Accuracy (average cross validation accuracy with cv = 5)
Logistic Regression	0.605
Linear Support Vector Machine	0.6136
Neural Networks	0.6231

As we can see from the above table the bi-gram model is giving decent results. But I am concerned whether the classifiers will generalize well to the unseen data from a different set and that the model is not overfitting the training data.