DIC Lab 5
Programming The data flow for Big Data Analytics using Apache Spark

Group:
Arnav Ahire (5020 8006) ubit-arnavane
Ritika (5020 6346) ubit-ritika

(Readme)

Files provided:

1. *titanic.csv*: Required as input for Spark_Vignette.ipynb.
2. *Spark_Vignette.ipynb*: The Vignette notebook for Apache Spark.
3. *new_lemmatizer.csv*: The file to extract lemmas from.
4. '*Input*' folder: This folder contains all the input files for our word co-occurrence program. (For both Bi-grams i.e n=2 and Tri-grams i.e n=3)
5. *'Output'* folder: This folder contains the output that we had received for the specified input from *'Input'* folder.
6. *DIC_Lab5_FeaturedActivity.ipynb*: This notebook contains the code for Bi-grams and Tri-grams Word co-occurrence program of MapReduce.

Steps to be performed:

1. Spark installation is a prerequisite.
2. Open Jupyter notebook via terminal for executing Vignette and Featured Activity. Run it by setting *Python 2* as the kernel.
3. For Spark Vignette run all the cells of *Spark_Vignette.ipynb* notebook. Make sure the file titanic.csv stays in the same folder as this Vignette.
4. For performing Featured Activity, open *DIC_Lab5_FeaturedActivity.ipynb* and execute all the cells one by one in sequence.
5. The input taken for this is present in the *Input* folder. It contains four data files that will be used as reference to test our word co-occurrence program.
6. The output for bi-grams is given in 2 folders, *BiGramsMapperOutput* which stores the output of the mapper function and *BiGramsReducerOutput* which stores the output of reducer function.
7. Similarly, output for tri-grams is given in 2 folders, *TriGramsMapperOutput* which stores the output of the mapper function and *TriGramsReducerOutput* which stores the output of reducer function.
8. In order to refer the obtained output, you can see the sample output that we obtained from the *'Output'* folder.

Expected Output:

The final expected output from *Reducer* should have a format as this:

Bi-grams:

```
(u'qui coegisti.', u'<aus. biss. praef>,<aus. biss. praef>,<aus. biss. praef>,<aus. biss. praef>')
(u'atque caerula,', u'<aus. biss. 3>')
(u'poeta p.', u'<sen. eld. fr. 1.4>')
('nescio ut', u'<aus. biss. 3>')
(u'cera misce,', u'<aus. biss. 5>')
('uerus hic', u'<sen. eld. fr. 1.2>')
(u'sine posset,', u'<aus. biss. praef>')
('quod solacium', u'<aus. biss. praef>,<aus. biss. praef>')
(u'ubi Quint.', u'<sen. eld. fr. 1.3>')
('patria libero', u'<aus. biss. 3>')
('negotium potis', u'<mac. frag 10>')
('neo intro', u'<aus. biss. praef>')
('ilico ut', u'<aus. biss. 3>')
(u'naturale puellas:', u'<aus. biss. 5>')
(u'innocens, pudicus', u'<prud. epil. 2>')
(u'capto Germana', u'<aus. biss. 3>')
('capta lingua', u'<aus. biss. 3>')
('tenuis pone', u'<aus. biss. 2>')
(u'temperies age,', u'<aus. biss. 5>')
('neque dies', u'<aus. biss. praef>')
('ut hic', u'<aus. biss. 3>,<aus. biss. 3>,<sen. eld. fr. 1.2>')
(u'induco duc,', u'<sen. eld. fr. 1.2>')
('quamuis abscido', u'<aus. biss. praef>')
(u'quod cantilenae,', u'<aus. biss. praef>,<aus. biss. praef>')
```

Tri-grams:

```
('sinis induco uis', u'<sen. eld. fr. 1.2>')
(u'operio sine supergressus,', u'<aus. biss. praef>')
(u'Libero, non fas', u'<aus. biss. praef>')
('arcanum ius intro', u'<aus. biss. praef>')
(u'aut,  tuus', u'<aus. biss. praef>')
(u'irrumpis, iugo tuus', u'<aus. biss. praef>')
(u'praecipuus et Quint.', u'<sen. eld. fr. 1.2>,<sen. eld. fr. 1.2>,<sen. eld. fr. 1.2>,<sen. eld. fr. 1.2>')
('tu arcanus lorum', u'<aus. biss. praef>')
(u'Cereri, tu et', u'<aus. biss. praef>,<aus. biss. praef>')
(u'ingressu, quis arcanum', u'<aus. biss. praef>,<aus. biss. praef>')
('initium suus ne', u'<aus. biss. praef>')
('quis quamquam solacium', u'<aus. biss. praef>')
(u'quisque Libero, iura', u'<aus. biss. praef>')
('controuersia uerus nox', u'<sen. eld. fr. 1.2>')
(u'nascor Danuuii, fortuna', u'<aus. biss. 3>')
('quamquam sub ludo', u'<aus. biss. praef>')
('neque alumnus securitas', u'<aus. biss. praef>')
('bellicum nescio hic', u'<aus. biss. 3>,<aus. biss. 3>')
('hic haud missa', u'<aus. biss. 2>')
```

Word Co-occurrence Table:

We have mentioned some sample entries of our bi-grams and tri-grams word co-occurrence problems in an output table. It consists of keys i.e n-grams in one column and their values i.e the location of the n-grams in another column. Note here that are n-grams are space separated (“ ”) and so our key looks like this:

- Word<space>Neighbour for bigrams
- Word<space>Neighbour1<space>Neighbour2 for trigrams

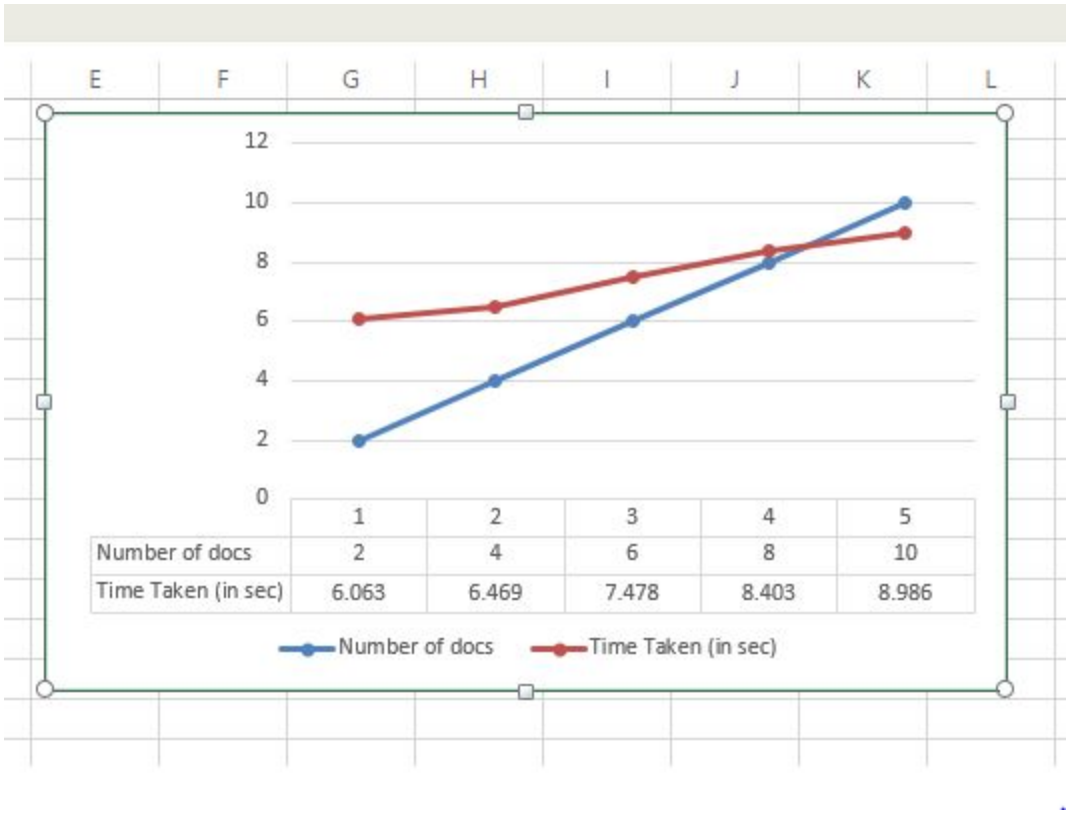
The output table of bi-grams and tri-grams is as follows:

N-gram (n=2)	**Location**
nescio ut	<aus. biss. 3>
uerus hic	<sen. eld. fr. 1.2>
quod solacium	<aus. biss. praef>,<aus. biss. praef>
iura etiam	<aus. biss. praef>
N-gram (n=3)	**Location**
operto arcanum qui	<aus. biss. praef>
quidam quis quicumque	<sen. eld. fr. 1.2>,<sen. eld. fr. 1.2>
ubi grauis dico	<sen. eld. fr. 1.3>
et atque Rheno	<aus. biss. 3>

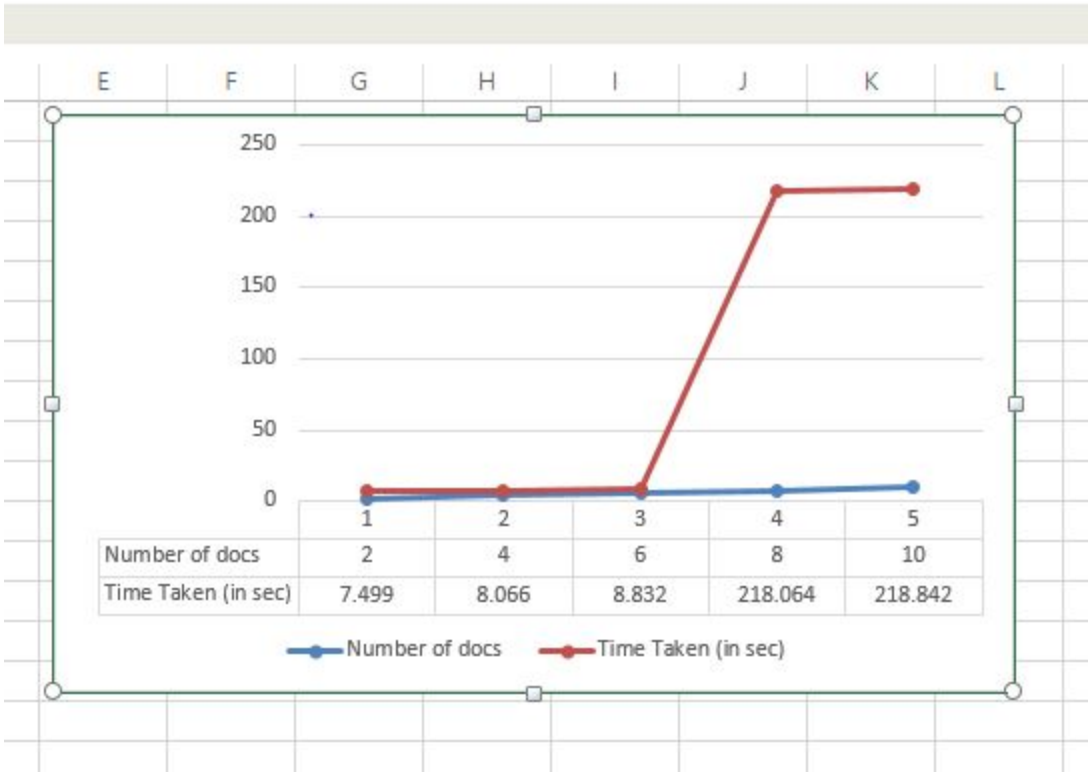
Plot and Observations:

We ran our bi-grams and tri-grams word co-occurrence programs for different number of documents and measured the time taken by each run of Apache Spark. Typically, we initially checked for 2 documents, then gradually increased it to 4, then 6 and 8 upto a total of 10 documents in a run, and recorded the time for each run and created the plots for both of them. We got the following plots:

Bi-Grams: (n=2)



Tri-Grams: (n=3)



Observation:

1. We can observe from both the graphs in general that as the number of documents to be processed increase, the time required to process the documents increases.
2. We can observe this relation in case of bi-grams easily.
3. We can infoer in case of tri-grams also the same, except for the fact that as the number of documents increases from 6 to 8 and 10, the increase in the processing time becomes exponential. Thus we can infer that as the number of documents increase and if the documents are large enough, then the amount of processing time grows exponentially in comparison to the increase in number of documents.