Final Project

May 8, 2016

1 Setting up global variables

```
In [ ]: !pip install beautifulsoup4
        from bs4 import BeautifulSoup
        import urllib
        import string
        import collections
        import requests
        import json

        debates= ['https://www.washingtonpost.com/news/the-fix/wp/2016/02/25/the-cnntelemundo-republica
                  'https://www.washingtonpost.com/news/the-fix/wp/2016/03/03/the-fox-news-gop-debate-tr
                  'https://www.washingtonpost.com/news/the-fix/wp/2016/03/10/the-cnn-miami-republican-d
```

2 Defining global functions.

```
In [ ]: def openFile(url):
            #opening files
            file0 = BeautifulSoup(urllib.urlopen(url).read(),"lxml")
            return file0

        def cleanFile(file1):
            #function extracts text lines from html
            textLines = []

            paragraph = file1.find_all("p")
            for lines in paragraph:
                textLines.append(lines.get_text())

            return textLines

        def lines2words(file1):
            #function takes a list of strings and returns it as a string of words
            return_string = ''

            for line in file1:
                return_string += line

            words = return_string.split()

            return words
```

```
def getCandidate(candidate,file2):
    #function gets cadidate lines
    i = 0
    candidateLines = []

    for line in file2:
        if candidate in line:
            candidateLines.append(line)
            i += 1
        elif candidate in file2[i] :
            candidateLines.append(line)
            i += 1
        else:
            i += 1

    return candidateLines

print openFile(debates[0])
```

3 Function to compile all keywords due to API time out limit

```
In [ ]: def keywordAnalysis (textFile):
    #Calculating how many passes to send the API through according to dictionaryt length.
        multiple= 0
        multiple= len(textFile)/50
        remainder= len(textFile)%50
        if remainder > 0:
            multiple += 1

        temp=[]


        url = "http://access.alchemyapi.com/calls/text/TextGetRankedKeywords"
        api_key = '1f62c149d1b092011db73353df8215f6a5d3eb9e'
        headers = {"Accept": "application/json"}

        i=0
        while i != multiple:
            text = textFile[i*50:i*50+50]

            parameters = {
                'outputMode': 'json',
                'apikey' : api_key,
                'maxRetrieve' : 200,
                'sentiment': 1,
                'text': text}

            resp = requests.post(url, params=parameters, headers=headers)
            temp.append(json.loads(resp.text))
            i += 1

            #printing out relevenaces
```

```
            #j=0
            #while j != len(data['keywords']):
                #print "Keyword: ", data['keywords'][j]['text'], ", Relevance: ", data['keywords'][
                #j+=1

        return temp
```

4 Final Function.

You choose a candidate name and provide a source of text. The function parses the data and returns back keywords according to relevace.

```
In [ ]: def getCandidateKeywords(candidateName,URL):

            File1 = openFile(URL)
            textFile = cleanFile(File1)

            candidateLines= getCandidate(candidateName, textFile)

            word_counts = collections.Counter(lines2words(candidateLines))

            #return keywordAnalysis(candidateLines)
            return word_counts

            #for word, count in word_counts.most_common():
                #print word, count


        '''debate1={}
        debate1['Trump']= getCandidateKeywords('TRUMP', debates[0])
        debate1['Cruz']= getCandidateKeywords('CRUZ', debates[0])
        debate1['Rubio']= getCandidateKeywords('RUBIO', debates[0])

        debate2={}
        debate2['Trump']= getCandidateKeywords('TRUMP', debates[1])
        debate2['Cruz']= getCandidateKeywords('CRUZ', debates[1])
        debate2['Rubio']= getCandidateKeywords('RUBIO', debates[1])

        debate3={}
        debate3['Trump']= getCandidateKeywords('TRUMP', debates[2])
        debate3['Cruz']= getCandidateKeywords('CRUZ:', debates[2])
        debate3['Rubio']= getCandidateKeywords('RUBIO', debates[2])'''

        #for line in debate1['Cruz']:
        #    i = 0
        #    while i != len(line['keywords']):
        #        print '\n' , "Keyword: ", line['keywords'][i]['text'], ", Relevance: ", line['keywords
        #        print "Sentiment: ", line['keywords'][i]['sentiment']
        #        i+=1

        #i=0
        #for line in info:
```

```
        #    print info[i], '\n'
        #    i += 1
```

5 Creating SQL Tables

```
In [ ]: import MySQLdb as mdb
        import sys

        con = mdb.connect(host = 'localhost', user = 'root', passwd = 'dwdstudent2015', charset='utf8',

        # Query to create a database
        db_name = 'Final_Project'
        create_db_query = "CREATE DATABASE IF NOT EXISTS {0} DEFAULT CHARACTER SET 'utf8'".format(db_nar

        # Create a database
        cursor = con.cursor()
        cursor.execute(create_db_query)
        cursor.close()
```

6 Creating Debate Sentiment Tables

```
In [ ]: cursor = con.cursor()
        table_name = 'Debate1'
        # Create a table
        # The {0} and {1} are placeholders for the parameters in the format(....) statement
        create_table_query = '''CREATE TABLE IF NOT EXISTS {0}.{1}
                                        (candidate varchar(250),
                                        keyword varchar(250),
                                        relevance varchar(250),
                                        sentiment varchar(250)
                                        )'''.format(db_name, table_name)

        cursor.execute(create_table_query)
        cursor.close()
```

7 Creating Debate Words Table

```
In [ ]: cursor = con.cursor()
        table_name = 'Debate3_Words'
        # Create a table
        # The {0} and {1} are placeholders for the parameters in the format(....) statement
        create_table_query = '''CREATE TABLE IF NOT EXISTS {0}.{1}
                                        (candidate varchar(250),
                                        word varchar(250))'''.format(db_name, table_name)

        cursor.execute(create_table_query)
        cursor.close()
```

8 Function to send sentiment analysis to SQL

```
In [ ]: def send2SQL (candidateName, DebateSent, DebateDic):
            cursor = con.cursor()
```

```
        query_template = 'INSERT INTO Final_Project.'+ DebateSent + '(candidate, keyword, relevance

        for line in DebateDic[candidateName]:

            i = 0
            while i != len(line['keywords']):
                candidate = candidateName
                keyword = line['keywords'][i]['text']
                relevance= line['keywords'][i]['relevance']
                sentiment= line['keywords'][i]['sentiment']['type']
                print "Inserting ", candidate, keyword

                query_parameters = (candidate, keyword, relevance, sentiment)
                cursor.execute(query_template, query_parameters)
                con.commit()
                i+=1


        cursor.close()
```

9 Function to send words to SQL

```
In [ ]: def send2SQLwords (candidate, DebateSent, word):
            cursor = con.cursor()

            query_template = 'INSERT INTO Final_Project.'+ DebateSent + '(candidate, word) VALUES (%s, %

            for line in word:

                candidate = candidate
                word = line
                print "Inserting ", candidate, word

                query_parameters = (candidate, word)
                cursor.execute(query_template, query_parameters)
                con.commit()


            cursor.close()

In [ ]: FILE=getCandidateKeywords('RUBIO',debates[2])

        send2SQLwords ('Rubio', 'Debate3_Words', FILE)

In [ ]: File1 = openFile(debates[2])
        textFile = cleanFile(File1)

        candidateLines= getCandidate('KASICH', textFile)
        print candidateLines

        #sending JSON
        #kasichFile=
```

10 Cleaning up string in Debates

```
In [ ]: %sql update Debate1_Words set word=  REPLACE(word, 'TRUMP', ' ')
        %sql update Debate1_Words set word=  REPLACE(word, 'CRUZ', ' ')
        %sql update Debate1_Words set word=  REPLACE(word, 'RUBIO', ' ')
        %sql update Debate1_Words set word=  REPLACE(word, 'KASICH', ' ')
```

11 Alchemy API wouldn't work for Cruz Debate 2

```
In [ ]: print len(kasichFile['keywords'])


        #for line in cruzFile['keywords']:
        #    print '\n' , "Keyword: ", line['text'], ", Relevance: ", line['relevance']
        #    print "Sentiment: ", line['sentiment']



        cursor = con.cursor()

        query_template = 'INSERT INTO Final_Project.Debate3(candidate, keyword, relevance, sentiment) VA

        for line in kasichFile['keywords']:

            candidate = 'Kasich'
            keyword = line['text']
            relevance= line['relevance']
            sentiment= line['sentiment']['type']
            print "Inserting ", candidate, keyword

            query_parameters = (candidate, keyword, relevance, sentiment)
            cursor.execute(query_template, query_parameters)
            con.commit()



        cursor.close()


In [ ]: %load_ext sql
        %sql mysql://root:dwdstudent2015@localhost:3306/Final_Project?charset=utf8

        %sql select distinct word, count(word), candidate from conWords group by candidate, word order b

In [ ]: a = %sql select candidate, word from Debate1_Words
        b= %sql select candidate, word from Debate2_Words
        c= %sql select candidate, word from Debate3_Words


        cursor = con.cursor()

        query_template = 'INSERT INTO Final_Project.conWords(candidate, word) VALUES (%s, %s)'

        for line in c:
```

```
        candidate = line[0]
        word = line[1]
        print "Inserting ", candidate, word
        query_parameters = (candidate, word)
        cursor.execute(query_template, query_parameters)
        con.commit()

cursor.close()
```