THE UNIVERSITY OF SYDNEY

School of Information Technologies
Faculty of Engineering & IT

ASSIGNMENT/PROJECT COVERSHEET - GROUP ASSESSMENT

Unit of Study: COMP5349

Assignment name: Assignment 1

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.

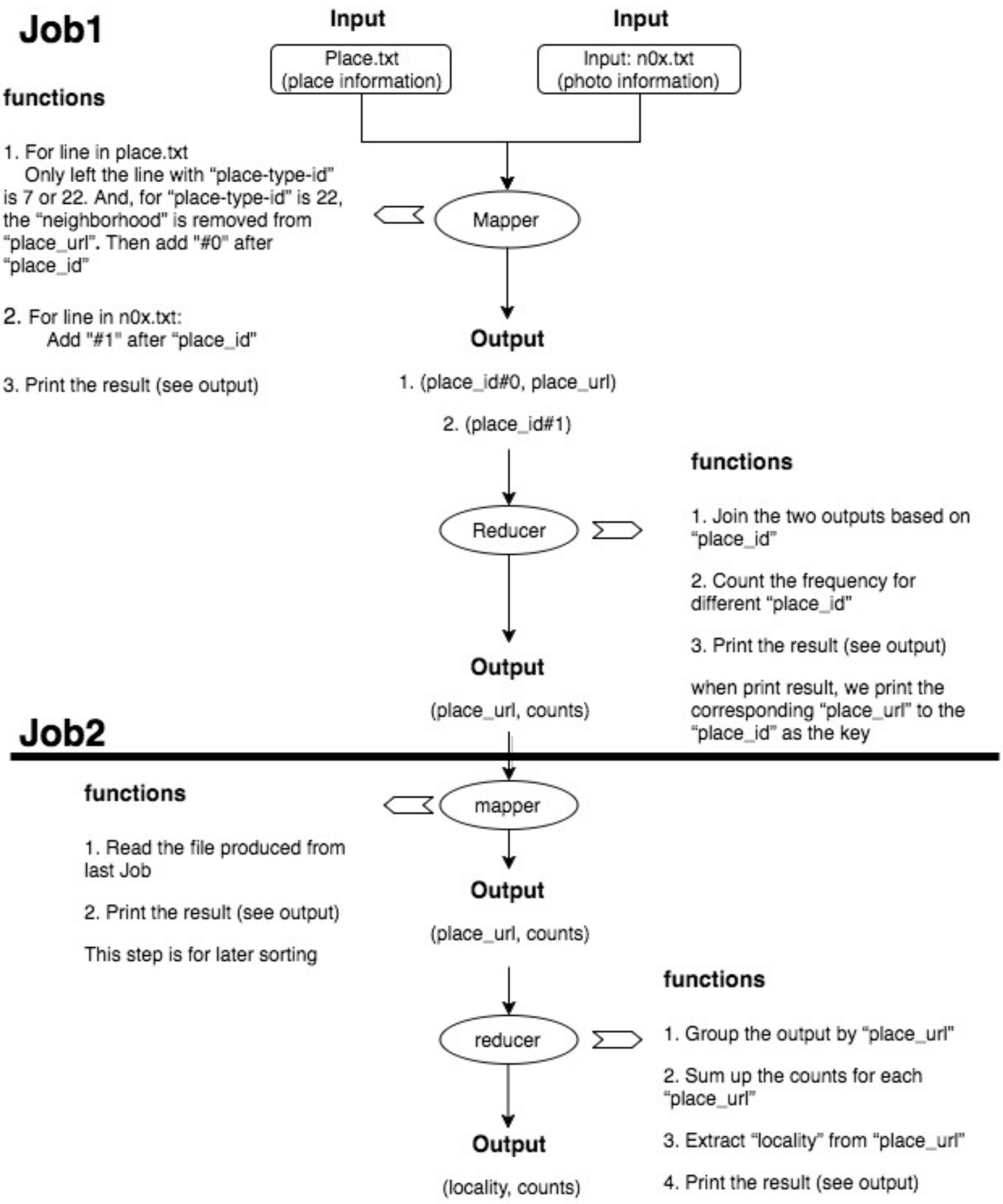
Project team members				
Student name	**Student ID**	**Participated**	**Agree to share**	**Signature**
1. Shaowei Zhang	470144491	Yes / No	Yes/No	Zhang Shaowei
2. Binbin Song	450621769	Yes / No	Yes / No	Binbin Song
3.		Yes / No	Yes / No	
4.		Yes / No	Yes / No	
5.		Yes / No	Yes / No	
6.		Yes / No	Yes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

Job Design Documentation

Task 1

- Number of Photo taken by Locality

All mappers and reducers functions are described in the flowchart below. The reason why we need the second Job is because we want to aggregate the count frequency for places with the same locality but different place_id. For example, the place with neighborhood ‘Newtown’ and the place with locality ‘Sydney’ should be regarded as the same object in this task.



functions

1. For line in place.txt
   Only left the line with “place-type-id” is 7 or 22. And, for “place-type-id” is 22, the “neighborhood” is removed from “place_url”. Then add "#0" after “place_id”
2. For line in n0x.txt:
   Add "#1" after “place_id”
3. Print the result (see output)

Output

1. (place_id#0, place_url)
2. (place_id#1)

functions

1. Join the two outputs based on “place_id”
2. Count the frequency for different “place_id”
3. Print the result (see output)

when print result, we print the corresponding “place_url” to the “place_id” as the key

Output

(place_url, counts)

functions

1. Read the file produced from last Job
2. Print the result (see output)

This step is for later sorting

Output

(place_url, counts)

functions

1. Group the output by “place_url”
2. Sum up the counts for each “place_url”
3. Extract “locality” from “place_url”
4. Print the result (see output)

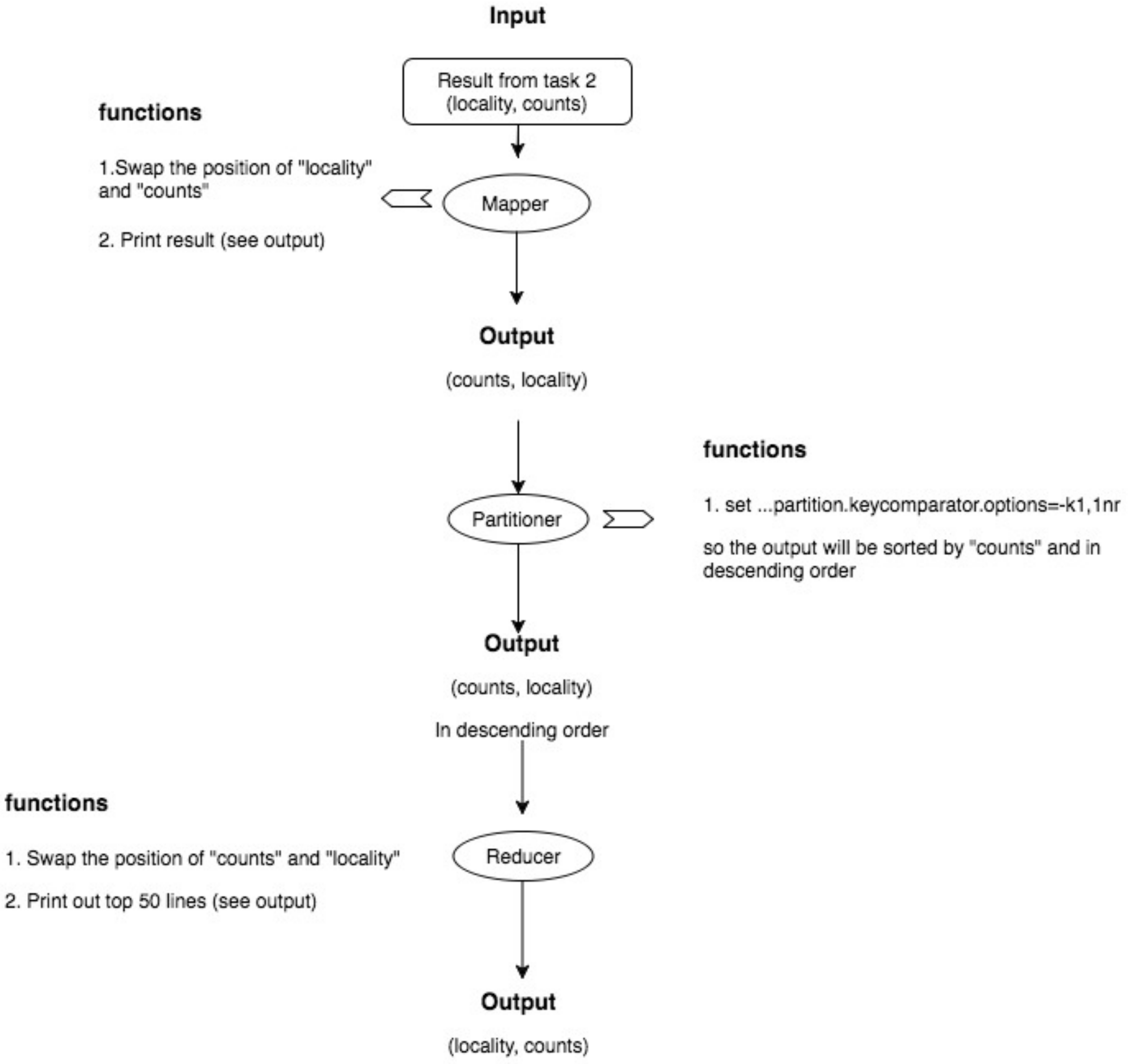
Output

(locality, counts)

Task 2

- Top 50 most popular locality based on the counts

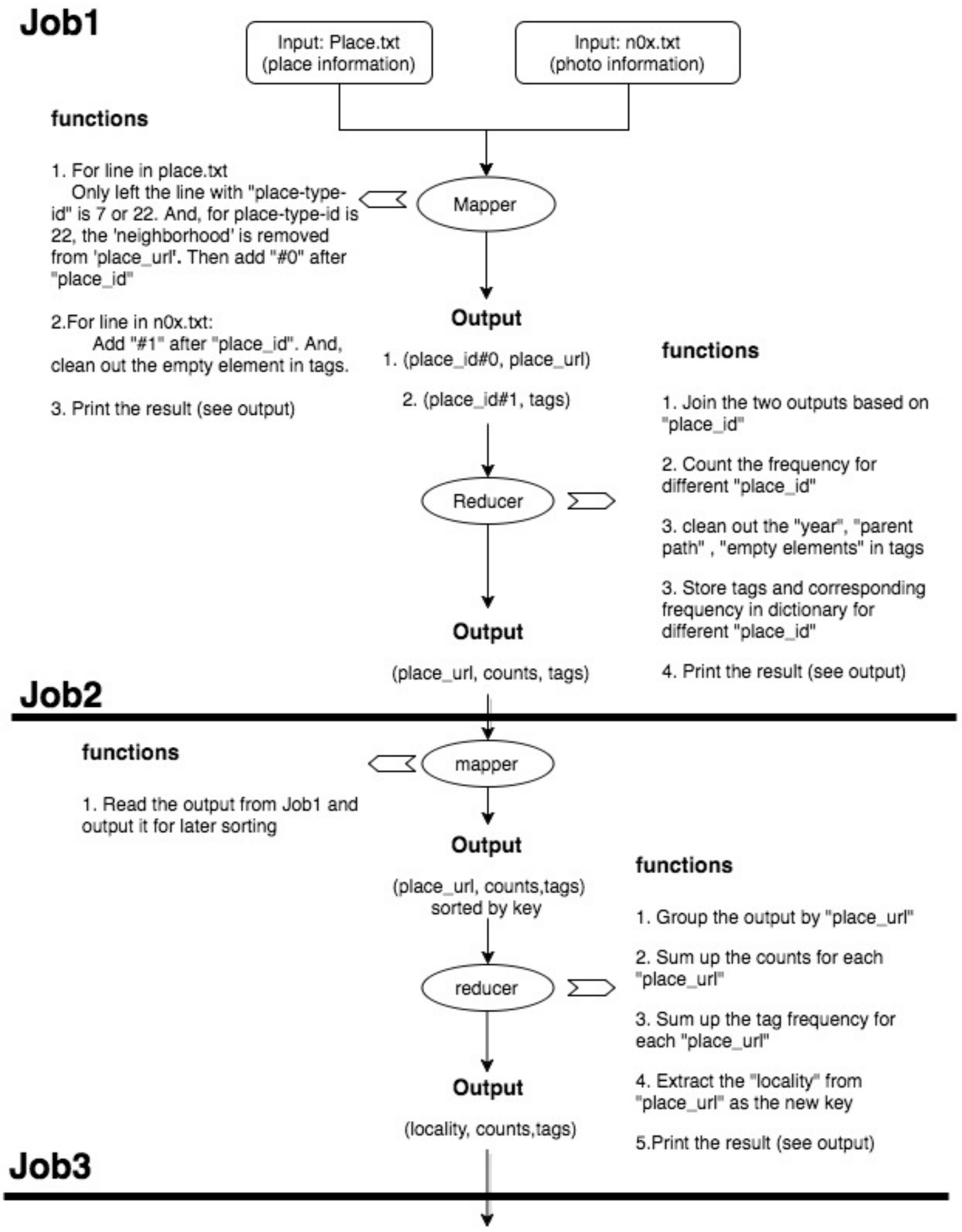
All mappers and Reducers functions are explained in the flowchart below. Our input is the result from task1. You can also think there are two jobs (one job is from task1 and the other one is job below) in task2. The key idea in task2 is to achieve the descending sort. And, we simply redefine the Partitioner. In the right side of Partitioner box below, the functions description, -n is numeric sorting, -r specifies the result should be reversed (which is in descending order).


Input
Result from task 2 (locality, counts)
functions
1.Swap the position of "locality" and "counts"
2. Print result (see output)
Mapper
Output
(counts, locality)
functions
1. set ...partition.keycomparator.options=-k1,1nr
so the output will be sorted by "counts" and in descending order
Partitioner
Output
(counts, locality)
In descending order
functions
1. Swap the position of "counts" and "locality"
2. Print out top 50 lines (see output)
Reducer
Output
(locality, counts)

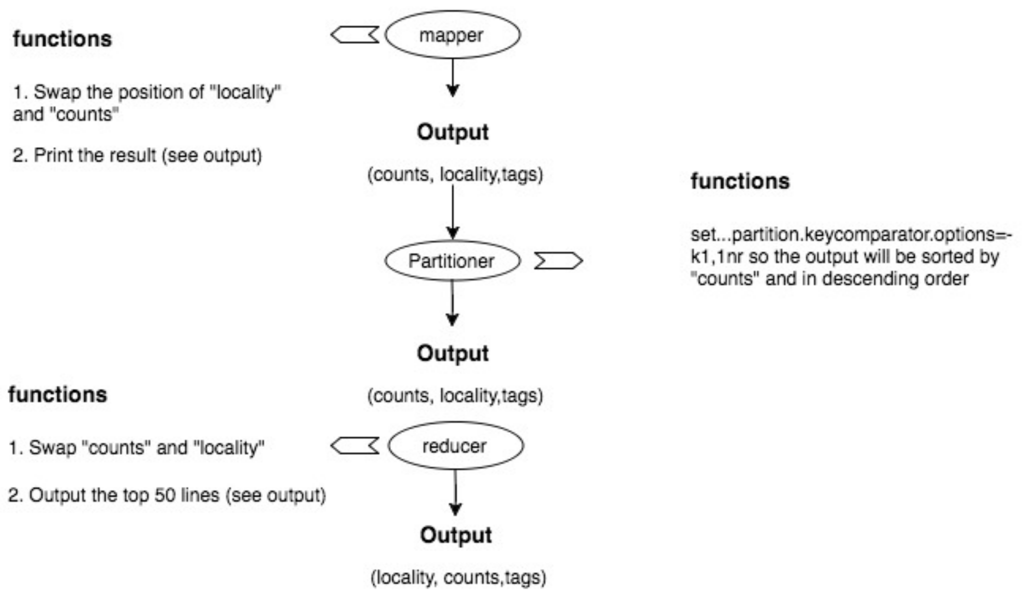
Task3

- Ten most popular tags for each Top-50 localities

All the procedure to design the program are displayed in the flowchart below. And, all mappers and reducers functions are explained below. We implement three jobs to solve the Task3. Job1 and Job2 are pretty similar with Task 1 except tags are not processed in task1. And, Job3 is similar with task2 except tags is not involved. When dealing with tags, some tags are empty which is “”, so we put a judgement statement in Job1 Mapper to drop those tags. Then in Job1 reducer, we clean out some unrelated strings like year, parent path in tags before aggregating them together.



Job1
Input: Place.txt (place information)
Input: n0x.txt (photo information)
functions
1. For line in place.txt Only left the line with "place-type-id" is 7 or 22. And, for place-type-id is 22, the 'neighborhood' is removed from 'place_url'. Then add "#0" after "place_id"
2.For line in n0x.txt: Add "#1" after "place_id". And, clean out the empty element in tags.
3. Print the result (see output)
Mapper
Output
1. (place_id#0, place_url)
2. (place_id#1, tags)
Reducer
functions
1. Join the two outputs based on "place_id"
2. Count the frequency for different "place_id"
3. clean out the "year", "parent path" , "empty elements" in tags
3. Store tags and corresponding frequency in dictionary for different "place_id"
4. Print the result (see output)
Output
(place_url, counts, tags)
Job2
functions
1. Read the output from Job1 and output it for later sorting
mapper
Output
(place_url, counts,tags) sorted by key
reducer
functions
1. Group the output by "place_url"
2. Sum up the counts for each "place_url"
3. Sum up the tag frequency for each "place_url"
4. Extract the "locality" from "place_url" as the new key
5.Print the result (see output)
Output
(locality, counts,tags)
Job3



functions

1. Swap the position of "locality" and "counts"

2. Print the result (see output)

functions

set...partition.keycomparator.options=-k1,1nr so the output will be sorted by "counts" and in descending order

functions

1. Swap "counts" and "locality"

2. Output the top 50 lines (see output)

Appendix

Task1 – output - path:

/user/szha5691/Final-1-1x

Final Output /user/szha5691/Final-1-2x

Task2 – output – path:

Final Output /user/szha5691/Final-2x

Task3 – output – path:

/user/szha5691/Final-3-2x

/user/szha5691/Final-3-1x

Final Output /user/szha5691/Final-3-3x