THE UNIVERSITY OF
SYDNEY

School of Information Technologies
Faculty of Engineering & IT

ASSIGNMENT/PROJECT COVERSHEET - GROUP ASSESSMENT

Unit of Study: COMP5349

Assignment name: Assignment 3

Tutorial time: Thursday **Tutor name:** 4:00 PM - 6:00 PM

DECLARATION
We the undersigned declare that we have read and understood the *University of Sydney Academic Dishonesty and Plagiarism in Coursework Policy*, an, and except where specifically acknowledged, the work contained in this assignment/project is our own work, and has not been copied from other sources or been previously submitted for award or assessment.

We understand that failure to comply with the *Academic Dishonesty and Plagiarism in Coursework Policy* can lead to severe penalties as outlined under Chapter 8 of the *University of Sydney By-Law 1999* (as amended). These penalties may be imposed in cases where any significant portion of my submitted work has been copied without proper acknowledgement from other sources, including published works, the internet, existing programs, the work of other students, or work previously submitted for other awards or assessments.

We realise that we may be asked to identify those portions of the work contributed by each of us and required to demonstrate our individual knowledge of the relevant material by answering oral questions or by undertaking supplementary work, either written or in the laboratory, in order to arrive at the final assessment mark.

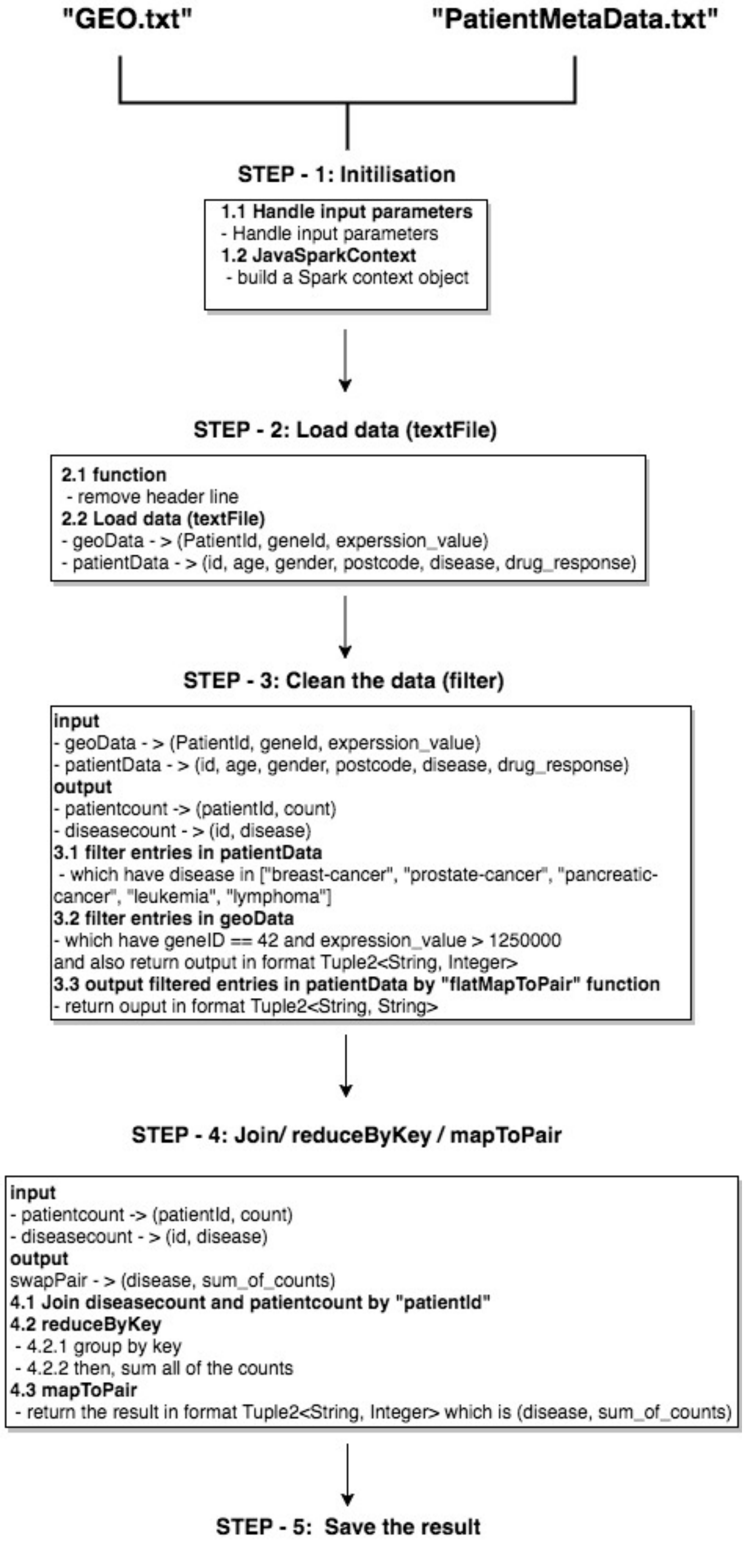
Project team members

Student name	**Student ID**	**Participated**	**Agree to share**	**Signature**
1. Shaowei Zhang	470144491	Yes / No	Yes/No	Zhang Shaowei
2. Binbin Song	450621769	Yes / No	Yes / No	Binbin Song
3.		Yes / No	Yes / No	
4.		Yes / No	Yes / No	
5.		Yes / No	Yes / No	
6.		Yes / No	Yes / No	
7.		Yes / No	Yes / No	
8.		Yes / No	Yes / No	
9.		Yes / No	Yes / No	
10.		Yes / No	Yes / No	

Cloud Computing – assignment 3

Task 1: Number of cancer patients with certain active genes per cancer type

Below flowchart describes the full design of task1.


"GEO.txt"
"PatientMetaData.txt"
STEP - 1: Initilisation
1.1 Handle input parameters
- Handle input parameters
1.2 JavaSparkContext
- build a Spark context object
STEP - 2: Load data (textFile)
2.1 function
- remove header line
2.2 Load data (textFile)
- geoData - > (PatientId, geneId, experssion_value)
- patientData - > (id, age, gender, postcode, disease, drug_response)
STEP - 3: Clean the data (filter)
input
- geoData - > (PatientId, geneId, experssion_value)
- patientData - > (id, age, gender, postcode, disease, drug_response)
output
- patientcount -> (patientId, count)
- diseasecount - > (id, disease)
3.1 filter entries in patientData
- which have disease in ["breast-cancer", "prostate-cancer", "pancreatic-cancer", "leukemia", "lymphoma"]
3.2 filter entries in geoData
- which have geneID == 42 and expression_value > 1250000
and also return output in format Tuple2<String, Integer>
3.3 output filtered entries in patientData by "flatMapToPair" function
- return ouput in format Tuple2<String, String>
STEP - 4: Join/ reduceByKey / mapToPair
input
- patientcount -> (patientId, count)
- diseasecount - > (id, disease)
output
swapPair - > (disease, sum_of_counts)
4.1 Join diseasecount and patientcount by "patientId"
4.2 reduceByKey
- 4.2.1 group by key
- 4.2.2 then, sum all of the counts
4.3 mapToPair
- return the result in format Tuple2<String, Integer> which is (disease, sum_of_counts)
STEP - 5: Save the result

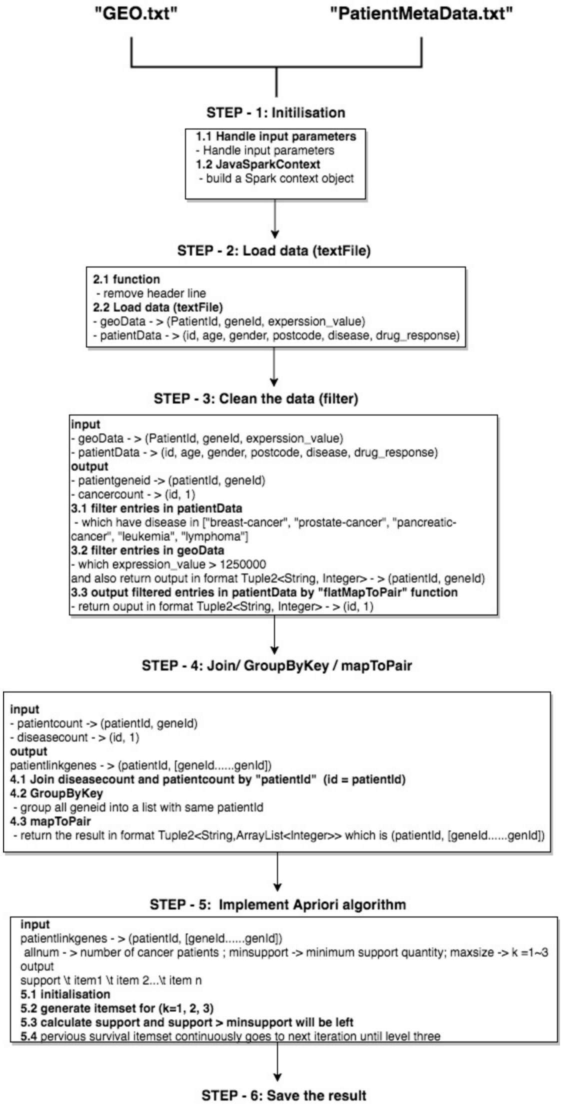
1.1 Task1 – flowchart

Task 2: Frequent Itemset Mining

The figure below indicates the design of task2. And, we also list some main User Defined Function (UDF) in the Table (Table 2.1):

Class/ UDF/ Operator	Step	Functions
ItemSetReduceFunction	5	Sum the frequency of itemset
ItemSetCalculateFrequency	5	Calculate the frequency of each itemset
ItemSetFrequencyFilterFunction	5	Filter itemset frequency > minsupport (0.3)

Table 2.1 UDF in task2



"GEO.txt"
"PatientMetaData.txt"
STEP - 1: Initilisation
1.1 Handle input parameters
- Handle input parameters
1.2 JavaSparkContext
- build a Spark context object
STEP - 2: Load data (textFile)
2.1 function
- remove header line
2.2 Load data (textFile)
- geoData - > (PatientId, geneId, experssion_value)
- patientData - > (id, age, gender, postcode, disease, drug_response)
STEP - 3: Clean the data (filter)
input
- geoData - > (PatientId, geneId, experssion_value)
- patientData - > (id, age, gender, postcode, disease, drug_response)
output
- patientgeneid -> (patientId, geneId)
- cancercount - > (id, 1)
3.1 filter entries in patientData
- which have disease in ["breast-cancer", "prostate-cancer", "pancreatic-cancer", "leukemia", "lymphoma"]
3.2 filter entries in geoData
- which expression_value > 1250000
and also return output in format Tuple2<String, Integer> - > (patientId, geneId)
3.3 output filtered entries in patientData by "flatMapToPair" function
- return ouput in format Tuple2<String, Integer> - > (id, 1)
STEP - 4: Join/ GroupByKey / mapToPair
input
- patientcount -> (patientId, geneId)
- diseasecount - > (id, 1)
output
patientlinkgenes - > (patientId, [geneId......genId])
4.1 Join diseasecount and patientcount by "patientId" (id = patientId)
4.2 GroupByKey
- group all geneid into a list with same patientId
4.3 mapToPair
- return the result in format Tuple2<String,ArrayList<Integer>> which is (patientId, [geneId......genId])
STEP - 5: Implement Apriori algorithm
input
patientlinkgenes - > (patientId, [geneId......genId])
allnum - > number of cancer patients ; minsupport -> minimum support quantity; maxsize -> k =1~3
output
support \t item1 \t item 2...\t item n
5.1 initialisation
5.2 generate itemset for (k=1, 2, 3)
5.3 calculate support and support > minsupport will be left
5.4 pervious survival itemset continuously goes to next iteration until level three
STEP - 6: Save the result

Figure 2.1 Task2 Design

Task 3: Association Rule Generation

The figure below indicates the design of task3. And, we also list some main User Defined Function (UDF) in the Table (Table 3.1):

Class/ UDF/ Operator	Step	Functions
InputtoItemset	2	Transform String into itemset
SubpatternFunction	3	break each a set of geneId down into multiple sub-itemsets
GenerateRules	3	Calculate confidence value for each sub-itemset and cross out those with low confidence (below 60%)

Table 3.1 UDF in task3

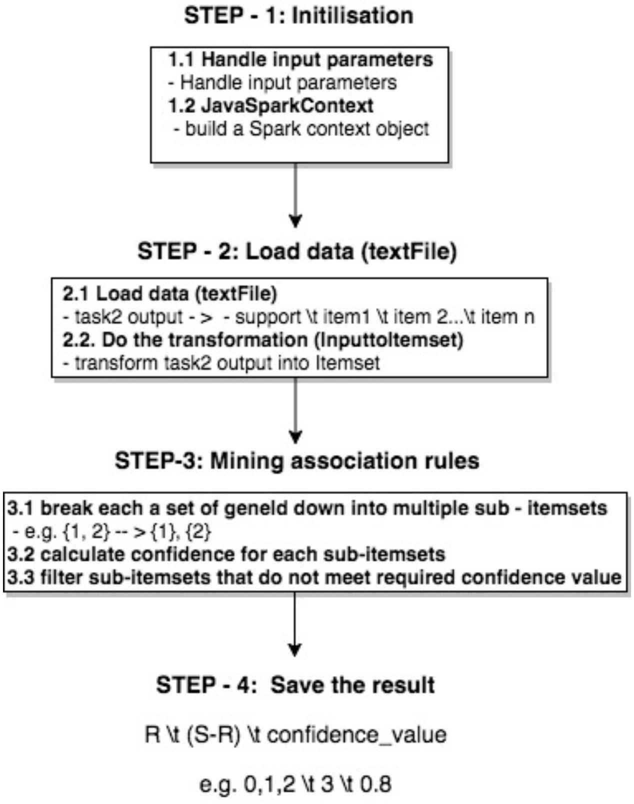

STEP - 1: Initilisation
1.1 Handle input parameters
- Handle input parameters
1.2 JavaSparkContext
- build a Spark context object
STEP - 2: Load data (textFile)
2.1 Load data (textFile)
- task2 output - > - support \t item1 \t item 2...\t item n
2.2. Do the transformation (InputtoItemset)
- transform task2 output into Itemset
STEP-3: Mining association rules
3.1 break each a set of geneId down into multiple sub - itemsets
- e.g. {1, 2} -- > {1}, {2}
3.2 calculate confidence for each sub-itemsets
3.3 filter sub-itemsets that do not meet required confidence value
STEP - 4: Save the result
R \t (S-R) \t confidence_value
e.g. 0,1,2 \t 3 \t 0.8

Figure 3.1 Task3 Design

Appendix:

Final output path (using “test” dataset)

Task1 output:

/user/szha5691/assignment3/task1

Task2 output:

/user/szha5691/assignment3/task2

Task3 output:

/user/szha5691/assignment3/task3

Attention

The “test” data is too small and the threshold (minimum support = 30% & minimum confidence = 60%) is too high, which causes limited output. So we change the threshold (minimum support = 0% & minimum confidence = 0%) to show that our program works. The low- threshold output path is showed as below:

Task1 output:

/user/szha5691/assignment3lowthre/task1

Task2 output:

/user/szha5691/assignment3lowthre/task2

Task3 output:

/user/szha5691/assignment3lowthre/task3

Execution times

The executors are marked as failed when we are trying “small” dataset and “large” dataset. We cannot get the execution time for them. Only the approximate execution time of “test” dataset is showed:

Task item	test dataset (ms)	small dataset (ms)	large dataset (ms)
Task1	10000	Unknown	Unknown
Task2	20000	Unknown	Unknown
Task3	10000	Unknown	Unknown