Instructions for Lab 4 Execution

Aneesh Bhatnagar (50208162, aneeshbh@buffalo.edu)

1. Word count on tweets:

For collecting the tweets, run the file **tweets.py,** in the terminal in the following form:
python/python2.7 tweets.py <SearchTerm> <OutputFile>

Then, run the **preprocess.py** file using the following form, where the input file is the output from the 1st Python run:
python/python2.7 preprocess.py <InputFile> <OutputFile>

This output file will go into the HDFS for the wordcount program and the output of that will go into the Jupyter notebook for making the word cloud.
hadoop jar wc.jar WordCount ~/input/tweets ~/output

Output Format for Hadoop:

```
word1  count1
word2  count2
.
.
.
word_n count_n
```

2. Word co-occurrence on tweets

For collecting the tweets, run the file **tweets.py,** in the terminal in the following form:
python/python2.7 tweets.py <SearchTerm> <OutputFile>

Then, run the **cleanTweets.py** file using the following form, where the input file is the output from the 1st Python run:
python/python2.7 cleanTweets.py <InputFile> <OutputFile>

This output file will go into the HDFS for the wordcount program and the output of that will go into the Jupyter notebook for making the word cloud.

hadoop jar PairCooccurence.jar PairCooccurence ~/input/tweets ~/output

hadoop jar StripeCooccurence.jar StripeCooccurence ~/input/tweets ~/output

Output Format:
This is the output format for pairs:
word1, word2 <count1>
word3, word4 <count2>
.
.
.
word_x, word_y <count_n>

For the Stripes, the output is as follows:
word1 {neighbour1:count1; neighbour2:count2; … ; neighbour_n:count_n}
word2 {neighbour1:count1; neighbour2:count2; … ; neighbour_n:count_n}
.
.
.
word_n {neighbour1:count1; neighbour2:count2; … ; neighbour_n:count_n}

3. Wordcount on Classical Latin text

Assuming the latin files are already in the folder called ~/input/latin (any number of files), use the following command to run the WordCount/Location using jar file.
The code assumes that the new_lemmatizer.csv file is present in the same working directory as the jar file.

```
hadoop jar LemmaLocation.jar LemmaLocation ~/input/latin ~/output <number_of_files>
```

- <number_of_files> is the number of files you want to run the code for. If you enter a value more than the maximum number of files, it will use all the files. The files are selected in alphabetical order, 1 to n.

Output Format:
The output format is as shown below, one word per line.
word1 <location1>, <location2>, <location3> .... <location-n>
word2 <location1>, <location2>, <location3> .... <location-n>
.
.
.
word_n <location1>, <location2>, <location3> .... <location-n>

4. Word co-occurrence among multiple documents

a) For bi-gram (2-gram):
Assuming the latin files are already in the folder called ~/input/latin (any number of files), use the following command to run the Bi-Gram using the jar file.
The code assumes that the new_lemmatizer.csv file is present in the same working directory as the jar file.

```
hadoop jar Word2grams.jar Word2grams ~/input/latin ~/output <number_of_files>
```

- <number_of_files> is the number of files you want to run the code for. If you enter a value more than the maximum number of files, it will use all the files. The files are selected in alphabetical order, 1 to n.

Output Format:
The output format is as shown below, one pair per line.

```
word1,word2 <location1>, <location2>, <location3> ....  <location-n>
word3, word4 <location1>, <location2>, <location3> ....  <location-n>
.
.
.
word_x, word_y <location1>, <location2>, <location3> ....  <location-n>
```

b) For tri-gram (3-gram):
Assuming the latin files are already in the folder called ~/input/latin (any number of files), use the following command to run the Tri-Gram using jar file.
The code assumes that the new_lemmatizer.csv file is present in the same working directory as the jar file.

```
hadoop jar Word3grams.jar Word3grams ~/input/latin ~/output <number_of_files>
```

- <number_of_files> is the number of files you want to run the code for. If you enter a value more than the maximum number of files, it will use all the files. The files are selected in alphabetical order, 1 to n.

Output Format:
The output format is as shown below, one triplet per line.

```
word1,word2,word3 <location1>, <location2>, <location3> ....  <location-n>
word4, word5, word6 <location1>, <location2>, <location3> ....  <location-n>
.
.
.
word_x, word_y, word_z <location1>, <location2>, <location3> ....  <location-n>
```