CS_583_Assignment_4

December 3, 2021

1 CS 583 Section B Assignment 4

Task Description:

Given a sentence, predict if the sentence is a question or not

There are two steps: 1) detect a question and 2) detect the type of the question. We provide the code for step1, and require you to finish step 2.

By

- Aditya Nittala(10466689)
- Govinda Tanikella(10448181)

[ ]:
```
!pip install pycorenlp
```

[55]:
```
#import pandas as pandas
from pycorenlp import StanfordCoreNLP
from sklearn.feature_extraction.text import TfidfVectorizer,CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.multiclass import OutputCodeClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn import svm
class_to_int = {'what': 0, 'affirmation': 1, 'unknown': 2, 'who': 3, 'when': 4}
class isQuestionBasic():

    #Init Constructur
    #Initialize stanford core nlp local instance on port 9000
    def __init__(self):
        self.nlp = StanfordCoreNLP("http://localhost:9000")

    def isQuestion(self, sentence):
        if '?' in sentence:
            return 1
        output = self.nlp.annotate(sentence, properties={
            'annotators': 'parse',
            'outputFormat': 'json',
            'timeout': 1000
        })
```

```
        if('SQ' or 'SBARQ') in output['sentences'][0]["parse"]:
            return 1
        else:
            return 0



    def parse_and_train_test_split(self):
        with open("sample.txt") as f:
            contents = f.read()
            contents = contents.split("\n")[:-1]
            texts = [x.split(",,,")[0].strip() for x in contents]
            labels = [class_to_int[x.split(",,,")[1].strip()] for x in contents]
            X_train, X_test, y_train, y_test = train_test_split(texts, labels,␣
↪test_size=0.2)
            return X_train, X_test, y_train, y_test

    def MVNVClassifier(self):
        X_train, X_test, y_train, y_test = self.parse_and_train_test_split()
        cv = TfidfVectorizer(min_df=1)
        X_train = cv.fit_transform(X_train)
        X_test = cv.transform(X_test)
        mnb = MultinomialNB()
        mnb.fit(X_train, y_train)
        score = mnb.score(X_test, y_test)
        return score
    def SVMClassifier(self):
        X_train, X_test, y_train, y_test = self.parse_and_train_test_split()
        cv = TfidfVectorizer(min_df=1)
        X_train = cv.fit_transform(X_train)
        X_test = cv.transform(X_test)
        svmc = svm.SVC()
        svmc.fit(X_train, y_train)
        score = svmc.score(X_test, y_test)
        return score
    def Modified_SVM(self):
        X_train, X_test, y_train, y_test = self.parse_and_train_test_split()
        cv = CountVectorizer(min_df=1)
        X_train = cv.fit_transform(X_train)
        X_test = cv.transform(X_test)
        svmc = svm.SVC(kernel='sigmoid')
        svmc.fit(X_train, y_train)
        score = svmc.score(X_test, y_test)
        return score
    def Modified_NB(self):
      #Implementing Improved Multinomial NB
      X_train, X_test, y_train, y_test = self.parse_and_train_test_split()
```

```
      cv = CountVectorizer(min_df=1)
      X_train = cv.fit_transform(X_train)
      X_test = cv.transform(X_test)
      clf = OutputCodeClassifier(estimator=MultinomialNB(),code_size=2)
      clf.fit(X_train, y_train)
      score = clf.score(X_test, y_test)
      return score


if __name__=='__main__':
  BQ = isQuestionBasic()
  MNNBClassifer = BQ.MVNVClassifier()
  SVMC = BQ.SVMClassifier()
  print("Accuracy of the Mutlinomial Naive Bayes is: ", MNNBClassifer*100)
  print("Accuracy of the SVM classifer is: ",SVMC*100)
  Mod_SVM = BQ.Modified_SVM()
  Modified_NB = BQ.Modified_NB()
  print("The improved accuracy for MNB is: ", Modified_NB*100)
  print("The improved SVM accuracy is: ",Mod_SVM*100)
```

```
Accuracy of the Mutlinomial Naive Bayes is:  75.42087542087542
Accuracy of the SVM classifer is:  93.60269360269359
The improved accuracy for MNB is:  90.57239057239057
The improved SVM accuracy is:  96.63299663299664
```