1. WordCount on Collected Tweets

Proccess:

1. First we collected tweets on a topic here it is "MachinLearning" in R.
2. Then we generated a text file using the collected tweet text named tweetText.txt.
3. We then run this file in hadoop server using our WordCount1 program.
4. The file which is generated from the hadoop is collected and again fed in Jupyter to make a word cloud. The generated file is named as hashtags.txt

Jupyter Code :

Name : Jupyter/DIC_LAB_4_QNS_1.ipynb

Input :

Input/Activity1/tweetText/tweetText.txt

Sample Output :

Output/Activity1/wordCloud/hashtags .txt

Jar and Source Files :

Jar File : Jar/Activity1/wc1.jar
Source Code : Jar/Activity1/WordCount1.java

To run :

```
hadoop com.sun.tools.javac.Main WordCount1.java
jar cf wc1.jar WordCount1*.class
hadoop jar wc1.jar WordCount1 ~/input/tweetText ~/output1
```

Output Format :

```
<word> <count>
Eg: ##MachineLearning:      1
```

2. Word Co-occurance on Collected Tweets using pairs and stripes method

Proccess:

1. First we collected tweets on a topic here it is "MachinLearning" in R.
2. Then we generated a text file using the collected tweet text named tweetText.txt.
3. We then run this file in hadoop server using our PairsOccurrence.java program.

Jupyter Code :

Name : Jupyter/DIC_LAB_4_QNS_1.ipynb

Input :

Input/Activity2/tweetText/tweetText.txt

Sample Output :

Output/Activity2/pairs/pairs.txt
Output/Activity2/stripes/stripes.txt

Jar and Source Files :

For Pairs Method : Jar/Activity2/po.jar
For Stripes Method : Jar/Activity2/so.jar

Pairs Source Code : Jar/Activity2/PairsOccurrence.java
Stripes Source Code : Jar/Activity2/StripesOccurrence.java

To run :

For Pairs :

```
hadoop com.sun.tools.javac.Main PairsOccurrence.java
jar cf so.jar PairsOccurrence*.class
hadoop jar so.jar PairsOccurrence ~/input/tweetText ~/output1
```

For Stripes :

```
hadoop com.sun.tools.javac.Main StripesOccurrence.java
jar cf so.jar StripesOccurrence*.class
hadoop jar so.jar StripesOccurrence ~/input/tweetText ~/output1
```

Output Format :

For Pairs :

```
<<word1> <word2> >  #Count
Eg : <"#AI #BigData>          1
```

For Stripes :

```
<word1> {word2 = #word2count      word3 = #word3count  ………… wordN = #wordNcount}
Eg : "#AI{#MachineLearning" = 1    #bigdata = 34    fuel<U+2026> = 56    }
```

3. Featured Activity 1: Wordcount on Classical Latin text

Input :

Input/Featured Activity1/

Output :

Output/Featured Activity1/lemmatization.txt

Jar and Source Files :

Jar File : Jar/Featured Activity1/lemma.jar
Source Code : Jar/Featured Activity1/Lemmatization.java

To run :

```
hadoop com.sun.tools.javac.Main Lemmatization.java
jar cf lemma.jar Lemmatization*.class
hadoop jar lemma.jar Lemmatization ~/input/latin ~/output1
```

Output Format :

```
<word> <<docId> [#lineNumber , #positionInLine] > <<docId [#lineNumber , #positionInLine] >> ….. count : #Count
Eg: abigo <verg. aen. [261, 6]> <verg. aen. [407, 7]> count : 2
```

4. Featured Activity 2: Word co-occurrence among multiple documents.

4.A : 2-word co-occurrence

Input :

Input/Featured Activity2/2
Input/Featured Activity2/4
Input/Featured Activity2/6
Input/Featured Activity2/8
Input/Featured Activity2/10
Input/Featured Activity2/10

Output :

Output/Featured Activity2/4a/2/CooccurrencePerformance.txt
Output/Featured Activity2/4a/4/CooccurrencePerformance.txt
Output/Featured Activity2/4a/6/CooccurrencePerformance.txt
Output/Featured Activity2/4a/8/CooccurrencePerformance.txt
Output/Featured Activity2/4a/10/CooccurrencePerformance.txt
Output/Featured Activity2/4a/15/CooccurrencePerformance.txt

Jar and Source Files :

Jar File : Jar/Featured Activity2/4a/cop.jar
Source Code : Jar/Featured Activity2/4a/CooccurrencePerformance.java

To run :

```
hadoop com.sun.tools.javac.Main CooccurrencePerformance.java
jar cf cop.jar CooccurrencePerformance*.class
time hadoop jar cop.jar CooccurrencePerformance ~/input/latin/2 ~/output1
```

Output Format :

```
{<word1> <word2>} <<docId1> <docId2> ….. <docIdN>>
Eg: {a aba}             <ter. heaut. 4.3.18> <ter. Heaut. 4.3.18>
```

Plot :

File Name : Jupyter/4a.tiff (Graph between number of files and time taken)

4.b : 3-word co-occurrence

Input :

Input/Featured Activity2/2
Input/Featured Activity2/4
Input/Featured Activity2/6
Input/Featured Activity2/8
Input/Featured Activity2/10
Input/Featured Activity2/15

Output :

Output/Featured Activity2/4b/2.1/CooccurrencePerformance4b .txt
Output/Featured Activity2/4b/4.1/CooccurrencePerformance4b .txt
Output/Featured Activity2/4b/6.1/CooccurrencePerformance4b .txt
Output/Featured Activity2/4b/8.1/CooccurrencePerformance4b .txt
Output/Featured Activity2/4b/10.1/CooccurrencePerformance4b .txt
Output/Featured Activity2/4b/15.1/CooccurrencePerformance4b .txt

Jar and Source Files :

Jar File : Jar/Featured Activity2/4b/cop4b.jar
Source Code : Jar/Featured Activity2/4b/CooccurrencePerformance4b.java

To run :

```
hadoop com.sun.tools.javac.Main CooccurrencePerformance4b.java
jar cf cop4b.jar CooccurrencePerformance4b*.class
time hadoop jar cop4b.jar CooccurrencePerformance4b ~/input/latin/2 ~/output1
```

Output Format :

```
{<word1> <word2> <word3>} <<docId1> <docId2> …… <docIdN>>
Eg: {Ab nostro hic}          <ter. hec. 5.3.9> <ter. Hec. 5.3.9>
```

Plot :

File Name : Jupyter/4b.tiff (Graph between number of files and time taken)