MIDS-W261-HW-03-TEMPLATE

September 20, 2016

1 DATSCIW261 ASSIGNMENT

Version 2016-01-27 (FINAL) Week 3 ASSIGNMENTS

Link: https://docs.google.com/spreadsheets/d/1ncFQl5Tovn-16slD8mYjP_nzMTPSfiGeLLzW8v_sMjg/edit?usp=sharing

1.1 HW3.0.

1. How do you merge two sorted lists/arrays of records of the form [key, value]?
2. Where is this used in Hadoop MapReduce? [Hint within the shuffle]
3. What is a combiner function in the context of Hadoop?
4. Give an example where it can be used and justify why it should be used in the context of this problem.
5. What is the Hadoop shuffle?

1.2 HW3.1 consumer complaints dataset: 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 [3]:
```
%%bash

        sudo touch mapper31.py
        sudo touch reducer31.py
        chmod -R 777 /home/cloudera/share/HW3
```

In [9]:
```
%%writefile mapper31.py
        #!/usr/bin/env python
        import sys

        sys.stderr.write("reporter:counter:MapperCounters,Calls,1\n")

        for line in sys.stdin:
            if "debt" in line:
                sys.stderr.write("reporter:counter:DebtCounter,Total,1\n")
            elif "mortgage" in line:
                sys.stderr.write("reporter:counter:MortgageCounter,Total,1\n")
            else:
                sys.stderr.write("reporter:counter:OtherCounter,Total,1\n")
            print line
```

```
Overwriting mapper31.py
```

In [10]:
```
%%writefile reducer31.py
         #!/usr/bin/env python
         import sys

         sys.stderr.write("reporter:counter:ReducerCounters,Calls,1\n")
         for line in sys.stdin:
             print line
```

```
Overwriting reducer31.py
```

In [11]:
```
!hdfs dfs -rm Consumer_complaints.csv
         !hdfs dfs -copyFromLocal /home/cloudera/share/HW3/Consumer_complaints.csv
         !hdfs dfs -rm -r consumer-complaints
         #usr/local/Cellar/hadoop/2.6.0/libexec/share/hadoop/tools/lib
         dataDir = "/home/cloudera/share/HW3"

         !hadoop jar /usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-streaming-2.6.0-mr1-cdh5.8
            -mapper /home/cloudera/share/HW3/mapper31.py \
```

```
    -reducer /home/cloudera/share/HW3/reducer31.py \
    -combiner /home/cloudera/share/HW3/reducer31.py \
    -input Consumer_complaints.csv \
    -output consumer-complaints \
    -numReduceTasks 3
    #--D mapreduce.job.reduces=2  depecated
  #-input historical_tours.txt  file on Hadoop

Deleted Consumer_complaints.csv
Deleted consumer-complaints
packageJobJar: [] [/usr/lib/hadoop-mapreduce/hadoop-streaming-2.6.0-cdh5.8.0.jar] /tmp/streamjob4348378
16/09/18 09:59:16 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
16/09/18 09:59:17 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
16/09/18 09:59:17 INFO mapred.FileInputFormat: Total input paths to process : 1
16/09/18 09:59:17 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedException
        at java.lang.Object.wait(Native Method)
        at java.lang.Thread.join(Thread.java:1281)
        at java.lang.Thread.join(Thread.java:1355)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.java:862)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutputStream.java:600)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:789)
16/09/18 09:59:17 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedException
        at java.lang.Object.wait(Native Method)
        at java.lang.Thread.join(Thread.java:1281)
        at java.lang.Thread.join(Thread.java:1355)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.java:862)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutputStream.java:600)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:789)
16/09/18 09:59:17 INFO mapreduce.JobSubmitter: number of splits:2
16/09/18 09:59:18 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1474147998669_0005
16/09/18 09:59:18 INFO impl.YarnClientImpl: Submitted application application_1474147998669_0005
16/09/18 09:59:18 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/a
16/09/18 09:59:18 INFO mapreduce.Job: Running job: job_1474147998669_0005
16/09/18 09:59:28 INFO mapreduce.Job: Job job_1474147998669_0005 running in uber mode : false
16/09/18 09:59:28 INFO mapreduce.Job:  map 0% reduce 0%
16/09/18 09:59:46 INFO mapreduce.Job:  map 33% reduce 0%
16/09/18 09:59:47 INFO mapreduce.Job:  map 67% reduce 0%
16/09/18 09:59:50 INFO mapreduce.Job:  map 83% reduce 0%
16/09/18 09:59:51 INFO mapreduce.Job:  map 100% reduce 0%
16/09/18 10:00:11 INFO mapreduce.Job:  map 100% reduce 23%
16/09/18 10:00:15 INFO mapreduce.Job:  map 100% reduce 28%
16/09/18 10:00:16 INFO mapreduce.Job:  map 100% reduce 33%
16/09/18 10:00:17 INFO mapreduce.Job:  map 100% reduce 56%
16/09/18 10:00:21 INFO mapreduce.Job:  map 100% reduce 80%
16/09/18 10:00:25 INFO mapreduce.Job:  map 100% reduce 84%
16/09/18 10:00:26 INFO mapreduce.Job:  map 100% reduce 89%
16/09/18 10:00:28 INFO mapreduce.Job:  map 100% reduce 100%
16/09/18 10:00:28 INFO mapreduce.Job: Job job_1474147998669_0005 completed successfully
16/09/18 10:00:28 INFO mapreduce.Job: Counters: 55
        File System Counters
                FILE: Number of bytes read=56163339
                FILE: Number of bytes written=112921407
```

```
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=50910824
        HDFS: Number of bytes written=55600181
        HDFS: Number of read operations=15
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=6
Job Counters
        Killed reduce tasks=1
        Launched map tasks=2
        Launched reduce tasks=4
        Data-local map tasks=2
        Total time spent by all maps in occupied slots (ms)=39370
        Total time spent by all reduces in occupied slots (ms)=97236
        Total time spent by all map tasks (ms)=39370
        Total time spent by all reduce tasks (ms)=97236
        Total vcore-seconds taken by all map tasks=39370
        Total vcore-seconds taken by all reduce tasks=97236
        Total megabyte-seconds taken by all map tasks=40314880
        Total megabyte-seconds taken by all reduce tasks=99569664
Map-Reduce Framework
        Map input records=312913
        Map output records=625826
        Map output bytes=52126201
        Map output materialized bytes=56163357
        Input split bytes=242
        Combine input records=625826
        Combine output records=1251652
        Reduce input groups=625827
        Reduce shuffle bytes=56163357
        Reduce input records=1251652
        Reduce output records=2503304
        Spilled Records=2503304
        Shuffled Maps =6
        Failed Shuffles=0
        Merged Map outputs=6
        GC time elapsed (ms)=1195
        CPU time spent (ms)=21400
        Physical memory (bytes) snapshot=868319232
        Virtual memory (bytes) snapshot=7530622976
        Total committed heap usage (bytes)=513482752
DebtCounter
        Total=29092
MapperCounters
        Calls=2
MortgageCounter
        Total=120917
OtherCounter
        Total=162904
ReducerCounters
        Calls=9
Shuffle Errors
        BAD_ID=0
```

```
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=50910582
        File Output Format Counters
                Bytes Written=55600181
16/09/18 10:00:28 INFO streaming.StreamJob: Output directory: consumer-complaints
```

1.2.1 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.

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.

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 [15]:
```
%%bash
sudo touch generic_word_count.py
chmod -R 777 /home/cloudera/share/HW3/
```

In [28]:
```
%%writefile generic_word_count.py
#!/usr/bin/env python

#Mapper and reducer with mrjob just counting calls to each component, just using single line t

from mrjob.job import MRJob
from mrjob.step import MRStep
import re

wordRe = re.compile(r"[\w]+")

class MRWordFrequencyCount(MRJob):

    def mapper(self, _, line):
        self.increment_counter('group','num_mapper_calls',1)
        for word in wordRe.findall(line):
            yield word.lower(), 1
```

```
            def reducer(self, key, values):
                self.increment_counter('group','num_reducer_calls',1)
                yield key, sum(values)


        if __name__ == '__main__':
            MRWordFrequencyCount.run()

Overwriting generic_word_count.py
```

```
In [29]: !echo "foo foo quux labs foo bar quux" > oneLine.txt
         !chmod 777 oneLine.txt
         !python generic_word_count.py oneLine.txt

No configs found; falling back on auto-configuration
Creating temp directory /tmp/generic_word_count.cloudera.20160918.180325.192748
Running step 1 of 1...
Counters: 1
        group
                num_mapper_calls=1
Counters: 2
        group
                num_mapper_calls=1
                num_reducer_calls=4
Streaming final output from /tmp/generic_word_count.cloudera.20160918.180325.192748/output...
"bar"        1
"foo"        3
"labs"        1
"quux"        2
Removing temp directory /tmp/generic_word_count.cloudera.20160918.180325.192748...
```

We see that the mapper has been called one time while the reducer has been called four times. This makes sense because the one-line file is so sort that it only requires a single mapper, and there are four unique words, or keys, so each one will get shuffled to a separate reducer, thus giving us four reducer calls.

```
In [30]: %%writefile complaints_word_count.py
         #!/usr/bin/env python

         #Same as above but we run on complaints data and isolate the issue column

         from mrjob.job import MRJob
         import re
         from mrjob.step import MRStep

         wordRe = re.compile(r"[\w]+")

         class MRComplaintFrequencyCount(MRJob):

             def mapper(self, _, line):
                 self.increment_counter('group','num_mapper_calls',1)

                 #Issue is third column in csv
                 issue = line.split(",")[3]

                 for word in wordRe.findall(issue):
```

```
                    yield word.lower(), 1

            def reducer(self, key, values):
                self.increment_counter('group','num_reducer_calls',1)
                yield key, sum(values)


        if __name__ == '__main__':
            MRComplaintFrequencyCount.run()
```

```
Writing complaints_word_count.py
```

```
In [44]: %%bash
         sudo mkdir complaints
         sudo chmod -R 777 complaints/
         python complaints_word_count.py Consumer_complaints.csv --output-dir complaints | head -10
```

```
"a"        3503
"account"        20681
"acct"        163
"action"        2505
"advance"        240
"advertising"        1193
"amount"        98
"amt"        71
"an"        2505
"and"        16448

mkdir: cannot create directory ‘complaints’: File exists
No configs found; falling back on auto-configuration
Running step 1 of 1...
Creating temp directory /tmp/complaints_word_count.cloudera.20160920.132736.913573
Counters: 1
        group
                num_mapper_calls=312913
Counters: 2
        group
                num_mapper_calls=312913
                num_reducer_calls=172
Streaming final output from complaints...
Removing temp directory /tmp/complaints_word_count.cloudera.20160920.132736.913573...
```

```
In [33]: from complaints_word_count import MRComplaintFrequencyCount

         mrJob = MRComplaintFrequencyCount(args=["Consumer_complaints.csv"])

         with mrJob.make_runner() as runner:
             runner.run()
             print runner.counters()
```

```
[{'group': {'num_mapper_calls': 312913, 'num_reducer_calls': 172}}]
```

```
In [56]: %%writefile complaints_word_count_combiner.py
         #!/usr/bin/env python

         #Same as above but we run on complaints data and isolate the issue column
```

```
        from mrjob.job import MRJob
        import re
        from mrjob.step import MRStep

        wordRe = re.compile(r"[\w]+")

        class MRComplaintFrequencyCount(MRJob):

            def mapper(self, _, line):
                self.increment_counter('group','num_mapper_calls',1)

                #Issue is third column in csv
                issue = line.split(",")[3]

                for word in wordRe.findall(issue):
                    yield word.lower(), 1

            def reducer(self, key, values):
                self.increment_counter('group','num_reducer_calls',1)
                yield key, sum(values)

            def combiner(self, key, values):
                self.increment_counter('group','num_combiner_calls',1)
                yield key, sum(values)


        if __name__ == '__main__':
            MRComplaintFrequencyCount.run()
```

```
Overwriting complaints_word_count_combiner.py
```

```
In [57]: %%bash

         python complaints_word_count_combiner.py Consumer_complaints.csv --output-dir complaints
         sudo chmod -R 777 complaints/
```

```
"a"        3503
"account"        20681
"acct"        163
"action"        2505
"advance"        240
"advertising"        1193
"amount"        98
"amt"        71
"an"        2505
"and"        16448
"application"        8868
"applied"        139
"apply"        118
"apr"        3431
"arbitration"        168
"are"        3821
"atm"        2422
"attempts"        11848
```

```
"available"        274
"balance"        597
"bank"        202
"bankruptcy"        222
"being"        5663
"billing"        8158
"by"        5663
"can"        1999
"cancelling"        2795
"card"        4405
"cash"        240
"caused"        5663
"changes"        350
"charged"        976
"charges"        131
"checks"        75
"closing"        2795
"club"        12545
"collect"        11848
"collection"        1907
"communication"        6920
"company"        4858
"cont"        11848
"contact"        3053
"convenience"        75
"costs"        4350
"credit"        55251
"credited"        92
"customer"        2734
"d"        11848
"day"        71
"dealing"        1944
"debit"        2422
"debt"        19309
"decision"        2774
"decrease"        1149
"delay"        243
"delinquent"        1061
"deposits"        10555
"determination"        1490
"did"        139
"didn"        925
"disclosure"        5214
"disclosures"        64
"dispute"        904
"disputes"        6938
"embezzlement"        3276
"expect"        807
"false"        2508
"fee"        3198
"fees"        807
"for"        929
"forbearance"        350
"fraud"        3842
```

```
"funds"        5663
"get"        4357
"getting"        291
"health"        12545
"i"        925
"identity"        4729
"illegal"        2505
"improper"        4309
"incorrect"        29133
"increase"        1149
"info"        2896
"information"        29069
"interest"        4238
"investigation"        4858
"issuance"        640
"issue"        1099
"issues"        538
"late"        1797
"lease"        6337
"lender"        2165
"line"        1732
"loan"        119630
"low"        5663
"making"        3226
"managing"        5006
"marketing"        1193
"missing"        64
"modification"        70487
"money"        413
"monitoring"        1453
"my"        10731
"not"        12353
"of"        10885
"on"        29069
"opening"        16205
"or"        22533
"other"        7886
"out"        1242
"overlimit"        127
"owed"        11848
"pay"        3821
"payment"        92
"payments"        3226
"payoff"        1155
"plans"        350
"practices"        1003
"privacy"        240
"problems"        9484
"process"        5505
"processing"        243
"promised"        274
"protection"        4139
"rate"        3431
"receive"        139
```

```
"received"        216
"receiving"        3226
"relations"        1367
"repay"        1647
"repaying"        3844
"report"        34903
"reporting"        6559
"representation"        2508
"rewards"        1002
"s"        4858
"sale"        139
"scam"        566
"score"        4357
"service"        1518
"servicer"        1944
"servicing"        36767
"settlement"        4350
"sharing"        2832
"shopping"        672
"statement"        1220
"statements"        2508
"stop"        131
"t"        2924
"tactics"        6920
"taking"        3747
"terms"        350
"the"        6248
"theft"        3276
"threatening"        2505
"to"        8401
"transaction"        1485
"transfer"        597
"unable"        8178
"underwriting"        2774
"unsolicited"        640
"use"        1477
"using"        2422
"verification"        5214
"was"        274
"when"        4095
"with"        1944
"withdrawals"        10555
"workout"        350
"wrong"        169
"you"        3821
"your"        3844

No configs found; falling back on auto-configuration
Running step 1 of 1...
Creating temp directory /tmp/complaints_word_count_combiner.cloudera.20160920.135940.261521
Counters: 2
        group
                num_combiner_calls=318
                num_mapper_calls=312913
```

```
Counters: 3
        group
                num_combiner_calls=318
                num_mapper_calls=312913
                num_reducer_calls=172
Streaming final output from complaints...
Removing temp directory /tmp/complaints_word_count_combiner.cloudera.20160920.135940.261521...
```

In [9]:
```
%%bash

        cat complaints/* | sort -k2,2 -g | tail -50
        printf "\n"
        cat complaints/* | sort -k2,2 -g | head -10
```

```
"identity"        4729
"company"        4858
"investigation"        4858
"s"        4858
"managing"        5006
"disclosure"        5214
"verification"        5214
"process"        5505
"being"        5663
"by"        5663
"caused"        5663
"funds"        5663
"low"        5663
"the"        6248
"lease"        6337
"reporting"        6559
"communication"        6920
"tactics"        6920
"disputes"        6938
"other"        7886
"billing"        8158
"unable"        8178
"to"        8401
"application"        8868
"problems"        9484
"deposits"        10555
"withdrawals"        10555
"my"        10731
"of"        10885
"attempts"        11848
"collect"        11848
"cont"        11848
"d"        11848
"owed"        11848
"not"        12353
"club"        12545
"health"        12545
"opening"        16205
"and"        16448
"debt"        19309
"account"        20681
```

```
"or"        22533
"information"        29069
"on"        29069
"incorrect"        29133
"report"        34903
"servicing"        36767
"credit"        55251
"modification"        70487
"loan"        119630

"disclosures"        64
"missing"        64
"amt"        71
"day"        71
"checks"        75
"convenience"        75
"credited"        92
"payment"        92
"amount"        98
"apply"        118
```

3.2.1 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.**

```
In [114]: %%writefile complaints_word_count_combiner.py
          #!/usr/bin/env python

          #Same as above but we run on complaints data and isolate the issue column

          from mrjob.job import MRJob
          import re
          from mrjob.step import MRStep
          from collections import defaultdict

          wordRe = re.compile(r"[\w]+")

          class MRComplaintFrequencyCount(MRJob):

              def mapper(self, _, line):
                  self.increment_counter('group','num_mapper_calls',1)

                  #Issue is third column in csv
                  issue = line.split(",")[3]

                  for word in wordRe.findall(issue):
                      #Send all map outputs to same reducer
                      yield word.lower(), 1

              def reducer(self, key, values):
                  self.increment_counter('group','num_reducer_calls',1)
                  wordCounts = defaultdict(int)
                  total = 0
```

```
                    for value in values:
                        word, count = value
                        total+=count
                        wordCounts[word]+=count

                    for k,v in wordCounts.iteritems():
                        yield k, (v, float(v)/total)

                def combiner(self, key, values):
                    self.increment_counter('group','num_combiner_calls',1)
                    yield None, (key, sum(values))


            if __name__ == '__main__':
                MRComplaintFrequencyCount.run()

Overwriting complaints_word_count_combiner.py

In [121]: %%bash
            sudo rm complaints/part*
            sudo chmod -R 777 complaints/
            python complaints_word_count_combiner.py Consumer_complaints.csv  --jobconf mapred.reduce.tasl

"unsolicited"        [640, 0.0006399276881712367]
"being"        [5663, 0.005662360153302677]
"caused"        [5663, 0.005662360153302677]
"scam"        [566, 0.0005659360492264374]
"embezzlement"        [3276, 0.0032756298538265177]
"report"        [34903, 0.03489905640662605]
"attempts"        [11848, 0.011846661327270018]
"settlement"        [4350, 0.004349508505538874]
"underwriting"        [2774, 0.002773686573417204]
"issues"        [538, 0.0005379392128689459]

No configs found; falling back on auto-configuration
Running step 1 of 1...
Creating temp directory /tmp/complaints_word_count_combiner.cloudera.20160920.175242.965136
Counters: 2
        group
                num_combiner_calls=318
                num_mapper_calls=312913
Counters: 3
        group
                num_combiner_calls=318
                num_mapper_calls=312913
                num_reducer_calls=1
Streaming final output from complaints...
Removing temp directory /tmp/complaints_word_count_combiner.cloudera.20160920.175242.965136...
Traceback (most recent call last):
  File "complaints_word_count_combiner.py", line 42, in <module>
    MRComplaintFrequencyCount.run()
  File "/home/cloudera/anaconda2/lib/python2.7/site-packages/mrjob/job.py", line 429, in run
    mr_job.execute()
  File "/home/cloudera/anaconda2/lib/python2.7/site-packages/mrjob/job.py", line 447, in execute
    super(MRJob, self).execute()
```

```
  File "/home/cloudera/anaconda2/lib/python2.7/site-packages/mrjob/launch.py", line 158, in execute
    self.run_job()
  File "/home/cloudera/anaconda2/lib/python2.7/site-packages/mrjob/launch.py", line 238, in run_job
    self.stdout.flush()
IOError: [Errno 32] Broken pipe
```

```
In [25]: %%bash
         printf "Top 50\n\n"
         cat complaints/part* | sort -k3,3 -gr | head -50
         printf "\nBottom 10\n\n"
         cat complaints/part* | sort -k3,3 -gr | tail -10
```

```
Top 50

"loan"        [119630, 0.11961648333738288]
"modification"        [70487, 0.07047903586894681]
"credit"        [55251, 0.055244757342420306]
"servicing"        [36767, 0.03676284579842478]
"report"        [34903, 0.03489905640662605]
"incorrect"        [29133, 0.029129708342957247]
"on"        [29069, 0.029065715574140123]
"information"        [29069, 0.029065715574140123]
"or"        [22533, 0.02253045405869137]
"account"        [20681, 0.020678663311045852]
"debt"        [19309, 0.019306818329528762]
"and"        [16448, 0.016446141586000784]
"opening"        [16205, 0.016203169041898266]
"health"        [12545, 0.012543582575169005]
"club"        [12545, 0.012543582575169005]
"not"        [12353, 0.012351604268717635]
"owed"        [11848, 0.011846661327270018]
"d"        [11848, 0.011846661327270018]
"cont"        [11848, 0.011846661327270018]
"collect"        [11848, 0.011846661327270018]
"attempts"        [11848, 0.011846661327270018]
"of"        [10885, 0.010883770133974862]
"my"        [10731, 0.010729787534008658]
"withdrawals"        [10555, 0.010553807419761568]
"deposits"        [10555, 0.010553807419761568]
"problems"        [9484, 0.009482928429087514]
"application"        [8868, 0.008866998029222698]
"to"        [8401, 0.008400050794260249]
"unable"        [8178, 0.008177075990413084]
"billing"        [8158, 0.008157078250157733]
"other"        [7886, 0.007885108982684956]
"disputes"        [6938, 0.006937216094581312]
"tactics"        [6920, 0.006919218283514965]
"communication"        [6920, 0.006919218283514965]
"reporting"        [6559, 0.006558258916742408 5]
"lease"        [6337, 0.00633628399990801]
"the"        [6248, 0.006247294055771698]
"low"        [5663, 0.005662360153302677]
"funds"        [5663, 0.005662360153302677]
"caused"        [5663, 0.005662360153302677]
"by"        [5663, 0.005662360153302677]
```

```
"being"        [5663, 0.005662360153302677]
"process"        [5505, 0.005504378005285403]
"verification"        [5214, 0.005213410884570044]
"disclosure"        [5214, 0.005213410884570044]
"managing"        [5006, 0.005005434385914392]
"s"        [4858, 0.004857451108024794]
"investigation"        [4858, 0.004857451108024794]
"company"        [4858, 0.004857451108024794]
"identity"        [4729, 0.004728465683377778]

Bottom 10

"apply"        [118, 0.00011798666750657176]
"amount"        [98, 9.798892725122061e-05]
"payment"        [92, 9.198960517461527e-05]
"credited"        [92, 9.198960517461527e-05]
"convenience"        [75, 7.499152595756679e-05]
"checks"        [75, 7.499152595756679e-05]
"day"        [71, 7.099197790649657e-05]
"amt"        [71, 7.099197790649657e-05]
"missing"        [64, 6.399276881712366e-05]
"disclosures"        [64, 6.399276881712366e-05]
```

1.3 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 guided by the following questions:.

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 [26]: !wget https://www.dropbox.com/s/zlfyiwa70poqg74/ProductPurchaseData.txt?dl=0 > ProductPurchasel

--2016-09-20 13:50:17--  https://www.dropbox.com/s/zlfyiwa70poqg74/ProductPurchaseData.txt?dl=0
Resolving www.dropbox.com... 162.125.4.1
Connecting to www.dropbox.com|162.125.4.1|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://dl.dropboxusercontent.com/content_link/QcwNhjKDBiQDoItffqe7QmQhIaSkS6AkVVLrrUNiUcUhEYM
--2016-09-20 13:50:19--  https://dl.dropboxusercontent.com/content_link/QcwNhjKDBiQDoItffqe7QmQhIaSkS6Ak
Resolving dl.dropboxusercontent.com... 108.160.173.69
```

```
Connecting to dl.dropboxusercontent.com|108.160.173.69|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3458517 (3.3M) [text/plain]
Saving to: 'ProductPurchaseData.txt?dl=0'

ProductPurchaseData 100%[===================>]   3.30M  6.94MB/s    in 0.5s

2016-09-20 13:50:20 (6.94 MB/s) - 'ProductPurchaseData.txt?dl=0' saved [3458517/3458517]
```

```
In [27]: !sudo touch shopping_cart.py
         !sudo chmod 777 shopping_cart.py
```

```
Password:
Password:
```

```
In [140]: %%writefile shopping_cart.py
          #!/usr/bin/env python

          #Same as above but we run on complaints data and isolate the issue column

          from mrjob.job import MRJob
          import re
          from mrjob.step import MRStep
          from collections import defaultdict

          WORD_RE = re.compile(r"[\w']+")

          class MRShoppingCart(MRJob):

              def mapper_get_words(self, _, line):
                  # yield each word in the line
                  basket = len(WORD_RE.findall(line))
                  for word in WORD_RE.findall(line):
                      yield (word.lower(), (1,basket))

              def combiner_count_words(self, word, counts):
                  # sum the words we've seen so far
                  #I want to sum the words but also hold on to the basket
                  total=0
                  for count in counts:
                      sums, basket = count
                      total+=sums
                  yield word, (total, basket)

              def reducer_count_words(self, word, counts):
                  # send all (num_occurrences, word) pairs to the same reducer.
                  # num_occurrences is so we can easily use Python's max() function.
                  self.increment_counter('group','num_reducer_calls',1)
                  wordCount = 0
                  basketSize = 0
                  for count in counts:
                      sums, basket = count
                      basketSize = basket if basket > basketSize else basketSize
                      wordCount+=sums
```

```
                yield None, (word, wordCount, basketSize)

            # discard the key; it is just None
            def reducer_find_max_word(self, _, product_counts):
                # each item of word_count_pairs is (count, word),
                # so yielding one results in key=counts, value=word
                prodCounts = defaultdict(int)
                maxBasket = 0
                total=0
                for count in product_counts:
                    prod, prodCount, basket = count
                    maxBasket = basket if basket > maxBasket else maxBasket
                    prodCounts[prod]+=prodCount
                    total+=prodCount

                for k,v in prodCounts.iteritems():
                    yield k, (v, float(v)/total, maxBasket)

            def steps(self):
                return [
                    MRStep(mapper=self.mapper_get_words,
                           combiner=self.combiner_count_words,
                           reducer=self.reducer_count_words),
                    MRStep(reducer=self.reducer_find_max_word)
                ]


        if __name__ == '__main__':
            MRShoppingCart.run()

Overwriting shopping_cart.py

In [141]: !sudo chmod -R 777 shopping/
          !python shopping_cart.py ProductPurchaseData.txt  --jobconf mapred.reduce.tasks=1 --output-di

No configs found; falling back on auto-configuration
Running step 1 of 2...
Creating temp directory /tmp/shopping_cart.cloudera.20160920.195409.792109
Counters: 1
        group
                num_reducer_calls=12592
Running step 2 of 2...
Streaming final output from shopping...
"sna56940"      [2, 5.251769846438249e-06, 37]
"dai65887"      [1, 2.6258849232191246e-06, 37]
"gro63887"      [1, 2.6258849232191246e-06, 37]
"gro15558"      [1, 2.6258849232191246e-06, 37]
"fro15018"      [2, 5.251769846438249e-06, 37]
"sna27421"      [9, 2.3632964308972124e-05, 37]
"sna40906"      [2, 5.251769846438249e-06, 37]
"sna64909"      [1, 2.6258849232191246e-06, 37]
"fro65893"      [1, 2.6258849232191246e-06, 37]
"fro39997"      [7, 1.8381194462533874e-05, 37]
Removing temp directory /tmp/shopping_cart.cloudera.20160920.195409.792109...
Traceback (most recent call last):
```

```
  File "shopping_cart.py", line 68, in <module>
    MRShoppingCart.run()
  File "/home/cloudera/anaconda2/lib/python2.7/site-packages/mrjob/job.py", line 429, in run
    mr_job.execute()
  File "/home/cloudera/anaconda2/lib/python2.7/site-packages/mrjob/job.py", line 447, in execute
    super(MRJob, self).execute()
  File "/home/cloudera/anaconda2/lib/python2.7/site-packages/mrjob/launch.py", line 158, in execute
    self.run_job()
  File "/home/cloudera/anaconda2/lib/python2.7/site-packages/mrjob/launch.py", line 237, in run_job
    self.stdout.write(line)
IOError: [Errno 32] Broken pipe
```

```
In [28]: %%bash
         printf "Top 50\n\n"
         cat shopping//part* | sort -k3,3 -gr | head -50
         printf "\nBottom 10\n\n"
         cat shopping//part* | sort -k3,3 -gr | tail -10
```

```
Top 50

"dai62779"        [6667, 0.017506774783101905, 37]
"fro40251"        [3881, 0.010191059387013424, 37]
"ele17451"        [3875, 0.010175304077474108, 37]
"gro73461"        [3602, 0.009458437493435288, 37]
"sna80324"        [3044, 0.007993193706279015, 37]
"ele32164"        [2851, 0.007486397916097725, 37]
"dai75645"        [2736, 0.007184421149927526, 37]
"sna45677"        [2455, 0.006446547486502951, 37]
"fro31317"        [2330, 0.006118311871100561, 37]
"dai85309"        [2293, 0.006021154128941453, 37]
"ele26917"        [2292, 0.006018528244018234, 37]
"fro80039"        [2233, 0.005863601033548306, 37]
"gro21487"        [2115, 0.005553746612608449, 37]
"sna99873"        [2083, 0.005469718295065437, 37]
"gro59710"        [2004, 0.005262273386131126, 37]
"gro71621"        [1920, 0.005041690525807195, 37]
"fro85978"        [1918, 0.005036447282734282, 37]
"gro30386"        [1840, 0.00483162825872319, 37]
"ele74009"        [1816, 0.004768607020565931, 37]
"gro56726"        [1784, 0.004684578703022918 5, 37]
"dai63921"        [1773, 0.004655693968867509, 37]
"gro46854"        [1756, 0.004611053925172783, 37]
"ele66600"        [1713, 0.004498140873474361, 37]
"dai83733"        [1712, 0.004495514988551142, 37]
"fro32293"        [1702, 0.004469256139318951, 37]
"ele66810"        [1697, 0.004456126714702854 5, 37]
"sna55762"        [1646, 0.00432220658361868, 37]
"dai22177"        [1627, 0.004272314770077516, 37]
"fro78087"        [1531, 0.00402022981744848, 37]
"ele99737"        [1516, 0.003980841543600193, 37]
"gro94758"        [1489, 0.003909942650673277, 37]
"ele34057"        [1489, 0.003909942650673277, 37]
"fro35904"        [1436, 0.003770770749742663 5, 37]
"fro53271"        [1420, 0.003728756590971157, 37]
"sna93860"        [1407, 0.003694620086969308 5, 37]
```

```
"sna90094"        [1390, 0.0036499800432745837, 37]
"gro38814"        [1352, 0.0035501964161922567, 37]
"ele56788"        [1345, 0.003531815221729723, 37]
"gro61133"        [1321, 0.003468793983572464, 37]
"ele74482"        [1316, 0.0034556645589563684, 37]
"dai88807"        [1316, 0.0034556645589563684, 37]
"ele59935"        [1311, 0.003442535134340273, 37]
"sna96271"        [1295, 0.0034005209755687666, 37]
"dai43223"        [1290, 0.003387391550952671, 37]
"ele91337"        [1289, 0.0033847656660294517, 37]
"gro15017"        [1275, 0.003348003277104384, 37]
"dai31081"        [1261, 0.0033112408881793166, 37]
"gro81087"        [1220, 0.0032035796063273323, 37]
"dai22896"        [1219, 0.0032009537214041134, 37]
"gro85051"        [1214, 0.0031878242967880175, 37]

Bottom 10

"dai12448"        [1, 2.6258849232191246e-06, 37]
"dai12152"        [1, 2.6258849232191246e-06, 37]
"dai12139"        [1, 2.6258849232191246e-06, 37]
"dai11995"        [1, 2.6258849232191246e-06, 37]
"dai11946"        [1, 2.6258849232191246e-06, 37]
"dai11707"        [1, 2.6258849232191246e-06, 37]
"dai11582"        [1, 2.6258849232191246e-06, 37]
"dai11375"        [1, 2.6258849232191246e-06, 37]
"dai11273"        [1, 2.6258849232191246e-06, 37]
"dai11257"        [1, 2.6258849232191246e-06, 37]

sort: write failed: standard output: Broken pipe
sort: write error
```

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.

1.4 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 [142]:
```
%%bash

sudo touch shopping_pairs.py
sudo chmod 777 shopping_pairs.py
```

In [157]:
```
%%writefile shopping_pairs.py

#Pair, # occurrences, relative # occurrences

from mrjob.job import MRJob
from mrjob.step import MRStep
from collections import defaultdict
import itertools
import re
WORD_RE = re.compile(r"[\w']+")

class MRShoppingPairs(MRJob):

    def map_basket_pairs(self, _, basket):
        #Get all combinations of pairs, turn to set, then iterate and spit out
        combos = itertools.combinations(sorted(WORD_RE.findall(basket)), 2)
        for combo in combos:
            yield combo, 1
        yield "basket", 1

    def group_basket_pairs(self, key, values):
        total = 0
        for value in values:
            total += value
        if total>=100:
            yield None, (key, total)

    def final_basket_sum(self, _, values):
        values, valuesCp = itertools.tee(values)
        #[total for value in values for key, total in value if key=="basket"]
        for value in valuesCp:
            key, total = value
            if key == "basket":
                basketSize = total

        for value in values:
            key, total = value
            yield key, (total, float(total)/basketSize)

    def steps(self):
        return [MRStep(mapper=self.map_basket_pairs, reducer=self.group_basket_pairs),
                MRStep(reducer=self.final_basket_sum)]
if __name__=='__main__':
    MRShoppingPairs.run()
```

```
Overwriting shopping_pairs.py
```

```
In [158]: %%bash
          sudo chmod -R 777 shopping_pairs/
          python shopping_pairs.py ProductPurchaseData.txt  --jobconf mapred.reduce.tasks=4 --output-di
```

```
["SNA59903", "SNA72163"]        [310, 0.009967525159962702]
["SNA72163", "SNA80324"]        [116, 0.003729783608244108]
["SNA72163", "SNA93860"]        [121, 0.003890550143082216]
["SNA74022", "SNA96271"]        [107, 0.0034404038455355134]
["SNA80324", "SNA90094"]        [154, 0.00495160927301373]
["SNA80324", "SNA93860"]        [150, 0.004822996045143243]
["SNA80324", "SNA96271"]        [219, 0.007041574225909134]
["SNA80324", "SNA99873"]        [163, 0.005240989035722324]
["SNA90094", "SNA96271"]        [104, 0.0033439439246326485]
["SNA93860", "SNA99873"]        [105, 0.00337609723160027]

No configs found; falling back on auto-configuration
Running step 1 of 2...
Creating temp directory /tmp/shopping_pairs.cloudera.20160921.012658.327054
Running step 2 of 2...
Streaming final output from shopping_pairs...
Removing temp directory /tmp/shopping_pairs.cloudera.20160921.012658.327054...
```

```
In [34]: %%bash
         printf "Top 50\n\n"
         cat shopping_pairs/part* | sort -k4,4 -gr | head -50
```

```
Top 50

["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]
["FRO40251", "GRO73461"]        [882, 0.02835921674544227]
["DAI62779", "DAI75645"]        [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]
["DAI85309", "ELE17451"]        [482, 0.01549789395839362]
["DAI62779", "FRO78087"]        [482, 0.01549789395839362]
["DAI62779", "GRO94758"]        [479, 0.015401434037490756]
["GRO85051", "SNA80324"]        [471, 0.015144207581749784]
["DAI62779", "GRO21487"]        [471, 0.015144207581749784]
["ELE17451", "GRO30386"]        [468, 0.015047747660846917]
["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]
```

1.5 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

OPTIONAL: all HW below this are optional

1.6 HW3.6 Computing Relative Frequencies on 100K WikiPedia pages (93Meg)

Dataset description For this assignment you will explore a set of 100,000 Wikipedia documents:

https://www.dropbox.com/s/n5lfbnztclo93ej/wikitext_100k.txt?dl=0 s3://cs9223/wikitext_100k.txt, or https://s3.amazonaws.com/cs9223/wikitext_100k.txt Each line in this file consists of the plain text extracted from a Wikipedia document.

Task Compute the relative frequencies of each word that occurs in the documents in wikitext_100k.txt and output the top 100 word pairs sorted by decreasing order of relative frequency.

Recall that the relative frequency (RF) of word B given word A is defined as follows:

f(B|A) = Count(A, B) / Count (A) = Count(A, B) / sum_B'(Count (A, B')

where count(A,B) is the number of times A and B co-occur within a window of two words (co-occurrence window size of two) in a document and count(A) the number of times A occurs with anything else. Intuitively, given a document collection, the relative frequency captures the proportion of time the word B appears in the same document as A. (See Section 3.3, in Data-Intensive Text Processing with MapReduce).

In the async lecture you learned different approaches to do this, and in this assignment, you will implement them:

a. Write a mapreduce program which uses the Stripes approach and writes its output in a file named rfstripes.txt

b. Write a mapreduce program which uses the Pairs approach and writes its output in a file named rfpairs.txt

c. Compare the performance of the two approaches and output the relative performance to a file named rfcomp.txt. Compute the relative performance as follows: (running time for Pairs/ running time for Stripes). Also include an analysis comparing the communication costs for the two approaches. Instrument your mapper and reduces for counters where necessary to aid with your analysis.

NOTE: please limit your analysis to the top 100 word pairs sorted by decreasing order of relative frequency for each word (tokens with all alphabetical letters).

Please include markdown cell named rf.txt that describes the following:

the input/output format in each Hadoop task, i.e., the keys for the mappers and reducers the Hadoop cluster settings you used, i.e., number of mappers and reducers the running time for each approach: pairs and stripes

You can write your program using Python or MrJob (with Hadoop streaming) and you should run it on AWS. It is a good idea to develop and test your program on a local machine before deploying on AWS. Remember your notebook, needs to have all the commands you used to run each Mapreduce job (i.e., pairs and stripes) – include the Hadoop streaming commands you used to run your jobs.

In addition the All the following files should be compressed in one ZIP file and submitted. The ZIP file should contain:

A. The result files: rfstripes.txt, rfpairs.txt, rfcomp.txt

Prior to working with Hadoop, the corpus should first be preprocessed as follows: perform tokenization (whitespace and all non-alphabetic characters) and stopword removal using standard tools from the Lucene search engine. All tokens should then be replaced with unique integers for a more efficient encoding.

== Preliminary information for the remaing HW problems===

Much of this homework beyond this point will focus on the Apriori algorithm for frequent itemset mining and the additional step for extracting association rules from these frequent itemsets. Please acquaint yourself with the background information (below) before approaching the remaining assignments.

=== Apriori background information ===

Some background material for the Apriori algorithm is located at:

- Slides in Live Session #3
- https://en.wikipedia.org/wiki/Apriori_algorithm
- https://www.dropbox.com/s/k2zm4otych279z2/Apriori-good-slides.pdf?dl=0
- http://snap.stanford.edu/class/cs246-2014/slides/02-assocrules.pdf

Association Rules are frequently used for Market Basket Analysis (MBA) by retailers to understand the purchase behavior of their customers. This information can be then used for many different purposes such as cross-selling and up-selling of products, sales promotions, loyalty programs, store design, discount plans and many others. Evaluation of item sets: Once you have found the frequent itemsets of a dataset, you need to choose a subset of them as your recommendations. Commonly used metrics for measuring significance and interest for selecting rules for recommendations are: confidence; lift; and conviction.

1.7 HW3.7 Apriori Algorithm

What is the Apriori algorithm? Describe an example use in your domain of expertise and what kind of . Define confidence and lift.

NOTE: For the remaining homework use the online browsing behavior dataset located at (same dataset as used above):

`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

1.8 HW3.8. 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.

Suppose we want to recommend new products to the customer based on the products they have already browsed on the online website. Write a program using the A-priori algorithm to find products which are frequently browsed together. Fix the support to s = 100 (i.e. product sets need to occur together at least 100 times to be considered frequent) and find itemsets of size 2 and 3.

Then extract association rules from these frequent items.

A rule is of the form:

(item1, item5) $\Rightarrow$ item2.

List the top 10 discovered rules in descreasing order of confidence in the following format

(item1, item5) $\Rightarrow$ item2, supportCount ,support, confidence

1.9 HW3.8

Benchmark your results using the pyFIM implementation of the Apriori algorithm (Apriori - Association Rule Induction / Frequent Item Set Mining implemented by Christian Borgelt). You can download pyFIM from here:

http://www.borgelt.net/pyfim.html

Comment on the results from both implementations (your Hadoop MapReduce of apriori versus pyFIM) in terms of results and execution times.

2 END OF HOMEWORK