MIDS-W261-2016-HWK-Assignment3-Chandaria

February 3, 2016

=====DATSCIW261 ASSIGNMENT #3=====
MIDS UC Berkeley, Machine Learning at Scale
DATSCIW261 ASSIGNMENT #3
Version 2016-01-27 (FINAL)

Hetal Chandaria (hetalchandaria@berkeley.edu) & Patrick NG (patng@ischool.berkeley.edu)
W261 - 2 , ASSIGNMENT #3
Submission Date : 02/03/2016
Group : 4

HW3.0. What is a merge sort? Where is it used in Hadoop?

How is a combiner function in the context of Hadoop? Give an example where it can be used and justify why it should be used in the context of this problem.

What is the Hadoop shuffle?

0.0.1 Answer

Merge Sort Merge sort is a sort algorithm that sorts a dataset by divide-and-conquer. The data set is split continuously into sub-sets until each sub-set is size of 1 and therefore considered automatically sorted. Now, sub-sets are merged together to form a final fully sorted data-set. Given n sorted-subsets, the complexity to merge these sorted sub-sets is proportional to the total no. of elements in all the sub-sets combined.

Hadoop can be considered in some sense an implementation of merge sort as it first partitions the dataset into multiple map tasks, sorts each map output and eventually does a "merge" of each map's sorted output in the reducer. A lot of the heavy lifting of this implementation is done in the shuffle layer.

Combiner Function Given that data comes to the mapper in no defined order, the mapper output can be quite verbose as it can potentially not make any optimizations. The combiner function is run on mapper output and can "coalesce/combine" the mapper output so that the verbose map output does not fully need to be sent to the reducer. The combiner pretty much does the same work as that of the reducer. It has to process data in the format of the mapper output but unlike the reducer, it also generate output that is consumable by the reducer.

Let us consider a job that accepts the following input (word, priority) and the aim is to generate the avg. priority for each word. For a job without a combiner, the mapper could generate the following output (word, priority). This would be then averaged upon by the reducer. However, let's say that the word "foo" occurs a million times in the same mapper. In this scenario, without a combiner, 1 million rows will be sent to the reducer. A combiner if run could potentially optimize this into an key-value output such as (word, (sum(priority), frequency of word)). Using this tuple, the reducer can then calculate the avg. priority. Note, that the mapper output would also change to (word, (priority, 1)).

Not all operations support a combiner. Only commutative and associative operations can work as the output of the reducer has to be the same regardless of whether the combiner ran or not.

Hadoop shuffle The hadoop shuffle is the layer that transfers the data from the mappers to the reducers. Consider 4 maps and 2 reducers. Each mapper will generate data in 2 partitions considering that there are 2 reducers. This is governed by the partitioner function which can be overridden by the user. The shuffle layer will try to efficiently transfer the data from each mapper to a particular reducer, ensure that the data is sorted correctly by the configured keys (by defining an approrpiate comparator used to compare keys when sorting) and that data for a given key is provided in a sorted/grouped manner to the reducer. This is what enables us to write simple reducers without requiring us to do a lot of in-memory buffering. For example, consider the input data of format (word, priority) where the aim is to find max(priority) for each word. If the words came to the reducer in a random order, the reducer would need to keep a lookup table to track each word and its max priority given that there is no information on whether the word may or may not show up later. Given the sorted nature of input to the reducer, the simplistic reducer knows that as soon as a new word is seen, the previous word will never again show up and the max priority can then be generated for that word. This means no need to have a large lookup table for each seen word but rather just one object to track the data for the current word and its max priority.

The diagram below depicts the various components of hadoop shuffle.

```
In [ ]:
```

```
In [ ]: #### Create directories in HDFS for HW3
        !hdfs dfs -mkdir hw3
        !hdfs dfs -mkdir hw3/src
        !hdfs dfs -mkdir hw3/output
```

HW3.1 Use Counters to do EDA (exploratory data analysis and to monitor progress) Counters are lightweight objects in Hadoop that allow you to keep track of system progress in both the map and reduce stages of processing. By default, Hadoop defines a number of standard counters in "groups"; these show up in the jobtracker webapp, giving you information such as "Map input records", "Map output records", etc.

While processing information/data using MapReduce job, it is a challenge to monitor the progress of parallel threads running across nodes of distributed clusters. Moreover, it is also complicated to distinguish between the data that has been processed and the data which is yet to be processed. The MapReduce Framework offers a provision of user-defined Counters, which can be effectively utilized to monitor the progress of data across nodes of distributed clusters.

Use the Consumer Complaints Dataset provide here to complete this question:

`https://www.dropbox.com/s/vbalm3yva2rr86m/Consumer_Complaints.csv?dl=0`

The consumer complaints dataset consists of diverse consumer complaints, which have been reported across the United States regarding various types of loans. The dataset consists of records of the form:

Complaint ID,Product,Sub-product,Issue,Sub-issue,State,ZIP code,Submitted via,Date received,Date sent to company,Company,Company response,Timely response?,Consumer disputed?

Here's is the first few lines of the of the Consumer Complaints Dataset:

Complaint ID,Product,Sub-product,Issue,Sub-issue,State,ZIP code,Submitted via,Date received,Date sent to company,Company,Company response,Timely response?,Consumer disputed?

1114245,Debt collection,Medical,Disclosure verification of debt,Not given enough info to verify debt,FL,32219,Web,11/13/2014,11/13/2014,"Choice Recovery, Inc.",Closed with explanation,Yes,

1114488,Debt collection,Medical,Disclosure verification of debt,Right to dispute notice not received,TX,75006,Web,11/13/2014,11/13/2014,"Expert Global Solutions, Inc.",In progress,Yes,

1114255,Bank account or service,Checking account,Deposits and withdrawals,,NY,11102,Web,11/13/2014,11/13/2014,"FNIS (Fidelity National Information Services, Inc.)",In progress,Yes,

1115106,Debt collection,"Other (phone, health club, etc.)",Communication tactics,Frequent or repeated calls,GA,31721,Web,11/13/2014,11/13/2014,"Expert Global Solutions, Inc.",In progress,Yes,

User-defined Counters

Now, let's use Hadoop Counters to identify the number of complaints pertaining to debt collection, mortgage and other categories (all other categories get lumped into this one) in the consumer complaints

dataset. Basically produce the distribution of the Product column in this dataset using counters (limited to 3 counters here).

Hadoop offers Job Tracker, an UI tool to determine the status and statistics of all jobs. Using the job tracker UI, developers can view the Counters that have been created. Screenshot your job tracker UI as your job completes and include it here. Make sure that your user defined counters are visible.

In [65]:
```
%%writefile mapper.py
#!/usr/bin/python

#HW3.1 In this question ,we will only have a mapper which emit counters based on the type of c

import sys
import re
from csv import reader



# input comes from STDIN (standard input)
for token in reader(sys.stdin):
    key = None
    if(token[1]=='Debt collection'):
        sys.stderr.write("reporter:counter:Debt-counter,debt,1\n")
    elif(token[1]=='Mortgage'):
        sys.stderr.write("reporter:counter:Mortgage-counter,mortgage,1\n")
    else:
        sys.stderr.write("reporter:counter:Other-counter,other,1\n")
```

```
Overwriting mapper.py
```

In [66]:
```
# function to run the hadoop job
def hw3_1():

    #change properties of mapper.py
    !chmod a+x mapper.py

    #remove output directory if present else hadoop job gives error
    !hdfs dfs -rm -r hw3/output/hw31

    #remove header line from Consumer_Complaints.csv
    !tail -n $(($(wc -l < Consumer_Complaints.csv) - 1))  < Consumer_Complaints.csv > Consumer

    #move input file to hdfs
    !hdfs dfs -rm hw3/src/Consumer_Complaints_mod.csv
    !hdfs dfs -put Consumer_Complaints_mod.csv  hw3/src/

    # run map reduce job
    !hadoop jar /usr/local/Cellar/hadoop/2.7.1/libexec/share/hadoop/tools/lib/hadoop-*streamin
    -Dmapred.reduce.tasks=0 \
    -file mapper.py \
    -mapper mapper.py \
    -input hw3/src/Consumer_Complaints_mod.csv \
    -output hw3/output/hw31 \

hw3_1()
```

```
16/01/31 16:18:31 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
Deleted hw3/output/hw31
16/01/31 16:18:36 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
Deleted hw3/src/Consumer_Complaints_mod.csv
16/01/31 16:18:37 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
16/01/31 16:18:39 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files
16/01/31 16:18:39 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
packageJobJar: [mapper.py, /var/folders/91/cjfxt7ys6958ql16vjtgwwfw0000gn/T/hadoop-unjar6525908040360619
```

Output of the above from job tracker

HW 3.2 Analyze the performance of your Mappers, Combiners and Reducers using Counters
For this brief study the Input file will be one record (the next line only): foo foo quux labs foo bar quux

Perform a word count analysis of this single record dataset using a Mapper and Reducer based WordCount (i.e., no combiners are used here) using user defined Counters to count up how many time the mapper and reducer are called. What is the value of your user defined Mapper Counter, and Reducer Counter after completing this word count job. The answer should be 1 and 4 respectively. Please explain.

In [62]:
```
%%writefile mapper.py
#!/usr/bin/python
#HW3.2a In this question, we will emit a counter for everytime the mapper is called.
# output of mapper is word and 1
import sys
import re

sys.stderr.write("reporter:counter:HW_32a,num_mappers,1\n")
# input comes from STDIN (standard input)
for line in sys.stdin:
    # remove leading and trailing whitespace
    for token in line.strip().split(" "):
        print '%s,%s' % (token, 1)
```

```
Overwriting mapper.py
```

In [63]:
```
%%writefile reducer.py
#!/usr/bin/python
#HW3.2a In this question, we will emit a counter for everytime the reducer is called.

import sys

sys.stderr.write("reporter:counter:HW_32a,num_reducers,1\n")
last_key = None
word = None
total_count = 0
# input comes from STDIN (standard input)
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # parse the input we got from mapper.py
    word, count = line.split(',', 1)

    #if current key is same as last_key than increment count
    if(last_key == word):
```

```
                total_count += int(count)
            else:
                if (last_key):
                    print '%s\t%s' %(last_key,total_count)
                total_count = int(count)
                last_key = word
        if last_key == word:
            print '%s\t%s' %(last_key,total_count)
```

```
Overwriting reducer.py
```

```
In [64]: # function to run the hadoop job
         def hw3_2_a():

             #change properties of mapper.py
             !chmod a+x mapper.py;chmod a+x reducer.py

             #create inpur file and move it to hdfs
             !echo "foo foo quux labs foo bar quux" > hw32_a_input.txt

             #remove output directory if present else hadoop job gives error
             !hdfs dfs -rm -r hw3/output/hw32a

             #move input file to hdfs
             !hdfs dfs -rm hw3/src/hw32_a_input.txt
             !hdfs dfs -put hw32_a_input.txt  hw3/src/

             # run map reduce job. Here we have explicitly set the number of mappers and reducers
             !hadoop jar /usr/local/Cellar/hadoop/2.7.1/libexec/share/hadoop/tools/lib/hadoop-*streamin
             -Dmapred.map.tasks=1 \
             -numReduceTasks 4 \
             -file mapper.py \
             -mapper mapper.py \
             -file reducer.py \
             -reducer reducer.py \
             -input hw3/src/hw32_a_input.txt \
             -output hw3/output/hw32a

             print "\n"
             !echo "Word and its frequency"
             !hdfs dfs -cat hw3/output/hw32a/part*

         hw3_2_a()
```

```
16/02/01 21:13:17 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
rm: `hw3/output/hw32a': No such file or directory
16/02/01 21:13:19 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
Deleted hw3/src/hw32_a_input.txt
16/02/01 21:13:20 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
16/02/01 21:13:22 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files
16/02/01 21:13:22 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
packageJobJar: [mapper.py, reducer.py, /var/folders/91/cjfxt7ys6958ql16vjtgwwfw0000gn/T/hadoop-unjar1434


Word and its frequency
```

```
16/02/01 21:13:41 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
labs        1
quux        2
foo        3
bar        1
```

Output for first part

3.2.b Please use mulitple mappers and reducers for these jobs (at least 2 mappers and 2 reducers). Perform a word count analysis of the Issue column of the Consumer Complaints Dataset using a Mapper and Reducer based WordCount (i.e., no combiners used anywhere) using user defined Counters to count up how many time the mapper and reducer are called. What is the value of your user defined Mapper Counter, and Reducer Counter after completing your word count job.

```
In [138]: %%writefile mapper.py
          #!/usr/bin/python
          #HW3.2b In this question, we will emit a counter for everytime the mapper is called.
          # output of mapper is issue which is token[3] and 1

          import sys
          from csv import reader
          import re

          WORD_RE = re.compile(r"[\w']+")
          sys.stderr.write("reporter:counter:HW_32b,num_mappers,1\n")

          # input comes from STDIN (standard input)
          for token in reader(sys.stdin):
              #3rd token  contains text about issue
              issue = re.findall(WORD_RE,token[3])
              #now loop through all words in the issue and emit it
              for word in issue:
                  print '"%s",%s' %(word.lower(),1)

Overwriting mapper.py

In [139]: %%writefile reducer.py
          #!/usr/bin/python
          #HW3.2b In this question, we will emit a counter for everytime the reducer is called.

          import sys
          from csv import reader

          sys.stderr.write("reporter:counter:HW_32b,num_reducers,1\n")
          last_key = None #user for keeping track of which key is currently being aggregated
          word = None
          total_count = 0 #counter to increment frequency
          # input comes from STDIN (standard input)
          for token in reader(sys.stdin):
              word=token[0]
              count = int(token[1])

              #if current key is same as last_key than increment count
              if(last_key == word):
```

```
                total_count += int(count)
            else:
                if (last_key):
                    print '%s\t%s' %(last_key,total_count)
                total_count = int(count)
                last_key = word
        if last_key == word:
            print '%s\t%s' %(last_key,total_count)
```

```
Overwriting reducer.py
```

```
In [140]: # function to run the hadoop job
          def hw3_2_b():

              #change properties of mapper.py
              !chmod a+x mapper.py;chmod a+x reducer.py

              #remove output directory if present else hadoop job gives error
              !hdfs dfs -rm -r hw3/output/hw32b

              # run map reduce job
              !hadoop jar /usr/local/Cellar/hadoop/2.7.1/libexec/share/hadoop/tools/lib/hadoop-*streami
              -Dmapred.map.tasks=2 \
              -numReduceTasks 2 \
              -file mapper.py \
              -mapper mapper.py \
              -file reducer.py \
              -reducer reducer.py \
              -input hw3/src/Consumer_Complaints_mod.csv \
              -output hw3/output/hw32b

              print "\n"
              !echo "HDFS Output"
              !hdfs dfs -cat hw3/output/hw32b/part*

          hw3_2_b()
```

```
16/02/03 18:10:54 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
Deleted hw3/output/hw32b
16/02/03 18:10:56 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files
16/02/03 18:10:56 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
packageJobJar: [mapper.py, reducer.py, /var/folders/91/cjfxt7ys6958ql16vjtgwwfw0000gn/T/hadoop-unjar695


HDFS Output
16/02/03 18:11:18 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
action        2964
advance        240
advertising        1193
amount        98
amt        71
application        8868
apply        118
are        3821
atm        2422
```

```
attempts        17972
balance        597
bank        202
cancelling        2795
card        4405
charged        976
collect        17972
collection        72394
communication        8671
contact        3710
costs        4350
credited        92
customer        2734
day        71
dealing        1944
debit        2422
decision        2774
decrease        1149
didn't        925
disclosures        64
dispute        904
embezzlement        3276
fee        3198
forbearance        350
fraud        3842
funds        5663
get        4357
getting        291
identity        4729
illegal        2964
improper        4966
increase        1149
info        3553
information        29069
interest        4238
investigation        4858
issues        538
late        1797
lease        6337
lender        2165
line        1732
loan        119630
low        5663
managing        5006
marketing        1193
missing        64
modification        70487
money        3639
monitoring        1453
mortgage        8625
my        10731
opening        16205
originator        8625
other        7886
```

```
out         1242
pay         3821
payment       92
plans        350
practices       1003
privacy       240
problems       9484
rate        3431
receiving       3226
report       34903
reporting       6559
rewards       1002
scam        566
score        4357
sending       3226
servicing       36767
sharing       3489
shopping        672
statements       3621
stop        131
taking       4206
transaction       1485
underwriting       2774
using        2422
when        4095
with        1944
withdrawals       10555
a        3503
account        57448
acct        163
an        2964
and        16448
applied       139
apr        3431
arbitration       168
available       274
bankruptcy        222
being        5663
billing       8158
broker        8625
by        5663
can't        1999
cash        240
caused       5663
changes        350
charges        131
checks        75
closing        19000
company's       4858
cont'd        17972
convenience       75
credit        55251
debt        27874
delay        243
```

```
delinquent        1061
deposits        10555
determination        1490
did        139
disclosure        7655
disputes        6938
escrow        36767
expect        807
false        3621
fees        807
for        929
foreclosure        70487
i        925
incorrect        29133
issuance        640
issue        1098
making        3226
management        16205
not        18477
of        13983
on        29069
or        40508
overlimit        127
owed        17972
payments        39993
payoff        1155
process        5505
processing        243
promised        274
protection        4139
receive        139
received        216
relations        1367
repay        1647
repaying        3844
representation        3621
sale        139
service        1518
servicer        1944
settlement        4350
statement        1220
tactics        8671
terms        350
the        6248
theft        3276
threatening        2964
to        8401
transfer        597
unable        8178
unsolicited        640
use        1477
verification        7655
was        274
workout        350
```

```
wrong        169
you         3821
your         3844
```

Output for 32b of custome counters

3.2.c Perform a word count analysis of the Issue column of the Consumer Complaints Dataset using a Mapper, Reducer, and standalone combiner (i.e., not an in-memory combiner) based WordCount using user defined Counters to count up how many time the mapper, combiner, reducer are called. What is the value of your user defined Mapper Counter, and Reducer Counter after completing your word count job. Using a single reducer: What are the top 50 most frequent terms in your word count analysis? Present the top 50 terms and their frequency and their relative frequency. Present the top 50 terms and their frequency and their relative frequency. If there are ties please sort the tokens in alphanumeric/string order. Present bottom 10 tokens (least frequent items).

In [141]:
```
%%writefile mapper.py
#!/usr/bin/python
#HW3.2c In this question, we will emit a counter for everytime the mapper is called.
# output of mapper is issue which is token[3] and 1
import sys
from csv import reader
import re

WORD_RE = re.compile(r"[\w']+")
sys.stderr.write("reporter:counter:HW_32c,num_mappers,1\n")

# input comes from STDIN (standard input)
for token in reader(sys.stdin):
    #3rd token  contains text about issue
    issue = re.findall(WORD_RE,token[3])
    #now loop through all words in the issue and emit it
    for word in issue:
        print '"%s",%s' %(word.lower(),1)
```

```
Overwriting mapper.py
```

In [142]:
```
%%writefile combiner.py
#!/usr/bin/python
#HW3.2c In this question, we will emit a counter for everytime the combiner is called.
#the combiner will do intermediate aggregation of data and is similar to reducer in terms of
import sys
from csv import reader

sys.stderr.write("reporter:counter:HW_32c,num_combiners,1\n")
last_key = None
word = None
total_count = 0
# input comes from STDIN (standard input)
for token in reader(sys.stdin):
    word=token[0]
    count = int(token[1])

    #if current key is same as last_key than increment count
    if(last_key == word):
```

```
                total_count += int(count)
            else:
                if (last_key):
                    print '"%s",%s' %(last_key,total_count)
                total_count = int(count)
                last_key = word
        if last_key == word:
            print '"%s",%s' %(last_key,total_count)
```

```
Overwriting combiner.py
```

In [143]:
```
%%writefile reducer.py
#!/usr/bin/python
#HW3.2c In this question, we will emit a counter for everytime the reducer is called.

import sys
from csv import reader

sys.stderr.write("reporter:counter:HW_32c,num_reducers,1\n")
last_key = None
word = None
total_count = 0
# input comes from STDIN (standard input)
for token in reader(sys.stdin):
    word=token[0]
    count = int(token[1])

    #if current key is same as last_key than increment count
    if(last_key == word):
        total_count += int(count)
    else:
        if (last_key):
            print '"%s",%s' %(last_key,total_count)
        total_count = int(count)
        last_key = word
# do not forget to output the last word
if last_key == word:
    print '"%s",%s' %(last_key,total_count)
```

```
Overwriting reducer.py
```

In [145]:
```
# function to run the hadoop job
def hw3_2_c():

    #change properties of mapper.py
    !chmod a+x mapper.py;chmod a+x reducer.py

    #remove output directory if present else hadoop job gives error
    !hdfs dfs -rm -r hw3/output/hw32c

    # run map reduce job
    !hadoop jar /usr/local/Cellar/hadoop/2.7.1/libexec/share/hadoop/tools/lib/hadoop-*streami
    -Dmapred.map.tasks=4 \
    -numReduceTasks 2 \
    -file mapper.py \
```

```
            -mapper mapper.py \
            -file combiner.py \
            -combiner combiner.py \
            -file reducer.py \
            -reducer reducer.py \
            -input hw3/src/Consumer_Complaints_mod.csv \
            -output hw3/output/hw32c

            print "\n"
            !echo "HDFS Output"
            !hdfs dfs -cat hw3/output/hw32c/part* | head -20

        hw3_2_c()
```

```
16/02/03 18:12:37 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
Deleted hw3/output/hw32c
16/02/03 18:12:38 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files
16/02/03 18:12:39 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
packageJobJar: [mapper.py, combiner.py, reducer.py, /var/folders/91/cjfxt7ys6958qll6vjtgwwfw0000gn/T/had


HDFS Output
16/02/03 18:13:01 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
"action",2964
"advance",240
"advertising",1193
"amount",98
"amt",71
"application",8868
"apply",118
"are",3821
"atm",2422
"attempts",17972
"balance",597
"bank",202
"cancelling",2795
"card",4405
"charged",976
"collect",17972
"collection",72394
"communication",8671
"contact",3710
"costs",4350
cat: Unable to write to output stream.
```

Output of mapper, recucer and combiner counters

part 2 of 3_2_c Using a single reducer: What are the top 50 most frequent terms in your word count analysis? Present the top 50 terms and their frequency and their relative frequency. Present the top 50 terms and their frequency and their relative frequency. If there are ties please sort the tokens in alphanumeric/string order. Present bottom 10 tokens (least frequent items).

```
In [146]: %%writefile mapper.py
          #!/usr/bin/python
```

```
#HW3.2c_1 In this question, we will emit a counter for everytime the mapper is called.
# output of mapper is frequency and issue
# output of previous mapreduce job is used as input for this map reduce task
import sys
from csv import reader

sys.stderr.write("reporter:counter:Mapper-counter,num_mappers,1\n")

# input comes from STDIN (standard input)
for token in reader(sys.stdin):
    print '%s\t"%s"' %(token[1],token[0])
```

```
Overwriting mapper.py
```

In [147]:
```
%%writefile reducer.py
#!/usr/bin/python
#HW3.2c_1 In this question, we will emit a counter for everytime the reducer is called.
# In this case of mapreduce we are using secondary sort where in we first sort on the frequen
# and than sort on the issue in case of tie

import sys, Queue
import re
from csv import reader

sys.stderr.write("reporter:counter:Reducer-counter,num_reducers,1\n")

n_max = 50 # top n issues
n_min = 10 # bottom n issues
a_min = [] # list to hold the bototom issues
q_max = Queue.Queue(n_max) # Queue to hold the top 50 issues
total_count =0

# input comes from STDIN (standard input)
for line in sys.stdin:
    row = re.split(r'\t+',line.strip())
    freq = int(row[0])
    key = row[1]
    #increment the count
    total_count += freq

    # add the lowest 10 words as we are getting data in sorted order already
    if len(a_min) < n_min:
        a_min.append((key,freq))

    #Now add the top 10 words . In this case we use queue as its FIFO which will always pop t
    if q_max.full():
        q_max.get()
    q_max.put((key,freq))

print '\n Output = Word , Frequency and Relative Frequency'

print '\n%d Bottom issues:' %n_min
for record in a_min:
    print record[0] +"\t"+str(record[1])+"\t"+str(1.0*record[1]/total_count)
```

```
        print '\n%d Top issues:' %n_max
        for i in range(n_max):
            record = q_max.get()
            print record[0] +"\t"+str(record[1])+"\t"+str(1.0*record[1]/total_count)
```

```
Overwriting reducer.py
```

```
In [148]: # function to run the hadoop job
          def hw3_2_c():

              #change properties of mapper.py
              !chmod a+x mapper.py;chmod a+x reducer.py

              #remove output directory if present else hadoop job gives error
              !hdfs dfs -rm -r hw3/output/hw32c_2

              #run hadoop job
              !hadoop jar /usr/local/Cellar/hadoop/2.7.1/libexec/share/hadoop/tools/lib/hadoop-*streamin
              -D mapreduce.job.output.key.comparator.class=org.apache.hadoop.mapreduce.lib.partition.Key
              -D stream.num.map.output.key.fields=2 \
              -D mapreduce.partition.keypartitioner.options=-k1,1 \
              -D mapreduce.partition.keycomparator.options="-k1,1n -k2,2" \
              -D mapred.map.tasks=4 \
              -numReduceTasks 1 \
              -mapper mapper.py \
              -file mapper.py \
              -reducer reducer.py \
              -file reducer.py \
              -input hw3/output/hw32c/part* \
              -output hw3/output/hw32c_2


              print "\n"
              !echo "HDFS Output"
              !hdfs dfs -cat hw3/output/hw32c_2/part*

          hw3_2_c()
```

```
16/02/03 18:14:28 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
Deleted hw3/output/hw32c_2
16/02/03 18:14:29 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files
16/02/03 18:14:29 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
packageJobJar: [mapper.py, reducer.py, /var/folders/91/cjfxt7ys6958ql16vjtgwwfw0000gn/T/hadoop-unjar4130


HDFS Output
16/02/03 18:14:47 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...

 Output = Word , Frequency and Relative Frequency

10 Bottom issues:
"missing"        64        4.74668992545e-05
"disclosures"        64        4.74668992545e-05
"amt"        71        5.26585913604e-05
"day"        71        5.26585913604e-05
```

```
"checks"        75        5.56252725638e-05
"convenience"        75        5.56252725638e-05
"credited"        92        6.82336676783e-05
"payment"        92        6.82336676783e-05
"amount"        98        7.26836894834e-05
"apply"        118        8.75170955004e-05

50 Top issues:
"low"        5663        0.00420007891372
"caused"        5663        0.00420007891372
"funds"        5663        0.00420007891372
"being"        5663        0.00420007891372
"by"        5663        0.00420007891372
"the"        6248        0.00463395603972
"lease"        6337        0.00469996469649
"reporting"        6559        0.00486461550328
"disputes"        6938        0.00514570854731
"verification"        7655        0.00567748615302
"disclosure"        7655        0.00567748615302
"other"        7886        0.00584881199251
"billing"        8158        0.00605054631434
"unable"        8178        0.00606537972036
"to"        8401        0.00623077219745
"originator"        8625        0.00639690634484
"broker"        8625        0.00639690634484
"mortgage"        8625        0.00639690634484
"tactics"        8671        0.00643102317868
"communication"        8671        0.00643102317868
"application"        8868        0.00657713222795
"problems"        9484        0.00703400113327
"deposits"        10555        0.00782833002548
"withdrawals"        10555        0.00782833002548
"my"        10731        0.00795886399843
"of"        13983        0.0103707758168
"management"        16205        0.0120187672253
"opening"        16205        0.0120187672253
"and"        16448        0.0121989931084
"collect"        17972        0.0133292986469
"cont'd"        17972        0.0133292986469
"attempts"        17972        0.0133292986469
"owed"        17972        0.0133292986469
"not"        18477        0.0137038421488
"closing"        19000        0.0140917357162
"debt"        27874        0.0206733179659
"on"        29069        0.0215596139754
"information"        29069        0.0215596139754
"incorrect"        29133        0.0216070808747
"report"        34903        0.0258865185106
"escrow"        36767        0.0272689919514
"servicing"        36767        0.0272689919514
"payments"        39993        0.0296616203419
"or"        40508        0.0300435805469
"credit"        55251        0.0409780257923
"account"        57448        0.0426074754433
```

```
"modification"        70487        0.0522781144961
"foreclosure"        70487        0.0522781144961
"collection"        72394        0.0536924797598
"loan"        119630        0.0887260180908
```

In [ ]:

HW 3.2.1 OPTIONAL Using 2 reducers: What are the top 50 most frequent terms in your word count analysis? Present the top 50 terms and their frequency and their relative frequency. Present the top 50 terms and their frequency and their relative frequency. If there are ties please sort the tokens in alphanumeric/string order. Present bottom 10 tokens (least frequent items). Please use a combiner.

First create a job to generate the partition file

In [161]:
```
%%writefile mapper_3.2.1_1.py
#!/usr/bin/python
import sys
import re
import csv
import numpy as np

# The input is the result from hw3.2-part3, which has the format:
# word<tab>count

n = 0
rate = 5 # sample one out of every five counts
samples = []

# We want to sample the count
for line in sys.stdin:
    (word, count) = line.strip().split(",")

    if n % rate == 0:
        samples.append(int(count))

    n += 1

# Now we have a sample of counts.  Let's find the 50% percentile, as we only have 2 reducers.
p50 = np.percentile(samples, 50)

print p50
```

```
Overwriting mapper_3.2.1_1.py
```

In [172]:
```
%%writefile mapper_3.2.1_2.py
#!/usr/bin/python
import sys
import re
import csv
import argparse

parser = argparse.ArgumentParser()
parser.add_argument("--partitionFile", default="partitions.txt")

args = parser.parse_args()
```

```
        sys.stderr.write('reporter:counter:custom,mapper_called,1\n')


        # The input is the result from hw3.2-part3, which has the format:
        # word<tab>count

        # First read the partition file and get the 50th percentile
        with open(args.partitionFile, 'r') as f:
            p50 = float(f.readline())

        groups = ["g" + str(x) for x in range(2)] # names of the two groups

        for line in sys.stdin:
            (word, count) = line.strip().split(",")
            count = int(count)

            # Assign it to different reducers based on the count value
            if count >= p50:
                group = groups[0]
            else:
                group = groups[1]

            print group + "\t" + word+"\t"+str(count)

            # Use order inversion so that reducer can count the total word count in a single pass
            # Need to send it to each group
            for g in groups:
                print g + "\t" + str(count) + "\t" + str(sys.maxint)
```

```
Overwriting mapper_3.2.1_2.py
```

```
In [173]: # Quick test
          !hdfs dfs -cat hw3/output/hw32c/part* > t
          !cat t | python mapper_3.2.1_1.py > partitions.txt

          print "partitions.txt:"
          !cat partitions.txt
          print

          !head -n 30 t | \
          python mapper_3.2.1_2.py | \
          sort -k1,1 -k3,3nr
```

```
16/02/03 20:59:21 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
partitions.txt:
3276.0

reporter:counter:custom,mapper_called,1
g0        1149        9223372036854775807
g0        118        9223372036854775807
g0        1193        9223372036854775807
g0        17972        9223372036854775807
g0        17972        9223372036854775807
g0        1944        9223372036854775807
```

```
g0        202         9223372036854775807
g0        240         9223372036854775807
g0        2422         9223372036854775807
g0        2422         9223372036854775807
g0        2734         9223372036854775807
g0        2774         9223372036854775807
g0        2795         9223372036854775807
g0        2964         9223372036854775807
g0        3710         9223372036854775807
g0        3821         9223372036854775807
g0        4350         9223372036854775807
g0        4405         9223372036854775807
g0        597         9223372036854775807
g0        64         9223372036854775807
g0        71         9223372036854775807
g0        71         9223372036854775807
g0        72394          9223372036854775807
g0        8671         9223372036854775807
g0        8868         9223372036854775807
g0        904         9223372036854775807
g0        92         9223372036854775807
g0        925         9223372036854775807
g0        976         9223372036854775807
g0        98         9223372036854775807
g0        "collection"         72394
g0        "attempts"         17972
g0        "collect"         17972
g0        "application"         8868
g0        "communication"         8671
g0        "card"         4405
g0        "costs"         4350
g0        "are"         3821
g0        "contact"         3710
g1        1149         9223372036854775807
g1        118         9223372036854775807
g1        1193         9223372036854775807
g1        17972         9223372036854775807
g1        17972         9223372036854775807
g1        1944         9223372036854775807
g1        202         9223372036854775807
g1        240         9223372036854775807
g1        2422         9223372036854775807
g1        2422         9223372036854775807
g1        2734         9223372036854775807
g1        2774         9223372036854775807
g1        2795         9223372036854775807
g1        2964         9223372036854775807
g1        3710         9223372036854775807
g1        3821         9223372036854775807
g1        4350         9223372036854775807
g1        4405         9223372036854775807
g1        597         9223372036854775807
g1        64         9223372036854775807
g1        71         9223372036854775807
```

```
g1	71	9223372036854775807
g1	72394	9223372036854775807
g1	8671	9223372036854775807
g1	8868	9223372036854775807
g1	904	9223372036854775807
g1	92	9223372036854775807
g1	925	9223372036854775807
g1	976	9223372036854775807
g1	98	9223372036854775807
g1	"action"	2964
g1	"cancelling"	2795
g1	"decision"	2774
g1	"customer"	2734
g1	"atm"	2422
g1	"debit"	2422
g1	"dealing"	1944
g1	"advertising"	1193
g1	"decrease"	1149
g1	"charged"	976
g1	"didn't"	925
g1	"dispute"	904
g1	"balance"	597
g1	"advance"	240
g1	"bank"	202
g1	"apply"	118
g1	"amount"	98
g1	"credited"	92
g1	"amt"	71
g1	"day"	71
g1	"disclosures"	64
```

In [174]:
```
%%writefile reducer_3.2.1.py
#!/usr/bin/python
from __future__ import division # Use Python 3-style division
import sys, Queue, csv

sys.stderr.write('reporter:counter:custom,reducer_called,1\n')

wordCount = 0 # Count of each word
totalCount = 0 # Total number of words
curr = None # the current word

# input format:
# Total count: group \t count \t max.int
# Word: group \t word \t count
for fields in csv.reader(sys.stdin, delimiter='\t'):
    group = fields[0]
    word = fields[1]
    count = fields[2]
    count = int(count)

    # Find out the total word count.
    # We use count == sys.maxint as the special key for order inversion.
    if count == sys.maxint:
```

```
                totalCount += int(word) # The word is the count
                continue

            # If we have encountered a new word, output the answer of the current word
            if curr != word:
                if curr is not None:
                    print "\t".join([curr, str(wordCount), str(wordCount/totalCount)])
                    wordCount = 0

            wordCount += count
            curr = word

        # Handle the last word seen
        if curr is not None:
            print "\t".join([curr, str(wordCount), str(wordCount/totalCount)])
```

```
Overwriting reducer_3.2.1.py
```

In [175]:
```
# Quick test
!hdfs dfs -cat hw3/output/hw32c/part* > t
!cat t | python mapper_3.2.1_1.py > partitions.txt

print "partitions.txt:"
!cat partitions.txt
print

!head -n 30 t | \
python mapper_3.2.1_2.py | \
sort -k1,1 -k3,3nr | \
python reducer_3.2.1.py
```

```
16/02/03 20:59:41 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
partitions.txt:
3276.0

reporter:counter:custom,reducer_called,1
reporter:counter:custom,mapper_called,1
collection        72394        0.433709965372
attempts        17972        0.107669634192
collect        17972        0.107669634192
application        8868        0.0531278831522
communication        8671        0.0519476629243
card        4405        0.026390203573
costs        4350        0.0260607004637
are        3821        0.0228914796487
contact        3710        0.0111132412322
action        2964        0.00887861105453
cancelling        2795        0.00837237445932
decision        2774        0.00830946932027
customer        2734        0.00818965000779
atm        2422        0.00725505937047
debit        2422        0.00725505937047
dealing        1944        0.00582321858637
advertising        1193        0.00357361099462
decrease        1149        0.0034418097509
```

```
charged        976        0.00292359122443
didn't         925        0.00277082160103
dispute        904        0.00270791646198
balance        597        0.00178830323872
advance        240        0.000718915874861
bank        202        0.000605087528008
apply        118        0.000353466971807
amount        98        0.000293557315568
credited        92        0.000275584418697
amt        71        0.000212679279646
day        71        0.000212679279646
disclosures        64        0.000191710899963
```

```
In [176]: # First job - generate partitions file
          !hdfs dfs -rm -r hw3.2.1-part1

          !hadoop jar /usr/local/Cellar/hadoop/2.7.1/libexec/share/hadoop/tools/lib/hadoop-*streaming*.j
          -D mapreduce.job.name="hw3.2.1-part1" \
          -D mapred.map.tasks=1 \
          -D mapred.reduce.tasks=0 \
          -file mapper_3.2.1_1.py -mapper mapper_3.2.1_1.py \
          -input hw3/output/hw32c/part* \
          -output hw3.2.1-part1

          ! hdfs dfs -cat hw3.2.1-part1/part* > partitions.txt
```

```
16/02/03 20:59:52 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
Deleted hw3.2.1-part1
16/02/03 20:59:53 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files
16/02/03 20:59:53 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
packageJobJar: [mapper_3.2.1_1.py, /var/folders/91/cjfxt7ys6958qll6vjtgwwfw0000gn/T/hadoop-unjar44426557
16/02/03 21:00:05 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
```

```
In [177]: !hdfs dfs -rm -r hw3.2.1-part2

          !hadoop jar /usr/local/Cellar/hadoop/2.7.1/libexec/share/hadoop/tools/lib/hadoop-*streaming*.j
          -files "partitions.txt" \
          -D mapreduce.job.name="hw3.2.1-part2" \
          -D mapred.map.tasks=1 \
          -D mapred.reduce.tasks=2 \
          -D mapreduce.partition.keypartitioner.options=-k1,1 \
          -D stream.num.map.output.key.fields=3 \
          -D mapred.output.key.comparator.class=org.apache.hadoop.mapred.lib.KeyFieldBasedComparator \
          -D mapred.text.key.comparator.options="-k3,3nr -k2,2" \
          -file mapper_3.2.1_2.py -mapper mapper_3.2.1_2.py \
          -file reducer_3.2.1.py -reducer reducer_3.2.1.py \
          -partitioner org.apache.hadoop.mapred.lib.KeyFieldBasedPartitioner \
          -input hw3/output/hw32c/part* \
          -output hw3.2.1-part2
```

```
16/02/03 21:00:10 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
Deleted hw3.2.1-part2
16/02/03 21:00:11 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
16/02/03 21:00:11 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files
packageJobJar: [mapper_3.2.1_2.py, reducer_3.2.1.py, /var/folders/91/cjfxt7ys6958qll6vjtgwwfw0000gn/T/ha
```

Output the results

```
In [178]: # Get the list of output files
          !hdfs dfs -ls hw3.2.1-part2/part* 2>/dev/null | \
          tr -s ' ' | cut -d ' ' -f 8 > hw3.2.1_output_files

          # For each file name, append the word count taken from the 1st line in the file content
          !rm -f hw3.2.1_output_files_with_size
          !for f in `cat hw3.2.1_output_files`; do hdfs dfs -cat $f 2>/dev/null | \
          head -n 1 | cut -d$'\t' -f 2 | \
          xargs echo "$f" >> hw3.2.1_output_files_with_size; done

          # Sort hw3.2.1_output_files_with_size by size, in descending order.
          # Then generate a sorted list of output files, based on the sorted size
          !sort -t ' ' -k2,2nr hw3.2.1_output_files_with_size | \
          cut -d ' ' -f1 > hw3.2.1_output_files_sorted

          # Output the top 50
          print "Top 50 words:"
          !sed -n 1p hw3.2.1_output_files_sorted | xargs hdfs dfs -cat 2>/dev/null | head -n 50

          print

          print "Bottom 10 words:"
          !sed -n 2p hw3.2.1_output_files_sorted | xargs hdfs dfs -cat 2>/dev/null | tail -n 10
```

```
Top 50 words:
loan        119630        0.0887260180908
collection        72394        0.0536924797598
foreclosure        70487        0.0522781144961
modification        70487        0.0522781144961
account        57448        0.0426074754433
credit        55251        0.0409780257923
or        40508        0.0300435805469
payments        39993        0.0296616203419
escrow        36767        0.0272689919514
servicing        36767        0.0272689919514
report        34903        0.0258865185106
incorrect        29133        0.0216070808747
information        29069        0.0215596139754
on        29069        0.0215596139754
debt        27874        0.0206733179659
closing        19000        0.0140917357162
not        18477        0.0137038421488
attempts        17972        0.0133292986469
collect        17972        0.0133292986469
cont'd        17972        0.0133292986469
owed        17972        0.0133292986469
and        16448        0.0121989931084
management        16205        0.0120187672253
opening        16205        0.0120187672253
of        13983        0.0103707758168
my        10731        0.00795886399843
deposits        10555        0.00782833002548
withdrawals        10555        0.00782833002548
```

```
problems        9484        0.00703400113327
application        8868        0.00657713222795
communication        8671        0.00643102317868
tactics        8671        0.00643102317868
broker        8625        0.00639690634484
mortgage        8625        0.00639690634484
originator        8625        0.00639690634484
to        8401        0.00623077219745
unable        8178        0.00606537972036
billing        8158        0.00605054631434
other        7886        0.00584881199251
disclosure        7655        0.00567748615302
verification        7655        0.00567748615302
disputes        6938        0.00514570854731
reporting        6559        0.00486461550328
lease        6337        0.00469996469649
the        6248        0.00463395603972
being        5663        0.00420007891372
by        5663        0.00420007891372
caused        5663        0.00420007891372
funds        5663        0.00420007891372
low        5663        0.00420007891372

Bottom 10 words:
apply        118        8.75170955004e-05
amount        98        7.26836894834e-05
credited        92        6.82336676783e-05
payment        92        6.82336676783e-05
checks        75        5.56252725638e-05
convenience        75        5.56252725638e-05
amt        71        5.26585913604e-05
day        71        5.26585913604e-05
disclosures        64        4.74668992545e-05
missing        64        4.74668992545e-05
```

In [ ]:

HW3.3. Shopping Cart Analysis Product Recommendations: The action or practice of selling additional products or services to existing customers is called cross-selling. Giving product recommendation is one of the examples of cross-selling that are frequently used by online retailers. One simple method to give product recommendations is to recommend products that are frequently browsed together by the customers.

For this homework use the online browsing behavior dataset located at:

`https://www.dropbox.com/s/zlfyiwa70poqg74/ProductPurchaseData.txt?dl=0`

Each line in this dataset represents a browsing session of a customer. On each line, each string of 8 characters represents the id of an item browsed during that session. The items are separated by spaces.

Here are the first few lines of the ProductPurchaseData FRO11987 ELE17451 ELE89019 SNA90258 GRO99222 GRO99222 GRO12298 FRO12685 ELE91550 SNA11465 ELE26917 ELE52966 FRO90334 SNA30755 ELE17451 FRO84225 SNA80192 ELE17451 GRO73461 DAI22896 SNA99873 FRO86643 ELE17451 ELE37798 FRO86643 GRO56989 ELE23393 SNA11465 ELE17451 SNA69641 FRO86643 FRO78087 SNA11465 GRO39357 ELE28573 ELE11375 DAI54444

Do some exploratory data analysis of this dataset.

How many unique items are available from this supplier?

Using a single reducer: Report your findings such as number of unique products; largest basket; report the top 50 most frequently purchased items, their frequency, and their relative frequency (break ties by sorting the products alphabetical order) etc. using Hadoop Map-Reduce.

In [84]:
```
%%writefile mapper1.py
#!/usr/bin/python
#HW3.3 In this question, we will emit a counter for everytime the reducer is called.
# Here the mapper reads each session information and emits product along with the count.
# We also emit the number of products in each basket and basket count as for each line read.
import sys
import re

sys.stderr.write("reporter:counter:HW3_3,num_mapper,1\n")
# input comes from STDIN (standard input)
for line in sys.stdin:
    count = 0 #used to determine basket length
    # remove leading and trailing whitespace
    for token in line.strip().split(" "):
        print '%s\t%s' % (token, 1)
        count += 1
    print '%s\t%s'%('*BASKET',count) # total number of products in the basket
    print '%s\t%s'%('*NUM_OF_BASKET',1) # basket count
```

```
Writing mapper1.py
```

In [85]:
```
%%writefile combiner.py
#!/usr/bin/python
#HW3.3 In this question, we will emit a counter for everytime the combiner is called.
# Here the combiner code does intermediate aggregation for product, number of baskets
import sys
import re

sys.stderr.write("reporter:counter:HW3_3,num_combiners,1\n")

last_key = None
word = None
total_count = 0
max_basket = -1
basket_cnt = 0

# input comes from STDIN (standard input)
for line in sys.stdin:
    row = re.split(r'\t+',line.strip())
    product = row[0]
    count = int(row[1])

    #if its basket count determine which whether the basket is largest so far and continue
    if (product =='*BASKET'):
        if(count > max_basket):
            max_basket = count
        continue
    #if the data is about number of baskets increment the count and continue
    elif (product =='*NUM_OF_BASKET'):
        basket_cnt += count
        continue
```

```
            #else aggregate the product data
            else:
                #if current key is same as last_key than increment count
                if(last_key == product):
                    total_count += count
                else:
                    if (last_key):
                        print '%s\t%s' %(last_key,total_count)
                    total_count = count
                    last_key = product
        if last_key == product:
            print '%s\t%s' %(last_key,total_count)

        if(max_basket != -1):
            print '%s\t%s' %('*BASKET',max_basket)
        print '%s\t%s' %('*NUM_OF_BASKET',basket_cnt)
```

```
Overwriting combiner.py
```

In [86]:
```
%%writefile reducer1.py
#!/usr/bin/python
#HW3.3 In this question, we will emit a counter for everytime the reducer is called.
#here we have used heapq to store the 50 top products frequently purchased.

import sys
import re
import heapq

sys.stderr.write("reporter:counter:HW3_3,num_reducers,1\n")

last_key = None
word = None
total_count = 0
max_basket = -1
n_max = 100 #top n products purchased
unique_cnt = 0 #number of unique products
max_queue =  [] # hold top n products
num_baskets = 0

#function to add to queue.
#we have used heapq here so we can remove the smallest element if the queue is full during ins
def add_to_max_Q(max_q,product,cnt):
    # add elements if q is not full
    if (len(max_q) < n_max):
        heapq.heappush(max_q, (cnt, product))
    else:
        # add new element and then remove smallest element
        heapq.heappush(max_q, (cnt, product))
        heapq.heappop(max_q)

#function to print the queue data
def print_topn(max_q,total_bsk):
    cnt = len(max_q)
    if(cnt > n_max):
        cnt = n_max
```

```
            s = heapq.nlargest(cnt, max_q)
            for row in s:
                print '%s\t%s\t%s'%(row[1] ,row[0],(1.0*row[0]/total_bsk))


        # input comes from STDIN (standard input)
        for line in sys.stdin:
            row = re.split(r'\t+',line.strip())
            product = row[0]
            count = int(row[1])

            #if data is about basket length, determine which is largest and continue
            if (product =='*BASKET'):
                if(count > max_basket):
                    max_basket = count
                continue
            #if data is about number of baskets , than increment total count
            elif (product == '*NUM_OF_BASKET'):
                num_baskets += count
                continue
            #aggregate products
            else:
                #if current key is same as last_key than increment count
                if(last_key == product):
                    total_count += count
                else:
                    if (last_key):
                        #increment unique product count and add to our queue
                        unique_cnt += 1
                        add_to_max_Q(max_queue,last_key,total_count)
                    total_count = count
                    last_key = product
        if last_key == product:
            unique_cnt += 1
            #uncomment below line if you want use heapq
            add_to_max_Q(max_queue,last_key,total_count)


        print '%s\t%s\t%s' %('*LARGEST_BASKET',max_basket,1)
        print '%s\t%s\t%s' %('*UNIQUE_CNT',unique_cnt,1)
        print_topn(max_queue,num_baskets)

        # print "\n Largest Basket : %s" %(max_basket)
        # print "\n Number of unique products: %s" %(unique_cnt)
        # print "\n Top 50 products (product, frequency, relative frequency )\n "
```

```
Overwriting reducer1.py
```

In [ ]:

In [87]:
```
%%writefile mapper2.py
#!/usr/bin/python
#HW3.3 Writing a second map reduce job to do sorting on the output from the previous job
import sys
import re
```

```
        sys.stderr.write("reporter:counter:HW3_3b,num_mapper,1\n")
        # input comes from STDIN (standard input)
        for line in sys.stdin:
            # remove leading and trailing whitespace
            token = line.strip().split('\t')
            print '%s\t%s\t%s' %(token[1],token[0],token[2])
```

```
Overwriting mapper2.py
```

```
In [88]: %%writefile reducer2.py
        #!/usr/bin/python
        #HW3.3 Writing a second  reduce job to print the 50 top products
        import sys
        import re

        sys.stderr.write("reporter:counter:HW3_3b,num_mapper,1\n")

        print "\n Top 50 products (product, frequency, relative frequency )\n "
        counter = 0
        max_basket=0
        unique_cnt =0
        n_max=50

        # input comes from STDIN (standard input)
        for line in sys.stdin:
            token = re.split(r'\t+',line.strip())

            if(token[1]=='*LARGEST_BASKET'):
                max_basket = token[0]
                continue
            elif(token[1]=='*UNIQUE_CNT'):
                unique_cnt = token[0]
                continue;
            else:
                if(counter<n_max):
                    print '%s\t%s\t%s'%(token[1] ,token[0],token[2])
                    counter += 1

        print "\n Largest Basket : %s" %(max_basket)
        print "\n Number of unique products: %s" %(unique_cnt)
```

```
Overwriting reducer2.py
```

```
In [89]: # function to run the hadoop job
        def hw3_3():

            #change properties of mapper.py
            !chmod a+x mapper1.py;chmod a+x reducer1.py;chmod a+x mapper2.py;chmod a+x reducer2.py;chm

            #remove output directory if present else hadoop job gives error
            !hdfs dfs -rm -r hw3/output/hw33b
            !hdfs dfs -rm -r hw3/output/hw33

            #move input file to hdfs
```

```
            !hdfs dfs -rm hw3/src/ProductPurchaseData.txt
            !hdfs dfs -put ProductPurchaseData.txt  hw3/src/

            #run hadoop job with mapper1 and reducer1
            !hadoop jar /usr/local/Cellar/hadoop/2.7.1/libexec/share/hadoop/tools/lib/hadoop-*streaming
            -file mapper1.py \
            -mapper mapper1.py \
            -file combiner.py \
            -combiner combiner.py \
            -file reducer1.py \
            -reducer reducer1.py \
            -input hw3/src/ProductPurchaseData.txt \
            -output hw3/output/hw33


            #run hadoop job with mapper2 and reducer 2
            !hadoop jar /usr/local/Cellar/hadoop/2.7.1/libexec/share/hadoop/tools/lib/hadoop-*streaming
            -D mapreduce.job.output.key.comparator.class=org.apache.hadoop.mapreduce.lib.partition.KeyF
            -D stream.num.map.output.key.fields=2 \
            -D mapreduce.partition.keypartitioner.options=-k1,1 \
            -D mapreduce.partition.keycomparator.options="-k1,1nr -k2,2" \
            -D mapred.reduce.tasks=1 \
            -mapper mapper2.py \
            -file mapper2.py \
            -reducer reducer2.py \
            -file reducer2.py \
            -input hw3/output/hw33/part* \
            -output hw3/output/hw33b


            print "\n"
            !echo "HDFS Output"
            !hdfs dfs -cat hw3/output/hw33b/part*

        hw3_3()
```

```
16/02/01 21:22:04 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
Deleted hw3/output/hw33b
16/02/01 21:22:05 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
Deleted hw3/output/hw33
16/02/01 21:22:07 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
Deleted hw3/src/ProductPurchaseData.txt
16/02/01 21:22:08 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
16/02/01 21:22:10 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files
16/02/01 21:22:10 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
packageJobJar: [mapper1.py, combiner.py, reducer1.py, /var/folders/91/cjfxt7ys6958qll6vjtgwwfw0000gn/T/h
16/02/01 21:22:28 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files
16/02/01 21:22:28 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
packageJobJar: [mapper2.py, reducer2.py, /var/folders/91/cjfxt7ys6958qll6vjtgwwfw0000gn/T/hadoop-unjar55


HDFS Output
16/02/01 21:22:47 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
```

Top 50 products (product, frequency, relative frequency)

DAI62779	6667	0.214366097553
FRO40251	3881	0.124786984341
ELE17451	3875	0.1245940645
GRO73461	3602	0.115816211697
SNA80324	3044	0.0978746664094
ELE32164	2851	0.0916690781647
DAI75645	2736	0.0879714478634
SNA45677	2455	0.0789363686055
FRO31317	2330	0.0749172052346
DAI85309	2293	0.0737275328768
ELE26917	2292	0.0736953795698
FRO80039	2233	0.0717983344587
GRO21487	2115	0.0680042442365
SNA99873	2083	0.0669753384136
GRO59710	2004	0.0644352271631
GRO71621	1920	0.0617343493778
FRO85978	1918	0.0616700427639
GRO30386	1840	0.0591620848204
ELE74009	1816	0.0583904054532
GRO56726	1784	0.0573614996302
DAI63921	1773	0.0570078132536
GRO46854	1756	0.0564612070351
ELE66600	1713	0.0550786148355
DAI83733	1712	0.0550464615286
FRO32293	1702	0.0547249284589
ELE66810	1697	0.0545641619241
SNA55762	1646	0.0529243432687
DAI22177	1627	0.0523134304363
FRO78087	1531	0.0492267129674
ELE99737	1516	0.0487444133629
ELE34057	1489	0.0478762740748
GRO94758	1489	0.0478762740748
FRO35904	1436	0.0461721488055
FRO53271	1420	0.045657695894
SNA93860	1407	0.0452397029034
SNA90094	1390	0.044693096685
GRO38814	1352	0.0434712710202
ELE56788	1345	0.0432461978715
GRO61133	1321	0.0424745185042
DAI88807	1316	0.0423137519694
ELE74482	1316	0.0423137519694
ELE59935	1311	0.0421529854346
SNA96271	1295	0.0416385325231
DAI43223	1290	0.0414777659882
ELE91337	1289	0.0414456126813
GRO15017	1275	0.0409954663837
DAI31081	1261	0.0405453200862
GRO81087	1220	0.0392270345005
DAI22896	1219	0.0391948811935
GRO85051	1214	0.0390341146587

Largest Basket : 37

```
Number of unique products: 12591
```

In [ ]:

3.3.1 OPTIONAL Using 2 reducers: Report your findings such as number of unique products; largest basket; report the top 50 most frequently purchased items, their frequency, and their relative frequency (break ties by sorting the products alphabetical order) etc. using Hadoop Map-Reduce.

In [ ]:

HW3.4. (Computationally prohibitive but then again Hadoop can handle this) Pairs Suppose we want to recommend new products to the customer based on the products they have already browsed on the online website. Write a map-reduce program to find products which are frequently browsed together. Fix the support count (cooccurence count) to s = 100 (i.e. product pairs need to occur together at least 100 times to be considered frequent) and find pairs of items (sometimes referred to itemsets of size 2 in association rule mining) that have a support count of 100 or more.

List the top 50 product pairs with corresponding support count (aka frequency), and relative frequency or support (number of records where they coccur, the number of records where they coccur/the number of baskets in the dataset) in decreasing order of support for frequent (100>count) itemsets of size 2.

Use the Pairs pattern (lecture 3) to extract these frequent itemsets of size 2. Free free to use combiners if they bring value. Instrument your code with counters for count the number of times your mapper, combiner and reducers are called.

Please output records of the following form for the top 50 pairs (itemsets of size 2):

```
item1, item2, support count, support
```

Fix the ordering of the pairs lexicographically (left to right), and break ties in support (between pairs, if any exist) by taking the first ones in lexicographically increasing order.

Report the compute time for the Pairs job. Describe the computational setup used (E.g., single computer; dual core; linux, number of mappers, number of reducers) Instrument your mapper, combiner, and reducer to count how many times each is called using Counters and report these counts.

In [125]:
```
%%writefile mapper1.py
#!/usr/bin/python
#HW3.4 In this question, we will emit a counter for everytime the mapper is called.
# In this mapper code, we are first taking unique products for each basket ( if there are dup
# and than sorting it. We have used python function of combinations to emit all tuples of len
# We also emit for everyline read a count for the number of baskets

import sys
import re
import itertools

sys.stderr.write("reporter:counter:HW3_4,num_mapper,1\n")
# input comes from STDIN (standard input)
for line in sys.stdin:
    # emit count for basket
    print '%s\t%s\t%s'%('*','BASKET',1)

    # remove leading and trailing whitespace and tokenize
    token  =  line.strip().split(" ")
    for subset in itertools.combinations(sorted(set(token)), 2):
        print '%s\t%s\t%s' % (subset[0],subset[1], 1)
```

```
Overwriting mapper1.py
```

```
In [126]: %%writefile combiner.py
          #!/usr/bin/python

          #HW3.4 In this question, we will emit a counter for everytime the combiner is called.
          # In this combiner code we aggregate the basket count and products that occur together.
          # This is done for intermediate aggregation

          import sys
          import re

          sys.stderr.write("reporter:counter:HW3_4,num_combiner,1\n")
          last_key = None
          total_count = 0
          basket_cnt = 0

          # input comes from STDIN (standard input)
          for line in sys.stdin:
              row = re.split(r'\t+',line.strip())
              product = (row[0],row[1]) #combine product1 and product2 as tuple to be used as key
              count = int(row[2])
              #increment total number of baskets if incoming data is for baskets
              if (product == ('*','BASKET')):
                  basket_cnt += count
                  continue
              else:
                  #if current key is same as last_key than increment count
                  if(last_key == product):
                      total_count += count
                  else:
                      if (last_key):
                          print '%s\t%s\t%s' %(last_key[0],last_key[1],total_count)
                      total_count = count
                      last_key = product
          if last_key == product:
              print '%s\t%s\t%s' %(last_key[0],last_key[1],total_count)

          print '%s\t%s\t%s' %('*','BASKET',basket_cnt)
```

```
Overwriting combiner.py
```

```
In [127]: %%writefile reducer1.py
          #!/usr/bin/python

          #HW3.4 In this question, we will emit a counter for everytime the reducer is called.
          # In this reducer code we aggregate the basket count and products that occur together.

          import sys
          import re


          sys.stderr.write("reporter:counter:HW3_4,num_reducers,1\n")
          last_key = None
          total_count = 0
```

```
            max_basket = -1
            basket_cnt = 0
            sup_cnt = 100
            # input comes from STDIN (standard input)
            for line in sys.stdin:
                row = re.split(r'\t+',line.strip())
                product = (row[0],row[1]) #combine product1 and product2 as tuple to be used as key
                count = int(row[2])
                #increment total number of baskets if incoming data is for baskets
                if (product ==('*','BASKET')):
                    basket_cnt += count
                    continue
                #increment the count for products that occur together
                else:
                    #if current key is same as last_key than increment count
                    if(last_key == product):
                        total_count += count
                    else:
                        #emit pair of product if they are greater than 100
                        if (last_key and total_count > sup_cnt):
                            print '%s\t%s\t%s' %(last_key[0],last_key[1],total_count)
                        total_count = count
                        last_key = product
            if (last_key == product and total_count > sup_cnt ):
                print '%s\t%s\t%s' %(last_key[0],last_key[1],total_count)
            #emit the total number of baskets
            print '%s\t%s\t%s' %('*','NUM_OF_BASKET',basket_cnt)
```

```
Overwriting reducer1.py
```

In [ ]:

Implementing second mapreduce job to determine top 50 product pairs which coccur

In [128]:
```
%%writefile mapper2.py
#!/usr/bin/python

#HW3.4 In this question, we will emit a counter for everytime the mapper is called is called.
# In the mapper code we will emit the frequency first followed by the product pair
# we are also going to tell hadoop to sort the key as numeric and in reverse order

import sys
import re

sys.stderr.write("reporter:counter:HW3_4b,num_mappers,1\n")
# input comes from STDIN (standard input)
for line in sys.stdin:
    #remove leading and trailing spaces
    row = re.split(r'\t',line.strip())
    print '%s\t%s\t%s' %(row[2],row[0],row[1])
```

```
Overwriting mapper2.py
```

In [129]:
```
%%writefile reducer2.py
#!/usr/bin/python
```

```
        #HW3.4 In this question, we will emit a counter for everytime the reducer is called is called
        # As we are sorting the data in descending order based on the frequency, we will get the numbe
        # after that we enter the top 50 product pair in the list and print the record
        import sys
        import re

        sys.stderr.write("reporter:counter:HW3_4b,num_reducers,1\n")

        n_max = 50
        a_max = []
        num_baskets =0

        # input comes from STDIN (standard input)
        for line in sys.stdin:
            row = re.split(r'\t+',line.strip())
            freq = int(row[0])
            key = (row[1],row[2])

            if(key==('*','NUM_OF_BASKET')):
                num_baskets = freq
                continue
            else:
                # add the lowest 10 words
                if len(a_max) < n_max:
                    a_max.append((key,freq,1.0*freq/num_baskets))

        for record in a_max:
            print record
```

```
Overwriting reducer2.py
```

```
In [135]: # function to run the hadoop job
          import time
          def hw3_4():

              #change properties of mapper.py
              !chmod a+x mapper1.py;chmod a+x reducer1.py;chmod a+x mapper2.py;chmod a+x reducer2.py;chm

              #remove output directory if present else hadoop job gives error
              !hdfs dfs -rm -r hw3/output/hw34b
              !hdfs dfs -rm -r hw3/output/hw34a

              start = time.time()

              #run hadoop job with mapper1 and reducer1
              !hadoop jar /usr/local/Cellar/hadoop/2.7.1/libexec/share/hadoop/tools/lib/hadoop-*streamin
              -D mapreduce.job.output.key.comparator.class=org.apache.hadoop.mapreduce.lib.partition.Key
              -D stream.num.map.output.key.fields=2 \
              -D mapreduce.partition.keypartitioner.options=-k1,1 \
              -D mapreduce.partition.keycomparator.options="-k1,1 -k2,2" \
              -D mapred.reduce.tasks=1 \
              -file mapper1.py \
              -mapper mapper1.py \
              -file reducer1.py \
```

```
        -reducer reducer1.py \
        -input hw3/src/ProductPurchaseData.txt \
        -output hw3/output/hw34a

        #run hadoop job with mapper2 and reducer2
        !hadoop jar /usr/local/Cellar/hadoop/2.7.1/libexec/share/hadoop/tools/lib/hadoop-*streamin
        -D mapreduce.job.output.key.comparator.class=org.apache.hadoop.mapreduce.lib.partition.Key
        -D stream.num.map.output.key.fields=3 \
        -D mapreduce.partition.keypartitioner.options=-k1,1 \
        -D mapreduce.partition.keycomparator.options="-k1,1nr -k2,2 -k3,3" \
        -D mapred.reduce.tasks=1 \
        -mapper mapper2.py \
        -file mapper2.py \
        -reducer reducer2.py \
        -file reducer2.py \
        -input hw3/output/hw34a/part* \
        -output hw3/output/hw34b

        print"\n Time taken to run this section of code in seconds: "
        print time.time() - start

        print "\n"
        !echo "HDFS Output"
        !hdfs dfs -cat hw3/output/hw34b/part*

    hw3_4()
```

```
16/02/01 22:20:19 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
rm: ‘hw3/output/hw34b’: No such file or directory
16/02/01 22:20:20 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
Deleted hw3/output/hw34a
16/02/01 22:20:22 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files
16/02/01 22:20:22 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
packageJobJar: [mapper1.py, reducer1.py, /var/folders/91/cjfxt7ys6958qll6vjtgwwfw0000gn/T/hadoop-unjar28
16/02/01 22:20:54 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files
16/02/01 22:20:54 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
packageJobJar: [mapper2.py, reducer2.py, /var/folders/91/cjfxt7ys6958qll6vjtgwwfw0000gn/T/hadoop-unjar57

 Time taken to run this section of code in seconds:
59.0430419445


HDFS Output
16/02/01 22:21:21 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
(('DAI62779', 'ELE17451'), 1592, 0.05118806469245362)
(('FRO40251', 'SNA80324'), 1412, 0.04540046943828173)
(('DAI75645', 'FRO40251'), 1254, 0.04032024693739751)
(('FRO40251', 'GRO85051'), 1213, 0.039001961351725026)
(('DAI62779', 'GRO73461'), 1139, 0.03662261663612103)
(('DAI75645', 'SNA80324'), 1130, 0.03633323687341243)
(('DAI62779', 'FRO40251'), 1070, 0.03440403845535513)
(('DAI62779', 'SNA80324'), 923, 0.029677502331114755)
(('DAI62779', 'DAI85309'), 918, 0.029516735796276648)
(('ELE32164', 'GRO59710'), 911, 0.029291662647503297)
```

```
(('DAI62779', 'DAI75645'), 882, 0.02835921674544227)
(('FRO40251', 'GRO73461'), 882, 0.02835921674544227)
(('DAI62779', 'ELE92920'), 877, 0.02819845021060416)
(('FRO40251', 'FRO92469'), 835, 0.026848011317964052)
(('DAI62779', 'ELE32164'), 832, 0.026751551397061188)
(('DAI75645', 'GRO73461'), 712, 0.022893154560946594)
(('DAI43223', 'ELE32164'), 711, 0.022861001253978972)
(('DAI62779', 'GRO30386'), 709, 0.02279669464004373)
(('ELE17451', 'FRO40251'), 697, 0.022410854956432268)
(('DAI85309', 'ELE99737'), 659, 0.021189029291662647)
(('DAI62779', 'ELE26917'), 650, 0.020899649528954053)
(('GRO21487', 'GRO73461'), 631, 0.02028873669656924)
(('DAI62779', 'SNA45677'), 604, 0.019420597408443457)
(('ELE17451', 'SNA80324'), 597, 0.019195524259670107)
(('DAI62779', 'GRO71621'), 595, 0.019131217645734864)
(('DAI62779', 'SNA55762'), 593, 0.01906691103179962)
(('DAI62779', 'DAI83733'), 586, 0.01884183788302627)
(('ELE17451', 'GRO73461'), 580, 0.018648918041220538)
(('GRO73461', 'SNA80324'), 562, 0.01807015851580335)
(('DAI62779', 'GRO59710'), 561, 0.01803800520883573)
(('DAI62779', 'FRO80039'), 550, 0.01768431883219189)
(('DAI75645', 'ELE17451'), 547, 0.017587858911289025)
(('DAI62779', 'SNA93860'), 537, 0.01726632584161281)
(('DAI55148', 'DAI62779'), 526, 0.016912639464968973)
(('DAI43223', 'GRO59710'), 512, 0.01646249316742227)
(('ELE17451', 'ELE32164'), 511, 0.016430339860454647)
(('DAI62779', 'SNA18336'), 506, 0.01626957332561654)
(('ELE32164', 'GRO73461'), 486, 0.015626507186264106)
(('DAI62779', 'FRO78087'), 482, 0.01549789395839362)
(('DAI85309', 'ELE17451'), 482, 0.01549789395839362)
(('DAI62779', 'GRO94758'), 479, 0.015401434037490756)
(('DAI62779', 'GRO21487'), 471, 0.015144207581749784)
(('GRO85051', 'SNA80324'), 471, 0.015144207581749784)
(('ELE17451', 'GRO30386'), 468, 0.01504774760846917)
(('FRO85978', 'SNA95666'), 463, 0.01488698112600881)
(('DAI62779', 'FRO19221'), 462, 0.014854827819041188)
(('DAI62779', 'GRO46854'), 461, 0.014822674512073567)
(('DAI43223', 'DAI62779'), 459, 0.014758367898138324)
(('ELE92920', 'SNA18336'), 455, 0.014629754670267838)
(('DAI88079', 'FRO40251'), 446, 0.014340374907559243)
```

In [ ]:

Computational setup used

1. Single node haddop cluster , running hadoop version 2.7.1. This was setup on my mac (intel i7, 4 cores)
2. Yarn is enabled and all mapreduce jobs are running in yarn mode
3. Yarn configured with a single node manager / slave. This node manager has 8GB capacity. Yarn vcore /cpu support was not enabled
4. Each map task is configured to use 1 GB resource (mapreduce.map.memory.mb = 1024, default value)
5. Each reduce task is configured to use 1 GB resource (mapreduce.reduce.memory.mb = 1024, default value)
6. Given this, it means that my yarn cluster could run at max 6 maps/reduce tasks concurrently (yarn.app.mapreduce.am.resource.mb = 1536, MR application master is using 2GB)

Total time taken = 59.0430419445 secs
From Job a :
Number of mapper = 2
Number of reducers = 1
Number of combiners = 0
From Job b :
Number of mapper = 2
Number of reducers = 1
Number of combiners = 0
Image from Job a
Image from Job b

In [ ]:

HW3.5: Stripes Repeat 3.4 using the stripes design pattern for finding cooccuring pairs.

Report the compute times for stripes job versus the Pairs job. Describe the computational setup used (E.g., single computer; dual core; linux, number of mappers, number of reducers)

Instrument your mapper, combiner, and reducer to count how many times each is called using Counters and report these counts. Discuss the differences in these counts between the Pairs and Stripes jobs

In [109]:
```
%%writefile mapper1.py
#!/usr/bin/python
import sys
import re

sys.stderr.write("reporter:counter:HW3_5a,num_mappers,1\n")
# input comes from STDIN (standard input)
for line in sys.stdin:
    # Emit * , BAKSET for counting total number of baskets
    print '%s\t%s\t%s'%('*','BASKET',1)
    # remove leading and trailing whitespace and tokenize
    token  =  line.strip().split(" ")
    uniq_tokens = sorted(set(token))
    for i in range (0,len(uniq_tokens)-1):
        key = uniq_tokens[i]
        value = ""
        for j in range(i+1, len(uniq_tokens)):
            value += uniq_tokens[j] + "\t1\t"
        print '%s\t%s' %(key,value)
```

```
Overwriting mapper1.py
```

In [110]:
```
%%writefile reducer1.py
#!/usr/bin/python
#H3.5 This reducer functions gets the total number of baskets as the first element from mappe
# which we are able to use it to calculate relative frequncy. I have used heapq to insert top
# This code was written before we got instructions to use 2 map reduce jobs. I have not remov
# Given that we need to sort data based  lexicographically the 2nd map reduce job helps with


import sys
import re
import heapq
import operator
```

```
sys.stderr.write("reporter:counter:HW3_5a,num_reducers,1\n")

last_key = None
word = None
total_count = 0
max_basket = -1
basket_cnt = 0
sup_cnt = 100 # variable to hold the minimum support count
tmp_dict ={} # for aggregation
n_max = 50 # top n product pairs
max_q = []

#function to add to queue.
#we have used heapq here so we can remove the smallest element if the queue is full during ins
def add_to_max_Q(p1,p2,cnt):
    # add elements if q is not full
    if (len(max_q) < n_max):
        heapq.heappush(max_q, (cnt, (p1,p2)))
    else:
        # add new element and then remove smallest element
        heapq.heappush(max_q, (cnt, (p1,p2)))
        heapq.heappop(max_q)

#function to print the queue data
def print_topn(total_bsk):
    cnt = len(max_q)
    if(cnt > n_max):
        cnt = n_max
    s = heapq.nlargest(cnt, max_q)
    for row in s:
        print '%s\t%s\t%s\t%s'%(row[1][0] ,row[1][1],row[0],(1.0*row[0]/total_bsk))

def merge_dict(merged,row):
    for product in row:
        if product in merged:
            merged[product] += row[product]
        else:
            merged[product] = row[product]
    return merged

def emit_dict(key,merged):
    output = key
    cnt =0
    merged_sorted = sorted(merged.items(), key=operator.itemgetter(0))
    for tup in merged_sorted:
        # for prd, value in merged_sort.iteritems():
        prd = tup[0]
        value = tup[1]
        if(value < sup_cnt):
            continue
        cnt += 1
        add_to_max_Q(key,prd,value)
        output += "\t" + prd + "\t" + str(value)
```

```
# input comes from STDIN (standard input)
for line in sys.stdin:
    row = re.split(r'\t',line.strip())
    product = row[0]
    #increment the total basket if we encounter * BASKET tuple
    if(row[0]=='*' and row[1]=='BASKET'):
        basket_cnt += int(row[2])

        continue

    #increment the count for products that occur together.
    else:
        row_dict ={}
        #parse the remaining data elements and increment counters by 2
        for x in range (1,len(row),2):
            prd = row[x]
            prd_cnt = int(row[x+1])
            row_dict[prd]=prd_cnt

        if(last_key == product):
            #update dictionary
            tmp_dict=merge_dict(tmp_dict,row_dict)

        #emit the data is we find new product key
        else:
            #emit pair of product if they are greater than 100
            if (last_key):
                #print dictionary
                emit_dict(last_key,tmp_dict)
            #create dictonary
            tmp_dict=row_dict
            last_key = product


if (last_key == product):
    #print dictionary
    emit_dict(last_key,tmp_dict)

#emit the total number of baskets
print '%s\t%s\t%s\t%s' %('*','NUM_OF_BASKET',basket_cnt,1)
# "\n %d Top product pairs that occur together" %(n_max)
print_topn(basket_cnt)
```

```
Overwriting reducer1.py
```

In [ ]:

In [111]:
```
%%writefile mapper2.py
#!/usr/bin/python

#HW3.5 In this question, we will emit a counter for everytime the mapper is called is called.
# In the mapper code we will emit the same data out as from previous output
# we are also going to tell hadoop to sort the key as numeric and in reverse order

import sys
```

```
import re

sys.stderr.write("reporter:counter:HW3_5b,num_mappers,1\n")
# input comes from STDIN (standard input)
for line in sys.stdin:
    #remove leading and trailing spaces
    row = re.split(r'\t+',line.strip())
    print '%s\t%s\t%s\t%s' %(row[0],row[1],row[2],row[3])
```

```
Overwriting mapper2.py
```

In [112]:
```
# function to run the hadoop job
import time
def hw3_5_b():

    #change properties of mapper.py
    !chmod a+x mapper1.py;chmod a+x reducer1.py;chmod a+x mapper2.py

    #remove output directory if present else hadoop job gives error
    !hdfs dfs -rm -r hw3/output/hw35b
    !hdfs dfs -rm -r hw3/output/hw35a

    start = time.time()
    #run hadoop job
    !hadoop jar /usr/local/Cellar/hadoop/2.7.1/libexec/share/hadoop/tools/lib/hadoop-*streamin
    -file mapper1.py \
    -mapper mapper1.py \
    -file reducer1.py \
    -reducer reducer1.py \
    -input hw3/src/ProductPurchaseData.txt \
    -output hw3/output/hw35a

    #run hadoop job
    !hadoop jar /usr/local/Cellar/hadoop/2.7.1/libexec/share/hadoop/tools/lib/hadoop-*streamin
    -D mapreduce.job.output.key.comparator.class=org.apache.hadoop.mapreduce.lib.partition.Key
    -D stream.num.map.output.key.fields=3 \
    -D mapreduce.partition.keypartitioner.options=-k1,1 \
    -D mapreduce.partition.keycomparator.options="-k3,3nr -k1,1 -k2,2" \
    -D mapred.reduce.tasks=1 \
    -mapper mapper2.py \
    -file mapper2.py \
    -reducer mapper2.py \
    -input hw3/output/hw35a/part* \
    -output hw3/output/hw35b

    print"\n Time taken to run this section of code in seconds: "
    print time.time() - start

    print "\n"
    !echo "HDFS Output"
    !hdfs dfs -cat hw3/output/hw35b/part*

hw3_5_b()
```

```
16/02/01 21:35:26 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
```

```
Deleted hw3/output/hw35b
16/02/01 21:35:28 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
Deleted hw3/output/hw35a
16/02/01 21:35:29 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files
16/02/01 21:35:29 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
packageJobJar: [mapper1.py, reducer1.py, /var/folders/91/cjfxt7ys6958qll6vjtgwwfw0000gn/T/hadoop-unjar68
16/02/01 21:35:51 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files
16/02/01 21:35:51 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
packageJobJar: [mapper2.py, /var/folders/91/cjfxt7ys6958qll6vjtgwwfw0000gn/T/hadoop-unjar84930405028939

 Time taken to run this section of code in seconds:
41.5408499241


HDFS Output
16/02/01 21:36:11 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
*       NUM_OF_BASKET       31101       1
DAI62779        ELE17451        1592        0.0511880646925
FRO40251        SNA80324        1412        0.0454004694383
DAI75645        FRO40251        1254        0.0403202469374
FRO40251        GRO85051        1213        0.0390019613517
DAI62779        GRO73461        1139        0.0366226166361
DAI75645        SNA80324        1130        0.0363332368734
DAI62779        FRO40251        1070        0.0344040384554
DAI62779        SNA80324        923         0.0296775023311
DAI62779        DAI85309        918         0.0295167357963
ELE32164        GRO59710        911         0.0292916626475
DAI62779        DAI75645        882         0.0283592167454
FRO40251        GRO73461        882         0.0283592167454
DAI62779        ELE92920        877         0.0281984502106
FRO40251        FRO92469        835         0.026848011318
DAI62779        ELE32164        832         0.0267515513971
DAI75645        GRO73461        712         0.0228931545609
DAI43223        ELE32164        711         0.022861001254
DAI62779        GRO30386        709         0.02279669464
ELE17451        FRO40251        697         0.0224108549564
DAI85309        ELE99737        659         0.0211890292917
DAI62779        ELE26917        650         0.020899649529
GRO21487        GRO73461        631         0.0202887366966
DAI62779        SNA45677        604         0.0194205974084
ELE17451        SNA80324        597         0.0191955242597
DAI62779        GRO71621        595         0.0191312176457
DAI62779        SNA55762        593         0.0190669110318
DAI62779        DAI83733        586         0.018841837883
ELE17451        GRO73461        580         0.0186489180412
GRO73461        SNA80324        562         0.0180701585158
DAI62779        GRO59710        561         0.0180380052088
DAI62779        FRO80039        550         0.0176843188322
DAI75645        ELE17451        547         0.0175878589113
DAI62779        SNA93860        537         0.0172663258416
DAI55148        DAI62779        526         0.016912639465
DAI43223        GRO59710        512         0.0164624931674
ELE17451        ELE32164        511         0.0164303398605
DAI62779        SNA18336        506         0.0162695733256
```

```
ELE32164        GRO73461        486        0.0156265071863
DAI62779        FRO78087        482        0.0154978939584
DAI85309        ELE17451        482        0.0154978939584
DAI62779        GRO94758        479        0.0154014340375
DAI62779        GRO21487        471        0.0151442075817
GRO85051        SNA80324        471        0.0151442075817
ELE17451        GRO30386        468        0.0150477476608
FRO85978        SNA95666        463        0.014886981126
DAI62779        FRO19221        462        0.014854827819
DAI62779        GRO46854        461        0.0148226745121
DAI43223        DAI62779        459        0.0147583678981
ELE92920        SNA18336        455        0.0146297546703
DAI88079        FRO40251        446        0.0143403749076
```

Computational setup used

1. Single node haddop cluster , running hadoop version 2.7.1. This was setup on my mac (intel i7, 4 cores)
2. Yarn is enabled and all mapreduce jobs are running in yarn mode
3. Yarn configured with a single node manager / slave. This node manager has 8GB capacity. Yarn vcore /cpu support was not enabled
4. Each map task is configured to use 1 GB resource (mapreduce.map.memory.mb = 1024, default value)
5. Each reduce task is configured to use 1 GB resource (mapreduce.reduce.memory.mb = 1024, default value)
6. Given this, it means that my yarn cluster could run at max 6 maps/reduce tasks concurrently (yarn.app.mapreduce.am.resource.mb = 1536, MR application master is using 2GB)

Total time taken =
From Job a :
a.Time Taken : 23.9189631939 secs

b. Number of mapper = 2

c. Number of reducers = 1

d. Number of combiners = 0

From Job b :

a. Time Taken : 17.4585750103 secs

b. Number of mapper = 2

c. Number of reducers = 1

d. Number of combiners = 0

Image from Job a
Image from Job b

In [ ]: