ReadMe

Note: To run the jars provided please type the appropriate commands:

hadoop jar wordcount.jar sample.WordCount

hadoop jar pairs.jar trends.PairsDriver

hadoop jar stripes.jar stripes.StripesDriver

hadoop jar kmeans.jar kmeanCluster.KmeansDriver

hadoop jar shortestpathlargegraph.jar shortestpath.ShortestPath

hadoop jar shortestpathsmallgraph.jar shortestpath.ShortestPath

hadoop jar shortestpathwithdistance.jar shortestpath.ShortestPath

Part 1:

Phase 1: Cleaning the data

1) Import the file ' **CleanData.zip** ' from ' **part 1** ' folder.
2) Open the source code of the file and change the input and output folders in the code as necessary.
3) The code cam run on multiple files located in a folder, so only adjust the folder name as needed.
4) After the code is run, you will get the output in the specified.

Phase 2: Running Word Count on the output file

1) Upload output file from Phase 1, on to the hdfs into a folder named ' **/inputforfirstpart** '.
2) Import the ' **WordCount.zip** ' from ' **part 1** ' folder
3) Create a jar at any specified location of you could just use the jar that has been provided in the uploaded zip.
4) Run the jar, you will get the output of this operation in ' **/output** ' folder on the hdfs.
5) The class name for wordcount is **sample.WordCount**

Phase 3: Parsing the data into four files containing Hashtags, Words, Trends and Users

1) Download the output from the previous Phase namely ' **/output/part-r-00000** ' into any folder.
2) Import ' **Parser.zip** ' from ' **part 1** ' folder

3) Edit the source code of the file to change the input and the output folder and run the code.
4) You will get 4 files namely: **HashTags.txt** , **MostOccuringWords.txt** , **trends.txt**, **Users.txt**

Phase 4: Sorting the out of previous phase in descending order

1) Import ' **SortDocument.zip** ' from ' **part 1** ' folder
2) Change the input and output folders as needed.
3) Copy the data from previous phase in the input folder and run the code.
4) Four new files will be generated at the output folder, each file corresponding to the files generated from the previous phase.
5) The contents of the file will be sorted in descending order, i.e. the most occurring words/hashtags/users will be at the top.

Part 2:

Pairs Approach:

1) Create a folder named ' **/newinput** ' on your hdfs.
2) Upload the document that was obtained after 'Phase 2' of part 1;
3) Import ' **Pairs.zip** ' from ' **part 2** ' folder.
4) Export it as jar file.
5) Run the jar.
6) The class name for Pairs approach is ' **trends.PairsDriver** '
7) The output will be obtained in ' **/newoutput** ' folder
8) The value of number of counters can be set in the **PairsDriver.java** file by modifying **job.setNumReducetasks(number)**
   By Default the number is set to 3.

Stripes Approach:

1) If you follow the instructions provided for running Pairs Approach, you can skip the first two steps from Pairs Approach.
2) Import ' **Stripes.zip** ' from ' **part 2** ' folder
3) Export it as jar and run it.
4) Class name for Stripes approach is ' **stripes.StripesDriver** '
5) The output will be obtained in ' **/newoutput** ' folder.

Part 3:

1) Create a folder named ' **/inputforkmeans** ' on your hdfs.
2) Create a folder named ' **/Centroids** ' and upload your initial 3 centroids in a file named ' **Centroid1.txt** '
3) Upload the followers count list for your part. Make sure there is no space in your followers count list.
4) Import ' **k-means.zip** ' from ' **part 3** ' folder.
5) Export as jar and run it.
6) The class name for k-means is ' **kmeanCluster.KmeansDriver** '
7) The output will be obtained in ' **/outputforkmens** '
8) **Please Note that code will work for 3 centroids only since the constraint is hardcoded.**

Part 4:

Shortest Path with Distance

1) Import ' **ShortestPathWithDistance.zip** ' from ' **part 4** ' folder
2) Create a folder named ' **/inputforSPDistance** '
3) Upload the ' **input-graph-distance-small** ' file to the folder
4) Export jar and run it
5) The output will be obtained In ' **/outputforSPDistance** '

Shortest Path with Large Graph

1) Import ' **ShortestPathLargeGraph.zip** ' from ' **part 4** ' folder
2) Create a folder named ' **/inputforSPLargeGraph** '
3) Upload the ' **input-graph-large '** file to the folder
4) Export jar and run it
5) The output will be obtained In ' **/outputforSPLargeGraph** '

Shortest Path with Small Graph

1) Import ' **ShortestPathSmallGraph.zip** ' from ' **part 4** ' folder
2) Create a folder named ' **/inputforSPSmallGraph** '
3) Upload the ' **input-graph-small** ' file to the folder
4) Export jar and run it
5) The output will be obtained In ' **/outputforSPSmallGraph** '