RBDA Project Readme

Technology Trend Analysis Based on Text Mining

Jiachen Zhu, Yahui Cui, Ranjita Rajeeva Shetty

1. Where is our data located?

If you want to download the data directly from the internet, you could do it by using following URLs:

stackoverflow.com:

https://archive.org/download/stackexchange/stackoverflow.com-Posts.7z

Libraries.io:

https://zenodo.org/record/1196312/files/Libraries.io-open-data-1.2.0.tar.gz

arXiv.org:

We used a Python script "`get_arxiv.py`" to download it (the script is attached with this Readme file). To run the script, you need have the following packages installed:

atoma: https://github.com/NicolasLM/atoma

requests: https://github.com/requests/requests

You could also access our data on NYU HPC Dumbo:

HDFS FileSystem:

stackoverflow.com: `/user/jz3224/RBDAData/Ranjita`

Libraries.io: `/user/jz3224/RBDAData/Lily`

arXiv.org: `/user/jz3224/RBDAData/Jason`

Regular FileSystem (you probably need administrative rights to access it):

stackoverflow.com: `/archive/j/jz3224/RBDAProject/Ranjita`

Libraries.io: `/archive/j/jz3224/RBDAProject/Lily`

arXiv.org: `/archive/j/jz3224/RBDAProject/Jason`

2. How to clean the data?

Under the folder `NewDataCleaning` of our final code drop, you could see three folders. Each folder is for one dataset.

You need the following command to run the code:

stackoverflow.com:

```
javac -classpath `yarn classpath`:. -d . *.java
jar -cvf DataClean.jar *.class
hadoop jar DataClean.jar DataClean pathToInput pathToOutput
```

Libraries.io:

```
javac -classpath `yarn classpath`:./opencsv-2.4.jar:. -d . *.java
jar -cvf MRCleaning.jar *.class
export LIBJARS=./opencsv-2.4.jar
export HADOOP_CLASSPATH=./opencsv-2.4.jar
hadoop jar MRCleaning.jar MRCleaning -libjars ${LIBJARS} -input pathToInput pathToOutput
```

arXiv.org:

```
javac -classpath `yarn classpath`:. -d . *.java
jar -cvf DataCleaning.jar *.class
hadoop jar DataCleaning.jar DataCleaning pathToInput pathToOutput
```

3. How to do text normalization and lemmatization?

After getting the output from the previous step, you could start to do text normalization and lemmatization by using the code under the folder SentenceCleaningWithNLP.

All those code should be run in the `spark-shell` by inputing the command line by line. Also, you should use the following command to start `spark-shell`:

```
module load spark/2.3.0
spark-shell --packages JohnSnowLabs:spark-nlp:1.6.0
```

One thing you should pay attention to is that you need to change the input and output path for each file. The input path could be changed in line 6 and the output path could be changed in line 42.

4. What's next?

After having the result from the perviously step, you have three choices:

(1) You could use code under the folders WordCountYear and DemoCode to get the same analytics result we presented during the symposium. If you want to do this, go to step 5.

(2) You could use code under the folder TFIDF to get the tf-idf analytics result, which is not very successful. If you want to do this, got the step 6.

(3) You could use code under Word2vec to get the Word2vec analytics result, which we think it has great potential but we haven't figured out how to integrate it with other result. If you want to do this, go to step 7.

5. How to use the code under the folders WordCountYear and DemoCode?

By using the following command you could compile and run the WordCountYear code:

```
javac -classpath `yarn classpath`:. -d . *.java
jar -cvf WordCountYear.jar *.class
```

`hadoop jar DataWordCountYearCleaning.jar WordCountYear` **`pathToInput pathToOutput`**

You need run the code three times, and one for each dataset. The input should be the result of step 3.

After you run the code above, download the final output to your local mechine, then you could use it with the two jupyter notebook files under the folder DemoCode.

6. How to use the code under the folder TFIDF?

By using the following command you could compile and run the TFIDF code:

```
javac -classpath `yarn classpath`:. -d . *.java
jar -cvf tfidf.jar *.class
hadoop jar tfidf.jar TFIDFWordCount pathToInput1 pathToOutput
hadoop jar tfidf.jar TFIDFGetWord pathToOutput pathToOutput2
```

Since tfidf is a failed attempt, we only do it with arXiv.org data.

7. How to use the code under the folder Word2vec?

The code of "`readme.scala`" should be run in the spark-shell by inputing the command line by line. Also, you should use the following command to start `spark-shell`:

```
module load spark/2.3.0
spark-shell --driver-memory 40g
```

One thing you should pay attention to is that you need to change the input and output path for each run. The input path could be changed in line 6 and the output path could be changed in line 27.

After the run the code above, download the final output to your local mechine, then you could use it with "`cleaned.ipynb`" to explore the result.

Contact Information:

Jiachen Zhu: jz3224@nyu.edu
Yahui Cui: yc3329@nyu.edu
Ranjita Rajeeva Shetty: rrs462@nyu.edu

If you find any difficulty to run our code, you could always contact us directly. Thank you very much.