# IERG4300 / ESTR4300 Fall 2020 Homework 2

Release date: Oct 19, 2020

Due date: Nov 6, 2020 (Friday) 11:59pm

*The solution will be posted right after the deadline, so no late homework will be accepted!*

**Every Student MUST include the following statement, together with his/her signature in the submitted homework.**

*I declare that the assignment submitted on Elearning system is original except for source material explicitly acknowledged, and that the same or related material has not been previously submitted for another course. I also acknowledge that I am aware of University policy and regulations on honesty in academic work, and of the disciplinary guidelines and procedures applicable to breaches of such policy and regulations, as contained in the website http://www.cuhk.edu.hk/policy/academichonesty/.*

Signed (Student [illegible]) Date: 6-11-2020

Name Chan Kai Yin SID 1155124983

**Submission notice:**

- Submit your homework via the elearning system

**General homework policies:**

A student may discuss the problems with others. However, the work a student turns in must be created COMPLETELY by oneself ALONE. A student may not share ANY written work or pictures, nor may one copy answers from any source other than one's own brain.

Each student **MUST LIST** on the homework paper the **name of every person he/she has discussed or worked with**. If the answer includes content from any other source, the student **MUST STATE THE SOURCE**. Failure to do so is cheating and will result in sanctions. Copying answers from someone else is cheating even if one lists their name(s) on the homework.

If there is information you need to solve a problem but the information is not stated in the problem, try to find the data somewhere. If you cannot find it, state what data you need, make a reasonable estimate of its value, and justify any assumptions you make. You will be graded not only on whether your answer is correct, but also on whether you have done an intelligent analysis.

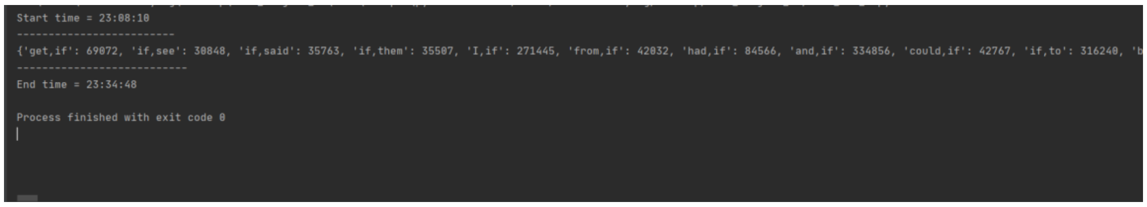
Q1.

a) False, cause even there are some false positives in the first or second map reduce, those false positives cases will be filtered in the last map-reduce job.

b) False, cause when a file gets cut into several chunk, there will be at least one chunk that the frequent item pairs will pass the threshold, so it will get into the list and will be checked in later map-reduce job.

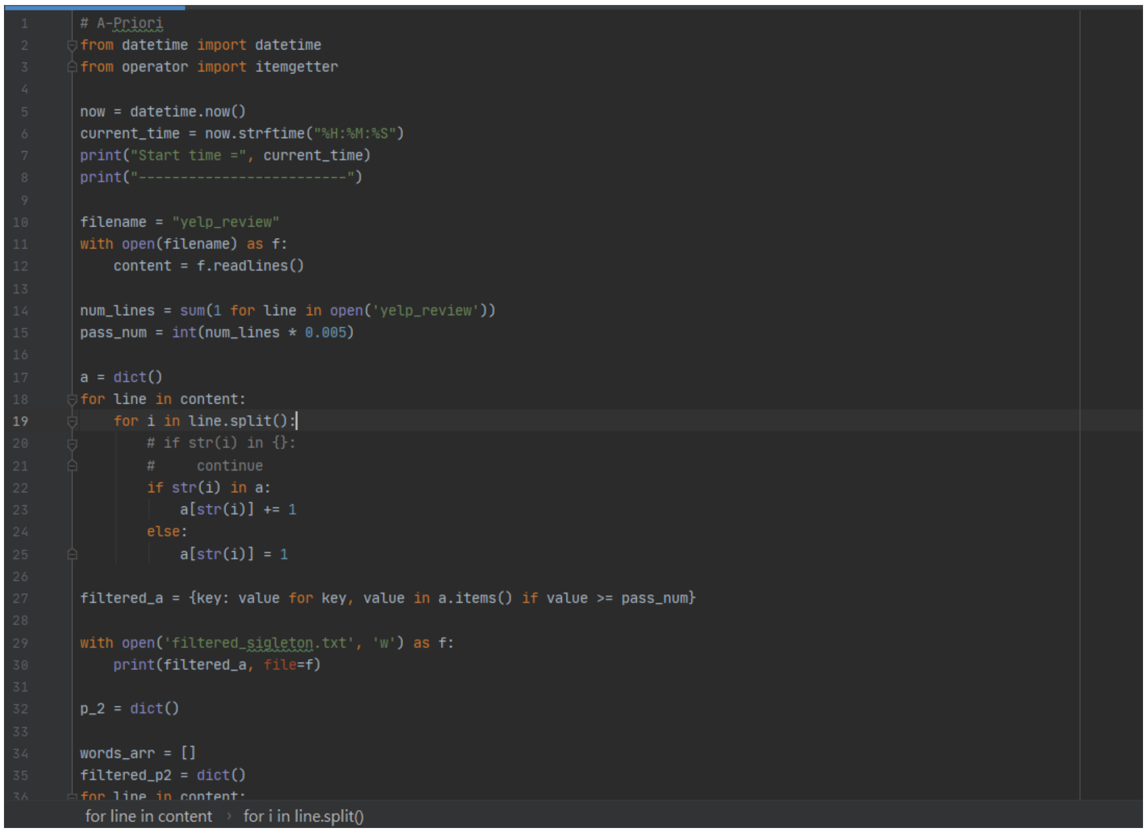
Q2.

a)

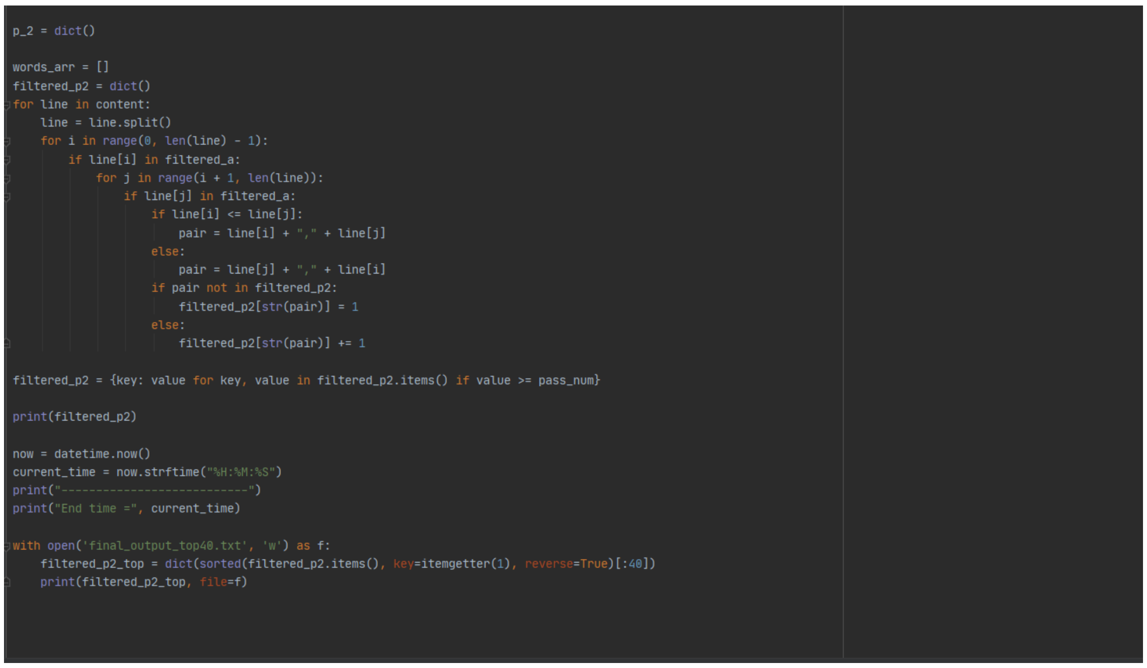


```
Start time = 23:08:10
------------------------
{'get,if': 69072, 'if,see': 30848, 'if,said': 35763, 'if,them': 35507, 'I,if': 271445, 'from,if': 42032, 'had,if': 84566, 'and,if': 334856, 'could,if': 42767, 'if,to': 316240, 'b
--------------------------
End time = 23:34:48

Process finished with exit code 0
```

The A-priori takes about 26 mins to run. And I print out the top 40 pairs for reference.



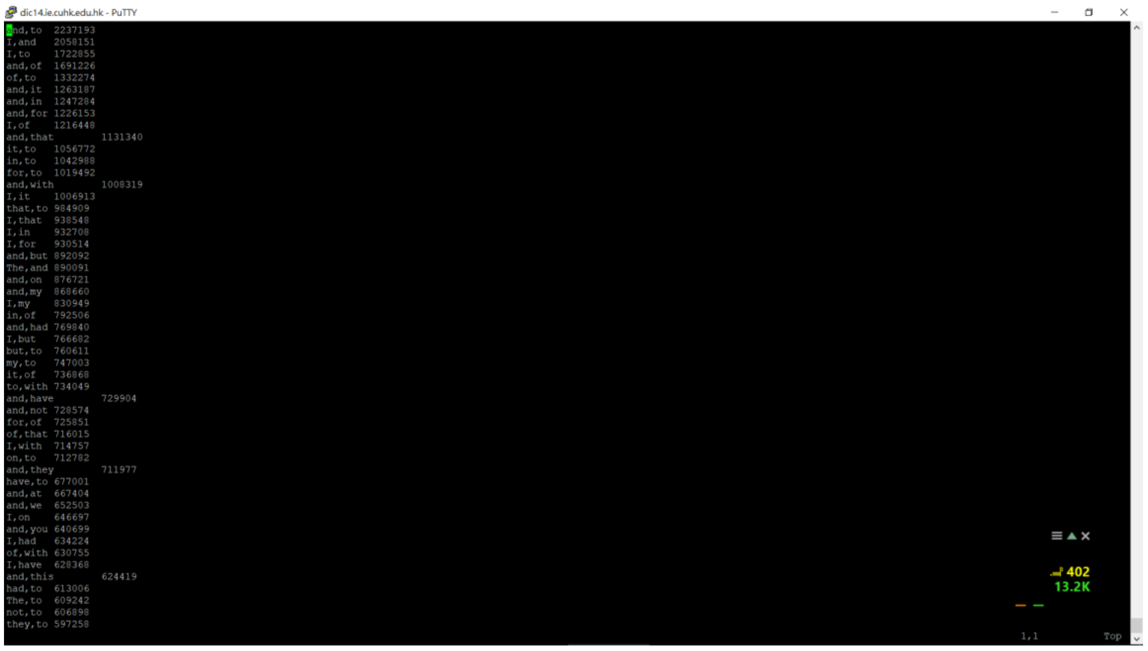
```python
# A-Priori
from datetime import datetime
from operator import itemgetter

now = datetime.now()
current_time = now.strftime("%H:%M:%S")
print("Start time =", current_time)
print("------------------------")

filename = "yelp_review"
with open(filename) as f:
    content = f.readlines()

num_lines = sum(1 for line in open('yelp_review'))
pass_num = int(num_lines * 0.005)

a = dict()
for line in content:
    for i in line.split():
        # if str(i) in {}:
        #     continue
        if str(i) in a:
            a[str(i)] += 1
        else:
            a[str(i)] = 1

filtered_a = {key: value for key, value in a.items() if value >= pass_num}

with open('filtered_sigleton.txt', 'w') as f:
    print(filtered_a, file=f)

p_2 = dict()

words_arr = []
filtered_p2 = dict()
for line in content:
```



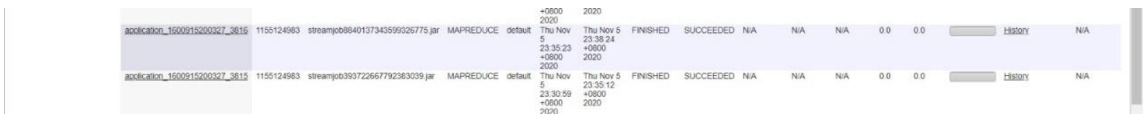
```python
p_2 = dict()

words_arr = []
filtered_p2 = dict()
for line in content:
    line = line.split()
    for i in range(0, len(line) - 1):
        if line[i] in filtered_a:
            for j in range(i + 1, len(line)):
                if line[j] in filtered_a:
                    if line[i] <= line[j]:
                        pair = line[i] + "," + line[j]
                    else:
                        pair = line[j] + "," + line[i]
                    if pair not in filtered_p2:
                        filtered_p2[str(pair)] = 1
                    else:
                        filtered_p2[str(pair)] += 1

filtered_p2 = {key: value for key, value in filtered_p2.items() if value >= pass_num}

print(filtered_p2)

now = datetime.now()
current_time = now.strftime("%H:%M:%S")
print("----------------------------")
print("End time =", current_time)

with open('final_output_top40.txt', 'w') as f:
    filtered_p2_top = dict(sorted(filtered_p2.items(), key=itemgetter(1), reverse=True)[:40])
    print(filtered_p2_top, file=f)
```

Top40:

'and,to': 2237193, 'I,and': 2058151, 'I,to': 1722855, 'and,of': 1691226, 'of,to': 1332274, 'and,it': 1263187, 'and,in': 1247284, 'and,for': 1226153, 'I,of': 1216448, 'and,that': 1131340, 'it,to': 1056772, 'in,to': 1042988, 'for,to': 1019492, 'and,with': 1008319, 'I,it': 1006913, 'that,to': 984909, 'I,that': 938548, 'I,in': 932708, 'I,for': 930514, 'and,but': 892092, 'The,and': 890091, 'and,on': 876721, 'and,my': 868660, 'I,my': 830949, 'in,of': 792506, 'and,had': 769840, 'I,but': 766682, 'but,to': 760611, 'my,to': 747003, 'it,of': 736868, 'to,with': 734049, 'and,have': 729904, 'and,not': 728574, 'for,of': 725851, 'of,that': 716015, 'I,with': 714757, 'on,to': 712782, 'and,they': 711977, 'have,to': 677001, 'and,at': 667404 b.)


dic14.ie.cuhk.edu.hk - PuTTY

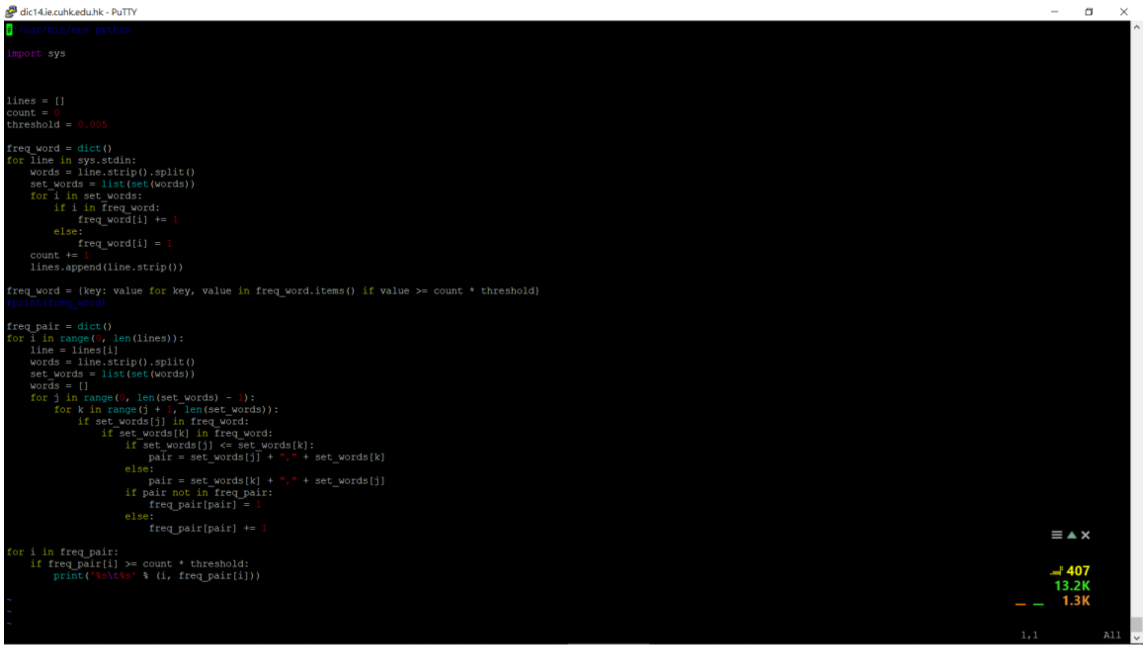
```
and,to  2237193
I,and   2058151
I,to    1722855
and,of  1691226
of,to   1332274
and,it  1263187
and,in  1247284
and,for 1226153
I,of    1216448
and,that        1131340
it,to   1056772
in,to   1042988
for,to  1019492
and,with        1008319
I,it    1006913
that,to 984909
I,that  938548
I,in    932708
I,for   930514
and,but 892092
The,and 890091
and,on  876721
and,my  868660
I,my    830949
in,of   792506
and,had 769840
I,but   766682
but,to  760611
my,to   747003
it,of   736868
to,with 734049
and,have        729904
and,not 728574
for,of  725851
of,that 716015
I,with  714757
on,to   712782
and,they        711977
have,to 677001
and,at  667404
and,we  652503
I,on    646697
and,you 640699
I,had   634224
of,with 630755
I,have  628368
and,this        624419
had,to  613006
The,to  609242
not,to  606898
they,to 597258
                                                                1,1           Top
```



| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | +0800 2020 | 2020 | | | | | | | | | | |
| application_1600915200327_3816 | 1155124983 | streamjob8840137343599326775.jar | MAPREDUCE | default | Thu Nov 5 23:35:23 +0800 2020 | Thu Nov 5 23:38:24 +0800 2020 | FINISHED | SUCCEEDED | N/A | N/A | N/A | 0.0 | 0.0 | | History | N/A |
| application_1600915200327_3815 | 1155124983 | streamjob3937226677923830399.jar | MAPREDUCE | default | Thu Nov 5 23:30:59 +0800 2020 | Thu Nov 5 23:35:12 +0800 2020 | FINISHED | SUCCEEDED | N/A | N/A | N/A | 0.0 | 0.0 | | History | N/A |

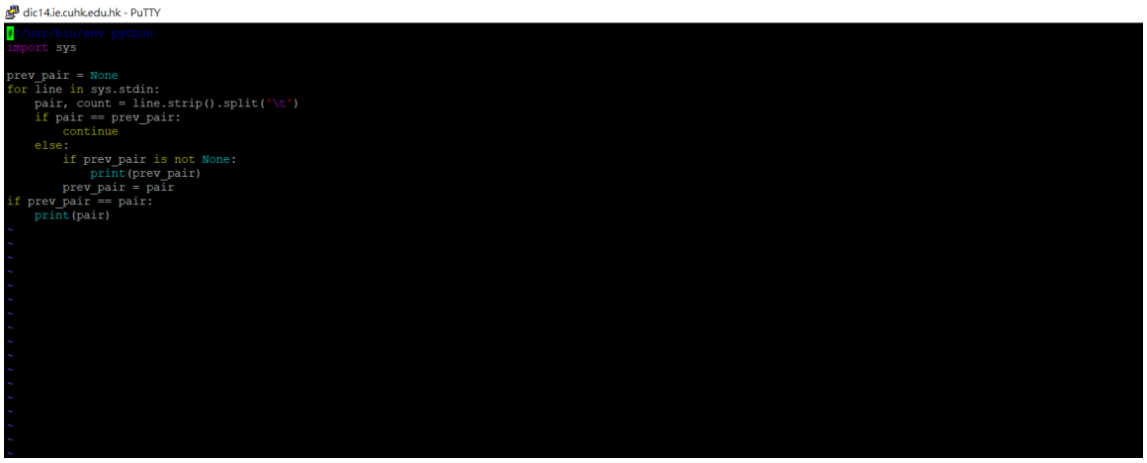
It takes about 8 min to run the two map-reduce job.

The first Mapper:


dic14.ie.cuhk.edu.hk - PuTTY

```python
#!/usr/bin/env python

import sys



lines = []
count = 0
threshold = 0.005

freq_word = dict()
for line in sys.stdin:
    words = line.strip().split()
    set_words = list(set(words))
    for i in set_words:
        if i in freq_word:
            freq_word[i] += 1
        else:
            freq_word[i] = 1
    count += 1
    lines.append(line.strip())

freq_word = {key: value for key, value in freq_word.items() if value >= count * threshold}
#print(freq_word)

freq_pair = dict()
for i in range(0, len(lines)):
    line = lines[i]
    words = line.strip().split()
    set_words = list(set(words))
    words = []
    for j in range(0, len(set_words) - 1):
        for k in range(j + 1, len(set_words)):
            if set_words[j] in freq_word:
                if set_words[k] in freq_word:
                    if set_words[j] <= set_words[k]:
                        pair = set_words[j] + "," + set_words[k]
                    else:
                        pair = set_words[k] + "," + set_words[j]
                    if pair not in freq_pair:
                        freq_pair[pair] = 1
                    else:
                        freq_pair[pair] += 1

for i in freq_pair:
    if freq_pair[i] >= count * threshold:
        print('%s\t%s' % (i, freq_pair[i]))

~
~
~
                                                                1,1           All
```

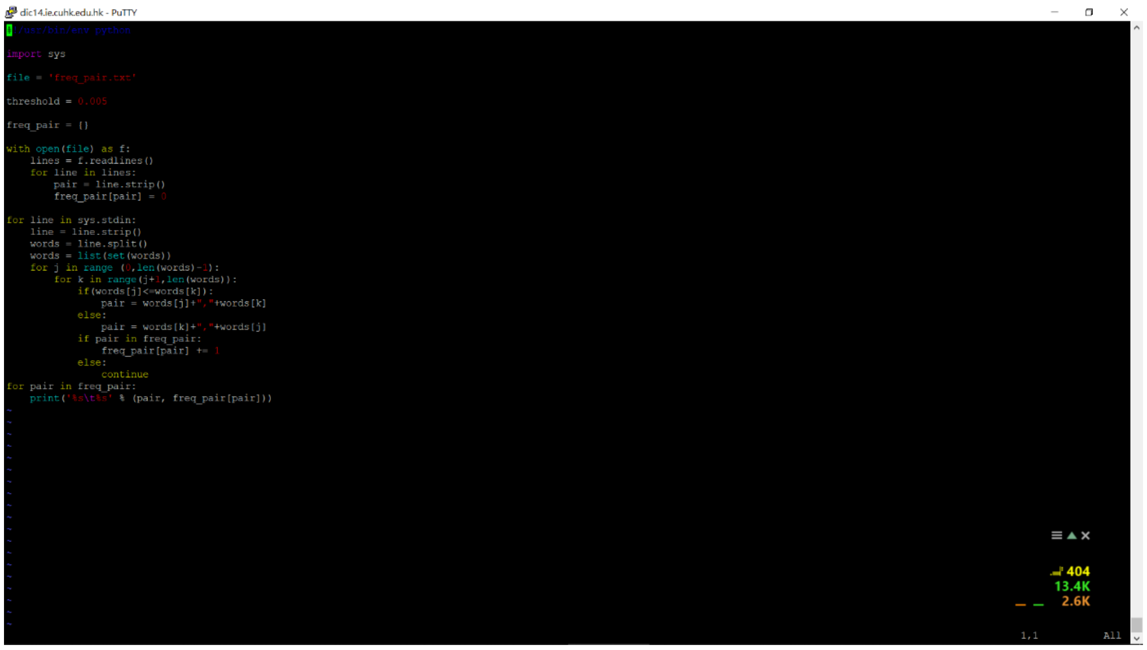
The first reducer:



dic14.ie.cuhk.edu.hk - PuTTY

```
#!/usr/bin/env python
import sys

prev_pair = None
for line in sys.stdin:
    pair, count = line.strip().split('\t')
    if pair == prev_pair:
        continue
    else:
        if prev_pair is not None:
            print(prev_pair)
        prev_pair = pair
if prev_pair == pair:
    print(pair)
~
```

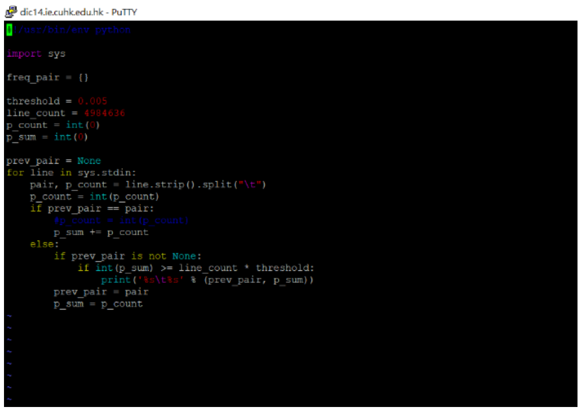
The second Mapper:



dic14.ie.cuhk.edu.hk - PuTTY

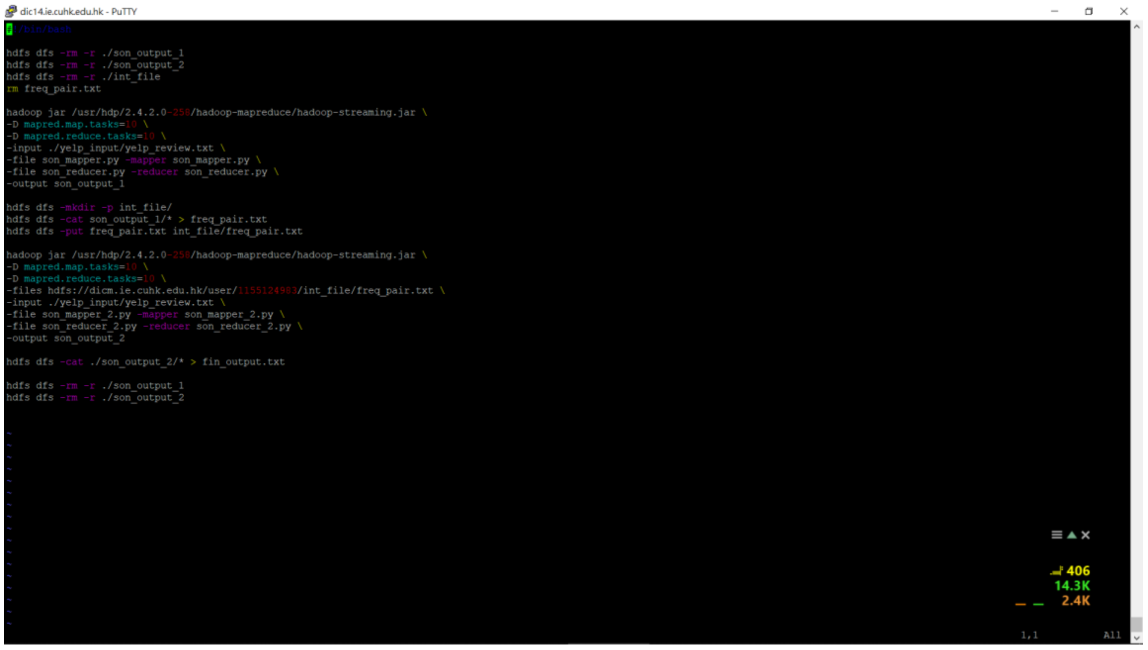
```
#!/usr/bin/env python

import sys

file = 'freq_pair.txt'

threshold = 0.005

freq_pair = {}

with open(file) as f:
    lines = f.readlines()
    for line in lines:
        pair = line.strip()
        freq_pair[pair] = 0

for line in sys.stdin:
    line = line.strip()
    words = line.split()
    words = list(set(words))
    for j in range (0,len(words)-1):
        for k in range(j+1,len(words)):
            if(words[j]<=words[k]):
                pair = words[j]+","+words[k]
            else:
                pair = words[k]+","+words[j]
            if pair in freq_pair:
                freq_pair[pair] += 1
            else:
                continue
for pair in freq_pair:
    print('%s\t%s' % (pair, freq_pair[pair]))
~
                                                                    1,1           All
```

The second reducer:



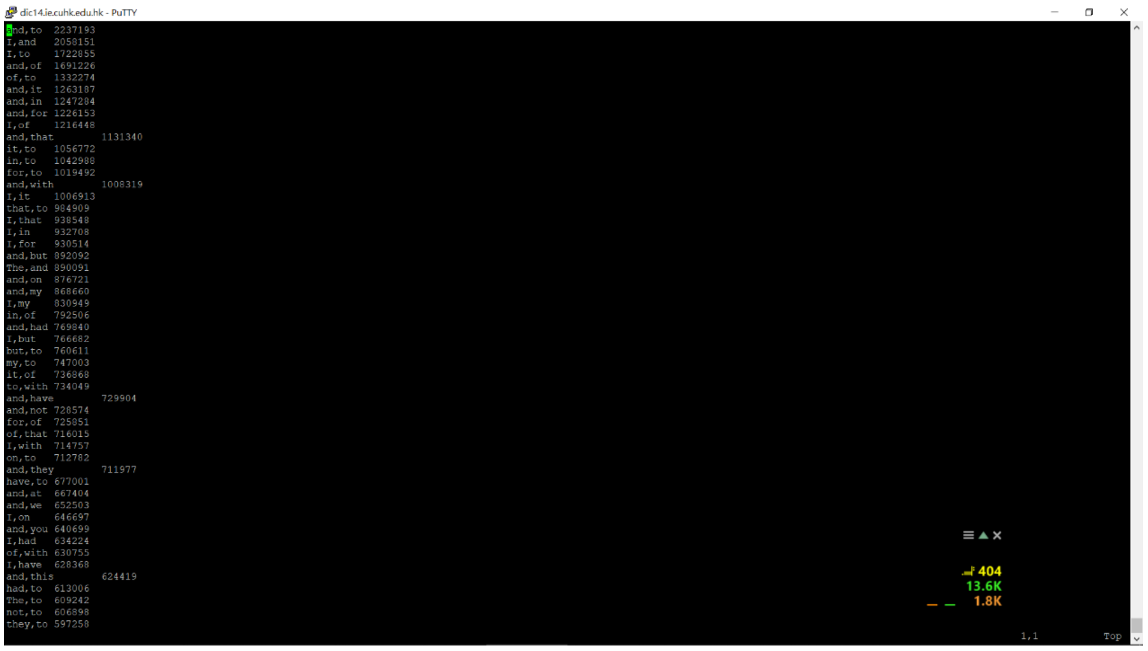
```
#!/usr/bin/env python

import sys

freq_pair = {}

threshold = 0.005
line_count = 4984636
p_count = int(0)
p_sum = int(0)

prev_pair = None
for line in sys.stdin:
    pair, p_count = line.strip().split("\t")
    p_count = int(p_count)
    if prev_pair == pair:
        #p_count = int(p_count)
        p_sum += p_count
    else:
        if prev_pair is not None:
            if int(p_sum) >= line_count * threshold:
                print('%s\t%s' % (prev_pair, p_sum))
        prev_pair = pair
        p_sum = p_count
~
~
~
~
~
~
~
~
```

The .sh file:

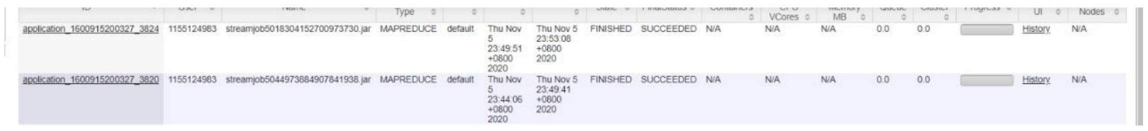


```
#!/bin/bash

hdfs dfs -rm -r ./son_output_1
hdfs dfs -rm -r ./son_output_2
hdfs dfs -rm -r ./int_file
rm freq_pair.txt

hadoop jar /usr/hdp/2.4.2.0-258/hadoop-mapreduce/hadoop-streaming.jar \
-D mapred.map.tasks=10 \
-D mapred.reduce.tasks=10 \
-input ./yelp_input/yelp_review.txt \
-file son_mapper.py -mapper son_mapper.py \
-file son_reducer.py -reducer son_reducer.py \
-output son_output_1

hdfs dfs -mkdir -p int_file/
hdfs dfs -cat son_output_1/* > freq_pair.txt
hdfs dfs -put freq_pair.txt int_file/freq_pair.txt

hadoop jar /usr/hdp/2.4.2.0-258/hadoop-mapreduce/hadoop-streaming.jar \
-D mapred.map.tasks=10 \
-D mapred.reduce.tasks=10 \
-files hdfs://dicm.ie.cuhk.edu.hk/user/1155124983/int_file/freq_pair.txt \
-input ./yelp_input/yelp_review.txt \
-file son_mapper_2.py -mapper son_mapper_2.py \
-file son_reducer_2.py -reducer son_reducer_2.py \
-output son_output_2

hdfs dfs -cat ./son_output_2/* > fin_output.txt

hdfs dfs -rm -r ./son_output_1
hdfs dfs -rm -r ./son_output_2

~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
                                                                    1,1           All
```

c.)


dic14.ie.cuhk.edu.hk - PuTTY

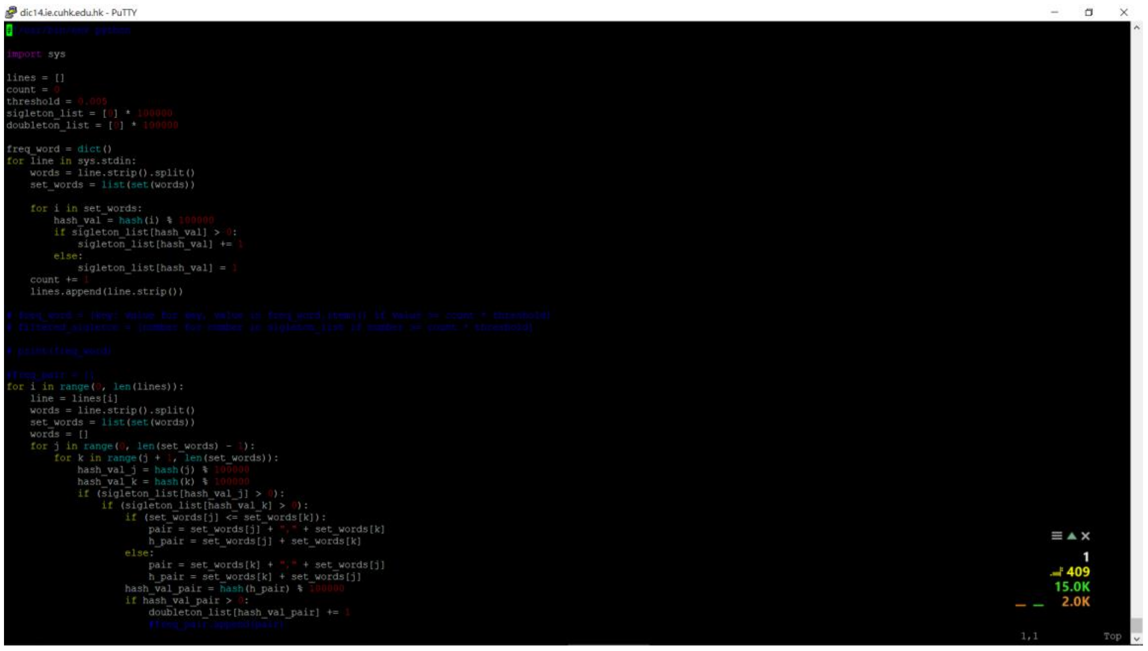
```
and,to  2237193
I,and   2058151
I,to    1722855
and,of  1691226
of,to   1332274
and,it  1263187
and,in  1247284
and,for 1226153
I,of    1216448
and,that        1131340
it,to   1056772
in,to   1042988
for,to  1019492
and,with        1008319
I,it    1006913
that,to 984909
I,that  938548
I,in    932708
I,for   930514
and,but 892092
The,and 890091
and,on  876721
and,my  868660
I,my    830949
in,of   792506
and,had 769840
I,but   766682
but,to  760611
my,to   747003
it,of   736868
to,with 734049
and,have        729904
and,not 728574
for,of  725851
of,that 716015
I,with  714757
on,to   712782
and,they        711977
have,to 677001
and,at  667404
and,we  652503
I,on    646697
and,you 640699
I,had   634224
of,with 630755
I,have  628368
and,this        624419
had,to  613006
The,to  609242
not,to  606898
they,to 597258
                                                                 1,1           Top
```



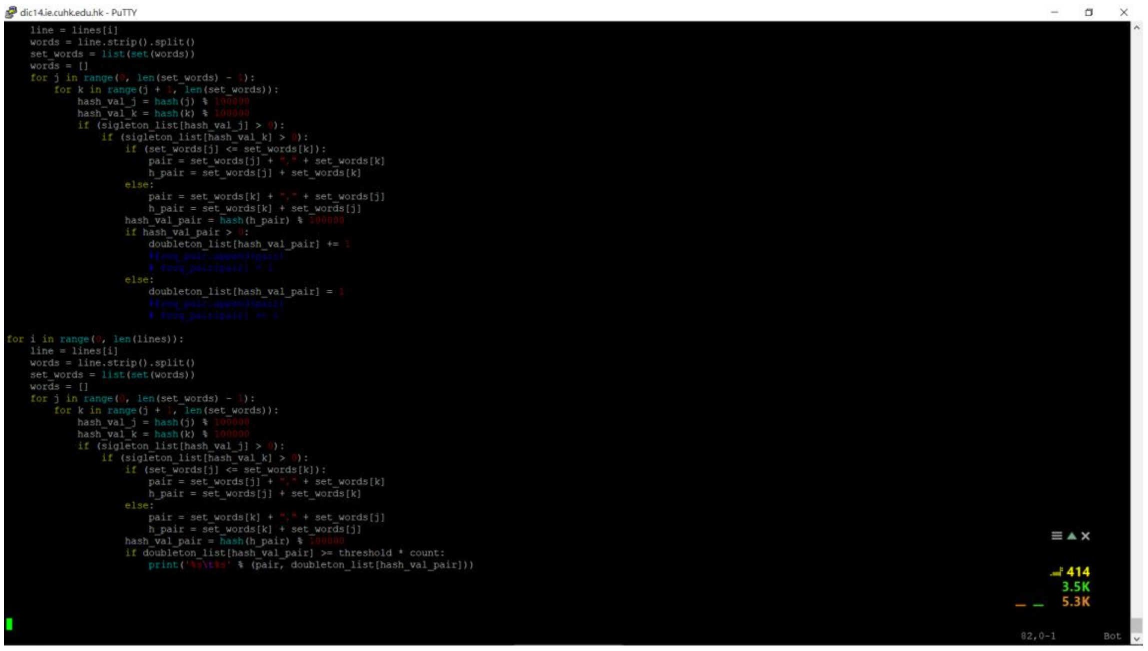
| | | | Type | | | | | | | | VCores | MB | | | | UI | Nodes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| application_1600915200327_3824 | 1155124983 | streamjob5018304152700973730.jar | MAPREDUCE | default | Thu Nov 5 23:49:51 +0800 2020 | Thu Nov 5 23:53:08 +0800 2020 | FINISHED | SUCCEEDED | N/A | | N/A | N/A | 0.0 | 0.0 | | History | N/A |
| application_1600915200327_3820 | 1155124983 | streamjob5044973884907841938.jar | MAPREDUCE | default | Thu Nov 5 23:44:06 +0800 2020 | Thu Nov 5 23:49:41 +0800 2020 | FINISHED | SUCCEEDED | N/A | | N/A | N/A | 0.0 | 0.0 | | History | N/A |

It takes about 9 mins to run the two map-reduce job.

First Mapper for PCY-version:


dic14.ie.cuhk.edu.hk - PuTTY

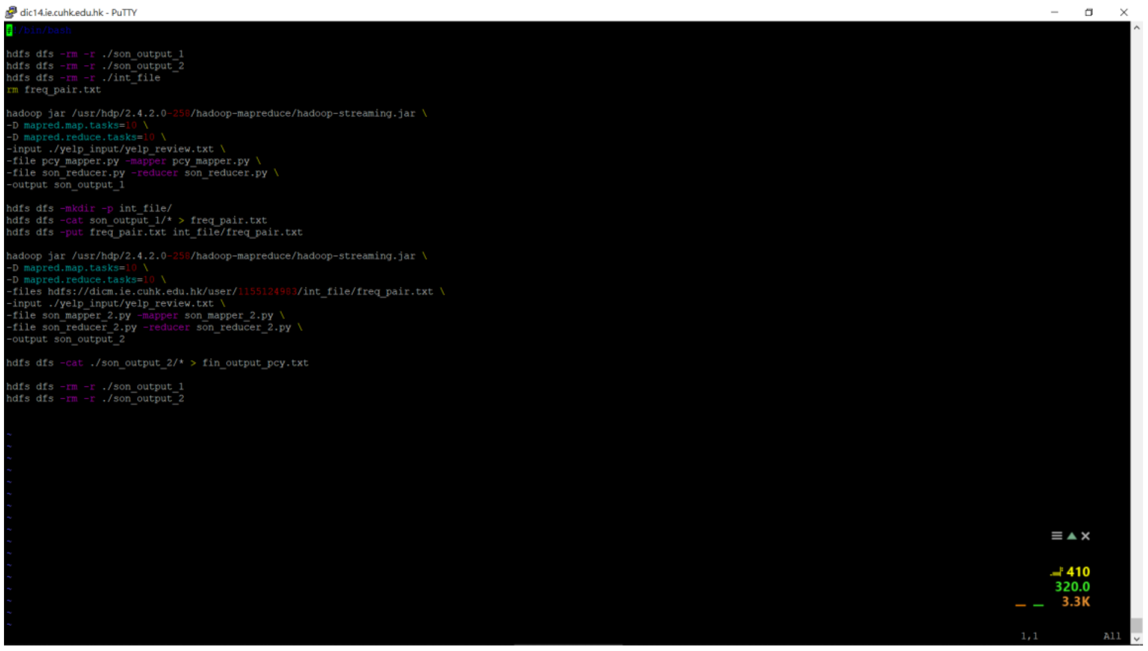
```python
#!/usr/bin/env python

import sys

lines = []
count = 0
threshold = 0.005
sigleton_list = [0] * 100000
doubleton_list = [0] * 100000

freq_word = dict()
for line in sys.stdin:
    words = line.strip().split()
    set_words = list(set(words))

    for i in set_words:
        hash_val = hash(i) % 100000
        if sigleton_list[hash_val] > 0:
            sigleton_list[hash_val] += 1
        else:
            sigleton_list[hash_val] = 1
    count += 1
    lines.append(line.strip())

# freq_word = {key: value for key, value in freq_word.items() if value >= count * threshold}
# filtered_sigleton = [number for number in sigleton_list if number >= count * threshold]

# print(freq_word)

#freq_pair = []
for i in range(0, len(lines)):
    line = lines[i]
    words = line.strip().split()
    set_words = list(set(words))
    words = []
    for j in range(0, len(set_words) - 1):
        for k in range(j + 1, len(set_words)):
            hash_val_j = hash(j) % 100000
            hash_val_k = hash(k) % 100000
            if (sigleton_list[hash_val_j] > 0):
                if (sigleton_list[hash_val_k] > 0):
                    if (set_words[j] <= set_words[k]):
                        pair = set_words[j] + "," + set_words[k]
                        h_pair = set_words[j] + set_words[k]
                    else:
                        pair = set_words[k] + "," + set_words[j]
                        h_pair = set_words[k] + set_words[j]
                    hash_val_pair = hash(h_pair) % 100000
                    if hash_val_pair > 0:
                        doubleton_list[hash_val_pair] += 1
                        #freq_pair.append(pair)
                                                                 1,1           Top
```



```
    line = lines[i]
    words = line.strip().split()
    set_words = list(set(words))
    words = []
    for j in range(0, len(set_words) - 1):
        for k in range(j + 1, len(set_words)):
            hash_val_j = hash(j) % 100000
            hash_val_k = hash(k) % 100000
            if (sigleton_list[hash_val_j] > 0):
                if (sigleton_list[hash_val_k] > 0):
                    if (set_words[j] <= set_words[k]):
                        pair = set_words[j] + "," + set_words[k]
                        h_pair = set_words[j] + set_words[k]
                    else:
                        pair = set_words[k] + "," + set_words[j]
                        h_pair = set_words[k] + set_words[j]
                    hash_val_pair = hash(h_pair) % 100000
                    if hash_val_pair > 0:
                        doubleton_list[hash_val_pair] += 1
                        #freq_pair.append(pair)
                        # freq_pair[pair] = 1
                    else:
                        doubleton_list[hash_val_pair] = 1
                        #freq_pair.append(pair)
                        # freq_pair[pair] += 1

for i in range(0, len(lines)):
    line = lines[i]
    words = line.strip().split()
    set_words = list(set(words))
    words = []
    for j in range(0, len(set_words) - 1):
        for k in range(j + 1, len(set_words)):
            hash_val_j = hash(j) % 100000
            hash_val_k = hash(k) % 100000
            if (sigleton_list[hash_val_j] > 0):
                if (sigleton_list[hash_val_k] > 0):
                    if (set_words[j] <= set_words[k]):
                        pair = set_words[j] + "," + set_words[k]
                        h_pair = set_words[j] + set_words[k]
                    else:
                        pair = set_words[k] + "," + set_words[j]
                        h_pair = set_words[k] + set_words[j]
                    hash_val_pair = hash(h_pair) % 100000
                    if doubleton_list[hash_val_pair] >= threshold * count:
                        print('%s\t%s' % (pair, doubleton_list[hash_val_pair]))

                                                                  82,0-1        Bot
```

The reducer and second map-reduce program as the same as part b.

The .sh file:



```
#!/bin/bash

hdfs dfs -rm -r ./son_output_1
hdfs dfs -rm -r ./son_output_2
hdfs dfs -rm -r ./int_file
rm freq_pair.txt

hadoop jar /usr/hdp/2.4.2.0-258/hadoop-mapreduce/hadoop-streaming.jar \
-D mapred.map.tasks=10 \
-D mapred.reduce.tasks=10 \
-input ./yelp_input/yelp_review.txt \
-file pcy_mapper.py -mapper pcy_mapper.py \
-file son_reducer.py -reducer son_reducer.py \
-output son_output_1

hdfs dfs -mkdir -p int_file/
hdfs dfs -cat son_output_1/* > freq_pair.txt
hdfs dfs -put freq_pair.txt int_file/freq_pair.txt

hadoop jar /usr/hdp/2.4.2.0-258/hadoop-mapreduce/hadoop-streaming.jar \
-D mapred.map.tasks=10 \
-D mapred.reduce.tasks=10 \
-files hdfs://dicm.ie.cuhk.edu.hk/user/1155124983/int_file/freq_pair.txt \
-input ./yelp_input/yelp_review.txt \
-file son_mapper_2.py -mapper son_mapper_2.py \
-file son_reducer_2.py -reducer son_reducer_2.py \
-output son_output_2

hdfs dfs -cat ./son_output_2/* > fin_output_pcy.txt

hdfs dfs -rm -r ./son_output_1
hdfs dfs -rm -r ./son_output_2
```

As a conclusion, the son algorithm is the fastest. The second fast is the PCY algorithm. The third one is A-Priori because it runs on a single machine. The son algorithm and PCY algorithm are using map-reduce program, having similar running speed.

Q3.

a) Jaccard similarities for

(x,y) = 0.2

(x,z) = 0.5

(y,z) = 0.5

b)

using abcde as indication:

| | | |
|---|---|---|
| a | b | a |
| b | a | a |
| a | b | b |
| b | a | a |
| a | a | b |

using 12345 for (abcde) as indication:

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 1 | 1 |

|  |  |  |
|---|---|---|
| 1 | 2 | 2 |
| 2 | 1 | 1 |
| 1 | 1 | 2 |

c.) Jaccard similarities according to signatures:

(x,y) = 1/5

(x,z) = 1/5

(y,z) = 3/5

d.) ((1-s)^r))^b