Assignment A

Analysis of Twitter datasets using MapReduce

<u>Weightage</u>: 150 points (15%), with extra credit of 35 points

<u>Posted date:</u> Wed 25 Jan, 2017

<u>Due date</u>: Sun 12 Feb, 2017 by 11:59PM IST

Introduction

In this assignment, you will learn to apply MapReduce and Hadoop for analyzing tweets collected during a 2 month period, from Oct 17 to Dec 16, 2016. This dataset is about 1TB in size and is a 1% sample of all tweets in that period posted on Twitter using Twitters API, and the dataset is in JSON[1]. Review Twitter's JSON object description. While some fields have been extracted for you as JSON attributes, you will need to parse the tweet to get others. E.g. you will need to parse to extract the hashtag: any sequence of letters that begin with a '#' and whose last letter is succeeded with a whitespace character (space, tab, newline). You may reuse any open-source parsing libraries for the Twitter objects, with clear attribution in your report and code.

For each of these tasks below, write a ***MapReduce job in Java*** with the given filename. This should contains the core logic classes – Map, Reduce, Combiner, Partitioner, for the application. Other driver code, etc. should be separated out in a different file, **`Driver.java`**. You will be submitting these source files, along with the location of the log files and output files from their runs present in HDFS. You must develop the idea for and write all the core MapReduce logic code by yourself, without external help. The goal is to help you think in terms of MapReduce and develop algorithms.

Also, you will submit a ***report*** that contains a brief intuition of your Map/Reduce/Combiner/Partitioner logic, plots of the results, and a brief discussion of the observations. Document your code. Unless otherwise stated, you should run your jobs and provide results using at least 10% of the entire dataset – more the better, but given the shared resource, make sure to try on smaller before you use larger data. Clearly mention the dataset range you used for each task. Equal weightage is given to the source code and the report.

1. Simple Statistics on Tweets *[Solve any 3 out of 4, 3x15 points]*

i. Frequency distribution of total tweets by user. X Axis has buckets with number of tweets by a user (choose an appropriate static bucket size), Y axis has frequency of users who have tweeted as many times as the X axis, in the whole dataset. (**`FreqUserTweet.java`**)

ii. Frequency distribution of total hashtags by tweet. X Axis has buckets with number of tweets for a hashtag (choose an appropriate static bucket size), Y axis has frequency of hashtags that were tweeted as many times as the X Axis, in the whole dataset. (**`FreqTagTweet.java`**)

[1] https://dev.twitter.com/overview/api/

iii. Frequency distribution of total tweets by language. X Axis has buckets with the language ID (use "lang" property in JSON), Y axis has frequency of tweets in that language, in the whole dataset. (**`FreqLangTweet.java`**)

iv. Which pairs of hastags co-occur the most together? Identify and plot the frequency of the top 100 most frequently co-occuring hashtags in the entire dataset. (**`TopCooccurrence.java`**)

2. Temporal Statistics on Tweets *[Solve any 2 out of 3, 2x15 points]*

Unless noted otherwise, use the universal coordinated time (UTC) in all these temporal statistics. A week ranges from Sunday 00:00 - Saturday 23:59:59, UTC.

i. What are the top 10 trending hashtags that were popular (most frequent) in each whole week of the dataset? (**`TimeTop10Tags.java`**)

ii. What were the top 3 tweets that were retweeted the most in each whole week of the dataset? (**`TimeTopTweets.java`**)

iii. For each "time_zone", how does the number of tweets per hour change in a 24 hour cycle? Is there any trend you observe for weekdays and weekends? (**`TimeTweetFreq.java`**)

3. Network Statistics on Tweets *[Solve any 1 out of 2, 15 points]*

i. Frequency distribution of total followers by user, and followees by user. X Axis has buckets with number of followers/followees per user (choose an appropriate static bucket size), Y axis has two bars with frequency of users who have as many followers/folowees as the X axis, in the whole dataset. (**`FreqUserFollow.java`**)

ii. How correlated are the number of followers and followees for a user? Show a bubble plot where X Axis has buckets with number of followers, Y Axis has buckets with number of followees, and you plot a bubble whose size corresponds to the number of users with that many followers and followees. (**`FreqUserFollowCorr.java`**)

4. Scalability *[30 points]*

Does Hadoop/MapReduce weakly scale? Demonstrate using 2 examples from the above solutions. Plot the data size/slot count in X Axis, time taken in Y axis. Discuss. (**`ScaleXYZ.java`**, where **`XYZ`** is the file name of the earlier task you are scaling)

5. Influence Propagation *[30 points]*

How quickly does an idea propagate on Twitter, and how quickly does a "hot" idea die away? Select 6 hashtags from among the 100 most frequent in the dataset you have. Examine the temporal pattern for their growth in popularity. What affects this growth -- Who tweets it? Which country/timezone it emerged in? The language they were tweeted in? Whether had an image/URL link or not? Refer to research literature where relevant. (**`Influence.java`**)

Extra Credit

6. Sentiment Analysis *[20 points]*

Can you use twitter to gauge the sentiment of the broader community towards a particular topic? Pick a topic that was "controversial" during the period of data collection, 17 Oct to 16 Dec, 2017, e.g. on politics (US elections), sports, movies, etc. Use a sentiment analysis library to see how the public sentiment changed during that period, i.e., how did the number of positive vs. negative sentiments change on a daily basis in those ~60 days? Did they represent the eventual outcome? (**`Sentiment.java`**)

References

http://stanfordnlp.github.io/CoreNLP/
https://blog.openshift.com/day-20-stanford-corenlp-performing-sentiment-analysis-of-twitter-using-java/
https://lizrush.gitbooks.io/algorithms-for-webdevs-ebook/content/chapters/sentiment-analysis.html
http://www.nltk.org/howto/sentiment.html
https://www.google.co.in/trends/hottrends

7. Similarity in User Interest *[15 points]*

Do frequently followed users have similar interests? For the 1000 most followed users in the dataset, find the top 10 pairs having similar interests using a cosine similarity distance function between users that is based on the all the #hashtags and @mentions present in all tweets of a given user in the dataset. (**`Similarity.java`**)

References

http://pulkitgoyal.in/similarity-metrics-twitter/
https://alexn.org/blog/2012/01/16/cosine-similarity-euclidean-distance.html s

Submission Instructions

For all tasks, you should submit the source files integrated with Maven's pom.xml that compiles without error, the log files generated for your job, the list of HDFS output file directories along with the MD5 checksum of its files.

i. Name your source folder as **`username-ds256-alpha/`**. Replace "username" you're your cluster account username. Make sure the root of the folder contains the **`pom.xml`**, and an **`output.csv`** CSV file with path of HDFS output files and their checksums, **`log/`** folder containing logs generated for the final output reported, and your assignment report **`username-ds56-alpha.pdf`**. Do not include jar or class files in this folder.

ii. The source files should only contain the Java files that are mentioned above in the tasks along with a **`Driver.java`** file. Any shared libraries of logic can be placed in **`Utils.java`**.

iii. Tar your folder into a single file with the filename as below.

```
tar cvf username-ds256-alpha-src.tar username-ds256-alpha/
```

iv. Calculate its MD5 checksum for the tarred tile

```
md5sum username-ds256-alpha-src.tar
```

v. Zip the tarred file with a *strong password*.

```
zip –e username-ds256-alpha-src.tar.zip
username-ds256-alpha-src.tar
```

Copy the encrypted file to **`~ds256/submission-alpha/`** folder in the head node and email the *password* and *MD5 checksum* to Jayanth by the deadline with the subject line "**`assignment-ds256-alpha username`**".

If you upload an unencrypted file to the folder, or you use a weak password, you will get 0 (zero) points.

Rules

- You are working in a shared cluster. So make sure that your source files are kept secure, and are NOT readable by any other student account. Disable group and world read and execute permissions on your home folder. Make sure the jar submitted to Hadoop does NOT contain source files. Do not store source files in /tmp, on HDFS or any other publicly readable locations. Any student who violates these rules will get an automatic 0 (zero) for that assignment.
- Do NOT look into the source code of others, even if others are in violation and source code is "lying around" in some folder. If it is not yours or a shared dataset, do not be curious. If you come across such violations, please bring it to the attention of the TA and Yogesh immediately.
- We will pass the submissions through *plagiarism checks*. If there are noticeable similarities between different submissions, you will get an automatic 0 (zero) for that assignment. Repeat offenders will get grade point reductions or failing grades.

Guidelines

- Given human readable names to your jobs and include your username in the job. This helps identify issues.
- Run your experiments on a subset of the data initially before you run it on the full dataset. Since it may take 10's of minutes or hours to analyze the whole data, limit your full dataset analytics (i.e. all ~4000 files) to 1 or 2 runs for the final results. Instead use, say 100 files, for testing and debugging.
- Watch your jobs for exceptions and errors. An incorrect map or reduce method can appear to run slowly while it generates a lot of exceptions in the background, overflowing the log files. Monitor the status using Yarn application status, and tail the log output file watching for exceptions.
- If you notice someone else's application is taking up a lot of resources, is behaving abnormally or causing the cluster to be unstable, bring it to the attention of the TA and Yogesh. Do NOT kill others' job.
- Do NOT submit a large number of MR jobs at the same time. While the batch system does schedule jobs, do not hog the cluster with many long running jobs. If we find students dominating the cluster, they will be warned initially, and if repeated, their jobs may be involuntarily terminated.
- Think about the number of resource slots you will be using, the number of Mapper and Reducer tasks, and the number of Map and Reduce key-value pairs you will be processing. You should have a reasonable estimate of the quantity of resources you need and the time required.