Excercise - Working With CSV

May 2, 2019

1 Excercise - Working With CSV

1.1 Using the CSV module

1. A CSV file is often used exchange format for spreadsheets and databases.
2. Each line is called a record and each field within a record is seperated by a delimiter such as comma, tab etc.
3. We use the module "CSV" which is not included in the standard library of Python.

Note: Keep in mind that Mac uses a different delimiter to determine the end of a row in a CSV file than Microsoft. Since the CSV python module we will use works well with Windows CSV files, we will save and use a Windows CSV file in our program. So in MAC, you have to save the CSV file as "windows csv" file rather than just csv file.

Let us write a program to read a CSV file (word_sentiment.csv). This file contains a list of 2000 + words and its sentiment ranging form -5 to +5. Write a function "word_sentiment" which checks if the entered word is found in the sentiment_csv file and returns the corresponding sentiment. If the word is not found it returns 0.

1.1.1 Step 1:Import the module CSV.

Note: If any module is not included in the computer, we will need to do "pip install csv" in the terminal (in case of mac) or in the command prompt (in case of windows).

```
In [1]: import csv
```

1.1.2 Step 2: Assign the path of the file to a global variable "SENTIMENT_CSV"

```
In [2]: SENTIMENT_CSV = "C:\\Users\kmpoo\Dropbox\HEC\Teaching\Python for PhD May 2019\python4pl
```

1.1.3 Step 3: Open the file using the "with open()" command and read the file

Before we read a file, we need to open it. The "with open()" command is very handy since it can open the file and give you a handler with which you can read the file. One of the benefits of the "with"command is that (unlike the simple open() command) it can automaticaly close the file, allowing write operations to be completed. The syntax is :

***with open('filename', 'mode', 'encoding') as* fileobj* ***

Where ***fileobj*** is the file object returned by open(); ***filename*** is the string name of the file. ***mode*** indicates what you want to do with the file and ***ecoding*** defines the type of encoding with which you want to open the file.

Mode could be: * w -> write. if the file exists it is overwritten * r -> read * a -> append. Write at the end of the file * x - > write. Only if the file does not exist. It does not allow a file to be re-written

For each, adding a subfix 't' refers to read/write as text and the subfix 'b' refers to read/write as bytes.

Encoding could be: * 'ascii' * 'utf-8' * 'latin-1' * 'cp-1252' * 'unicode-escape'

After opening the file, we call the **csv.reader()** function to read the data. It assigns a data structure (similar to a multidimentional list) which we can use to read any cell in the csv file.

```
In [ ]: with open(SENTIMENT_CSV, 'rt',encoding = 'utf-8') as senti_data:
            sentiment = csv.reader(senti_data)
            for data_row in sentiment:
                print(data_row)
```

1.1.4 The full code

Let us package all of this into a nice function which - reads the word_sentiment.csv file - searches for a particualr given word - returns the sentiment value of the word given to it. If the word is not found it returns 0 .

```
In [5]: import csv
        SENTIMENT_CSV = "C:/Users/kmpoo/Dropbox/HEC/Teaching/Python for PhD Mar 2018/python4pho
        '''Updated the path to point to your file. The path provided changes based on your ope

        def word_sentiment(word):
            '''This function uses the word_sentiment.csv file to find the sentiment of the wor
            entered'''
            with open(SENTIMENT_CSV, 'rt',encoding = 'utf-8') as senti_data:
                sentiment = csv.reader(senti_data)

                for data_row in sentiment:
                    if data_row[0] == word:
                        sentiment_val = data_row[1]
                        return sentiment_val

                return 0

        def main():
            word_in = input("enter the word: ").lower()
            return_val = word_sentiment(word_in)
            print("the sentiment of the word ",word_in ," is: ",return_val)
        main()
```

```
enter the word: good
the sentiment of the word  good  is:  3
```

Now let us update this code so that we ask the user to enter a sentence. We then break the sentence into words and find the sentiment of each word. We then aggregate the sentiments across all the words to calcuate the sentiment of the sentence and tell if the sentence entered is positive or negative. Hint: Use the split() command we saw in lesson 1.

```
In [6]: import csv
        SENTIMENT_CSV = "C:/Users/kmpoo/Dropbox/HEC/Teaching/Python for PhD Mar 2018/python4pho
        '''The path provided changes based on your operating system. For a windows system the
        of the path will be "C:/Users/User/Desktop/word_sentiment.csv" '''

        def word_sentiment(word):
            """This function uses the word_sentiment.csv file to find the sentiment of the wor
            entered"""
            with open(SENTIMENT_CSV, 'rt',encoding = 'utf-8') as senti_data:
                sentiment = csv.reader(senti_data)

                for data_row in sentiment:
                    if data_row[0] == word:
                        sentiment_val = data_row[1]
                        return sentiment_val

                return 0

        def main():
            """This function asks the user to input a sentence and tries to calculate the sent
            of the sentence"""
            sentiment = 0
            sentence_in = input("enter the sentence: ").lower()
            words_list = sentence_in.split()
            for word in words_list:
                sentiment = sentiment + int(word_sentiment(word))

            if sentiment > 0:
                print("The entered sentence has a positive sentiment")
            elif sentiment == 0:
                print("The entered sentence has a neutral sentiment")
            else:
                print("The entered sentence has a negative sentiment")

        main()
```

```
enter the sentence: Poonacha is bad
The entered sentence has a negative sentiment
```

Can you improve this code to handle double like "not" ? eg. "poonacha is not good" should return a negative sentiment rather than positive .

```
In [ ]: # Enter code here
```

Do you think we can build a rudimentary learning algorithm to imporve the corpus of sentiments ?