By the end of this activity, you will be able to:

1. Import and query text documents with Lucene
2. Perform weighted queries to see how rankings change
3. View the Term Frequency-Inverse Document Frequency (TF-IDF)

NOTE: if you get the error *Exception in thread "main" java.lang.NoClassDefFoundError* when running the commands in this activity, you will need to download Lucene by running these commands:

```
cd $HOME/Downloads
wget http://archive.apache.org/dist/lucene/java/5.5.0/lucene-5.5.0.tgz
tar -xvzf lucene-5.5.0.tgz
```

Step 1. Open a terminal shell. Open a terminal shell by clicking on the square black box on the top left of the screen.

Change into the vector directory:

```
cd Downloads/big-data-2/vector
```

Run *ls* to see the scripts and data directory:

```
[cloudera@quickstart vector]$ ls
data  LuceneQuery.class  LuceneTFIDF.class  runLuceneQuery.sh  runLuceneTFIDF.sh
[cloudera@quickstart vector]$ ls data/
news1.csv  news2.csv  news3.csv
```

The data directory contains three CSV files, which contain textual data from the news.

Step 2. Import and query text documents. Run *runLuceneQuery.sh data* to import the documents in the data directory:

```
./runLuceneQuery.sh data
```

```
[cloudera@quickstart vector]$ ./runLuceneQuery.sh data
Index Location:data/index
Skipping (not csv/htm/html/xml/txt) : write.lock
Indexed : data/news1.csv
Indexed : data/news2.csv
Indexed : data/news3.csv

***************************************************************
3 new documents added.
***************************************************************
```

Enter *voters* to query for that term:

```
Enter query for Lucene (q=quit):
voters
****************************************
Displaying 3 results.
****************************************
1) data/news1.csv score :0.043995064
2) data/news2.csv score :0.024887364
3) data/news3.csv score :0.011129968
```

The output shows the rankings and score for each of the three CSV files for the term *voters*. This shows that *news1.csv* is ranked first, *news2.csv* is second, and *news3.csv* is third.

Next, enter *delegates* to query for that term:

```
Enter query for Lucene (q=quit):
delegates
****************************************
Displaying 2 results.
****************************************
1) data/news2.csv score :0.041339863
2) data/news1.csv score :0.01953125
```

The output shows that *news2.csv* is ranked first, *news1.csv* is ranked second, and *news3.csv* is not shown since the term *delegates* does not appear in this document.

We can query for multiple terms by entering them together; enter *voters delegates* to query for both terms:

```
Enter query for Lucene (q=quit):
voters delegates
****************************************
Displaying 3 results.
****************************************
1) data/news2.csv score :0.04811
2) data/news1.csv score :0.041432917
3) data/news3.csv score :0.0032286723
```

The output shows that *news2.csv* is ranked first, *news1.csv* ranked second, and *news3.csv* ranked third.

Step 3. Perform weighted queries. We can perform a weighted query (or "boosting") to give one term more importance than the others. Enter *voters^5 delegates* to give the term *voters* a boost factor of 5:

```
Enter query for Lucene (q=quit):
voters^5 delegates
****************************************
Displaying 3 results.
****************************************
1) data/news1.csv score :0.047636837
2) data/news2.csv score :0.035135828
3) data/news3.csv score :0.005357802
```

The output shows that *news1.csv* is ranked first and *news2.csv* is ranked second. Note that these two rankings are reversed from when we performed the same query without boosting.

Enter *q* to quit this script.

Step 4. View the TF-IDF. Run *runLuceneTF-IDF.sh data* to see the TF-IDF for terms in the documents:

```
./runLuceneTFIDF.sh data
```

```
[cloudera@quickstart vector]$ ./runLuceneTFIDF.sh  data
Index Location:data/index
Skipping (not csv,htm,html,xml,txt : write.lock
Indexed : data/news1.csv
Indexed : data/news2.csv
Indexed : data/news3.csv
*******************************************************
3 new documents added.
*******************************************************
```

Enter *voters* to see the TF-IDF for that term:

```
Enter a term to calculate TF-IDF (q=quit):
voters
Doc # 0: data/news1.csv   TF-IDF = 2.252547264099121
Doc # 1: data/news2.csv   TF-IDF = 1.5927913188934326
Doc # 2: data/news3.csv   TF-IDF = 0.712317943572998
```

Enter *delegates* to see the TF-IDF for that term:

```
Enter a term to calculate TF-IDF (q=quit):
delegates
Doc # 0: data/news1.csv   TF-IDF = 1.0
Doc # 1: data/news2.csv   TF-IDF = 2.6457512378692627
```

Enter *q* to quit this script.

Mark as completed