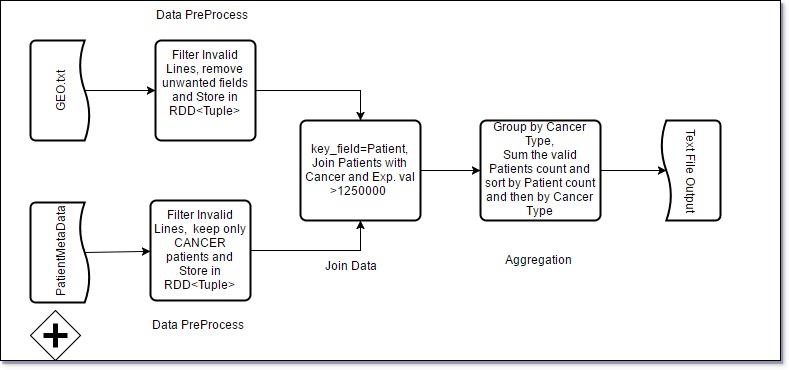
Cloud Computing Assignment 3 using Spark - Report

Task1: Number of cancer patients with certain active genes per cancer type:



Steps for Computation:

1. Invoke an SparkConf variable to start a SparkConfig and then create a SparkContext instance.
2. Data pre-processing: Read files using sc.textFile data files and Filter the header row and unwanted columns. Store GEO data and PatientMetaData into respective Tuple<fields> RDD.
3. Call 2 custom filter() methods which checks for valid gene Expression value and Cancer Patients and filters out invalid records.
4. Using PairRDD.join(), both gene Expression value and Cancer Patients data are joined on a common field PatientId.
5. Using the Aggregation functions available in spark, we groupBy() on CancerType and then count the sum of Patients per each CancerType. We then sort() the number of Patients and CancerType and write it to a TXT file using the saveAsText function.

Output:

breast-cancer 74

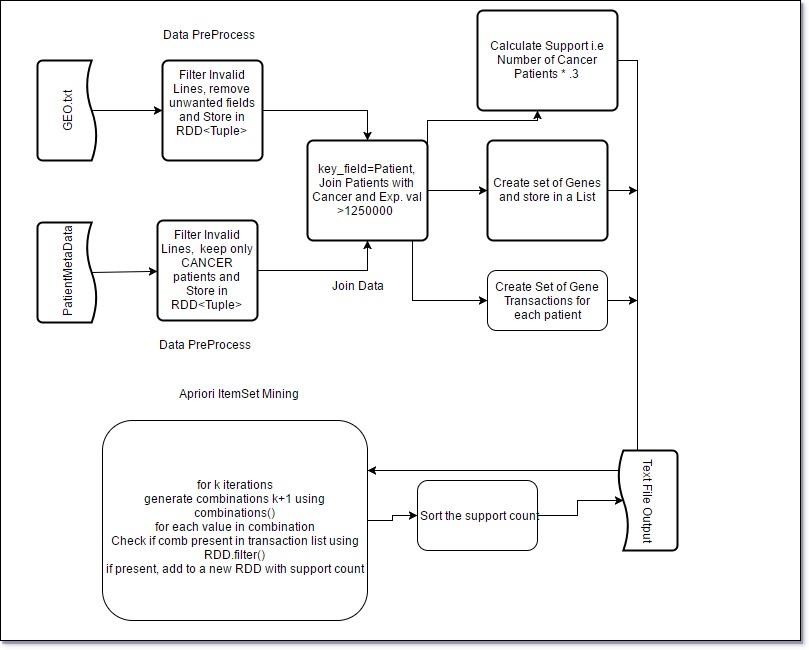
pancreatic-cancer 68

prostate-cancer 65

lymphoma 65

leukemia 60

# Task2: Frequent Itemset Mining



Steps for Computation:

1. Follow same set of steps as in Task1 and get a list of Cancer Patients with respective Stron Gene combinations.
2. Compute the minimum support for -> (occurances of each geneset)/(total of Cancer Patients) and eliminate the genes below minimum support count using a custom filter function().
3. Apriori Implementation: For iter < num\_iter

Generate item\_set combinations using itertools.combinatorics flatMap (itemset,transactionList) and return (item, count) reduce (item,count)

filter(count,min\_support) previousSet.union(newSet)

1. Finally sort using SortBy(descending) support count and saveAsText(Output)

Output (Sample):

|  |  |  |
| --- | --- | --- |
| 295 | 42 |  |
| 261 | 299 |
| 251 | 199 |
| 198 | 42 | 299 |
| 188 | 42 | 199 |
| 162 | 199 | 299 |
| 149 | 234 |  |
| 140 | 887 |  |
| 138 | 798 |  |
| 138 | 883 |  |
| 137 | 917 |  |
| 136 | 747 |  |
| 136 | 772 |  |

# Task3: Association Rule Generation

Steps for Computation:

1. Read Task2Output file and Map(geneset, support) pairs.
2. Filter out genesets < 2 using Filter().
3. Generate subsets and calculate confidence by using custom FlatMap(conf\_calc\_map)
4. Filter out genes with < 60% using Spark filter().
5. Sort confidence using SortBy(descending).
6. Format Output using Map(OutputFormat).
7. Save Output to text File. Output:

(u'42', u'299') (u'299',)-(u'42', u'299') 0.76

(u'42', u'199') (u'199',)-(u'42', u'199') 0.75

(u'42', u'299') (u'42',)-(u'42', u'299') 0.67

(u'199', u'299') (u'199',)-(u'199', u'299') 0.65

(u'42', u'199') (u'42',)-(u'42', u'199') 0.64

(u'199', u'299') (u'299',)-(u'199', u'299') 0.62

Task Time:

|  |  |
| --- | --- |
| Task | Small Data Runtime |
| 1 | 4 Seconds |
| 2 | 12 seconds |
| 3 | 6 seconds |
|  |  |

Execution Commands:

spark-submit --master yarn --num-executors 3 task1.py --output hdfs:///user/lban2974/3Task1/ hdfs:///share/genedata/small/

spark-submit --master yarn --num-executors 3 task2.py --output hdfs:///user/lban2974/3Task2/ hdfs:///share/genedata/small/

spark-submit --master yarn --num-executors 3 task3.py --output hdfs:///user/lban2974/3Task3/ hdfs:///user/lban2974/3Task2/

Output:

http://soit-hdp-pro-1.ucc.usyd.edu.au:50070/explorer.html#/user/lban2974/3Task1

http://soit-hdp-pro-1.ucc.usyd.edu.au:50070/explorer.html#/user/lban2974/3Task2

http://soit-hdp-pro-1.ucc.usyd.edu.au:50070/explorer.html#/user/lban2974/3Task3