EE412 Foundation of Big Data Analytics, Fall 2018
HW1

Due date: 10/03/2018 (11:59pm)

Submission instructions: Use KAIST KLMS to submit your homeworks. Your submission should be one gzipped tar file whose name is `YourStudentID_hw1.tar.gz`. For example, if your student ID is 20161234, and it is for homework #1, please name the file as `20161234_hw1.tar.gz`.

Submitting writeup: Prepare answers to the homework questions into a single PDF file. You can use the following template. Please write as succinctly as possible.

Submitting code: Each problem is accompanied by a programming part. Put all the code for each question into a single file. Good coding style (including comments) will be one criterion for grading. Please make sure your code is well structured and has descriptive comments.

Ethics Oath: For every homework submission, please fill out and submit the **PDF** version of this document that pledges your honor that you did not violate any ethics rules required by this course and KAIST. You can either scan a printed version into a PDF file or make the Word document into a PDF file after filling it out. Please sign on the document and submit it along with your other files.

Discussions with other people are permitted and encouraged. However, when the time comes to write your solution, such discussions (except with course staff members) are no longer appropriate: you must write down your own solutions independently. If you received any help, you must specify on the top of your written homework any individuals from whom you received help, and the nature of the help that you received. *Do not, under any circumstances, copy another person's solution.* We check all submissions for plagiarism and take any violations seriously.

1 MapReduce and Spark (40 pts)

(a) [15 pts] Solve the following problems which are based on the exercises in the Mining of Massive Datasets 2nd edition (MMDS) textbook

- Exercise 2.2.1
  Suppose we execute the word-count MapReduce program described in this section on a large repository such as a copy of the Web. We shall use 100 Map tasks and some number of Reduce tasks.

  (a) Suppose we do not use a combiner at the Map tasks. Do you expect there to be significant skew in the times taken by the various reducers to process their value list? Why or why not?
  (b) If we combine the reducers into a small number of Reduce tasks, say 10 tasks, at random, do you expect the skew to be significant? What if we instead combine the reducers into 10,000 Reduce tasks?
  (c) Suppose we do use a combiner at the 100 Map tasks. Do you expect skew to be significant? Why or why not?

- Exercise 2.3.3 (Slightly changed)
  In the form of relational algebra implemented in SQL, relations are not sets, but bags; that is, tuples are allowed to appear more than once. There are extended definitions of union, intersection, and difference for bags, which we shall define below. Write the MapReduce algorithms (using **pseudo code**) for computing the following operations on bags R and S:

  (a) *Bag Union*, defined to be the bag of tuples in which tuple t appears the sum of the numbers of times it appears in R and S.
  (b) *Bag Intersection*, defined to be the bag of tuples in which tuple t appears the minimum of the numbers of times it appears in R and S.
  (c) *Bag Difference*, defined to be the bag of tuples in which the number of times a tuple t appears is equal to the number of times it appears in R minus the number of times it appears in S. A tuple that appears more times in S than in R does not appear in the difference.

  R and S can be considered as inputs in your pseudo code. Your answer should contain explanations of the algorithm.

- Exercise 2.4.1
  Suppose a job consists of n tasks, each of which takes time t seconds. Thus, if there are no failures, the sum over all compute nodes of the time taken to execute tasks at that node is nt. Suppose also that the probability of a task failing is p per job per second, and when a task fails, the overhead of management of the restart is such that it adds $10t$ seconds to the total execution time of the job. What is the total expected execution time of the job?

(b) [25 pts] Find potential friends in a social network using Spark

Social networks like Facebook, Twitter, and LiveJournal have users who are connected as friends in the form of a graph. In this problem, you will use a real dataset containing a LiveJournal friends graph to discover potential friends who have many mutual friends.

The dataset can be downloaded from this link:
http://www.di.kaist.ac.kr/~swhang/ee412/hw1q1.zip[1]

Each line is in the following format:

```
<User><TAB><Friends>
```

where `<User>` is an integer ID corresponding to a user, `<TAB>` is the tab character, and `<Friends>` is a comma-separated list of IDs of friends.

We are interested in finding pairs of users who are 1) not friends with each other, but 2) have many common friends. For example, if A is a friend of B, B is a friend of C, but A is not a friend of C, then we are interested in the pair (A, C), which has one common friend B.

The output should be in this format:

```
<User><TAB><User><TAB><Count>
```

where the first user ID is always smaller than the second user ID, and `<Count>` contains the number of mutual friends between the two users. The lines must be sorted by their counts in descending order. In case there are ties in counts, the lines are sorted by the first user ID integer in ascending order and then the second user ID integer in ascending order.

Here are some hints:

- The Spark Programming Guide may be useful:
  https://spark.apache.org/docs/2.2.0/rdd-programming-guide.html.
- Before running on the entire dataset, debug your code on a small test dataset.
- If you run out of memory running Spark, try adding the flag --driver-memory 8G (on a Spark cluster, --executor-memory 8G). Also consider using a machine in the Haedong lounge.
- Although this problem requires thought, the code itself does not have to be long.
- Good code style will be part of the grading, so make sure your code is well structured and add enough descriptive comments.

Please submit the following results:

[1] This dataset is from the Stanford Large Network Dataset Collection

- A short description on your algorithm for generating the output.
- The top-10 user pairs with their counts.
- Your source code in one file.

2 Frequent Itemsets (30 pts)

(a) [15 pts] Solve the following problems which are based on the exercises in the MMDS textbook

- Exercise 6.1.1 (Slightly changed)
  Suppose there are 100 kinds of items, numbered 1 to 100, and also 100 baskets, also numbered 1 to 100. Item i is in basket b if and only if i divides b with no remainder. Thus, item 1 is in all the baskets, item 2 is in all fifty of the even-numbered baskets, and so on. Basket 12 consists of items (1, 2, 3, 4, 6, 12), since these are all the integers that divide 12. Answer the following questions:

  (a) If the support threshold is 5, which items are frequent?
  (b) If the support threshold is 5, which pairs of items are frequent?
  (c) What is the sum of the sizes of all the baskets?

- Exercise 6.2.3 (Slightly changed)
  Let there be I kinds of items in a market-basket data set of B baskets. Suppose that every basket contains exactly K items. For example, if we have 2 baskets (x, y, w) and (x, y, z) where x, y, w, z are items, then I is 4, B is 2, and K is 3. As a function of I, B, and K:

  (a) How much space does the triangular-matrix method take to store the counts of all pairs of items, assuming four bytes per array element?
  (b) What is the largest possible number of pairs with a nonzero count?
  (c) Under what circumstances can we be certain that the triples method will use less space than the triangular array?

- Exercise 6.2.7
  Suppose we have market baskets that satisfy the following assumptions:
  (1) The support threshold is 10,000.
  (2) There are one million kinds of items, represented by the integers 0, 1, ..., 999999.
  (3) There are N frequent items, that is, items that occur 10,000 times or more.
  (4) There are one million pairs that occur 10,000 times or more.
  (5) There are $2M$ pairs that occur exactly once. Of these pairs, M consist of two frequent items; the other M each have at least one nonfrequent item.
  (6) No other pairs occur at all.
  (7) Integers are always represented by 4 bytes.

  Suppose we run the A-Priori Algorithm and can choose on the second pass between the triangular-matrix method for counting candidate pairs and a hash table of item-item-count triples. Neglect in the first case the space needed to

translate between original item numbers and numbers for the frequent items, and in the second case neglect the space needed for the hash table. As a function of N and M, what is the minimum number of bytes of main memory needed to execute the A-Priori Algorithm on this data?

(b) [15pts] Find frequent itemsets using the A-Priori algorithm

Suppose you are an online retailer like Amazon and want to improve the shopping experience by analyzing customer behavior. In this problem, implement the A-Priori algorithm described in Chapter 6.2.5 to find frequent items and item pairs in an online browsing dataset.

The dataset can be downloaded from this link:
http://www.di.kaist.ac.kr/~swhang/ee412/browsing.txt[2]

Each line represents a browsing session of a customer where item id strings are delimited by spaces.

To store counts of pairs, please use the triangular method (and not the triples method). Also use a support threshold of 200.

The output should contain the number of frequent items, the number of frequent pairs, and the top-10 most frequent pairs. For any ties, sort by the first id and then the second id in alphabetical order.

This problem does not require Spark programming, and your code does not have to be long. To execute your Python program, run: `python path/to/myapp.py`.

Please submit the following results:

- The number of frequent items.
- The number of frequent pairs.
- The top-10 most frequent pairs.
- Your source code in one file.

3 Finding Similar Items (30 pts)

(a) [15 pts] Solve the following exercises in the MMDS textbook

- Exercise 3.3.2
  Using the data from Fig. 3.4 in the textbook, add to the signatures of the columns the values of the following hash functions:
  (a) $h_3(x) = 2x + 4 \bmod 5$
  (b) $h_4(x) = 3x - 1 \bmod 5$

[2]This dataset is from the Stanford Large Network Dataset Collection

- Exercise 3.4.2
  For each of the (r, b) pairs in Exercise 3.4.1, compute the threshold, that is, the value of s for which the value of $1-(1-s^r)^b$ is exactly $1/2$. How does this value compare with the estimate of $(1/b)^{1/r}$ that was suggested in Section 3.4.2?

- Exercise 3.6.1
  What is the effect on probability of starting with the family of minhash functions and applying:

  (a) A 2-way AND construction followed by a 3-way OR construction.
  (b) A 3-way OR construction followed by a 2-way AND construction.
  (c) A 2-way AND construction followed by a 2-way OR construction, followed by a 2-way AND construction.
  (d) A 2-way OR construction followed by a 2-way AND construction, followed by a 2-way OR construction followed by a 2-way AND construction.

(b) [15pts] Find similar documents using minhash-based LSH

Suppose you are looking for very similar articles within a large article set. In this problem, implement the minhash-based LSH algorithm in Chapter 3.4.3 to efficiently find articles that have high Jaccard similarities.

The dataset can be downloaded from this link:
http://www.di.kaist.ac.kr/∼swhang/ee412/articles.txt[3]

Each line is in the following format:

```
<ARTICLE ID><SPACE><TEXT>
```

where `<ARTICLE ID>` is the article ID, `<SPACE>` is a single space, and `<TEXT>` is the text of the article.

When extracting k-shingles,

- Ignore non-alphabet characters
- Convert all characters to lower case
- Extract 3-shingles (i.e., set k=3)

When generating random hash functions, use the hash function $(ax+b)\%c$ as in the textbook. Let n be the number of rows as in Figure 3.4. Then set c to be the smallest prime number larger than or equal to n. Then set a to be a random integer between $[0, c-1]$ and b to be another random integer between $[0, c-1]$.

Set b and r so that the threshold is about 0.9. You can set $b = 6$ and $r = 20$, unless you prefer a different setting.

[3]This dataset is from the University of Edinburgh

The output should contain the candidate article ID pairs whose signature components agree by at least 0.9. Computing the actual Jaccard similarities is optional.

This problem does not require Spark programming, and your code does not have to be long.

Please submit the following results:

- The candidate article ID pairs.
- The runtime of your code on the dataset.
- Your source code in one file.