HW 5.0

In an enterprise setting, a data warehouse serves as a data repository. It can store not only relational data, but also semi-structured and unstructured data. It is a system for reporting and data analysis.

A Star Schema is a combination of facts and dimensions for storing data. It consists of one or more fact tables referencing any number of dimension tables.

For example, one database of sales can be keeping the main data about each sales, and then, it can be referencing to store, product and date databases.

HW 5.1

In the database world, 3NF (Third Normal Form) is a normal form used in normalizing a database design to reduce the duplication of data and ensure referential integrity by ensuring that the entity is in second normal form and that all the attributes in a table are determined only by the candidate keys of that table and not by any non-prime attributes. It was designed to improve database processing while minimizing storage costs.

Machine Learning also consumes data with joining. One would use denormalized log files to extract features or variables from these logs.

HW 5.2

Using MRJob, implement a hashside join (memory-backed map-side) for left, right and inner joins. Run your code on the data used in HW 4.4: (Recall HW 4.4: Find the most frequent visitor of each page using mrjob and the output of 4.2 (i.e., transfromed log file). In this output please include the webpage URL, webpageID and Visitor ID.) :

Justify which table you chose as the Left table in this hashside join.

Please report the number of rows resulting from:

(1) Left joining Table Left with Table Right

(2) Right joining Table Left with Table Right

(3) Inner joining Table Left with Table Right

In [ ]:
```
## first, we get the data from our database
```

In [20]:
```
!wget https://www.dropbox.com/sh/m0nxsf4vs5cyrp2/AADCHtrJ4CBCDO1po_OAWg0ia/a
```

```
--2016-07-16 13:45:24--  https://www.dropbox.com/sh/m0nxsf4vs5cyrp2/AADCH
trJ4CBCDO1po_OAWg0ia/anonymous-msweb.data?dl=0 (https://www.dropbox.com/s
h/m0nxsf4vs5cyrp2/AADCHtrJ4CBCDO1po_OAWg0ia/anonymous-msweb.data?dl=0)
Resolving www.dropbox.com (www.dropbox.com)... 162.125.65.1
Connecting to www.dropbox.com (www.dropbox.com)|162.125.65.1|:443... conn
ected.
HTTP request sent, awaiting response... 302 Found
Location: https://dl.dropboxusercontent.com/content_link/4fs1tHJE0tBZeTzt
iQLvEGJZkUlrXMn647C6Fm0pwKyleryHrXxVRpJ3IUXZuuHM/file (https://dl.dropbox
usercontent.com/content_link/4fs1tHJE0tBZeTztiQLvEGJZkUlrXMn647C6Fm0pwKyl
eryHrXxVRpJ3IUXZuuHM/file) [following]
--2016-07-16 13:45:25--  https://dl.dropboxusercontent.com/content_link/4
fs1tHJE0tBZeTztiQLvEGJZkUlrXMn647C6Fm0pwKyleryHrXxVRpJ3IUXZuuHM/file (htt
ps://dl.dropboxusercontent.com/content_link/4fs1tHJE0tBZeTztiQLvEGJZkUlrX
Mn647C6Fm0pwKyleryHrXxVRpJ3IUXZuuHM/file)
Resolving dl.dropboxusercontent.com (dl.dropboxusercontent.com)... 108.16
0.173.165
Connecting to dl.dropboxusercontent.com (dl.dropboxusercontent.com)|108.1
60.173.165|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1423098 (1.4M) [text/plain]
Saving to: ‘anonymous-msweb.data?dl=0.1’

100%[======================================>] 1,423,098   3.22MB/s   in
 0.4s

2016-07-16 13:45:27 (3.22 MB/s) - ‘anonymous-msweb.data?dl=0.1’ saved [14
23098/1423098]
```

In [21]:
```
!mv anonymous-msweb.data?dl=0 anonymous-msweb.data
```

In [22]:
```
#first check how does the file look like
! head -55 anonymous-msweb.data
```

```
I,4,"www.microsoft.com","created by getlog.pl"
T,1,"VRoot",0,0,"VRoot"
N,0,"0"
N,1,"1"
T,2,"Hide1",0,0,"Hide"
N,0,"0"
N,1,"1"
A,1287,1,"International AutoRoute","/autoroute"
A,1288,1,"library","/library"
A,1289,1,"Master Chef Product Information","/masterchef"
A,1297,1,"Central America","/centroam"
A,1215,1,"For Developers Only Info","/developer"
A,1279,1,"Multimedia Golf","/msgolf"
A,1239,1,"Microsoft Consulting","/msconsult"
A,1282,1,"home","/home"
A,1251,1,"Reference Support","/referencesupport"
A,1121,1,"Microsoft Magazine","/magazine"
A,1083,1,"MS Access Support","/msaccesssupport"
A,1145,1,"Visual Fox Pro Support","/vfoxprosupport"
A,1276,1,"Visual Test Support","/vtestsupport"
A,1200,1,"Benelux Region","/benelux"
A,1259,1,"controls","/controls"
A,1155,1,"Sidewalk","/sidewalk"
A,1092,1,"Visual FoxPro","/vfoxpro"
A,1004,1,"Microsoft.com Search","/search"
A,1057,1,"MS PowerPoint News","/powerpoint"
A,1140,1,"Netherlands (Holland)","/netherlands"
A,1198,1,"Picture It","/pictureit"
A,1147,1,"Microsoft Financial Forum","/msft"
A,1005,1,"Norway","/norge"
A,1026,1,"Internet Site Construction for Developers","/sitebuilder"
A,1119,1,"Corporation Information","/corpinfo"
A,1216,1,"Virtual Reality Markup Language","/vrml"
A,1218,1,"MS Publisher Support","/publishersupport"
A,1205,1,"Hardware Supprt","/hardwaresupport"
A,1269,1,"Customer Guides","/business"
A,1031,1,"MS Office","/msoffice"
A,1003,1,"Knowledge Base","/kb"
A,1238,1,"Excel Development","/exceldev"
A,1118,1,"SQL Server","/sql"
A,1242,1,"MS Garden","/msgarden"
A,1171,1,"MS Merchant","/merchant"
A,1175,1,"MS Project Support","/msprojectsupport"
A,1021,1,"Visual C","/visualc"
A,1222,1,"MS Office News","/msofc"
A,1284,1,"partner","/partner"
A,1294,1,"Bookshelf","/bookshelf"
A,1053,1,"Jakarta","/visualj"
A,1293,1,"Encarta","/encarta"
A,1167,1,"Windows Hardware Testing","/hwtest"
A,1202,1,"Advanced Technology","/advtech"
A,1234,1,"Office Free Stuff News","/off97cat"
A,1054,1,"Exchange","/exchange"
A,1262,1,"Chile","/chile"
A,1074,1,"Windows NT Workstation","/ntworkstation"
```

In [23]:
```
import re
open("anonymous-msweb-preprocessed.data", "w").close
custID = "NA"
with open("anonymous-msweb.data", "r") as IF:
    for line in IF:
        line = line.strip()
        data = re.split(",",line)
        if data[0] == "C":
            custID = data[1]
            custID = re.sub("\"","",custID)
        if data[0] == "V" and not custID == "NA":
            with open("anonymous-msweb-preprocessed.data", "a") as OF:
                OF.writelines(line+","+"C"+","+custID+"\n")
```

In [24]:
```
! head anonymous-msweb-preprocessed.data
```

```
V,1000,1,C,10001
V,1001,1,C,10001
V,1002,1,C,10001
V,1001,1,C,10002
V,1003,1,C,10002
V,1001,1,C,10003
V,1003,1,C,10003
V,1004,1,C,10003
V,1005,1,C,10004
V,1006,1,C,10005
```

We can work with this data in the same way as in the class example. The difference is that we are dealing with a dataset of visitors and visits.

(1) Left join:

We want to print out what visitor visited what page. This is the same as our class example.

In [31]:
```
%%writefile reducersideleftjoin.py
from mrjob.job import MRJob
from mrjob.step import MRStep
from mrjob.compat import jobconf_from_env
import re

class leftjoin(MRJob):
    def mapper(self, _, line):
        x = line.split(",")
        if len(x) == 5:
            ## if we're dealing with the page visiting history:
            if x[0]=="V":
                yield x[0], ("lefttable", x[1], x[2], x[3], x[4])
            ## if we're dealing with the page info:
            if x[0] == "A":
                yield x[0], ("righttable", x[1], x[2], x[3], x[4])

    def reducer(self, key, values):
        pagename = list()
        pagevisits = list()
        for val in values:
            if val[0]== u'lefttable':
                pagevisits.append(val)
            else:
                pagename.append(val)
        # this is what makes the result of the query different
        # based on the table on which we are joining:
        for c in pagename:
            if len(pagevisits)==0:
                yield None, [key] + c[1:] + [None]
            for o in pagevisits:
                yield None, [key] + c[1:] + o[1:]

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

```
Overwriting reducersideleftjoin.py
```

In [32]:
```
from reducersideleftjoin import leftjoin
mr_job = leftjoin(args=['anonymous-msweb-preprocessed.data','anonymous-msweb
with mr_job.make_runner() as runner:
    runner.run()
    count = 0
    # stream_output: get access of the output
    for line in runner.stream_output():
        key,value =  mr_job.parse_output_line(line)
        print value
        count = count + 1
print "\n"
print "There are %s records" %count
```

```
[u'A', u'1000', u'1', u'"regwiz"', u'"/regwiz"', None]
[u'A', u'1001', u'1', u'"Support Desktop"', u'"/support"', None]
[u'A', u'1002', u'1', u'"End User Produced View"', u'"/athome"', None]
[u'A', u'1003', u'1', u'"Knowledge Base"', u'"/kb"', None]
[u'A', u'1004', u'1', u'"Microsoft.com Search"', u'"/search"', None]
[u'A', u'1005', u'1', u'"Norway"', u'"/norge"', None]
[u'A', u'1006', u'1', u'"misc"', u'"/misc"', None]
[u'A', u'1007', u'1', u'"International IE content"', u'"/ie_intl"', None]
[u'A', u'1008', u'1', u'"Free Downloads"', u'"/msdownload"', None]
[u'A', u'1009', u'1', u'"Windows Family of OSs"', u'"/windows"', None]
[u'A', u'1010', u'1', u'"Visual Basic"', u'"/vbasic"', None]
[u'A', u'1011', u'1', u'"MS Office Development"', u'"/officedev"', None]
[u'A', u'1012', u'1', u'"Outlook Development"', u'"/outlookdev"', None]
[u'A', u'1013', u'1', u'"Visual Basic Support"', u'"/vbasicsupport"', Non
e]
[u'A', u'1014', u'1', u'"Office Free Stuff"', u'"/officefreestuff"', Non
e]
[u'A', u'1015', u'1', u'"Excel"', u'"/msexcel"', None]
[u'A', u'1016', u'1', u'"MS Excel"', u'"/excel"', None]
```

(2) Right join:

In [44]:
```
%%writefile reducersiderightjoin.py
from mrjob.job import MRJob
from mrjob.step import MRStep
from mrjob.compat import jobconf_from_env

class rightjoin(MRJob):
    def mapper(self, _, line):
        #x = line.strip()
        #data = re.split(",",line)
        x = line.split(",")
        #x = re.split(",",line)
        if len(x) >1:
            ## if we're dealing with the page visiting history:
            if x[0]=="V":
                yield x[0], ("lefttable", x[1:])
            ## if we're dealing with the page info:
            if x[0] == "A":
                yield x[0], ("righttable", x[1:])

    def reducer(self, key, values):
        pagename = list()
        pagevisits = list()
        for val in values:
            if val[0]== u'righttable':
                pagevisits.append(val)
            else:
                pagename.append(val)
        # this is what makes the result of the query different
        # based on the table on which we are joining:
        for o in pagevisits:
            if len(pagevisits)==0:
                yield None, [key] + [None, None, None] + o[1:]
            for c in pagename:
                yield None, [key] + c[1:] + o[1:]

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

```
Overwriting reducersiderightjoin.py
```

In [45]:
```
from reducersiderightjoin import rightjoin
mr_job = rightjoin(args=['anonymous-msweb.data','anonymous-msweb-preprocesse
with mr_job.make_runner() as runner:
    runner.run()
    count = 0
    # stream_output: get access of the output
    for line in runner.stream_output():
        key,value =  mr_job.parse_output_line(line)
        print value
        count = count + 1
print "\n"
print "There are %s records" %count
```

```
There are 0 records
```

(3) Inner join:

In [29]:
```
%%writefile reducersideinnerjoin.py
from mrjob.job import MRJob
from mrjob.step import MRStep
from mrjob.compat import jobconf_from_env

class innerjoin(MRJob):
    def mapper(self, _, line):
        #x = line.strip()
        #data = re.split(",",line)
        x = line.split(",")
        #x = re.split(",",line)
        if len(x) == 5:
            ## if we're dealing with the page visiting history:
            if x[0]=="V":
                yield x[0], ("lefttable", x[1], x[2], x[3], x[4])
            ## if we're dealing with the page info:
            if x[0] == "A":
                yield x[0], ("righttable", x[1], x[2], x[3], x[4])

    def reducer(self, key, values):
        pagename = list()
        pagevisits = list()
        for val in values:
            if val[0]== u'lefttable':
                pagevisits.append(val)
            else:
                pagename.append(val)
        # inner join:

        for o in pagevisits:
            for c in pagename:
                yield None, [key] + c[1:] + o[1:]

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

```
Overwriting reducersideinnerjoin.py
```

Now, we run the code through python driver.

In [30]:
```
from reducersideinnerjoin import innerjoin
mr_job = innerjoin(args=['anonymous-msweb-preprocessed.data','anonymous-mswe
with mr_job.make_runner() as runner:
    runner.run()
    count = 0
    # stream_output: get access of the output
    for line in runner.stream_output():
        key,value =  mr_job.parse_output_line(line)
        print value
        count = count + 1
print "\n"
print "There are %s records" %count
```

```
There are 0 records
```

HW 5.3 EDA of Google n-grams dataset

Do some EDA on this dataset using mrjob, e.g.,

- Longest 5-gram (number of characters). Note if there are ties pick the ngram sort alphabetical order [HINT: think secondary sort where primary sort key is length of ngram in characters, and the secondary sort is ngram string itself]

In [35]:
```
!which bash
```

```
/usr/bin/bash
```

In [36]:
```
!aws s3 sync s3://filtered-5grams/
```

```
/usr/bin/sh: aws: command not found
```

First, we download all the ngram data files, and save them where they can be processed by the mapper and reducer.

In [37]:
```
%%writefile download.sh
#!/usr/bin/bash
url=$(awk -F = '{print $2}' url.txt)
for i in $(cat file.txt);
do
wget "${url}${i}"
done
```

```
Writing download.sh
```

In [38]:
```
!wget https://www.dropbox.com/sh/0cv65h44zylqwe3/AADM7tG85Qvup0Ok6wp0WJlua/f
```

```
--2016-07-16 14:22:06--  https://www.dropbox.com/sh/0cv65h44zylqwe3/AADM7
tG85Qvup0Ok6wp0WJlua/filtered-5Grams/googlebooks-eng-all-5gram-20090715-5
-filtered.txt (https://www.dropbox.com/sh/0cv65h44zylqwe3/AADM7tG85Qvup0O
k6wp0WJlua/filtered-5Grams/googlebooks-eng-all-5gram-20090715-5-filtered.
txt)
Resolving www.dropbox.com (www.dropbox.com)... 162.125.65.1
Connecting to www.dropbox.com (www.dropbox.com)|162.125.65.1|:443... conn
ected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: ‘googlebooks-eng-all-5gram-20090715-5-filtered.txt.1’

    [    <=>                                 ] 166,941      240KB/s   in
 0.7s

2016-07-16 14:22:08 (240 KB/s) - ‘googlebooks-eng-all-5gram-20090715-5-fi
ltered.txt.1’ saved [166941]
```

In [39]:
```
! head googlebooks-eng-all-5gram-20090715-0-filtered.txt
```

```
GBMModel=gbm(DaysInHospital~ ., distribution  ="gaussian",
                        trainData[,-c(memberIndex,droppedFeatures)], n.tr
ees = numOfTrees,
                        shrinkage  =  GBM_SHRINKAGE,
                        interaction.depth=GBM_DEPTH,n.minobsinnode  =  GB
M_MINOBS,verbose  =  TRUE, keep.data=FALSE)A BILL FOR ESTABLISHING RELIGI
OUS     59      59      54
A Biography of General George   92      90      74
A Case Study in Government      102     102     78
A Case Study of Female  447     447     327
A Case Study of Limited 55      55      43
A Child's Christmas in Wales    1099    1061    866
A Circumstantial Narrative of the       62      62      50
```

In [58]:
```
!wget https://www.dropbox.com/sh/0cv65h44zylqwe3/AABkuM-jAXNVfTaXw6Prj1P1a/f
```

```
--2016-06-25 19:09:28--  https://www.dropbox.com/sh/0cv65h44zylqwe3/AABku
M-jAXNVfTaXw6Prj1P1a/filtered-5Grams?dl=0 (https://www.dropbox.com/sh/0cv
65h44zylqwe3/AABkuM-jAXNVfTaXw6Prj1P1a/filtered-5Grams?dl=0)
Resolving www.dropbox.com (www.dropbox.com)... 162.125.65.1
Connecting to www.dropbox.com (www.dropbox.com)|162.125.65.1|:443... conn
ected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: ‘filtered-5Grams?dl=0’

    [     <=>                                 ] 522,354      373KB/s   in
 1.4s

2016-06-25 19:09:31 (373 KB/s) - ‘filtered-5Grams?dl=0’ saved [522354]
```

In [40]:
```
%%writefile mapper.py
#!/w261/venv/bin/python

import sys
for line in sys.stdin:
    line=line.strip()
    ngram=line.split('\t')[0] #extract product field from second field
    length = len(ngram)

    print ngram, length
```

```
Overwriting mapper.py
```

In [47]:

```
%%writefile long5gram.py
#!/w261/venv/bin/python

import re

import mrjob
from mrjob.protocol import RawProtocol
from mrjob.job import MRJob
from mrjob.step import MRStep

class long5gram(MRJob):

    def steps(self):
        JOBCONF_STEP1 = {
            'mapreduce.job.output.key.comparator.class': 'org.apache.hadoop.
            'stream.map.output.field.separator':'\t',
            'mapreduce.partition.keycomparator.options': '-k1,1nr -k2',
            'mapreduce.job.reduces': '16'
        }
        JOBCONF_STEP2 = {
            'mapreduce.job.output.key.comparator.class': 'org.apache.hadoop.
            'stream.map.output.field.separator':'\t',
            'mapreduce.partition.keycomparator.options': '-k1,1nr -k2',
            'mapreduce.job.reduces': '1'
        }
        return [MRStep(
            jobconf = JOBCONF_STEP1,
            mapper_init = self.mapper_init,
            mapper = self.mapper,
            mapper_final = self.mapper_final,
            reducer_init = self.reducer_init,
            reducer = self.reducer,
            reducer_final = self.reducer_final
        ),MRStep(
            jobconf = JOBCONF_STEP2,
            mapper = self.mapper_sort,
            reducer = self.reducer_sort
        )]

    def mapper_init(self):
        self.longest = 0
        self.longngram = ""

    def mapper(self, _, line):
        line = line.strip()
        ngram=line.split('\t')[0] #extract product field from second field
        length = len(ngram)

    def mapper_final(self):
        yield self.longest, self.longngram

    def reducer_init(self):
        self.longest = 0
        self.longngram = None
        self.longngrams = {}

    def reducer(self, key, value):
```

```
        for v in value:
            if key >= self.longest:
                self.longest = key
                self.longngrams[v] = key

    def reducer_final(self):
        for key in self.longngrams.keys():
            yield self.longngrams[key], key

    def mapper_sort(self, key, value):
        yield key, value

    def reducer_sort(self, key, value):
        for v in value:
            yield key, v

if __name__ == '__main__':
    long5gram.run()
```

```
Writing long5gram.py
```

In [53]:
```
!hdfs dfs -rm -r HW5_3/long5gram
!python long5gram.py -r hadoop 'googlebooks-eng-all-5gram-20090715-0-filtere
```

```
rm: `HW5_3/long5gram': No such file or directory
No configs found; falling back on auto-configuration
Creating temp directory /tmp/long5gram.root.20160716.215005.058897
Looking for hadoop binary in $PATH...
Found hadoop binary: /usr/bin/hadoop
Using Hadoop version 2.6.0
Copying local files to hdfs:///user/root/tmp/mrjob/long5gram.root.2016071
6.215005.058897/files/...
STDERR: put: unexpected URISyntaxException
Traceback (most recent call last):
  File "long5gram.py", line 74, in <module>
    long5gram.run()
  File "/w261/venv/lib/python2.7/site-packages/mrjob/job.py", line 430, i
n run
    mr_job.execute()
  File "/w261/venv/lib/python2.7/site-packages/mrjob/job.py", line 448, i
n execute
    super(MRJob, self).execute()
  File "/w261/venv/lib/python2.7/site-packages/mrjob/launch.py", line 16
0, in execute
    self.run_job()
  File "/w261/venv/lib/python2.7/site-packages/mrjob/launch.py", line 23
0, in run_job
    runner.run()
  File "/w261/venv/lib/python2.7/site-packages/mrjob/runner.py", line 47
3, in run
    self._run()
  File "/w261/venv/lib/python2.7/site-packages/mrjob/hadoop.py", line 32
3, in _run
    self._upload_local_files_to_hdfs()
  File "/w261/venv/lib/python2.7/site-packages/mrjob/hadoop.py", line 35
4, in _upload_local_files_to_hdfs
    self._upload_to_hdfs(path, uri)
  File "/w261/venv/lib/python2.7/site-packages/mrjob/hadoop.py", line 35
8, in _upload_to_hdfs
    self.fs._put(path, target)
  File "/w261/venv/lib/python2.7/site-packages/mrjob/fs/hadoop.py", line
 317, in _put
    self.invoke_hadoop(['fs', '-put', local_path, target])
  File "/w261/venv/lib/python2.7/site-packages/mrjob/fs/hadoop.py", line
 179, in invoke_hadoop
    raise CalledProcessError(proc.returncode, args)
subprocess.CalledProcessError: Command '['/usr/bin/hadoop', 'fs', '-put',
 '/w261/coursework/Untitled Folder/long5gram.py', 'hdfs:///user/root/tmp/
mrjob/long5gram.root.20160716.215005.058897/files/long5gram.py']' returne
d non-zero exit status 1
```

In [41]:
```
# test of the first mapper

!chmod +x mapper.py
```

In [42]:
```
!python mapper.py googlebooks-eng-all-5gram-20090715-0-filtered.txt
```

```
^C
```

In [43]:
```
!cat googlebooks-eng-all-5gram-20090715-0-filtered.txt | ./mapper.py |sort -
```

```
diabetic glomerulopathy by pharmacological amelioration 55
differential reinforcement of successive approximations 55
oligonucleotide arrays using semiconductor photoresists 55
Prevention of experimental autoimmune encephalomyelitis 55
der Verfassungsgebenden Deutschen Nationalversammlung und 57
Guidelines for clinical intracardiac electrophysiological 57
Hydroxytryptamine stimulates inositol phosphate production 58
Interpersonal Communication Interpersonal communication is 58
E e E e E 9
interaction.depth=GBM_DEPTH,n.minobsinnode  =  GBM_MINOBS,verbose  =  TRU
E, keep.data=FALSE)A BILL FOR ESTABLISHING RELIGIOUS 125
```

In [ ]:
```
## now, we just need to find what caused the map-reduce job above to crash
## maybe it couldn't find the jar file, which is in
## HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-2*.jar
```

- Top 10 most frequent words (please use the count information), i.e., unigrams

In [52]:
```
%%writefile mostfrequentwords.py
#!/w261/venv/bin/python

import re

import mrjob
from mrjob.protocol import RawProtocol
from mrjob.job import MRJob
from mrjob.step import MRStep

class mostfrequentwords(MRJob):

    def steps(self):

        JOBCONF_STEP1 = {
            'stream.num.map.output.key.field':2,
            'mapreduce.job.output.key.comparator.class': 'org.apache.hadoop.
            'stream.map.output.field.separator':',',
            'mapreduce.partition.keycomparator.options': '-k2,2nr',
            'mapreduce.job.reduces': '1'
        }
        return [self.mr(mapper=self.mapper,
                        reducer=self.reducer),
                 self.mr(jobconf=JOBCONF_STEP1,
                        mapper=None,
                        reducer=self.reducer_sort)
                ]

    def mapper(self, _, line):
        line = line.strip()
        ngram=line.split('\t')
        words = ngram[0].split()
        for word in words:
            yield word.lower(), int(ngram[1])

    def reducer(self, key, count):
        yield key, sum(count)

    def reducer_sort(self, key, count):
        for c in count:
            yield key, c

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

```
Writing mostfrequentwords.py
```

In [54]:
```
!hdfs dfs -rm -r HW5_3/mostfrequentwords
!python mostfrequentwords.py -r hadoop googlebooks-eng-all-5gram-20090715-0-
```

```
rm: `HW5_3/mostfrequentwords': No such file or directory
No configs found; falling back on auto-configuration
Creating temp directory /tmp/mostfrequentwords.root.20160716.215742.37394
8
Looking for hadoop binary in $PATH...
Found hadoop binary: /usr/bin/hadoop
Using Hadoop version 2.6.0
Copying local files to hdfs:///user/root/tmp/mrjob/mostfrequentwords.roo
t.20160716.215742.373948/files/...
STDERR: put: unexpected URISyntaxException
Traceback (most recent call last):
  File "mostfrequentwords.py", line 43, in <module>
    mostfrequentwords.run()
  File "/w261/venv/lib/python2.7/site-packages/mrjob/job.py", line 430, i
n run
    mr_job.execute()
  File "/w261/venv/lib/python2.7/site-packages/mrjob/job.py", line 448, i
n execute
    super(MRJob, self).execute()
  File "/w261/venv/lib/python2.7/site-packages/mrjob/launch.py", line 16
0, in execute
    self.run_job()
  File "/w261/venv/lib/python2.7/site-packages/mrjob/launch.py", line 23
0, in run_job
    runner.run()
  File "/w261/venv/lib/python2.7/site-packages/mrjob/runner.py", line 47
3, in run
    self._run()
  File "/w261/venv/lib/python2.7/site-packages/mrjob/hadoop.py", line 32
3, in _run
    self._upload_local_files_to_hdfs()
  File "/w261/venv/lib/python2.7/site-packages/mrjob/hadoop.py", line 35
4, in _upload_local_files_to_hdfs
    self._upload_to_hdfs(path, uri)
  File "/w261/venv/lib/python2.7/site-packages/mrjob/hadoop.py", line 35
8, in _upload_to_hdfs
    self.fs._put(path, target)
  File "/w261/venv/lib/python2.7/site-packages/mrjob/fs/hadoop.py", line
 317, in _put
    self.invoke_hadoop(['fs', '-put', local_path, target])
  File "/w261/venv/lib/python2.7/site-packages/mrjob/fs/hadoop.py", line
 179, in invoke_hadoop
    raise CalledProcessError(proc.returncode, args)
subprocess.CalledProcessError: Command '['/usr/bin/hadoop', 'fs', '-put',
 '/w261/coursework/Untitled Folder/mostfrequentwords.py', 'hdfs:///user/r
oot/tmp/mrjob/mostfrequentwords.root.20160716.215742.373948/files/mostfre
quentwords.py']' returned non-zero exit status 1
```

In [ ]:
```
```

```
# it seems that the problem is the same as before,
# so it is something in the settings that is wrong.
```

- 20 Most/Least densely appearing words (count/pages_count) sorted in decreasing order of relative frequency

In [ ]:
```
## given that we must always have pages_count =< count:
## take the output of the reducer and count the number of words
```

In [59]:
```
%%writefile wordsdensity.py
#!/w261/venv/bin/python

import re
import numpy as np

import mrjob
from mrjob.protocol import RawProtocol
from mrjob.job import MRJob
from mrjob.step import MRStep

class wordsdensity(MRJob):

    def steps(self):

        JOBCONF_STEP1 = {
            'stream.num.map.output.key.field':2,
            'mapreduce.job.output.key.comparator.class': 'org.apache.hadoop.
            'stream.map.output.field.separator':',',
            'mapreduce.partition.keycomparator.options': '-k2,2nr',
            'mapreduce.job.reduces': '1'
        }
        return [self.mr(mapper=self.mapper,
                        reducer=self.reducer),
                self.mr(jobconf=JOBCONF_STEP2,
                        mapper=None,
                        reducer=self.reducer_sort)
               ]

    def mapper(self, _, line):
        line = line.strip()
        ngram=line.split('\t')
        words = ngram[0].split()
        for word in words:
            yield word.lower(), (int(ngram[1]), int(ngram[2]))

    def reducer(self, key, counts):
        ct = 0
        pages_ct = 0
        for c, pct in counts:
            ct += c
            pages_ct += pct
        yield key, (ct/float(pct))

    def reducer_sort(self, key, count):
        for c in count:
            yield key, c
```

```
Overwriting wordsdensity.py
```

In [60]:
```
!hdfs dfs -rm -r HW5_3/wordsdensity
!python wordsdensity.py -r hadoop googlebooks-eng-all-5gram-20090715-0-filte
```

```
rm: `HW5_3/wordsdensity': No such file or directory
```

- Distribution of 5-gram sizes (character length). E.g., count (using the count field) up how

many times a 5-gram of 50 characters shows up. Plot the data graphically using a histogram.

In [61]:
```
%%writefile histogram.py
#!/w261/venv/bin/python

import mrjob
from mrjob.protocol import RawProtocol
from mrjob.job import MRJob
from mrjob.step import MRStep

class histogram(MRJob):

    def steps(self):
        JOBCONF_STEP1 = {
            'mapred.reduce.tasks': '16'
        }

        JOBCONF_STEP2 = {
            'stream.num.map.output.key.field':2,
            'mapreduce.job.output.key.comparator.class': 'org.apache.hadoop.
            'stream.map.output.field.separator':',',
            'mapreduce.partition.keycomparator.options': '-k1,1nr',
            'mapreduce.job.reduces': '1'
        }
        return [self.mr(jobconf=JOBCONF_STEP1,
                        mapper=self.mapper,
                        combiner = self.reducer,
                        reducer=self.reducer),
                 self.mr(jobconf=JOBCONF_STEP2,
                        mapper=None,
                        reducer=self.reducer_sort)
                ]

    def mapper(self, _, line):
        line = line.strip()
        ngram=line.split('\t')
        yield len(ngram[0]), int(ngram[1])

    def reducer(self, key, count):
        yield key, sum(count)

    def reducer_sort(self, key, count):
        for c in count:
            yield key, c

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

```
Writing histogram.py
```

In [62]:
```
!hdfs dfs -rm -r HW5_3/histogram
!python histogram.py -r hadoop googlebooks-eng-all-5gram-20090715-0-filtered
```

```
rm: `HW5_3/histogram': No such file or directory
No configs found; falling back on auto-configuration
Creating temp directory /tmp/histogram.root.20160716.222949.179715
Looking for hadoop binary in $PATH...
Found hadoop binary: /usr/bin/hadoop
Using Hadoop version 2.6.0
Copying local files to hdfs:///user/root/tmp/mrjob/histogram.root.2016071
6.222949.179715/files/...
STDERR: put: unexpected URISyntaxException
Traceback (most recent call last):
  File "histogram.py", line 44, in <module>
    histogram.run()
  File "/w261/venv/lib/python2.7/site-packages/mrjob/job.py", line 430, i
n run
    mr_job.execute()
  File "/w261/venv/lib/python2.7/site-packages/mrjob/job.py", line 448, i
n execute
    super(MRJob, self).execute()
  File "/w261/venv/lib/python2.7/site-packages/mrjob/launch.py", line 16
0, in execute
    self.run_job()
  File "/w261/venv/lib/python2.7/site-packages/mrjob/launch.py", line 23
0, in run_job
    runner.run()
  File "/w261/venv/lib/python2.7/site-packages/mrjob/runner.py", line 47
3, in run
    self._run()
  File "/w261/venv/lib/python2.7/site-packages/mrjob/hadoop.py", line 32
3, in _run
    self._upload_local_files_to_hdfs()
  File "/w261/venv/lib/python2.7/site-packages/mrjob/hadoop.py", line 35
4, in _upload_local_files_to_hdfs
    self._upload_to_hdfs(path, uri)
  File "/w261/venv/lib/python2.7/site-packages/mrjob/hadoop.py", line 35
8, in _upload_to_hdfs
    self.fs._put(path, target)
  File "/w261/venv/lib/python2.7/site-packages/mrjob/fs/hadoop.py", line
 317, in _put
    self.invoke_hadoop(['fs', '-put', local_path, target])
  File "/w261/venv/lib/python2.7/site-packages/mrjob/fs/hadoop.py", line
 179, in invoke_hadoop
    raise CalledProcessError(proc.returncode, args)
subprocess.CalledProcessError: Command '['/usr/bin/hadoop', 'fs', '-put',
 '/w261/coursework/Untitled Folder/histogram.py', 'hdfs:///user/root/tmp/
mrjob/histogram.root.20160716.222949.179715/files/histogram.py']' returne
d non-zero exit status 1
```

In [ ]:
```
## it seems that this really is a configuration problem.
```

In [1]:
```
from IPython.display import display, Math, Latex
```

HW 5.4 Synonym detection over 2Gig of Data

Please unit test and system test your code with respect to SYSTEMS TEST DATASET and show the results. Please compute the expected answer by hand and show your hand calculations for the SYSTEMS TEST DATASET. Then show the results you get with you system.

Let's first start with calculation for this systems test dataset.

For the items in the example "DocA, DocB, DocC":

the initial mini-documents are:

DocA {X:20, Y:30, Z:5} DocB {X:100, Y:20} DocC {M:5, N:20, Z:5}

So then, their first stripes for inverted indices are

X: {(DocA, 3), (DocB, 2)} Y: {(DocA, 3), (DocB, 2)} Z: {(DocA, 3), (DocC, 3)} M: {(DocC, 3)} N: {(DocC, 3)}

Then, their Jaccard similarity is

$$(DocA, DocB) = \frac{2}{3+2-2} = \frac{2}{3} = 0.66

(DocB, DocC) = \frac{0}{2+3-0} = 0

(DocA, DocC) = \frac{1}{2+3-1} = \frac{1}{4} $$

and their Cosine similarity is

$$(DocA, DocB) = \frac{2}{\sqrt{3} \sqrt{2}} = \frac{2}{ \sqrt{6}

(DocB, DocC) = \frac{0}{\sqrt{2} \sqrt{3}} = 0

(DocA, DocC) = \frac{1}{ \sqrt{3} \sqrt{3}} = \frac{1}{3} $$

As the example code shows, this can also be used for detection of synonyms.

(1) Build stripes for the most frequent 10,000 words using cooccurence information based on the words ranked from 9001,-10,000 as a basis/vocabulary (drop stopword-like terms), and output to a file in your bucket on s3 (bigram analysis, though the words are non-contiguous).

==Design notes for (1)== For this task you will be able to modify the pattern we used in HW 3.2 (feel free to use the solution as reference). To total the word counts across the 5-grams, output the support from the mappers using the total order inversion pattern:

<*word,count>

to ensure that the support arrives before the cooccurrences.

In addition to ensuring the determination of the total word counts, the mapper must also output co-occurrence counts for the pairs of words inside of each 5-gram. Treat these words as a basket, as we have in HW 3, but count all stripes or pairs in both orders, i.e., count both orderings:

(word1,word2), and (word2,word1), to preserve symmetry in our output for (2).

In [ ]:
```
## Step 1: find the 10,000 most frequent words

## 1.1: create bigrams from all n-grams
## 1.2: drop all stop-like words
## 1.3: find 10,000 most frequent words
## 1.4: build stripes from them
```

Before everything, we got the complete data for all the n-grams.

First, we create bi-grams from all the n-grams.

In [ ]:
```
%%writefile mapper1.py
#!/w261/venv/bin/python

import sys
count = 0

for line in sys.stdin:
    line=line.strip()
    words=line.split()
    # parameter finding how long is the n-gram
    n = len(words) - 3
    #for word in words:
    if n > 2:
        for i in range(n-1):
            print words[i], words[i+1], words[n]
```

Then, we remove the stop words:

In [1]:
```
%%writefile reducer1.py
#!/w261/venv/bin/python

from nltk.corpus import stopwords

for line in sys.stdin:
    line=line.strip()
    words=line.split()
    if words[0] not in stopwords and words[1] not in stopwords:
        print words[0], words[1], words[2]
```

```
Writing reducer.py
```

In [ ]:
```
!chmod a+x mapper1.py
!chmod a+x reducer1.py
```

Now, we select the 10,000 bigrams with the highest count:

In [2]:
```
!pwd
```

```
/w261/coursework/Untitled Folder
```

In [ ]:
```
#Load the input data into HDFS and make sure the output directory is clear
!bin/hdfs dfs -put /w261/coursework/Untitled Folder/googlebooks-eng-all
!bin/hdfs dfs -rm -r /user/kuknina/googlebooks-bigrams
```

In [ ]:
```
%%bash
bin/hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-0.20-mapreduce/contrib/s
-D mapred.map.tasks=1 \
-D mapred.reduce.tasks=2 \
-file ./mapper1.py    -mapper ./mapper1.py \
-file ./reducer1.py   -reducer ./reducer1.py \
-input /googlebooks-eng-all -output /user/kuknina/googlebooks-bigrams
```

In [ ]:
```
## use this to build the stripes:
## http://nbviewer.jupyter.org/urls/dl.dropbox.com/s/qcirdwyelyebonp/Extende
```

In [ ]:
```
## we change the ngram structure to fit the Similarity tester
```

In [37]:
```
%%writefile mapper.py
#!/w261/venv/bin/python

#HW 2.2 - Mapper Function Code
import sys
doc_no = 0
for line in sys.stdin:
    output_file = open('normalized_stripes.txt', 'w')
    line=line.strip()
    words=line.split()
    # parameter finding how long is the n-gram
    n = len(words) - 3
    diction = {}
    try:
        val = int(words[n+2])
        for i in range(n):
            diction[words[i]] = val
        output_file.write(doc_no, diction)
        #print doc_no, diction
        doc_no +=1
    except:
        pass

    #print diction
        #if n > 2:
            #for i in range(n-1):
                #print words[i], words[i+1], words[n]
```

```
Overwriting mapper.py
```

In [38]:
```
!chmod a+x mapper.py
```

In [39]:
```
!cat googlebooks-eng-all-5gram-20090715-0-filtered.txt | ./mapper.py   | hea
```

In [40]:
```

```

```
#NormalizeStripes()
!cat normalized_stripes.txt
```

In [ ]:
```
# Normalization
%%writefile reducer.py
#!/w261/venv/bin/python

import ast
import math

# Create a method that normalizes the binary stripes.
def NormalizeStripes():
    # Create an output file to write normalized stripes to.
    output_file = open('normalized_stripes.txt', 'w')
    # Read each line of binarized stripes file.
    for line in open('binary_stripes.txt', 'r'):
        line = line.strip()
        contents = line.split("\t")
        word = contents[0]
        # Read in stripe as a dictionary
        stripe = dict(ast.literal_eval(contents[1]))
        # Get length of stripe to use for normalization.
        stripeLen = len(stripe)
        # Iterate through each value in stripes dictionary.
        for key in stripe:
            # Normalize each value by dividing by sqrt(stripeLength).
            stripe[key] = float(1)/math.sqrt(stripeLen)
        output_file.write("\t".join([word, str(stripe)]) + "\n")
```

In [ ]:
```
! chmod a+x reducer.py
```

In [ ]:
```
#now, when the first stage is done, we do the same process
#as we did for the words in the small system test.
```

In [ ]:

```
writefile stripebuilding.py
/w261/venv/bin/python

port re
port mrjob
port json
om mrjob.protocol import RawProtocol
om mrjob.job import MRJob
om mrjob.step import MRStep

ass stripebuilding(MRJob):
  def configure_options(self):
      super(stripebuilding, self).configure_options()
      self.add_passthrough_option(
          '--offset', type='int', default=9000)

  def steps(self):
      JOBCONF_STEP1 = {
      }
      JOBCONF_STEP2 = {
      }
      return [self.mr(jobconf=JOBCONF_STEP1,
                  mapper_init=self.mapper_init,
                  mapper=self.mapper,
                  combiner=self.reducer,
                  reducer=self.reducer)
          ]

  def mapper_init(self):
      stopwords = set(['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourse
      'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', 'her',
      'herself', 'it', 'its', 'itself', 'they', 'them', 'their', 'theirs', 't
      'what', 'which', 'who', 'whom', 'this', 'that', 'these', 'those', 'am',
      'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having',
      'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because',
      'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between
      'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from',
      'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then',
      'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each',
      'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'sa
      'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', 'should

      w = re.compile(r"[\w']+")
      x = 0

      self.topfreq10000 = set()
      self.basis1000 = set()
      with open("part-00000","r") as f:
          for line in f.readlines():
              if x < 10000:
                  line = line.strip().split("\t")
                  word = re.findall(w,line[0])[0]

                  if word not in stopwords:
                      self.topfreq10000.add(word)
                      if x >= self.options.offset:
                          self.basis1000.add(word)
```

```
                    x += 1

            else:
                continue


 def mapper(self,_,line):

     STRIPES = {}

     line = line.strip()
     ngram,count,page_count,book_count = line.split("\t")

     words = map(lambda x: x.lower(), ngram.split())

     for w1 in words:
         if w1 in self.mostFreq_10000:
             STRIPES.setdefault(w1,{})
             for w2 in words:
                 if w2 in self.vocab_basis_1000 and w2 != w1:
                     STRIPES[w1].setdefault(w2,0)
                     STRIPES[w1][w2] += int(count)

             yield w1, STRIPES[w1]

 def reducer(self,key,stripes):
     # make a combined stripe from the list of stripes
     aggregate_stripe = {}
     for stripe in stripes:
         for word in stripe:
             aggregate_stripe.setdefault(word, 0)
             aggregate_stripe[word] += stripe[word]
     yield key, aggregate_stripe

__name__ == '__main__':
 stripebuilding.run()
```

In [ ]:
```
#now, we run this, and output to a file in s3 bucket
```

In [ ]:
```
!aws s3 rm --recursive s3://w261-hw5.4/stripes
!python buildStripes.py -r emr s3://filtered-5grams/ \
    --cluster-id=j-10BB56N1HW4SH \
    --output-dir=s3://w261-hw5.4/stripes \
    --file=mostFrequent/part-00000 \
    --no-output
```

(2) Using two (symmetric) comparison methods of your choice (e.g., correlations, distances, similarities), pairwise compare all stripes (vectors), and output to a file in your bucket on s3.

==Design notes for (2)== For this task you will have to determine a method of comparison. Here are a few that you might consider:

- Jaccard
- Cosine similarity
- Spearman correlation

- Euclidean distance
- Taxicab (Manhattan) distance
- Shortest path graph distance (a graph, because our data is symmetric!)
- Pearson correlation
- Kendall correlation ...

However, be cautioned that some comparison methods are more difficult to parallelize than others, and do not perform more associations than is necessary, since your choice of association will be symmetric.

Please use the inverted index (discussed in live session #5) based pattern to compute the pairwise (term-by-term) similarity matrix.

Please report the size of the cluster used and the amount of time it takes to run for the index construction task and for the synonym calculation task. How many pairs need to be processed (HINT: use the posting list length to calculate directly)? Report your Cluster configuration!

In [ ]:
```
# we create the inverted indices and calculate similarities.
```

In [ ]:

```
%%writefile simcomparison.py
#!/w261/venv/bin/python

import collections
import re
import mrjob
import json
import math
import numpy as np
import itertools
from mrjob.protocol import RawProtocol
from mrjob.job import MRJob
from mrjob.step import MRStep

class simcomparison(MRJob):

    MRJob.SORT_VALUES = True
    def steps(self):
        JOBCONF_STEP1 = {
        }
        JOBCONF_STEP2 = {
        }
        JOBCONF_STEP3 = {
            'mapreduce.job.output.key.comparator.class': 'org.apache.hadoop.
            'mapreduce.partition.keycomparator.options':'-k1,1nr',
        }
        return [MRStep(jobconf=JOBCONF_STEP1,
                       mapper=self.mapper,
                       reducer=self.reducer)
                ,
                MRStep(jobconf=JOBCONF_STEP2,
                       mapper=self.sim_mapper,
                       reducer=self.sim_reducer)
                ,
                MRStep(jobconf=JOBCONF_STEP3,
                       mapper=None,
                       reducer=self.reducer_sort)
                ]



    def mapper(self,_,line):
        line = line.strip()
        key, stripe = line.split("\t")

        key = key.replace('"','')
        stripe = json.loads(stripe)
        l = len(stripe)
        for w in stripe:
            yield w, (key, l)

    def reducer(self,key,value):
        #this will output the invereted indices for each word
        d = collections.defaultdict(list)
        for v in value:
            d[key].append(v)
        yield key,d[key]
```

```
    def sim_mapper(self,key,inv_indx):

        X = map(lambda x: x[0]+"."+str(x[1]) , inv_indx)
        for subset in itertools.combinations(sorted(set(X)), 2):
            yield subset[0]+"."+subset[1], 1

    def sim_reducer(self,key,value):
        w1,l_w1,w2,l_w2 = key.split(".")
        s = sum(value)

        jaccard = s / ( int(l_w1) + int(l_w2) - s )
        cosine = s / ( math.sqrt(int(l_w1))*math.sqrt(int(l_w2)) )

        avg = (jaccard+cosine+dice)/3
        yield avg, (w1+" - "+w2,cosine,jaccard)


    def reducer_sort(self,key,value):
        for v in value:
            yield key,v

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

In [ ]:
```
!mrjob create-cluster --max-hours-idle 1

!aws s3 rm --recursive s3://w261-hw5.4/similarities
!python simcomparison.py -r emr s3://w261-hw5.4/stripes \
    --cluster-id=j-1BGSPMX02IDX \
    --output-dir=s3://w261-hw5.4/similarities \
    --no-output
```

HW 5.5 Evaluation of synonyms that you discovered

In this part of the assignment you will evaluate the success of you synonym detector (developed in response to HW5.4). Take the top 1,000 closest/most similar/correlative pairs of words as determined by your measure in HW5.4, and use the synonyms function in the accompanying python code:

nltk_synonyms.py

Note: This will require installing the python nltk package:

http://www.nltk.org/install.html (http://www.nltk.org/install.html)

and downloading its data with nltk.download().

For each (word1,word2) pair, check to see if word1 is in the list, synonyms(word2), and vice-versa. If one of the two is a synonym of the other, then consider this pair a 'hit', and then report the precision, recall, and F1 measure of your detector across your 1,000 best guesses. Report the macro averages of these measures.

In [ ]:
```
#I have to admit that it would be better to first get
#the previous part working and then evaluate the results.
```