

## Assignment 2

Language: Scala

IDE: IntelliJ

Platform: AWS

### Part 1

#### Tweet Processing & Classification using Pipelines

How to run:

1. Build and package the project in IntelliJ to get a jar file named **6350\_assn\_2\_2.11-0.1.jar** under the path `/target/scala-2.11/`.
2. Upload the jar file and the tweets.csv to AWS S3.
3. Run it on a cluster with the following parameters.

Add step

Step type

Spark application

Name

Tweet

Deploy mode

Cluster

Run your driver on a slave node (cluster mode) or on the master node as an external client (client mode).

Spark-submit options

--class Tweet

Specify other options for spark-submit.

Application location\*

s3://6350-assignment-2/6350\_assn\_2\_2.11-0.1.jar

Path to a JAR with your application and dependencies (client deploy mode only supports a local path).

Arguments

s3://6350-assignment-2/Tweets.csv  
s3://6350-assignment-2/tweet-output

Specify optional arguments for your application.

Action on failure

Continue

What to do if the step fails.

Cancel

Add

Result:

```
1 Accuracy: 0.7703381319049213
2 |
```

## Part 2

### PageRank for Airports

How to run:

1. Upload the airport.csv to AWS S3.
2. Run it on a cluster with the following parameters using the jar file from part 1.

The screenshot shows a 'Add step' dialog box with the following configuration:

- Step type:** Spark application
- Name:** PageRank
- Deploy mode:** Cluster
- Spark-submit options:** --class PageRank
- Application location\*:** s3://6350-assignment-2/6350\_assn\_2\_2.11-0.1.jar
- Arguments:** s3://6350-assignment-2/airport.csv, 10, s3://6350-assignment-2/pageRank-output
- Action on failure:** Continue

Buttons at the bottom: Cancel, Add

Part of Result:

```
1  ("ATL",36.62283163071528)
2  ("ORD",30.36223491322298)
3  ("DFW",26.708227044710444)
4  ("DEN",21.905119180659142)
5  ("CLT",21.66117970323455)
6  ("LAX",20.151478161520473)
7  ("PHX",17.048038979281507)
8  ("IAH",16.592653592224337)
9  ("LGA",16.118515405796312)
10 ("SFO",15.685822611800033)
```

## Part 3

### Topic Modeling from Classic Books

How to run:

1. Download a book from the Gutenberg project  
<http://www.gutenberg.org>. Here we choose Sherlock holmes.
2. Upload the Sherlock\_Holmes.txt to AWS S3.
3. Run it on a cluster with the following parameters using the jar file from part 1.

Add step

Step type
Spark application

Name
TopicModeling

Deploy mode
Cluster

Run your driver on a slave node (cluster mode) or on the master node as an external client (client mode).

Spark-submit options
--class TopicModeling

Specify other options for spark-submit.

Application location\*
s3://6350-assignment-2/6350\_assn\_2.11-0.1.jar

Path to a JAR with your application and dependencies (client deploy mode only supports a local path).

Arguments
s3://6350-assignment-2/Sherlock\_Holmes.txt  
s3://6350-assignment-2/topicModeling-output

Specify optional arguments for your application.

Action on failure
Continue

What to do if the step fails.

Cancel
Add

Result:

```

1 TOPIC 1:
2 took 0.003401641634888759
3 project 0.0033752398170424675
4 quite 0.0033725033017256356
5 without 0.003264842725623772
6 miss 0.0031649037539665556
7 good 0.003133985623192229
8 away 0.0031218365118898268
9 nothing 0.0031134844081667987
10 left 0.0030178158007912563
11 matter 0.002959855383350144
12
13 TOPIC 2:
14 quite 0.003434815321012206
15 nothing 0.0033264954557114717
16 good 0.0032908088048380977
17 tell 0.0032369444189662825
18 left 0.003206091796323911
19 every 0.0031140244169175196
20 face 0.0030624477458456715
21 make 0.0030245676409928718
22 took 0.0030059774811405674
23 away 0.0029483253710657726
24

```

```
25 TOPIC 3:
26 project 0.003408121720498146
27 quite 0.0032265720550807976
28 nothing 0.003146731978841166
29 every 0.0030595409592994558
30 tell 0.003008278458386203
31 face 0.002998439562566579
32 took 0.0029355547609897685
33 make 0.002899894972985773
34 good 0.0028683344047137405
35 last 0.0028535202974561413
36
37 TOPIC 4:
38 good 0.0035014259434134657
39 tell 0.0034594319567720815
40 project 0.0033519010169094096
41 every 0.0032213507641849983
42 quite 0.0031526040888385344
43 nothing 0.0031320610148372866
44 without 0.0031275376910154936
45 away 0.003124805150349058
46 face 0.003100970400044747
47 took 0.0030980735324592384
48
49 TOPIC 5:
50 miss 0.0035025775089876365
51 away 0.0033239803644040736
52 good 0.003259036216534128
53 take 0.003194759599812901
54 nothing 0.0031525390894828077
55 tell 0.0031449903465934057
56 young 0.0030756540109242716
57 quite 0.003049677855757757
58 door 0.003014718779274901
59 case 0.0029073308618538813
60
```