```python
# CIS 511 NLP - Assignment 4.1 - Natural Language Understanding for Dialog Systems

"""
Created on Sat April 11 20:30:18 2020

@author: Siyu Yang
@unique name: siyuya
@UMID:76998080
"""

from collections import defaultdict
import pandas
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import re
import sys


#=== Functions ===#

#   1)  find the middle content between begin word and end word
def middle_content(begin, end, content):
    mid_content  =''

    if content.find(begin): # find the line start with begin words
        beginword = content[content.find(begin):content.rfind(end)]
        mid_content = beginword[len(begin):]
        return mid_content


#   2)  convert I/O/B into number 1/0/2
def IOB_to_Num(x):
    IOB_Char = x[-1]
    if IOB_Char=='I':
        return 1
    elif IOB_Char =='B':
        return 2
    else :
        return 0



#   3)  Process Train File to get train data
def Process_TrainFile(trainfile):

    trainList = open(trainfile,'r').read() #read the file and remove the empty lines.
    name_list = []
    train_fea_dic = defaultdict(list) # use default dictionary as train file feature 
    dictionary


    for line in trainList.split("\n\n"):

        if("<class" in line):
            text = line.split("\n")
            text1 = text[0]
            text2 = re.sub('([.,!?()])', r' \1 ', text1)
            text3 = re.sub('\s{2,}', ' ', text2)
            value= text3.split(" ") #get the value from each line

            while "" in value: #remove empty lines
                value.remove("")

            for i in range(0, len(value)):
                # Feature 1:  the value of the token
                train_fea_dic['Value'].append(value[i])

                # Feature 2:  is token all uppercase?
                if(value[i].isupper):
                    train_fea_dic['UpperCase'].append(1)
                else:
                    train_fea_dic['UpperCase'].append(0)

                # Feature 3:  does token start with capital?
                if(value[i].istitle()):
                    train_fea_dic['Capital'].append(1)
                else:
                    train_fea_dic['Capital'].append(0)

                # Feature 4:  length of token
                train_fea_dic['Length'].append(len(value[i]))

                # Feature 5:  does the token consist only of numbers?
                if(value[i].isnumeric()):
                    train_fea_dic['Numeric'].append(1)
                else:
                    train_fea_dic['Numeric'].append(0)

                # Feature 6:  does token start with vowel?
                if value[i] in 'aeiou':
                    train_fea_dic['Vowel'].append(1)
                else:
                    train_fea_dic['Vowel'].append(0)

                # Feature 7:  length of left word < 4
                if(len(value[i])<4):
                    train_fea_dic['Length<Four'].append(1)
                else:
                    train_fea_dic['Length<Four'].append(0)

                # Feature 8:  if Right word contains "."
                if('.' in value[i]):
                    train_fea_dic['Period_in_token'].append(1)
                else:
                    train_fea_dic['Period_in_token'].append(0)




            if text[1].startswith("<class"):
                mid_content = middle_content("<class", ">", line)
                content = mid_content.split("\n")

                for num in range(0,len(content)):
                    if("id=" in content[num]):
                        ID_value = content[num].split("=")[1]
                    if("name=" in content[num]):
                        name_list = content[num].split("=")[1].split(" ")


            for i in range(0, len(value)):
                for j in range(0,len(name_list)):
                    if(value[i] == name_list[j]):
                        if(j==0):
                            value[i]=name_list[j]+("/B")
                        elif(j>0):
                            value[i]=name_list[j]+("/I")
                if(value[i] == ID_value):
                    value[i]=value[i]+("/B")

                else:
                    value[i]=value[i]+("/O")
                train_fea_dic['IOB'].append(value[i])


    train_data = pandas.DataFrame.from_dict(train_fea_dic)
    train_data['Value'] = train_data.index
    train_data['IOB'] = train_data['IOB'].map(IOB_to_Num)

    return train_data


#   4)  Process Test File to get test data
```

```
def Process_TestFile(testfile):
    testList = open(testfile,'r').read() #read the file and remove the empty lines.
    name_list = []
    test_fea_dic = defaultdict(list) # use default dictionary as test file feature
    dictionary

    for line in testList.split("\n\n"):

        if("<class" in line):
            testtext = line.split("\n")
            testtext1 = testtext[0]
            testtext2 = re.sub('([.,!?()])', r' \1 ', testtext1)
            testtext3 = re.sub('\s{2,}', ' ', testtext2)
            testvalue= testtext3.split(" ") #get the value from each line


            while "" in testvalue: #remove empty lines
                testvalue.remove("")

            for i in range(0, len(testvalue)):
                # Feature 1:  the value of the token
                test_fea_dic['Value'].append(testvalue[i])

                # Feature 2:  is token all uppercase?
                if(testvalue[i].isupper):
                    test_fea_dic['UpperCase'].append(1)
                else:
                    test_fea_dic['UpperCase'].append(0)

                # Feature 3:  does token start with capital?
                if(testvalue[i].istitle()):
                    test_fea_dic['Capital'].append(1)
                else:
                    test_fea_dic['Capital'].append(0)

                # Feature 4:  length of token
                test_fea_dic['Length'].append(len(testvalue[i]))

                # Feature 5:  does the token consist only of numbers?
                if(testvalue[i].isnumeric()):
                    test_fea_dic['Numeric'].append(1)
                else:
                    test_fea_dic['Numeric'].append(0)

                # Feature 6:  does token start with vowel?
                if testvalue[i] in 'aeiou':
                    test_fea_dic['Vowel'].append(1)
                else:
                    test_fea_dic['Vowel'].append(0)


                # Feature 7:  length of left word < 4
                if(len(testvalue[i])<4):
                    test_fea_dic['Length<Four'].append(1)
                else:
                    test_fea_dic['Length<Four'].append(0)


                # Feature 8:  if Right word contains "."
                if('.' in testvalue[i]):
                    test_fea_dic['Period_in_token'].append(1)
                else:
                    test_fea_dic['Period_in_token'].append(0)


            if testtext[1].startswith("<class"):
                mid_content = middle_content("<class", ">", line)
                content =mid_content.split("\n")


                for num in range(0,len(content)):
                    if("name=" in content[num]):
```

```python
                        name_list = content[num].split("=")[1].split(" ")

                    if("id=" in content[num]):
                        ID_value = content[num].split("=")[1]

            for i in range(0, len(testvalue)):
                for j in range(0,len(name_list)):
                    if(testvalue[i] == name_list[j]):
                        if(j==0):
                            testvalue[i]=name_list[j]+("/B")
                        elif(j>0):
                            testvalue[i]=name_list[j]+("/I")
                if(testvalue[i] == ID_value):
                    testvalue[i]=testvalue[i]+("/O")

                else:
                    testvalue[i]=testvalue[i]+("/O")
                test_fea_dic['IOB'].append(testvalue[i])


    test_data = pandas.DataFrame.from_dict(test_fea_dic)
    test_data['Value'] = test_data.index
    test_data['IOB'] = test_data['IOB'].map(IOB_to_Num )

    return test_data,test_fea_dic

#   5)  Calculate The Accuracy and Generate Output
def accuracy(train_data, test_data,test_fea_dic,output):

    features = ['Value','UpperCase','Capital','Length','Numeric','Vowel',
    'Length<Four','Period_in_token']
    X = train_data[features]
    Y = train_data['IOB']
    test_X = test_data[features]
    test_Y = test_data['IOB']
    X_train,  X_test, Y_train, Y_test = train_test_split(X, Y)

    # Create Decision Tree classifer object and get the predict value
    predict = DecisionTreeClassifier().fit(X_train,Y_train).predict(test_X)

    # Calculate and print the accuracy score
    accuracy = accuracy_score(test_Y,predict)
    print("Accuracy:",accuracy)


    for i in range(0,len(predict)):
        output.write('\n')
        if(predict[i] == 0):
            output.write(str(test_fea_dic['Value'][i]))
            output.write('/O')
        elif(predict[i] == 1):
            output.write(str(test_fea_dic['Value'][i]))
            output.write('/I')
        else:
            output.write(str(test_fea_dic['Value'][i]))
            output.write('/B')

    output.close()
    return accuracy




if __name__ == "__main__":

    # 1) load file
    trainfile = sys.argv[1]
    testfile = sys.argv[2]

    # 2) create outputfile
    outputName = testfile + ".out"
    output = open(outputName, "w")

    # 2)  Process Train File to get train data
    train_data= Process_TrainFile(trainfile)

    # 3)  Process Test File to get test data
    test_data,test_fea_dic = Process_TestFile(testfile)

    # 4) generate output and calculate accurancy
    accuracy(train_data, test_data,test_fea_dic, output)

```