Assignment 2

MET CS 777 - Big Data Analytics
kNN classifier
(20 points)

GitHub Classroom Invitation Link
`https://classroom.github.com/a/1NbdTBBy`

1 Description

In this assignment you will implement a k-nearest neighbors classifier (kNN classifier) in multiple steps. KNN is an algorithm that can find the top-k nearest objects to a specific query object.

In this assignment we want to use kNN to classify text documents. Given a specific search text like ”How many goals Vancouver score last year?” the algorithm can search in the document corpus and find the top K similar documents.

We will create out of a text corpus a TF-IDF (Term Frequency - Inverse Document Frequency) Matrix for the top 20k words of the corpus. The TF-IDF matrix will be used to compute similarity distances between the given query text and each of the documents in the corpus.

2 Wikipedia Data Set

You will use the Wikipedia data set. We have downloaded a copy of the whole Wikipedia data set from (`https://dumps.wikimedia.org`) and stored it a large file.

Each Wikipedia Page is a document and have a unique document ID and a specific URL.

For example

- **docID** 418348
- **URL** `https://en.wikipedia.org/wiki?curid=418348`

3 Obtaining the Dataset

3.1 Small Data Set - Wikipedia Pages

You can find a small data set (Only 1000 Wikipedia pages) data set here:
`https://s3.amazonaws.com/metcs777/WikipediaPagesOneDocPerLine1000LinesSmall.txt.bz2`

Google Storage:
`gs://metcs777/WikipediaPagesOneDocPerLine1000LinesSmall.txt`

AWS S3:
`s3://metcs777/WikipediaPagesOneDocPerLine1000LinesSmall.txt`
Small Data Set - Wikipedia Page Categories
Wikipedia Page Categories (for the above small data set)
`https://metcs777.s3.amazonaws.com/wiki-categorylinks-small.csv.bz2`

3.2 Large Data Set

Categories of the whole wikipedia
Google Storage:
`gs://metcs777/wiki-categorylinks.csv.bz2`
AWS S3:
`s3://metcs777/wiki-categorylinks.csv.bz2`
1 million pages - 2.2 GB, S3 URL:
Google Storage:
`gs://metcs777/WikipediaPagesOneDocPerLine1m.txt`
AWS S3:
`s3://metcs777/WikipediaPagesOneDocPerLine1m.txt`

Run your final implementation on the 1 million Wikipedia pages data set on AWS.

Data Description:
Each line is a single document in a pseudo XML format.

```
<doc id="431953" url="https://en.wikipedia.org/wiki?curid=431953"
 title="Doug Harvey (ice hockey)">Doug Harvey (ice hockey)Douglas Norman Harvey
     ...
</doc>
```

4 Assignment Tasks

Task 1 - Generate the Top 20K dictionary and Create the TF-IDF Array (4 Points)

Get the top 20,000 words in a local array and sort them based on the frequency of words. The top 20K most frequent words in the corpus is our dictionary and we want to go over each document and check if its words appear in the Top 20K words.

At the end, produce an RDD that includes the docID as key and a numpy array for the position of each words in the top 20K dictionary.

(docID, [dictionaryPos1, dictionaryPos2, dictionaryPos3...])

After having the top 20K words we want to create a large array that its columns are the word list in order and rows are documents.

The first step is to create the term frequency $TF(x, w)$ vector for each document

$$TF(x, w) = \frac{x[w]}{\sum_{w'} x[w']}$$

Inverse Document Frequency is:

$$IDF(w) = log\left(\frac{size\ of\ corpuse}{\sum_{"x\ in\ corpus"} 1\ if\ (x[w]{\geq}1),0\ otherwise}\right)$$

$$TF - IDF(w) = TF(x, w) \times IDF(w)$$

For more description about TF-IDF see the Wikipedia page:
`https://en.wikipedia.org/wiki/Tf-idf`

- In your code print out print(allDocsAsNumpyArrays.take(3))
- In your code print out print(allDocsAsNumpyArraysTFidf.take(2))

Task 2 - Implement the getPrediction function (6 Points)

Finally, you should implement the function getPrediction (textInput, k). This function will predict the membership of this text string in one of the 20 different newsgroups.

You should use the cosine similarity to calculate the distances.

$$Cos(x_1, x_2) = \sum_{i=1}^{d}(x_1 \times x_2)$$

Print out the results for the following queries:

```
print(getPrediction('Sport Basketball Volleyball Soccer', 10))

print(getPrediction('What is the capital city of Australia?', 10))

print(getPrediction('How many goals did Vancouver score last year?', 10))
```

Print the results on the output console, or alternatively store it on the cloud storage.

Task 3 - Using Dataframes (8 points)

Task 3.1 (2 point)

Use Spark Dataframe to provide summery statistics (max, average, median, std) about the number of wikipedia categories that are used for wikipedia pages. Print the results on the output console, or store it on the cloud storage.

Task 3.2. (3 points)

Use Spark Dataframe to find the top 10 mostly used wikipedia categories. Print the results on the output console, or store it on the cloud storage.

Task 3.3. (3 points)

Use Spark Dataframe to find out the wikipedia pages that are using the above top 10 most (from above question) and have top 10 largest number of categories?

This means those pages from all pages that are using the 10 frequent cats, and have the top-10 number of categories. Print the results on the output console, or store it on the cloud storage.

Task 4 - Removing Stop Words, do Stemming and redo the task 3 (2 points)

Task 4.1 - Remove Stop Words (1 point) Describe if removing the English Stop words (most common words like ”a, the, is, are, i, you, ...”) would change the final kNN results here.

You do not need to implement this task

Would your result be changed heavily after removing the stop words? Why? Provide reasons.

Task 4.2 - Do English word stemming (1 point) We can stem the words [”game”,”gaming”,”gamed”,”games”] to their root word ”game”. Read more about stemming here `https://en.wikipedia.org/wiki/Stemming`

Would stemming change your result heavily? Why? Provide reasons and describe it.

You do not need to implement this task, ony discuss your expected outcome results

5 Important Considerations

5.1 Machines to Use

One thing to be aware of is that you can choose virtually any configuration for your Cloud Cluster (Amazon or Google Cloud) - you can choose different numbers of machines, and different configurations of those machines. And each is going to cost you differently!

Pricing information is available at: `http://aws.amazon.com/elasticmapreduce/pricing/`

Since this is real money, it makes sense to develop your code and run your jobs locally, on your laptop, using the small data set. Once things are working, you’ll then move to Amazon EMR.

We are going to ask you to run your Spark jobs over the “real” data using 3 machines with **4 cores and 8GB RAM each** as workers. This provides 4 cores per machine (16 cores total) so it is quite a bit of horsepower. On the Google cloud take 3 machines with 4 cores and 8 GB of memory.

As you can see on EC2 Price list , this costs around 50 cents per hour. That is not much, but **IT WILL ADD UP QUICKLY IF YOU FORGET TO SHUT OFF YOUR MACHINES**. Be very careful, and stop your machine as soon as you are done working. You can always come back and start your machine or create a new one easily when you begin your work again. Another thing to be aware of is that Amazon charges you when you move data around. To avoid such charges, do everything in the ”N. Virginia” region. That’s where data is, and that’s where you should put your data and machines.

- You should document your code very well and as much as possible.
- You code should be compilable on a unix-based operating system like Linux or MacOS.

5.2 Academic Misconduct Regarding Programming

In a programming class like our class, there is sometimes a very fine line between ”cheating” and acceptable and beneficial interaction between peers. Thus, it is very important that you fully understand what is and what is not allowed in terms of collaboration with your classmates. We want to be 100% precise, so that there can be no confusion.

The rule on collaboration and communication with your classmates is very simple: you cannot transmit or receive code from or to anyone in the class in any way—visually (by showing someone your code), electronically (by emailing, posting, or otherwise sending someone your code), verbally (by reading code to someone) or in any other way we have not yet imagined. Any other collaboration is acceptable.

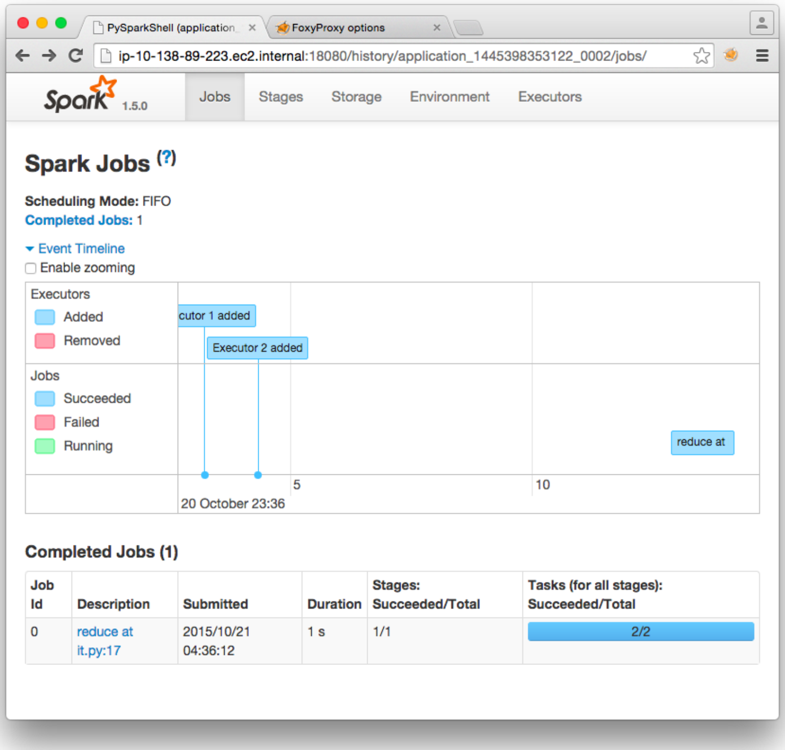
The rule on collaboration and communication with people who are not your classmates (or your TAs or instructor) is also very simple: it is not allowed in any way, period. This disallows (for example) posting any questions of any nature to programming forums such as **StackOverflow**. As far as going to the web and using Google, we will apply the **”two line rule”**. Go to any web page you like and do any search that you like. But you cannot take more than two lines of code from an external resource and actually include it in your assignment in any form. Note that changing variable names or otherwise transforming or obfuscating code you found on the web does not render the ”two line rule” inapplicable. It is still a violation to obtain more than two lines of code from an external resource and turn it in, whatever you do to those two lines after you first obtain them.

Furthermore, you should cite your sources. Add a comment to your code that includes the URL(s) that you consulted when constructing your solution. This turns out to be very helpful when you’re looking at something you wrote a while ago and you need to remind yourself what you were thinking.

5.3 Turnin

Create a single document that has results for all three tasks.

Also for each task, for each Spark job you ran, include a screen shot of the Spark History.


PySparkShell (application
FoxyProxy options
ip-10-138-89-223.ec2.internal:18080/history/application_1445398353122_0002/jobs/
Spark 1.5.0
Jobs
Stages
Storage
Environment
Executors

Spark Jobs (?)

Scheduling Mode: FIFO
Completed Jobs: 1

Event Timeline
□ Enable zooming

Completed Jobs (1)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	reduce at it.py:17	2015/10/21 04:36:12	1 s	1/1	2/2

Figure 1: Screenshot of Spark History

Please zip up all of your code and your document (use .zip only, please!), or else attach each piece of code as well as your document to your submission individually.

Please have the latest version of your code on the GitHub. Zip the files from GitHub and submit as your latest version of assignment work to the Blackboard. We will consider the latest version on the Blackboard but it should exactly match your code on the GitHub