587: Data Intensive Computing
Lab 5: Spark
README

Team Members:
Mythri Jonnavittula
Shawn Varghese

Environment Selection:

To run Spark, we configured a local installation of Jupyter with Spark. The submitted file is an IPython notebook. We used Jupyter as it is a popular package and works well with Spark. The specifications of the machine used for deployment are as follows:

CPU Cores: 4
RAM : 8 GB
Processor: i5
Operating System: Ubuntu 64-bit

Folder Structure:

1) Input - Contains input files for Titanic vignette, 2-gram input and 3-gram input
2) Output - Contains the output files for the program results
3) 2-gram.ipynb - IPython notebook for 2-gram
4) 3-gram.ipynb - IPython notebook for 3-gram
5) new_lemmatizer.csv - CSV file contains lemmas
6) Report.pdf - Statistics relating to execution time for programs based on number of files processed
7) Titanic_Vignette.ipynb - Titanic Vignette

Note: There are additional files and folders created while running pyspark. Please ignore these files, and leave them unmodified.

Program Execution Instructions:

1) Open the Jupyter notebook for 2-gram or 3-gram.
2) The 2nd cell contains the filepath for the input and output file. They can be modified if necessary
3) Click the ‘Cell’ menu and select ‘Run All’. The output will be saved into the path specified.
4) Output will be created by default in the Output folder, and will the folder will be of the convention “<n>_gram_output_<no of files>”

Program Flow:

1) Initially all the files to be processed are combined into a single RDD.

2) A lemma dictionary is created. Following that each line in the RDD is read, and word-pairs are created from them along with positional information. In this process, the words are checked for their corresponding lemmas, and replaced if necessary,
3) The output is fed to the flatFile() function to be processed for each tuple.
4) Each tuple is again mapped to produce the combination (pair,position)
5) This is passed to the reducer which combines all positions for a single pair.

Explanation of Output:

We found that the time taken for execution increases as the number of files is increased, as was expected. However, the time taken was not completely linear. As the number of files increases, it takes far longer for the program execution. Further, the reducer took the most time, and this is also expected, as the reducer needs to combiner the positions for all the pairs. The times have been plotted in the Report.pdf file.