CSE 587 Data Intensive Computing
Lab 5 Readme File

Aniruddh Chaturvedi [5020 6958]
Ashwin Nikam [5020 7368]

Environment chosen and how to run:

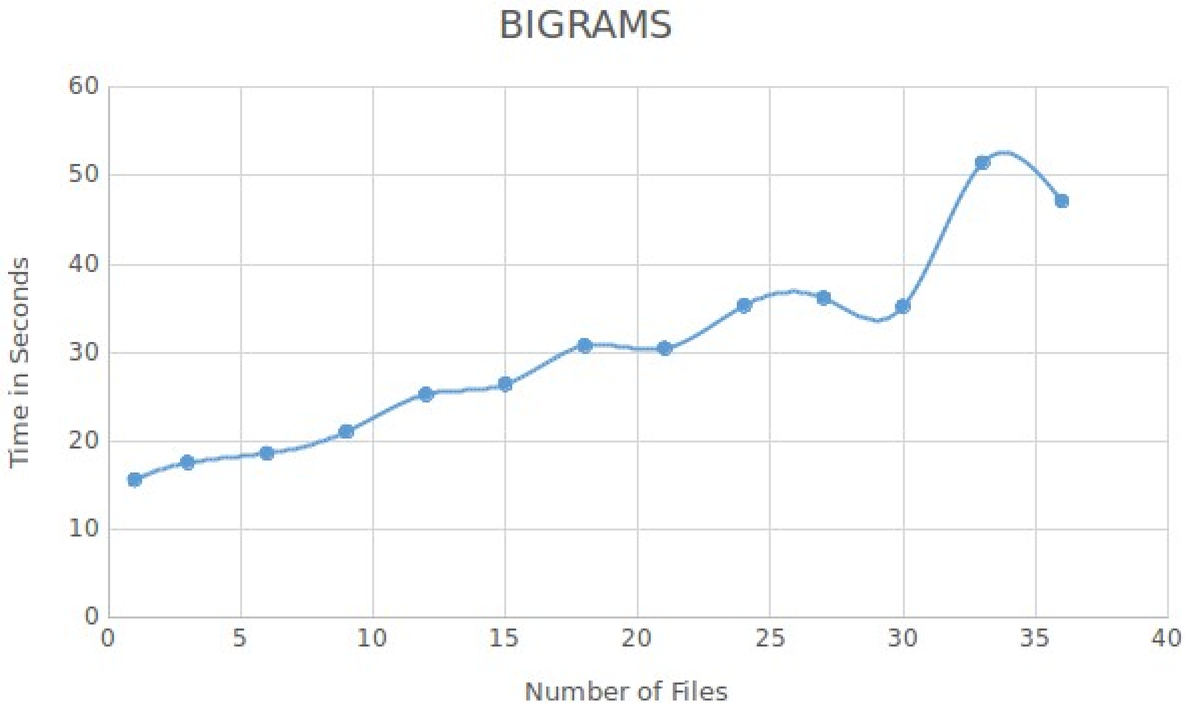
- The **python code** has been written in a **Jupyter notebook** called **WordCoOccurence.ipynb** which generates two output folders ‘paircount’ and ‘tricount’ for 2 grams and 3 grams respectively when the notebook is executed.
- The folder ‘co_occur_input’ and ‘new_lemmatizer.csv’ need to be in the same directory as WordCoOccurence.ipynb

Contents

- **WordCoOccurence.ipynb**
  - This is the main notebook which generates the bigrams and trigrams which are stored in paircount and tricount folders.
- **Vignette.ipynb**
  - The titanic vignette.
- **co_occur_input**
  - Input folder consisting of sample input files on which we want to compute bigrams and trigrams.
- **paircount**
  - Sample output folder for bigrams generated by WordCoOccurence.ipynb
- **tricount**
  - Sample output folder for trigrams generated by WordCoOccurence.ipynb
- **new_lemmatizer.csv**
  - The lemmatizer file which is used by WordCoOccurence.ipynb

Performance Evaluation for word co-occurrence using Spark

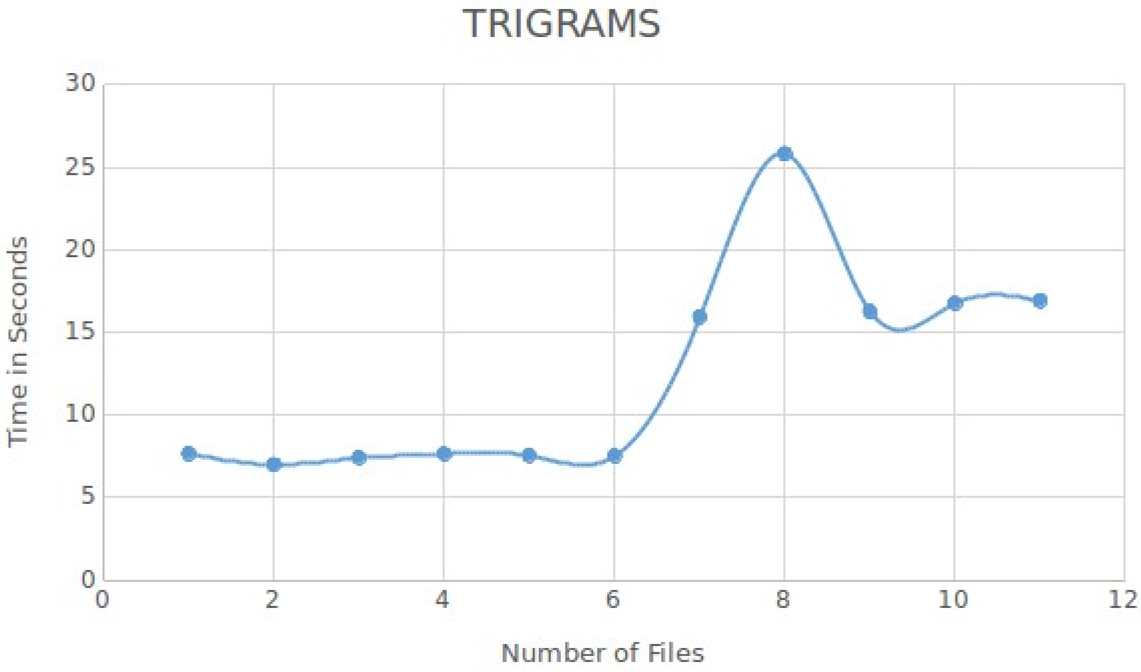
1. When n = 2 (bi-grams)



The plot above shows performance of spark for word occurrence using bi-grams where n = 2. The performance has been evaluated by checking the execution time and varying the number of input documents. We can see from the above plot that the execution time increases as we increase the number of documents to process.

No of Files	Time in seconds for bi-grams
1	15.624
3	17.58
6	18.62
9	21.081
12	25.273
15	26.447
18	30.744
21	30.432
24	35.283
27	36.146
30	35.187
33	51.446
36	47.095

2. When n = 3 (tri-grams)



The plot above shows performance of spark for word occurrence using tri-grams where n = 3. The performance has been evaluated by checking the execution time and varying the number of input documents. We can see from the above plot that the execution time increases as we increase the number of documents to process. Spark framework is much faster than Map-Reduce framework for this data. This is because Spark performs better when all the data fits in the memory. Hadoop MapReduce however, is designed for data that doesn’t fit in the memory and it can run well alongside other services.

No of Files	Time in seconds for tri-grams
1	7.653
2	6.996
3	7.412
4	7.635
5	7.543
6	7.525
7	15.913
8	25.83
9	16.253
10	16.747
11	16.897

Word Co Occurrence table (Format):

n-gram (n=2)	**Location**
('prae', 'hic')	<mac. frag 9>
('inpleo', 'uesica')	<mac. frag 5>
('congruo', 'erilem')	<prud. epil. 22>
n-gram (n=3)	**Location**
('uix', 'in', 'ibi')	<mac. frag 9>
('uenio', 'in', 'tristis')	<mac. frag 6>
('ne', 'lito', 'dum')	<mac. frag 4>