INF 553 - Spring 2016 Assignment 1

Overview of the Assignment

Access the Twitter Application Programming Interface(API) using python.

Estimate the public's perception (the *sentiment*) of the tweet's.

Calculate the TF-DF(term frequency-document frequency) of terms in a tweet.

Points to keep in mind:

This assignment is open-ended in several ways. You'll need to make some decisions about how best to solve the problem and implement them carefully.

It is perfectly acceptable to discuss your solution on the forum, but don't share code.

Each student must submit his/her own solution to the problem.

The code will be submitted to plagiarism detection software (Moss) to detect if students are submitting the same code.

You will submit your code, and the professor or grader will run your code against a test data set to see if your code produces the correct answer.

Your code should only use the Scala standard libraries unless you are specifically instructed otherwise. Your code should also not rely on any external libraries or web services.

What you will turn in:

A file called **lastname_firstname_assignment1.zip** that contains:

- A text file containing first 20 lines of downloaded twitter data:

  lastname_firstname_first20.txt

- Jar package to derive sentiment and calculate tf-df of term occurring in downloaded twitter data:

  lastname_firstname_tweets.jar

- The output from running your tweet sentiment program as:

  lastname_firstname_tweets_sentiment_first20.txt

- The output from running your tf-df program as:

  lastname_firstname_tweets_tfdf_first20.txt

The Twitter Application Programming Interface

Twitter provides a very rich REST API for querying the system, accessing data, and control your account. You can read more about the Twitter API (https://dev.twitter.com/docs)

Scala environment

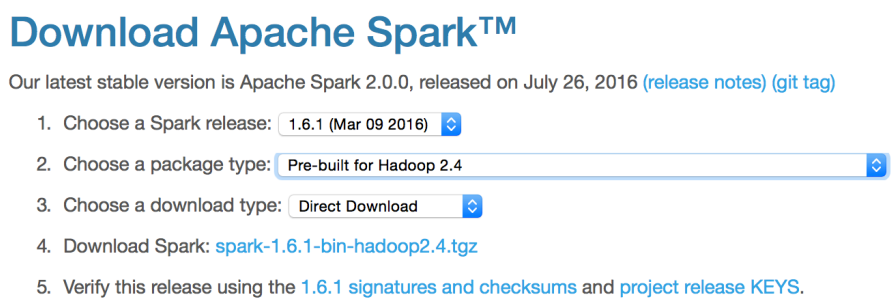
If you are new to Scala, you may find it valuable to work through a concise introduction to Scala. (https://www.cis.upenn.edu/~matuszek/Concise Guides/Concise Scala.html#maps)

You will need to establish a Scala programming environment to complete this assignment. First, you should download IntelliJ IDEA as compiler, which can be downloaded here https://www.jetbrains.com/idea/#chooseYourEdition. Second, you should install Scala plugin, the procedure is as follow: Open Preference -> Choose Plugins -> Click Install JetBrains plugin (at bottom) -> Search Scala and Install. Third, download SDK (Scala Development Kit), which can be downloaded here http://www.scala-lang.org/download/, version 2.10.2 is recommended. Forth, add SDK into external libraries.

Spark Installation

Spark can be downloaded from the official website http://spark.apache.org/downloads.html

Spark 1.6.1 combined with Hadoop 2.4 is recommended. The interface of Spark official website is shown in the following figure.



Packing Jar

The procedure is as follow: Open *File* -> Choose *Project Structure* -> Choose *Artifacts* -> Click + and Choose *JAR*-> Choose *from module with dependencies* -> enter main class name -> choose output file path -> click *apply* and *ok*

Running Command

The spark-submit is used to launch application, which is in Spark's bin director. Once you get the jar package, your application can be launched using the bin/spark-submit script. The following figure is an example.

```
Weiweis-MacBook-Pro:spark-1.6.1-bin-hadoop2.4 weiweiduan$ ./bin/spark-submit --c
lass test.TFDF --master local[1] /Users/weiweiduan/Desktop/test/out/artifacts/te
st_jar/test.jar "/Users/weiweiduan/Downloads/Assignment1_INF553/shortTwitter.txt
"
```

Some of the commonly used options are:
--class: The entry point for your application
--master: The master URL for the cluster. For our case, the master is your local machine, so you do not need to set master URL, but set the number of partitions as 1as what is shown in the above figure, so that the output is in one text file.
--application-jar: Path to a bundled jar including your application and all dependencies. The URL must be globally visible inside of your cluster, for instance, an hdfs:// path or a file:// path that is present on all nodes.
--application-arguments: Arguments passed to the main method of your main class, if any

Unicode strings

Strings in the twitter data prefixed with the letter "u" are unicode strings. For example:

u"This is a string"
Unicode is a standard for representing a much larger variety of characters beyond the roman alphabet (greek, russian, mathematical symbols, logograms from non-phonetic writing systems such as kanji, etc.)

In most circumstances, you will be able to use a unicode object just like a string.

If you encounter an error involving printing unicode, you can convert the string to UTF-8.

If you do not encode the strings as UTF-8, you will loose 30% score.

Get Twitter Data

You will be provided with **twitterstream.py** file which will help you to get live streaming twitter data. You just need to follow the instructions given below after putting in the required Keys.

The steps below will help you set up your twitter account to be able to access the live 1% stream.

To access the live stream, you will need to install the **oauth2 library** so you can properly authenticate. You can install it yourself in your Python environment. (The command $ pip install oauth2 should work for most environments, or try easy-install oauth2)

Create a twitter account if you do not already have one. Respond to the resulting email from Twitter to Confirm your Twitter Account.

Go to **https://dev.twitter.com/apps** and log in with your twitter credentials.

Click "**Create New App**"

Fill out the form and agree to the terms. Put in a dummy website if you don't have one you want to use.

On the next page, click the "**Keys & Access Tokens**" tab along the top, then scroll all the way down until you see the section "Your Access Token"

Click the button "**Create My Access Token**". You can **Read more about Oauth authorization.** (https://pypi.python.org/pypi/oauth2/)

You will now copy four values into **twitterstream.py**.

"**API Key(Consumer Key)**"

"**API secret(Consumer Secret)**"

"**Access token**"

"**Access token secret**".

Open twitterstream.py and you will see the code like below. Replace the corresponding values in the code.

```
api_key = "<Enter api key>"
api_secret = "<Enter api secret>"
access_token_key = "<Enter your access token key here>"
access_token_secret = "<Enter your access token secret here>"
```

Run the following and make sure you see data flowing and that no errors occur.

python twitterstream.py > output.txt
This command pipes the output to a file. Stop the program with Ctrl-C after running it for **1 minute** for data to accumulate. Keep the file output.txt for the duration of the assignment. Don't use someone else's file; we will check for uniqueness in other parts of the assignment.

What to turn in:

lastname_firstname_first20.txt containing the first 20 lines of the twitter data you downloaded from the web. Use the following command for the same.

head -n 20 output.txt > lastname_firstname_first20.txt

Submission Details

The jar package you submit should include the solutions to two problems.

Your codes will be tested by different tweets in the same format.

Finally, to facilitate our grading of your submission, we require that:

1. **Your output follow exactly the format as given in the solution files.**
2. **Add your first name and last name to the beginning of your jar, e.g.,john_smith_tweeets.jar.**
3. **Class name for the first problem is Sentiment, and for the second problem is TFDF**

Problem 1: Derive Tweet Sentiment Score (50 points)

For the first problem, the running command should have two parameters. The first parameter should be the first 20 tweets you collected. The second parameter should be **AFINN-111.txt.**
In **AFINN-111.txt**, contains a list of pre-computed sentiment scores. Each line in the file contains a word or phrase followed by a tab separated sentiment score.

Your Task:

We will run your code using the below given command:

```
./bin/spark-submit --class lastname_firstname_tweets.Sentiment --master local[
1] lastname_firstname_tweets.jar "lastname_firstname_first20.txt" "AFINN-111.
txt"
```

For the twitter data, you can also read the Twitter documentation (https://dev.twitter.com/docs/platform-objects/tweets) to understand what information each tweet contains and how to access it, but it's not too difficult to deduce the structure by direct inspection.

Now that the 'text' for each tweet is available, use that to calculate the sentiment score of each tweet. For that you have to lookup each term of the tweet in the AFINN-111.txt dictionary and add up their respective sentiment scores.

Keep in mind that all the terms in AFINN-111.txt are lower case. You will have to use the **string.toLowerCase()** function to convert your tweet to all lower case for this problem and convert to UTF-8.
Some tweets contain **URL's** (eg: https:\/\/t.co\/8wl9wrT), **hashtags**(starting with #,@) and prefix like "**RT@ManUtd:",** which should be ignored while calculating the sentiment score. Also the extra terms found in the tweets but not in AFINN-111.txt and should be given a **sentiment score of 0**.

The partition number should be 1. So you will get only one output file in the output folder and rename the file as **lastname_firstname_tweets_sentiment_first20.txt**

You can test your solution to this problem using the data file **shortTwitter.txt**:

You can verify your solution against **tweets_sentiment_shortTwitter_solution.txt**.

The format of output file, each line is a tuple, and it should be listed in ascending order. The sample of the output file is shown as following figure. If the format of your output file is not correct, you will loose 30% score.

```
(1,2)
(2,0)
(3,8)
(4,4)
(5,-1)
(6,0)
```

Skeleton of the codes:

```
object Sentiment {
  def func_name(para: type) : return_type{ //define your function here
    function body
  }
  def main(args: Array[String]) {
    for(line <- Source.fromFile(args(1)).getLines()){ // read the sentimental
                                                      // scores from AFINN-111.txt
     ...
    }
    val sparkConf = new SparkConf().setAppName("app_name")
    val sc = new SparkContext(sparkConf) //create SparkContext object
    val text = sc.textFile(args(0)) // create RDD
    val count = text.tranformation_operator_i(func_name_i)
                    .tranformation_operator_j(func_name_j) // tranformation
                    //operator, the function can be anonymous function
    counts.saveAsTextFile("lastname_firstname_tweets_sentiment_first20")
                    // action operator
  }
}
```

Problem 2:TF-DF of each term(term frequency – document frequency) (50 points)

Background: Given a document, ***tf*** (term frequency) of a term in the document is the number of occurrences of the term in the document. For example, consider the following short document D:

> *In a swanky hotel ballroom, he told the crowd of alumni and donors that Texas had become the largest feeder of **students** to USC after California, and that **students** from their state scored significantly higher on the SAT than the average of all applicants.*

The term "students" occurs twice in the documents, so its tf is 2 for this document.

Moreover, given a collection of documents, ***df*** (document frequency) of a term refers to the number of documents in the collection where the term occurs. For example, suppose that there are 10 documents in the collection and that "students" occur in three of them, e.g., one of which may be the document D above. Then the document frequency of "students" is 3.

tf and df of a term indicate the importance of the term in the document and across the documents. They are important statistics for effective information retrieval. (Note that df can be used to compute inverse document frequency or IDF)

Your Task:

We will run your code using the below given command:

```
./bin/spark-submit --class lastname_firstname_tweets.TFDF --master local[1]
lastname_firstname_tweets.jar "lastname_firstname_first20.txt"
```

The above given command has one argument:
lastname_firstname_first20.txt: The tweet data file created earlier.

You will have to use the **string.toLowerCase()** function to convert your tweet to all lower case for this problem and convert to UTF-8.

Some tweets contain **URL's** (eg: https:\/\/t.co\/8wl9wrT), **hashtags**(starting with #,@) and prefix like "**RT@ManUtd:",** which should be ignored while calculating the tf-df score.

You can test your solution to this problem using the data file **shortTwitter.txt**:

You can verify your solution against **tweets_sentiment_shortTwitter_solution.txt**.

The partition number should be 1. So you will get only one output file in the output folder and rename the file as **lastname_firstname_tweets_tfdf_first20.txt**

The format of output file, each line of your output is a 3-tuple. The first element is the word, the second element is its df, the third element is a list of tuple, each tuple has two elements. The first element is the order number of the tweet, the second element is the number of the word occurring in the tweet. The output file should be listed in alphabetic order and the last element of each line should listed in ascending order. The sample of output file is shown as following figure. If the format of your output file is not correct, you will loose 30% score.

```
(76ers,1,List((5,1)))
(a,5,List((1,1), (2,3), (3,1), (4,2), (6,1)))
(all,1,List((3,1)))
(and,1,List((6,1)))
(any,1,List((1,1)))
(as,1,List((1,1)))
(at,1,List((4,1)))
(bake,1,List((6,1)))
(battle,1,List((5,1)))
```

Skeleton of the codes:

```
object TFDF {
  def func_name(para: type) : return_type{ //define your function here
    function body
  }
  def main(args: Array[String]) {
    val sparkConf = new SparkConf().setAppName("app_name")
    val sc = new SparkContext(sparkConf) //create SparkContext object
    val text = sc.textFile(args(0)) // create RDD
    val count = text.tranformation_operator_i(func_name_i)
                .tranformation_operator_j(func_name_j) // tranformation
                //operator, the function can be anonymous function
    counts.saveAsTextFile("lastname_firstname_tweets_tfdf_first20")
                // action operator
  }
}
```