MIDS UC Berkeley - Machine Learning at Scale

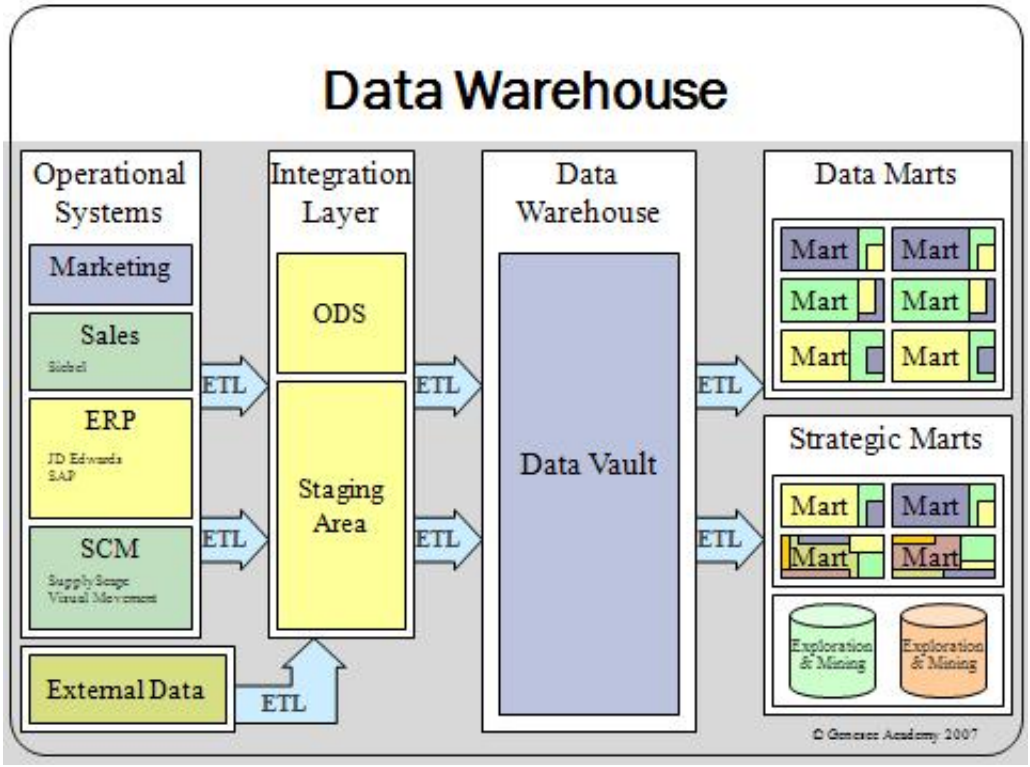
DATSCIW261 ASSIGNMENT #5

James Gray (https://github.com/jamesgray007)
jamesgray@ischool.berkeley.edu
Time of Initial Submission: HH:MM PM US Central, Friday, June 17, 2016
Time of **Resubmission**:
W261-1, Spring 2016
Week 5 Homework

HW 5.0

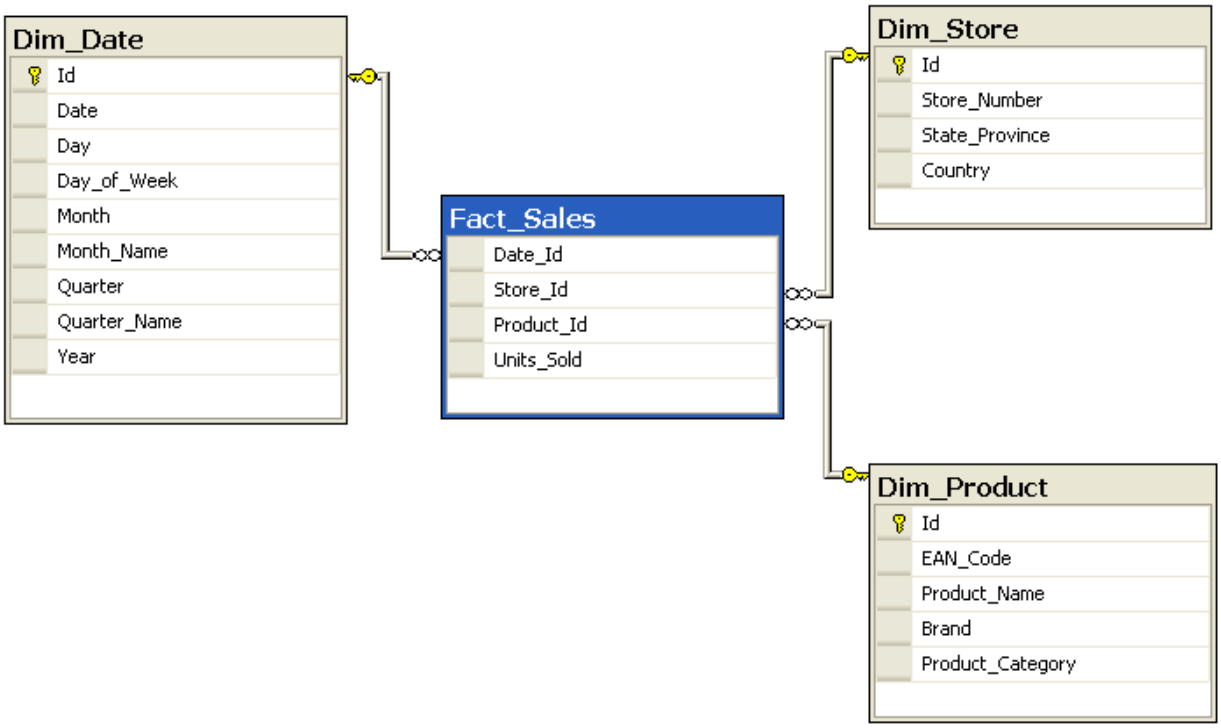
What is a data warehouse?

Bill Inmon's classic definition is that a data warehouse is a subject oriented, integrated, nonvolatile, time variant collection of data in support of management's decisions" (Building the Data Warehouse," John Wiley & Sons, Inc., 1992). A data warehouse integrates multiple data sources across the enterprise including OLTP and unstructured data such as weblogs. Data marts and cubes are built off the data warehouse to enable detailed analytics is specific subject domains such as sales reporting and analytics. A data warehouse is non-volatile and typically stores data over time periods. Data warehouses enable business intelligence but also machine learning workflows given the rich set of enterprise data.



What is a Star schema?

The star schema is one of the more popular Zdata models found within the data warehousing. The star schema is represented by facts and dimensions as shown in the picture below. The "fact" table contains the business measures of interest and the dimensions are foreign keys into tables that are attributes of the data and are used for slicing and dicing the measures. Facts are measures such as total sales, customer volume and units sold. The dimensions enable aggregation and hierarchies such as viewing sales over different time periods (e.g. year, month, week, day). Other typical dimensions include geopgrahy, customer type, product and store. Star schemas are denormalized constructs purposely built for analytics, performance and straighforward queries instead of detailed join conditions.



When is it used?

Star schemas are used for business reporting and analytics where there are well-defined measures and dimensions. They are used when high performance aggregations and query performance is required such as viewing sales over various time periods across various dimensions.

HW 5.1

In the database world What is 3NF? Does machine learning use data in 3NF? If so why?

Third normal form is a normal form that is used in normalizing a database design to reduce the duplication of data and ensure referential integrity by ensuring that (1) the entity is in second normal form, and (2) all the attributes in a table are determined only by the candidate keys of that table and not by any non-prime attributes (Wikipedia (https://en.wikipedia.org/wiki/Third_normal_form)). The normalized data is split across multiple tables and queries are required to assemble the denormalized data for viewing or reporting purposes. Machine learning does not use 3NF data given the normalized design and denormalized data sets are required to easily process the data.

In what form does ML consume data?

Machine learning models generally consume data as rows of tabular data where each line represents a complete record. This is represnted as denormalized data if the data was joined from multiple tables of a 3NF design.

Why would one use log files that are denormalized?

Web logs files by themselves could enable insight but denomalized log files that include other data elements such as user attributes (e.g., name, location, age, ethnicity) would enable deeper analysis such as segmentation, sales campaigns, targeting selling, etc.

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

==========

Join Data Analysis

- Web Page Master (msft_urls.txt) = 294 records (Note from HW4.4 that there are 9 pages did not have visitor)
- Transformed Log file (ms_logs.txt) = 98,654 records

Given that we are performing a memory-backed join, we will need to read the smaller file into memory. Therefor we will read the most frequent visitor file into memory and stream over the log file to perform the log.

LEFT TABLE

WEB PAGE MASTER (294 records)		
PAGE_ID (KEY)	WEB PAGE TITLE	PAGE_URL
1000	regwiz	/regwiz
1001	Support Desktop	/support
1002	End User Produced View	/athome

RIGHT TABLE

WEB PAGE VISITS (98,654 records)				
VISIT	**PAGE_ID (KEY)**	1	C	**VISITOR_ID (KEY)**
V	1000	1	C	10001
V	1001	1	C	10001
V	1001	1	C	10003

LEFT JOIN	JOIN TABLE (98,663) - include 9 records with 0 visits		
	PAGE_ID (KEY)	VISITOR_ID (KEY)	PAGE_URL
All rows from left table even if no matches on right	1000	36585	/regwiz
	1000	10001	/regwiz
	1001	23995	/support
	1294	-	/bookshelf

RIGHT JOIN	JOIN TABLE (98,654) - all web page visits		
	PAGE_ID	PAGE_URL	VISITOR_ID
All rows from right table even if no matches on left	1000	36585	/regwiz
	1000	10001	/regwiz
	1001	23995	/support

INNER JOIN	JOIN TABLE (98,654)		
	PAGE_ID	PAGE_URL	VISITOR_ID
All rows from both tables as long as there is a match between columns	1000	36585	/regwiz
	1000	10001	/regwiz
	1001	23995	/support

In [ ]:
```
!pip install mrjob
```

HW5.2 - Left Hashside Join (memory-backed map-side)

In [29]:
```
%%writefile LeftJoin.py
#!/usr/bin/python
## leftjoin.py
## Author: James Gray
## Description: perform left hashside join on web page master and web

import csv
#import LeftJoin
#reload(LeftJoin)

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

class LeftJoin(MRJob):

    def mapper_init(self):
        """Load file of page of master URLs into memory"""
        with open('msft_urls.txt','rb') as f:
            urls=csv.reader(f.readlines())
        self.url_dict={}
        for i in urls:
            #Saving the URLs into a dictionary will make it easy to ac
            self.url_dict[int(i[0])]=i[2]


    def mapper(self, _, line):
        """Extracts the page that was visited and the visitor id"""

        line=line.strip().split(',')
        page=line[1]
        visitor=line[4]
        #This is the "Left Join" logic that ensures that a row will be
        #every row in the master web page title
        try:
            url=self.url_dict[int(page)]
        except KeyError:
            url='NONE'
        yield page,(visitor,url)

    def steps(self):
        return [MRStep(
                mapper_init=self.mapper_init,
                mapper=self.mapper,
                )]

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

```
Overwriting LeftJoin.py
```

In [ ]:
```
!python leftjoin.py ms_logs.txt --file msft_urls.txt
```

Left Join Driver

In [30]:
```
%reload_ext autoreload
%autoreload 2
from leftjoin import LeftJoin

number_of_rows=0
mr_job = LeftJoin(args=['ms_logs.txt','--file','msft_urls.txt'])
with mr_job.make_runner() as runner:
    runner.run()
    # stream_output: get access of the output
    for line in runner.stream_output():
        #print mr_job.parse_output_line(line)
        number_of_rows+=1

print "Left Join returned rows = " + str(number_of_rows)
```

```
Left Join returned rows = 98654
```

HW5.2 - Right Hashside Join (memory-backed map-side)

In [34]:

```
%%writefile RightJoin.py
#!/usr/bin/python
## rightjoin.py
## Author: James Gray
## Description: perform right hashside join on web page master and web

import csv

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

class RightJoin(MRJob):

    def mapper_init(self):
        """Load file of page URLs into memory"""
        with open('msft_urls.txt','rb') as f:
            urls=csv.reader(f.readlines())
        self.url_dict={}
        for i in urls:
            #Saving the URLs into a dictionary will make it easy to ac
            #the second term here is a flag to see if we've emitted a
            self.url_dict[int(i[0])]=[i[2],0]


    def mapper(self, _, line):
        """Extracts the page that was visited and the visitor id"""
        line=line.strip().split(',')
        page=line[1]
        visitor=line[4]

        try:
            url=self.url_dict[int(page)][0]
            self.url_dict[int(page)][1]=1 #set flag to indicate we've
            yield page,(visitor,url)
        except KeyError:
            pass


    def mapper_final(self):
        """emit any records in the right table we haven't seen yet"""
        for i in self.url_dict.iteritems():
            if i[1][1]==0:
                page=i[0]
                url=i[1][0]
                yield page,('NONE',url)

    def steps(self):
        return [MRStep(
                mapper_init=self.mapper_init,
                mapper=self.mapper,
                mapper_final=self.mapper_final
                )]

if __name__ == '__main__':
```

```
    RightJoin.run()
```

```
Overwriting RightJoin.py
```

Right Join Driver

In [35]:
```
%reload_ext autoreload
%autoreload 2
from RightJoin import RightJoin

number_of_rows=0
mr_job = RightJoin(args=['ms_logs.txt','--file','msft_urls.txt'])
with mr_job.make_runner() as runner:
    runner.run()
    # stream_output: get access of the output
    for line in runner.stream_output():
        #print mr_job.parse_output_line(line)
        number_of_rows+=1

print "Right Join returned rows = " + str(number_of_rows)
```

```
Right Join returned rows = 98704
```

HW5.2 - Inner Hashside Join (memory-backed map-side)

In [36]:
```
%%writefile InnerJoin.py
#!/usr/bin/python
## innerjoin.py
## Author: James Gray
## Description: perform inner hashside join on web page master and web
import csv

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

class InnerJoin(MRJob):

    def mapper_init(self):
        """Load file of page URLs into memory"""
        with open('msft_urls.txt','rb') as f:
            urls=csv.reader(f.readlines())
        self.url_dict={}
        for i in urls:
            #Saving the URLs into a dictionary will make it easy to ac
            #the second term here is a flag to see if we've emitted a
            self.url_dict[int(i[0])]=[i[2],0]


    def mapper(self, _, line):
        """Extracts the page that was visited and the visitor id"""
        line=line.strip().split(',')
        page=line[1]
        visitor=line[4]
        #This is the "Inner Join" logic that emits a row for every rec
        #tables
        try:
            url=self.url_dict[int(page)][0]
            self.url_dict[int(page)][1]=1 #set flag to indicate we've
            yield page,(visitor,url)
        except KeyError:
            #Skip records that don't appear in both tables
            pass


    def steps(self):
        return [MRStep(
                mapper_init=self.mapper_init,
                mapper=self.mapper
            )]

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

```
Overwriting InnerJoin.py
```

Inner Join Driver

In [37]:
```
%reload_ext autoreload
%autoreload 2
from InnerJoin import InnerJoin

number_of_rows=0
mr_job = InnerJoin(args=['ms_logs.txt','--file','msft_urls.txt'])
with mr_job.make_runner() as runner:
    runner.run()
    # stream_output: get access of the output
    for line in runner.stream_output():
        #print mr_job.parse_output_line(line)
        number_of_rows+=1

print "Inner Join returned rows = " + str(number_of_rows)
```

```
Inner Join returned rows = 98654
```

HW 5.3 EDA of Google n-grams dataset

A large subset of the Google n-grams dataset https://aws.amazon.com/datasets/google-books-ngrams/ (https://aws.amazon.com/datasets/google-books-ngrams/)

which we have placed in a bucket/folder on Dropbox on s3:

https://www.dropbox.com/sh/tmqpc4o0xswhkvz/AACUifrl6wrMrlK6a3X3lZ9Ea?dl=0 (https://www.dropbox.com/sh/tmqpc4o0xswhkvz/AACUifrl6wrMrlK6a3X3lZ9Ea?dl=0)

s3://filtered-5grams/

In particular, this bucket contains (~200) files (10Meg each) in the format:

```
(ngram) \t (count) \t (pages_count) \t (books_count)
```

For HW 5.3-5.5, for the Google n-grams dataset unit test and regression test your code using the first 10 lines of the following file:

googlebooks-eng-all-5gram-20090715-0-filtered.txt

Once you are happy with your test results proceed to generating your results on the Google n-grams dataset.

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

1. Longest 5-gram (number of characters)
2. Top 10 most frequent words (please use the count information), i.e., unigrams
3. 20 Most/Least densely appearing words (count/pages_count) sorted in decreasing order of relative frequency
4. 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.

MRJob Configuration File (store on local machine ~/.mrjob.conf)

runners: emr: ec2_key_pair: w261
ec2_key_pair_file: ~/.aws/w261.pem
ssh_tunnel: true
aws_region: us-east-1
ec2_core_instance_type: m3.xlarge
ec2_master_instance_type: m3.xlarge
num_ec2_core_instances: 3

Create Amazon EMR Cluster

In [57]:
```
# create EMR Cluster

!mrjob create-cluster --max-hours-idle 1
```

```
Using configs in /Users/jamesgray/.mrjob.conf
Using s3://mrjob-5a4b4386e2160458/tmp/ as our temp dir on S3
Creating persistent cluster to run several jobs in...
Creating temp directory /var/folders/ld/9wpyxfw13t7_pdv_0b8958x40000
gn/T/no_script.jamesgray.20160619.143625.323507
Copying local files to s3://mrjob-5a4b4386e2160458/tmp/no_script.jam
esgray.20160619.143625.323507/files/...
Can't access IAM API, trying default instance profile: EMR_EC2_Defau
ltRole
Can't access IAM API, trying default service role: EMR_DefaultRole
j-1MTWMS2Y5VS2R
```

Unit Test File

In [39]:
```
!cat ngramtest.txt
```

```
A BILL FOR ESTABLISHING RELIGIOUS       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
A City by the Sea       62      60      49
A Collection of Fairy Tales     123     117     80
A Collection of Forms of        116     103     82
A Commentary on his Apology     110     110     69
A Comparative Study of Juvenile 68      64      44
A Comparison of the Properties  72      72      60
A Conceptual Framework and the  91      91      67
A Conceptual Framework for Life 49      49      40
A Concise Bibliography of the   145     143     122
A Continuation of the Letters   52      51      40
A Critical Review and a 197     194     155
A Critique and a Guide  42      42      42
A Defence of the Royal  153     153     120
```

HW5.3 Part 1 - Longest 5-gram (Number of Characters)

In [40]:
```
%%writefile longestfivegram.py
#!/usr/bin/python
## longestfivegram.py
## Author: James Gray
## Description: calculate the 5-gram with the largest number of charac

import csv

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

class LongestNgram(MRJob):

    def mapper(self, _, line):
        """Emit one record for each ngram length with its correspondin
        line=line.strip().split('\t')
        ngram=line[0]
        #We don't need keys here, since we want the overall max
        yield None,(len(ngram),ngram)

    def reducer(self, _, ngram_and_length):
        """Return only the ngram with the max character length"""
        yield None, max(ngram_and_length)

    def steps(self):
        return [
            MRStep(mapper=self.mapper
                   #Recycle the reducer for the combiner as well
                   ,combiner=self.reducer
                    ,reducer=self.reducer
                  )
        ]

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

```
Writing longestfivegram.py
```

Longest 5-gram Driver (Local Test)

In [41]:
```
#HW 5.3.A - Driver Function for Testing
from longestfivegram import LongestNgram

def run_53ngram():
    mr_job = LongestNgram(args=['ngramtest.txt'])
    with mr_job.make_runner() as runner:
        runner.run()
        for line in runner.stream_output():
            print mr_job.parse_output_line(line)

run_53ngram()
```

```
(None, [33, 'A Circumstantial Narrative of the'])
```

In [ ]:

Longest 5-gram - Run on Amazon EMR

In [56]:
```
!python ./longestfivegram.py \
    -r emr s3://filtered-5grams  \
    --cluster-id j-3NHTTKYIYQAHF \
    --output-dir=s3://jamesgray-w261/longestfivegram \
    --no-output \
    --no-strict-protocol
```

```
Using configs in /Users/jamesgray/.mrjob.conf
Using s3://mrjob-5a4b4386e2160458/tmp/ as our temp dir on S3
Creating temp directory /var/folders/ld/9wpyxfw13t7_pdv_0b8958x40000
gn/T/longestfivegram.jamesgray.20160618.165331.680114
Copying local files to s3://mrjob-5a4b4386e2160458/tmp/longestfivegr
am.jamesgray.20160618.165331.680114/files/...
Adding our job to existing cluster j-3NHTTKYIYQAHF
Waiting for step 1 of 1 (s-2Q83PWD5YS28W) to complete...
  Opening ssh tunnel to resource manager...
  Connect to resource manager at: http://localhost:40456/cluster (ht
tp://localhost:40456/cluster)
  RUNNING for 3.6s
   100.0% complete
  RUNNING for 36.2s
     0.0% complete
  RUNNING for 67.2s
     5.1% complete
  RUNNING for 98.5s
     9.0% complete
  RUNNING for 129.2s
    11.1% complete
  RUNNING for 160.2s
    14.5% complete
  RUNNING for 191.3s
    17.5% complete
  RUNNING for 222.6s
    20.6% complete
  RUNNING for 253.1s
    23.7% complete
  RUNNING for 284.1s
    27.6% complete
  RUNNING for 314.8s
    30.5% complete
  RUNNING for 345.9s
    34.1% complete
  RUNNING for 376.9s
    37.3% complete
  RUNNING for 408.2s
    39.7% complete
  RUNNING for 439.4s
    42.8% complete
  RUNNING for 470.6s
    45.6% complete
  RUNNING for 501.4s
    48.2% complete
  RUNNING for 533.1s
    51.2% complete
  RUNNING for 564.2s
    53.8% complete
  RUNNING for 600.1s
    82.7% complete
  COMPLETED
Attempting to fetch counters from logs...
Looking for step log in /mnt/var/log/hadoop/steps/s-2Q83PWD5YS28W on
```

```
ec2-54-165-245-217.compute-1.amazonaws.com...
  Parsing step log: ssh://ec2-54-165-245-217.compute-1.amazonaws.co
m/mnt/var/log/hadoop/steps/s-2Q83PWD5YS28W/syslog.2016-06-18-16
  Parsing step log: ssh://ec2-54-165-245-217.compute-1.amazonaws.co
m/mnt/var/log/hadoop/steps/s-2Q83PWD5YS28W/syslog
Counters: 55
        File Input Format Counters
                Bytes Read=2156069116
        File Output Format Counters
                Bytes Written=174
        File System Counters
                FILE: Number of bytes read=9044
                FILE: Number of bytes written=21003901
                FILE: Number of large read operations=0
                FILE: Number of read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=23450
                HDFS: Number of bytes written=0
                HDFS: Number of large read operations=0
                HDFS: Number of read operations=190
                HDFS: Number of write operations=0
                S3: Number of bytes read=2156069116
                S3: Number of bytes written=174
                S3: Number of large read operations=0
                S3: Number of read operations=0
                S3: Number of write operations=0
        Job Counters
                Data-local map tasks=188
                Launched map tasks=190
                Launched reduce tasks=11
                Other local map tasks=2
                Total megabyte-seconds taken by all map tasks=112381
32960
                Total megabyte-seconds taken by all reduce tasks=584
4150720
                Total time spent by all map tasks (ms)=7804259
                Total time spent by all maps in occupied slots (ms)=
351191655
                Total time spent by all reduce tasks (ms)=2029219
                Total time spent by all reduces in occupied slots (m
s)=182629710
                Total vcore-seconds taken by all map tasks=7804259
                Total vcore-seconds taken by all reduce tasks=202921
9
        Map-Reduce Framework
                CPU time spent (ms)=2947120
                Combine input records=58682266
                Combine output records=188
                Failed Shuffles=0
                GC time elapsed (ms)=55808
                Input split bytes=23450
                Map input records=58682266
                Map output bytes=2331730493
                Map output materialized bytes=48307
```

```
                Map output records=58682266
                Merged Map outputs=2090
                Physical memory (bytes) snapshot=100191006720
                Reduce input groups=1
                Reduce input records=188
                Reduce output records=1
                Reduce shuffle bytes=48307
                Shuffled Maps =2090
                Spilled Records=376
                Total committed heap usage (bytes)=125291200512
                Virtual memory (bytes) snapshot=409895063552
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
Removing s3 temp directory s3://mrjob-5a4b4386e2160458/tmp/longestfi
vegram.jamesgray.20160618.165331.680114/...
Removing temp directory /var/folders/ld/9wpyxfw13t7_pdv_0b8958x40000
gn/T/longestfivegram.jamesgray.20160618.165331.680114...
Killing our SSH tunnel (pid 28208)
```

Copy S3 Output to Local File System - Longest NGram

In [77]:
```
!aws s3 cp --recursive s3://jamesgray-w261/longestfivegram ./longestfi
!echo "LONGEST FIVE NGRAM:"
!cat ./longestfivegram/part-* | sort -k2nr | head -1
```

```
download: s3://jamesgray-w261/longestfivegram/part-00001 to longestf
ivegram/part-00001
download: s3://jamesgray-w261/longestfivegram/part-00000 to longestf
ivegram/part-00000
download: s3://jamesgray-w261/longestfivegram/part-00006 to longestf
ivegram/part-00006
download: s3://jamesgray-w261/longestfivegram/part-00003 to longestf
ivegram/part-00003
download: s3://jamesgray-w261/longestfivegram/part-00004 to longestf
ivegram/part-00004
download: s3://jamesgray-w261/longestfivegram/part-00005 to longestf
ivegram/part-00005
download: s3://jamesgray-w261/longestfivegram/part-00007 to longestf
ivegram/part-00007
download: s3://jamesgray-w261/longestfivegram/part-00008 to longestf
ivegram/part-00008
download: s3://jamesgray-w261/longestfivegram/part-00009 to longestf
ivegram/part-00009
download: s3://jamesgray-w261/longestfivegram/part-00010 to longestf
ivegram/part-00010
download: s3://jamesgray-w261/longestfivegram/part-00002 to longestf
ivegram/part-00002
download: s3://jamesgray-w261/longestfivegram/_SUCCESS to longestfiv
egram/_SUCCESS
LONGEST FIVE NGRAM:
null    [159, "ROPLEZIMPREDASTRODONBRASLPKLSON YHROACLMPARCHEYXMMIOU
DAVESAURUS PIOFPILOCOWERSURUASOGETSESNEGCP TYRAVOPSIFENGOQUAPIALLOBO
SKENUO OWINFUYAIOKENECKSASXHYILPOYNUAT"]
```

HW5.3 Part 2 - Top 10 most frequent words

In [1]:
```
%%writefile toptenwords.py
#!/usr/bin/python
## toptenwords.py
## Author: James Gray
## Description: calculate the top ten words across the n-gram corpus

import csv
import re

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

class TopTenWords(MRJob):

    def mapper(self, _, line):
        counts = {}
        line.strip()
        #Parse fields from each line
        [ngram,count,pages,books] = re.split("\t",line)
        count = int(count)
        words = re.split(" ",ngram)
        for word in words:
            # set words to lowercase
            counts.setdefault(word.lower(),0)
            counts[word.lower()] += count
        for word in counts.keys():
            yield word,counts[word]

    def combiner(self,word,count):
        yield word,sum(count)

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

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

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

```
Overwriting toptenwords.py
```

Top Ten Words Driver (Local Test)

In [2]:
```
from toptenwords import TopTenWords

def run_topten():
    mr_job = TopTenWords(args=['ngramtest.txt'])
    with mr_job.make_runner() as runner:
        runner.run()
        for line in runner.stream_output():
            print mr_job.parse_output_line(line)

run_topten()
```

```
('a', 3435)
('and', 330)
('apology', 110)
('bibliography', 145)
('bill', 59)
('biography', 92)
('by', 62)
('case', 604)
("child's", 1099)
('christmas', 1099)
('circumstantial', 62)
('city', 62)
('collection', 239)
('commentary', 110)
('comparative', 68)
('comparison', 72)
('conceptual', 140)
('concise', 145)
('continuation', 52)
('critical', 197)
('critique', 42)
('defence', 153)
('establishing', 59)
('fairy', 123)
('female', 447)
('for', 108)
('forms', 116)
('framework', 140)
('general', 92)
('george', 92)
('government', 102)
('guide', 42)
('his', 110)
('in', 1201)
('juvenile', 68)
('letters', 52)
('life', 49)
('limited', 55)
('narrative', 62)
('of', 1501)
('on', 110)
('properties', 72)
('religious', 59)
('review', 197)
('royal', 153)
('sea', 62)
('study', 672)
('tales', 123)
('the', 637)
('wales', 1099)
```

Top Ten Words - Amazon EMR

In [4]:
```
!python ./toptenwords.py \
    -r emr s3://filtered-5grams  \
    --cluster-id j-3NHTTKYIYQAHF \
    --output-dir=s3://jamesgray-w261/toptenwords \
    --no-output \
    --no-strict-protocol
```

```
Using configs in /Users/jamesgray/.mrjob.conf
Using s3://mrjob-5a4b4386e2160458/tmp/ as our temp dir on S3
Creating temp directory /var/folders/ld/9wpyxfw13t7_pdv_0b8958x400
00gn/T/toptenwords.jamesgray.20160618.175514.477196
Copying local files to s3://mrjob-5a4b4386e2160458/tmp/toptenword
s.jamesgray.20160618.175514.477196/files/...
Adding our job to existing cluster j-3NHTTKYIYQAHF
Waiting for step 1 of 1 (s-34LL837RG0FY2) to complete...
  Opening ssh tunnel to resource manager...
  Connect to resource manager at: http://localhost:40456/cluster
(http://localhost:40456/cluster)
  RUNNING for 18.2s
   100.0% complete
  RUNNING for 50.2s
     5.0% complete
  RUNNING for 81.0s
     5.6% complete
  RUNNING for 112.4s
     7.1% complete
```

S3 Output to Local File System - Top Ten Words

In [5]:
```
!aws s3 cp --recursive s3://jamesgray-w261/toptenwords ./toptenwords
!cat ./toptenwords/part-* | sort -k2nr | head -10000 > ./toptenwords.t
!echo "TOP TEN WORDS"
!cat ./toptenwords.txt | head -10
```

```
download: s3://jamesgray-w261/toptenwords/_SUCCESS to toptenwords/_S
UCCESS
download: s3://jamesgray-w261/toptenwords/part-00007 to toptenwords/
part-00007
download: s3://jamesgray-w261/toptenwords/part-00004 to toptenwords/
part-00004
download: s3://jamesgray-w261/toptenwords/part-00006 to toptenwords/
part-00006
download: s3://jamesgray-w261/toptenwords/part-00002 to toptenwords/
part-00002
download: s3://jamesgray-w261/toptenwords/part-00000 to toptenwords/
part-00000
download: s3://jamesgray-w261/toptenwords/part-00005 to toptenwords/
part-00005
download: s3://jamesgray-w261/toptenwords/part-00010 to toptenwords/
part-00010
download: s3://jamesgray-w261/toptenwords/part-00008 to toptenwords/
part-00008
download: s3://jamesgray-w261/toptenwords/part-00001 to toptenwords/
part-00001
download: s3://jamesgray-w261/toptenwords/part-00009 to toptenwords/
part-00009
download: s3://jamesgray-w261/toptenwords/part-00003 to toptenwords/
part-00003
sort: write failed: standard output: Broken pipe
sort: write error
TOP TEN WORDS
"the"   5490815394
"of"    3698583299
"to"    2227866570
"in"    1421312776
"a"     1361123022
"and"   1149577477
"that"  802921147
"is"    758328796
"be"    688707130
"as"    492170314
cat: stdout: Broken pipe
```

HW5.3 Part 3 - 20 Most/Least densely appearing words (count/pages_count) sorted in decreasing order of relative frequency

In [6]:
```
%%writefile density.py
#!/usr/bin/python
## density.py
## Author: James Gray
## Description: calculate 20 most/least densely appearing words

from __future__ import division
import csv

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

class Density(MRJob):

    def mapper(self, _, line):
        """Emit one record per word with count and page count """
        line=line.strip().split('\t')
        ngram=line[0]
        count=line[1]
        page_count=line[2]
        for word in ngram.split(' '):
            yield word,(count,page_count)

    def combiner(self,word,count):
        """Aggregate intermediate word counts and page counts"""
        word_count=0
        page_count=0
        for record in count:
            word_count+=int(record[0])
            page_count+=int(record[1])
        yield word,(word_count,page_count)

    def reducer(self,word,count):
        """Final aggregation of word counts and page counts, divided f
        word_count=0
        page_count=0
        for record in count:
            word_count+=int(record[0])
            page_count+=int(record[1])
        yield word,word_count/page_count

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

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

```
Overwriting density.py
```

Density Driver - Local Test

In [8]:
```
from density import Density

def run_density():
    mr_job = Density(args=['ngramtest.txt'])
    with mr_job.make_runner() as runner:
        runner.run()
        for line in runner.stream_output():
            print mr_job.parse_output_line(line)

run_density()
```

```
('A', 1.02272)
('Apology', 1.0)
('BILL', 1.0)
('Bibliography', 1.013986013986014)
('Biography', 1.0222222222222221)
('Case', 1.0)
("Child's", 1.0358152686145146)
('Christmas', 1.0358152686145146)
('Circumstantial', 1.0)
('City', 1.0333333333333334)
('Collection', 1.0863636363636364)
('Commentary', 1.0)
('Comparative', 1.0625)
('Comparison', 1.0)
('Conceptual', 1.0)
('Concise', 1.013986013986014)
('Continuation', 1.0196078431372548)
('Critical', 1.0154639175257731)
('Critique', 1.0)
('Defence', 1.0)
('ESTABLISHING', 1.0)
('FOR', 1.0)
('Fairy', 1.0512820512820513)
('Female', 1.0)
('Forms', 1.1262135922330097)
('Framework', 1.0)
('General', 1.0222222222222221)
('George', 1.0222222222222221)
('Government', 1.0)
('Guide', 1.0)
('Juvenile', 1.0625)
('Letters', 1.0196078431372548)
('Life', 1.0)
('Limited', 1.0)
('Narrative', 1.0)
('Properties', 1.0)
('RELIGIOUS', 1.0)
('Review', 1.0154639175257731)
('Royal', 1.0)
('Sea', 1.0333333333333334)
('Study', 1.0059880239520957)
('Tales', 1.0512820512820513)
('Wales', 1.0358152686145146)
('a', 1.0127118644067796)
('and', 1.0091743119266054)
('by', 1.0333333333333334)
('for', 1.0)
('his', 1.0)
('in', 1.0326741186586414)
('of', 1.0280821917808218)
('on', 1.0)
('the', 1.0079113924050633)
```

Density - Amazon EMR

In [9]:
```
!python ./density.py \
    -r emr s3://filtered-5grams  \
    --cluster-id j-3NHTTKYIYQAHF \
    --output-dir=s3://jamesgray-w261/density \
    --no-output \
    --no-strict-protocol
```

```
  RUNNING for 166.7s
     8.6% complete
  RUNNING for 198.0s
     9.1% complete
  RUNNING for 229.6s
    10.9% complete
  RUNNING for 260.0s
    12.2% complete
  RUNNING for 291.2s
    13.5% complete
  RUNNING for 321.7s
    14.2% complete
  RUNNING for 352.6s
    15.2% complete
  RUNNING for 383.8s
    16.6% complete
  RUNNING for 415.5s
    18.1% complete
  RUNNING for 446.1s
    19.0% complete
```

S3 Output to Local File System - Density

In [11]:
```
!aws s3 cp --recursive s3://jamesgray-w261/density ./density
!echo "TOP 20 HIGH DENSITY WORDS"
!cat ./density/part-* | sort -k2nr | head -20
!echo ""
!echo "BOTTOM 20 LOWEST DENSITY WORDS"
!cat ./density/part-* | sort -k2nr | tail -20
```

```
download: s3://jamesgray-w261/density/_SUCCESS to density/_SUCCESS
download: s3://jamesgray-w261/density/part-00005 to density/part-000
05
download: s3://jamesgray-w261/density/part-00004 to density/part-000
04
download: s3://jamesgray-w261/density/part-00002 to density/part-000
02
download: s3://jamesgray-w261/density/part-00000 to density/part-000
00
download: s3://jamesgray-w261/density/part-00001 to density/part-000
01
download: s3://jamesgray-w261/density/part-00010 to density/part-000
10
download: s3://jamesgray-w261/density/part-00008 to density/part-000
08
download: s3://jamesgray-w261/density/part-00006 to density/part-000
06
download: s3://jamesgray-w261/density/part-00003 to density/part-000
03
download: s3://jamesgray-w261/density/part-00009 to density/part-000
09
download: s3://jamesgray-w261/density/part-00007 to density/part-000
07
TOP 20 HIGH DENSITY WORDS
"xxxx"  11.557291666666666
"NA"    10.161726044782885
"blah"  8.074159907300116
"nnn"   7.533333333333333
"nd"    6.561143644505684
"ND"    5.40736428467472
"oooooooooooooooo"      4.921875
"PIC"   4.7272727272727275
"llll"  4.511627906976744
"LUTHER"        4.349498327759197
"oooooo"        4.207237859573151
"NN"    4.0908402725208175
"ooooo" 3.9492846924177396
"OOOOOO"        3.9313725490196076
"IIII"  3.7877030162412995
"lillelu"       3.7624521072796937
"OOOOO" 3.6570701447431206
"Sc"    3.6065625
"Madarassy"     3.576923076923077
"Pfeffermann"   3.576923076923077
sort: write failed: standard output: Broken pipe
sort: write error

BOTTOM 20 LOWEST DENSITY WORDS
"zusammen"      1.0
"zusammengenommen"      1.0
"zusammengetroffen"     1.0
"zwangloser"    1.0
"zwanziger"     1.0
"zweier"        1.0
```

```
"zweifache"     1.0
"zweigt"        1.0
"zwingst"       1.0
"zwischenstaatlicher"   1.0
"zwitterionic"  1.0
"zydeco"        1.0
"zygomaticofacial"      1.0
"zygomaticotemporal"    1.0
"zygosity"      1.0
"zylindrischen" 1.0
"zymogens"      1.0
"zymophore"     1.0
"zymosan"       1.0
"zymosis"       1.0
```

HW5.3 Part 4 - Distribution of 5-gram sizes (character length)

In [12]:

```
%%writefile distribution.py
#HW 5.3.D - Ngram Distribution MRJob Definition
from __future__ import division
import csv

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

class Distribution(MRJob):

    def mapper_init(self):
        self.count=0

    def mapper(self, _, line):
        """Emit records with ngrams and size"""
        line=line.strip().split('\t')
        ngram=line[0] #The text of the ngram
        size=len(ngram)
        ngram_count=int(line[1]) #The count of the ngram
        self.count+=ngram_count #Add the count to the running total of
        yield size,ngram_count #Yield the ngram and its count

    def mapper_final(self):
        """We needed this for the original statement of the problem, w
        required relative frequencies.  Though we've left the step in
        the result is no longer used."""
        yield '*count',self.count #Yield the total for order-inversion

    def reducer_init(self):
        self.total_count=None

    def reducer(self,size,ngram_count):
        total=sum(ngram_count)
        overall_total=None
        #Capture the totals for a relative frequency calcuation (no lo
        if size=='*count':
            overall_total=total
            self.total_count=total
        else:
            #Yield the character length and the number of ngrams with
            #(relative freq. calculation is commented out)
            yield size,(total)#, total/self.total_count)

    def steps(self):
        return [
            MRStep(
                mapper_init=self.mapper_init,
                mapper=self.mapper
                ,mapper_final=self.mapper_final
                ,reducer_init=self.reducer_init
                ,reducer=self.reducer
                  )
        ]
```

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

```
Writing distribution.py
```

Distribution Driver (Local Test)

In [14]:
```
from distribution import Distribution

def run_distribution():
    mr_job = Distribution(args=['ngramtest.txt'])
    with mr_job.make_runner() as runner:
        runner.run()
        for line in runner.stream_output():
            print mr_job.parse_output_line(line)

run_distribution()
```

```
(17, 62)
(22, 642)
(23, 252)
(24, 116)
(26, 102)
(27, 233)
(28, 1099)
(29, 289)
(30, 163)
(31, 117)
(33, 121)
```

Distribution - Amazon EMR

In [15]:
```
!python ./distribution.py \
    -r emr s3://filtered-5grams  \
    --cluster-id j-3NHTTKYIYQAHF \
    --output-dir=s3://jamesgray-w261/distribution \
    --no-output \
    --no-strict-protocol
```

```
Using configs in /Users/jamesgray/.mrjob.conf
Using s3://mrjob-5a4b4386e2160458/tmp/ as our temp dir on S3
Creating temp directory /var/folders/ld/9wpyxfw13t7_pdv_0b8958x400
00gn/T/distribution.jamesgray.20160618.185533.609256
Copying local files to s3://mrjob-5a4b4386e2160458/tmp/distributio
n.jamesgray.20160618.185533.609256/files/...
Adding our job to existing cluster j-3NHTTKYIYQAHF
Waiting for step 1 of 1 (s-2JEP66J97PZTJ) to complete...
  Opening ssh tunnel to resource manager...
  Connect to resource manager at: http://localhost:40456/cluster
(http://localhost:40456/cluster)
  RUNNING for 20.8s
   100.0% complete
  RUNNING for 53.0s
     5.0% complete
  RUNNING for 83.9s
     7.6% complete
  RUNNING for 115.1s
    10.8% complete
```

S3 Output to Local File System & Plot

In [16]:
```
!aws s3 cp --recursive s3://jamesgray-w261/distribution ./distribution
```

```
download: s3://jamesgray-w261/distribution/part-00005 to distributio
n/part-00005
download: s3://jamesgray-w261/distribution/part-00008 to distributio
n/part-00008
download: s3://jamesgray-w261/distribution/part-00003 to distributio
n/part-00003
download: s3://jamesgray-w261/distribution/part-00006 to distributio
n/part-00006
download: s3://jamesgray-w261/distribution/part-00004 to distributio
n/part-00004
download: s3://jamesgray-w261/distribution/part-00000 to distributio
n/part-00000
download: s3://jamesgray-w261/distribution/part-00007 to distributio
n/part-00007
download: s3://jamesgray-w261/distribution/part-00001 to distributio
n/part-00001
download: s3://jamesgray-w261/distribution/part-00002 to distributio
n/part-00002
download: s3://jamesgray-w261/distribution/part-00009 to distributio
n/part-00009
download: s3://jamesgray-w261/distribution/_SUCCESS to distribution/
_SUCCESS
download: s3://jamesgray-w261/distribution/part-00010 to distributio
n/part-00010
```

In [23]:
```
#%matplotlib inline

import os
import numpy as np
import matplotlib.pyplot as plt

def run_plot():
    lengths=[]
    totals=[]
    for i in os.listdir('./distribution'):
        if i.startswith('part'):
            #Load results from each output file we downloaded
            with open(i) as f:
                for line in f.readlines():
                    [length,total]=line.strip().split('\t')
                    #Save lengths and totals into separate vectors for
                    lengths.append(int(length))
                    totals.append(int(total))

    fig, chart = plt.subplots()
    #We already know the bar heights, so we can plot them directly rat
    chart.bar(lengths,totals)
    chart.set_ylabel('Count')
    chart.set_xlabel('N-gram length (in characters)')
    chart.set_title('Distribution of n-gram lengths')

    fig = plt.gcf()

run_plot()
```

```
---------------------------------------------------------------------
-------
IOError                                   Traceback (most recent cal
l last)
<ipython-input-23-5f6a939c3dfb> in <module>()
     27     fig = plt.gcf()
     28
---> 29 run_plot()

<ipython-input-23-5f6a939c3dfb> in run_plot()
     11         if i.startswith('part'):
     12             #Load results from each output file we downloade
d
---> 13             with open(i) as f:
     14                 for line in f.readlines():
     15                     [length,total]=line.strip().split('\t')

IOError: [Errno 2] No such file or directory: 'part-00000'
```

HW 5.3.1 OPTIONAL Question:

- Plot the log-log plot of the frequency distributuion of unigrams. Does it follow power law distribution?

For more background see: https://en.wikipedia.org/wiki/Log%E2%80%93log_plot (https://en.wikipedia.org/wiki/Log%E2%80%93log_plot)
https://en.wikipedia.org/wiki/Power_law (https://en.wikipedia.org/wiki/Power_law)

In [ ]:

HW 5.4 Synonym detection over 2Gig of Data

For the remainder of this assignment you will work with two datasets:

1: unit/systems test data set: SYSTEMS TEST DATASET

Three terms, A,B,C and their corresponding strip-docs of co-occurring terms

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

2: A large subset of the Google n-grams dataset as was described above

For each HW 5.4 - 5.5.1 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.

In this part of the assignment we will focus on developing methods for detecting synonyms, using the Google 5-grams dataset. To accomplish this you must script two main tasks using MRJob:

Task 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).

Task 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 (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).

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!

HW5.4 - Build Stripe Docs for Word Co-Occurences

We are using the vocabulary based on the words with frequency ranked 9001-10,000 taken from the "Top Ten Words" problem above that generated a text file of the top 10,000 words (toptenwords.txt). The vocabulary file is "bottomwords.txt" (words 9001-10000).

In [52]:

```
%%writefile stripes.py
#HW 5.4 - Stripes MRJob Definition
from __future__ import division
from itertools import combinations
import csv

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

class BuildStripes(MRJob):

    def jobconf(self):
        orig_jobconf = super(BuildStripes, self).jobconf()
        # Setting these high enough improves EMR job speed
        custom_jobconf = {
            "mapred.map.tasks":28,
            "mapred.reduce.tasks":28
        }
        return conf.combine_dicts(orig_jobconf, custom_jobconf)


    def mapper_init(self):
        """Load file for vocabulary -> words with frequency 9001-10000
        self.word_dict={}

        #This is the file of words with frequency ranked 9000-10000 th
        with open('bottomwords.txt','rb') as f:
            for row in f.readlines():
                line=row.strip().split('\t')
                self.word_dict[line[0][1:-1]]=line[1]


    def mapper(self, _, line):
        """
        Emit co-occurrence combinations for each pair of relevant word
        """
        line=line.strip().split('\t') # parse ngram line into tokens
        ngram=line[0].lower() # convert all tokens to lowercase
        count=int(line[1]) # count of occurrences in corpus
        ngram_words=ngram.split(" ") # individual words in our ngram
        output={}

        #Pull out words from ngram that we care about (those that appe
        words=[i for i in ngram_words if i in self.word_dict.keys()]

        # Update output stripe for each combination of co-occurring, r
        for word1,word2 in combinations(words,2):

            if word1 in output.keys():
                output[word1][word2]=output[word1].get(word2,0)+count
            else:
                output[word1]={word2:count}
```

```
            # maintain symmetry for opposite co-occurence
            if word2 in output.keys():
                output[word2][word1]=output[word2].get(word1,0)+count
            else:
                output[word2]={word1:count}

        # generate output of word, co-occurrence
        for word,cooccur in output.iteritems():
            yield word,cooccur

    def reducer(self,word,cos):
        """Aggregate stripes based on intermediate results from mapper
        output_dict={}
        for co in cos:
            # The second_word variable here is so named to distinguish
            # and refers to the words in the co-occurrence stripe
            for second_word,count in co.iteritems():
                output_dict[second_word] = output_dict.get(second_word
        yield word, output_dict


    def steps(self):
        return [
            MRStep(
                mapper_init=self.mapper_init,
                mapper=self.mapper
                #We can recycle the reducer as combiner here, which is
                ,combiner=self.reducer
                ,reducer=self.reducer
                  )
        ]

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

```
Overwriting stripes.py
```

Vocabulary (Most Frequent Words 9,001-10000)

In [71]:
```
!cat bottomwords.txt | head -15
```

```
"surveys"       169333
"jungle"        169314
"lacked"        169282
"correlate"     169273
"boxes" 169237
"escort"        169220
"disclosed"     169132
"shepherd"      169114
"commend"       169081
"zenith"        169049
"multiplication"        169025
"epic"  169004
"literacy"      168967
"atonement"     168908
"soda"  168897
```

In [70]:
```
!cat ngramtest.txt
```

```
A BILL FOR ESTABLISHING RELIGIOUS       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
A City by the Sea       62      60      49
A Collection of Fairy Tales     123     117     80
A Collection of Forms of        116     103     82
A Commentary on his Apology     110     110     69
A Comparative Study of Juvenile 68      64      44
A Comparison of the Properties  72      72      60
A Conceptual Framework and the  91      91      67
A Conceptual Framework for Life 49      49      40
A Concise Bibliography of the   145     143     122
A Continuation of the Letters   52      51      40
A Critical Review and a 197     194     155
A Critique and a Guide  42      42      42
A Defence of the Royal  153     153     120
```

In [72]:
```
%%writefile small_test.txt
atlas boon  50  50  50
boon cava dipped    10  10  10
atlas dipped    15  15  15
```

```
Writing small_test.txt
```

HW 5.4 Driver - Local Test for Creating Stripes

This creates stripes using the vocabulary of 1000 words.

In [76]:
```
%reload_ext autoreload
%autoreload 2

from stripes import BuildStripes

def run_5_4_stripe_test():
    mr_job = BuildStripes(args=['ngramtest.txt','--file','bottomwords.
    with mr_job.make_runner() as runner:
        runner.run()
        for line in runner.stream_output():
            print mr_job.parse_output_line(line)

run_5_4_stripe_test()
```

Unit Test Stripes on Amazon EMR

In [59]:
```
!python ./stripes.py \
    -r emr s3://jamesgray-w261/ngramtest.txt  \
    --file ./bottomwords.txt \
    --cluster-id j-1MTWMS2Y5VS2R \
    --output-dir=s3://jamesgray-w261/stripes_unittest \
    --no-output \
    --no-strict-protocol
```

```
Using configs in /Users/jamesgray/.mrjob.conf
Using s3://mrjob-5a4b4386e2160458/tmp/ as our temp dir on S3
Creating temp directory /var/folders/ld/9wpyxfw13t7_pdv_0b8958x40000
gn/T/stripes.jamesgray.20160619.143900.403013
Copying local files to s3://mrjob-5a4b4386e2160458/tmp/stripes.james
gray.20160619.143900.403013/files/...
Adding our job to existing cluster j-1MTWMS2Y5VS2R
Detected hadoop configuration property names that do not match hadoo
p version 2.4.0:
The have been translated as follows
 mapred.map.tasks: mapreduce.job.maps
mapred.reduce.tasks: mapreduce.job.reduces
Waiting for step 1 of 1 (s-61PMO8YM2A1S) to complete...
  PENDING (cluster is STARTING: Configuring cluster software)
  PENDING (cluster is BOOTSTRAPPING: Running bootstrap actions)
  PENDING (cluster is BOOTSTRAPPING: Running bootstrap actions)
  Opening ssh tunnel to resource manager...
  Connect to resource manager at: http://localhost:40176/cluster (ht
tp://localhost:40176/cluster)
  RUNNING for 22.9s
     5.0% complete
  RUNNING for 55.9s
    39.1% complete
  RUNNING for 87.4s
    82.1% complete
  COMPLETED
Attempting to fetch counters from logs...
Looking for step log in /mnt/var/log/hadoop/steps/s-61PMO8YM2A1S on
ec2-54-84-230-119.compute-1.amazonaws.com...
  Parsing step log: ssh://ec2-54-84-230-119.compute-1.amazonaws.com/
mnt/var/log/hadoop/steps/s-61PMO8YM2A1S/syslog
Counters: 54
        File Input Format Counters
                Bytes Read=11078
        File Output Format Counters
                Bytes Written=0
        File System Counters
                FILE: Number of bytes read=560
                FILE: Number of bytes written=5976323
                FILE: Number of large read operations=0
                FILE: Number of read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=2465
                HDFS: Number of bytes written=0
                HDFS: Number of large read operations=0
                HDFS: Number of read operations=29
                HDFS: Number of write operations=0
                S3: Number of bytes read=11078
                S3: Number of bytes written=0
                S3: Number of large read operations=0
                S3: Number of read operations=0
                S3: Number of write operations=0
        Job Counters
                Data-local map tasks=29
```

```
                Launched map tasks=29
                Launched reduce tasks=28
                Total megabyte-seconds taken by all map tasks=931648
320
                Total megabyte-seconds taken by all reduce tasks=103
2454080
                Total time spent by all map tasks (ms)=646978
                Total time spent by all maps in occupied slots (ms)=
29114010
                Total time spent by all reduce tasks (ms)=358491
                Total time spent by all reduces in occupied slots (m
s)=32264190
                Total vcore-seconds taken by all map tasks=646978
                Total vcore-seconds taken by all reduce tasks=358491
        Map-Reduce Framework
                CPU time spent (ms)=52510
                Combine input records=0
                Combine output records=0
                Failed Shuffles=0
                GC time elapsed (ms)=13785
                Input split bytes=2465
                Map input records=20
                Map output bytes=0
                Map output materialized bytes=12992
                Map output records=0
                Merged Map outputs=812
                Physical memory (bytes) snapshot=22207410176
                Reduce input groups=0
                Reduce input records=0
                Reduce output records=0
                Reduce shuffle bytes=12992
                Shuffled Maps =812
                Spilled Records=0
                Total committed heap usage (bytes)=25874661376
                Virtual memory (bytes) snapshot=147800072192
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
Removing s3 temp directory s3://mrjob-5a4b4386e2160458/tmp/stripes.j
amesgray.20160619.143900.403013/...
Removing temp directory /var/folders/ld/9wpyxfw13t7_pdv_0b8958x40000
gn/T/stripes.jamesgray.20160619.143900.403013...
Killing our SSH tunnel (pid 32789)
```

Full Data Set Stripes on Amazon EMR

In [60]:
```
!python ./stripes.py \
    -r emr s3://filtered-5grams  \
    --file ./bottomwords.txt \
    --cluster-id j-1MTWMS2Y5VS2R \
    --output-dir=s3://jamesgray-w261/stripes_full \
    --no-output \
    --no-strict-protocol
```

```
Using configs in /Users/jamesgray/.mrjob.conf
Using s3://mrjob-5a4b4386e2160458/tmp/ as our temp dir on S3
Creating temp directory /var/folders/ld/9wpyxfw13t7_pdv_0b8958x40000
gn/T/stripes.jamesgray.20160619.144912.668222
Copying local files to s3://mrjob-5a4b4386e2160458/tmp/stripes.james
gray.20160619.144912.668222/files/...
Adding our job to existing cluster j-1MTWMS2Y5VS2R
Detected hadoop configuration property names that do not match hadoo
p version 2.4.0:
The have been translated as follows
 mapred.map.tasks: mapreduce.job.maps
mapred.reduce.tasks: mapreduce.job.reduces
Waiting for step 1 of 1 (s-1O9UEVM1FB654) to complete...
  Opening ssh tunnel to resource manager...
  Connect to resource manager at: http://localhost:40176/cluster (ht
tp://localhost:40176/cluster)
  RUNNING for 21.1s
Unable to connect to resource manager
  RUNNING for 53.7s
  RUNNING for 84.8s
  RUNNING for 116.4s
  RUNNING for 147.1s
  RUNNING for 200.1s
  RUNNING for 341.1s
  RUNNING for 372.3s
  RUNNING for 414.0s
  RUNNING for 444.3s
  RUNNING for 476.0s
  RUNNING for 506.7s
  RUNNING for 537.6s
  RUNNING for 568.6s
  RUNNING for 857.1s
  RUNNING for 887.5s
  RUNNING for 919.0s
  RUNNING for 949.3s
  RUNNING for 980.2s
  RUNNING for 1010.7s
  RUNNING for 1042.2s
  RUNNING for 1072.5s
  RUNNING for 1104.0s
  RUNNING for 1134.7s
  RUNNING for 1166.1s
  RUNNING for 1197.1s
  RUNNING for 1228.7s
  RUNNING for 1259.2s
  RUNNING for 1290.4s
  RUNNING for 1321.6s
  RUNNING for 1352.7s
  RUNNING for 1383.7s
  RUNNING for 1414.9s
  RUNNING for 1445.4s
  RUNNING for 1476.3s
  RUNNING for 1507.1s
  RUNNING for 1537.8s
```

```
  COMPLETED
Attempting to fetch counters from logs...
Looking for step log in /mnt/var/log/hadoop/steps/s-1O9UEVM1FB654 on
ec2-54-84-230-119.compute-1.amazonaws.com...
  Parsing step log: ssh://ec2-54-84-230-119.compute-1.amazonaws.com/
mnt/var/log/hadoop/steps/s-1O9UEVM1FB654/syslog.2016-06-19-14
  Parsing step log: ssh://ec2-54-84-230-119.compute-1.amazonaws.com/
mnt/var/log/hadoop/steps/s-1O9UEVM1FB654/syslog
Counters: 55
        File Input Format Counters
                Bytes Read=2156069116
        File Output Format Counters
                Bytes Written=214733
        File System Counters
                FILE: Number of bytes read=297513
                FILE: Number of bytes written=23939371
                FILE: Number of large read operations=0
                FILE: Number of read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=23450
                HDFS: Number of bytes written=0
                HDFS: Number of large read operations=0
                HDFS: Number of read operations=190
                HDFS: Number of write operations=0
                S3: Number of bytes read=2156069116
                S3: Number of bytes written=214733
                S3: Number of large read operations=0
                S3: Number of read operations=0
                S3: Number of write operations=0
        Job Counters
                Data-local map tasks=188
                Launched map tasks=190
                Launched reduce tasks=28
                Other local map tasks=2
                Total megabyte-seconds taken by all map tasks=328332
29760
                Total megabyte-seconds taken by all reduce tasks=171
70038720
                Total time spent by all map tasks (ms)=22800854
                Total time spent by all maps in occupied slots (ms)=
1026038430
                Total time spent by all reduce tasks (ms)=5961819
                Total time spent by all reduces in occupied slots (m
s)=536563710
                Total vcore-seconds taken by all map tasks=22800854
                Total vcore-seconds taken by all reduce tasks=596181
9
        Map-Reduce Framework
                CPU time spent (ms)=12520850
                Combine input records=26439
                Combine output records=23092
                Failed Shuffles=0
                GC time elapsed (ms)=64209
                Input split bytes=23450
```

```
                Map input records=58682266
                Map output bytes=725208
                Map output materialized bytes=784186
                Map output records=26439
                Merged Map outputs=5320
                Physical memory (bytes) snapshot=114795364352
                Reduce input groups=997
                Reduce input records=23092
                Reduce output records=997
                Reduce shuffle bytes=784186
                Shuffled Maps =5320
                Spilled Records=46184
                Total committed heap usage (bytes)=131261267968
                Virtual memory (bytes) snapshot=464755920896
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
Removing s3 temp directory s3://mrjob-5a4b4386e2160458/tmp/stripes.j
amesgray.20160619.144912.668222/...
Removing temp directory /var/folders/ld/9wpyxfw13t7_pdv_0b8958x40000
gn/T/stripes.jamesgray.20160619.144912.668222...
Killing our SSH tunnel (pid 32828)
```

HW5.4 - Calculate Word Similarity for Unit Test Example

In [69]:
```
%%writefile simunit.txt
(DocA {X:20, Y:30, Z:5})
(DocB {X:100, Y:20})
(DocC {M:5, N:20, Z:5})
```

```
Writing simunit.txt
```

HW5.4 - Calculate Word Similarity Using Inverted Index and Cosine Simularity method

In [ ]:
```

```

HW5.4 - Calculate Word Similarity Using Inverted Index and Jaccard Simularity method

In [ ]:

HW 5.5 Evaluation of synonyms that your 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.

HW5.6 Optional

Repeat HW5 using vocabulary words ranked from 8001,-10,000; 7001,-10,000; 6001,-10,000; 5001,-10,000; 3001,-10,000; and 1001,-10,000;

Dont forget to report you Cluster configuration.

Generate the following graphs: -- vocabulary size (X-Axis) versus CPU time for indexing -- vocabulary size (X-Axis) versus number of pairs processed -- vocabulary size (X-Axis) versus F1 measure, Precision, Recall

HW 5.7 (optional)

Once again, benchmark your top 10,000 associations (as in 5.5), this time for your results from 5.6. Has your detector improved?

Type *Markdown* and LaTeX: α^2