Table of Contents

In [1]:
```
%%javascript
/*********************************************************************
*****************************
Known Mathjax Issue with Chrome - a rounding issue adds a border
to the right of mathjax markup
https://github.com/mathjax/MathJax/issues/1300
A quick hack to fix this based on stackoverflow discussions:
http://stackoverflow.com/questions/34277967/chrome-rendering-mat
hjax-equations-with-a-trailing-vertical-line
*********************************************************************
*****************************/

$('.math>span').css("border-left-color","transparent")
```

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

MIDS - w261 Machine Learning At Scale

Course Lead: Dr James G. Shanahan (**email** Jimi via James.Shanahan *AT* gmail.com)

Assignment - HW2

Name: Nilesh Bhoyar
Class: MIDS w261 ,Summer 2017 Group 2
Email: nilesh.bhoyar@iSchool.Berkeley.edu
StudentId 26302327 **End of StudentId**
Week: 2

NOTE: please replace `1234567` with your student id above
Due Time: HW is due the Tuesday of the following week by 8AM (West coast time)

Instructions

MIDS UC Berkeley, Machine Learning at Scale
DATSCIW261 ASSIGNMENT #2

Version 2017-16-5

IMPORTANT

This homework can be completed locally on your computer

=== INSTRUCTIONS for SUBMISSIONS ===

Follow the instructions for submissions carefully.

Each student has a `HW-<user>` repository for all assignments.

Push the following to your HW github repo into the master branch:

- Your local HW2 directory. Your repo file structure should look like this:

```
HW-<user>
    --HW3
       |__MIDS-W261-HW-03-<Student_id>.ipynb
       |__MIDS-W261-HW-03-<Student_id>.pdf
       |__some other hw3 file
    --HW4
       |__MIDS-W261-HW-04-<Student_id>.ipynb
       |__MIDS-W261-HW-04-<Student_id>.pdf
       |__some other hw4 file
    etc..
```

Useful References and Datasets

References

- See corresponding aysnc lecture and live session

Enron SPAM DATA SET

The dataset is a curated subset of the Enron email corpus. More details are given in the next section.

NOTE: This SPAM/HAM dataset for HW2 contains 100 records from the Enron SPAM/HAM corpus. Please limit your study to this unless otherwise instructed. There are about 93,000 emails in the original SPAM/HAM corpus. There are several versions of the SPAM/HAM corpus. Other Enron-Spam datasets are available from http://www.aueb.gr/users/ion/data/enron-spam/index.html (http://www.aueb.gr/users/ion/data/enron-spam/index.html) and http://www.aueb.gr/users/ion/publications.html (http://www.aueb.gr/users/ion/publications.html) in both raw and pre-processed form.

General information on the enronemail.txt data file

These data include email messages from 6 enron employees(in addition to various spam messages from a variety of sources) that were made publicly available after the company's collapse. These data were originally part of a much larger set that included many more individuals, but were distilled to the 6 for a publication developing personalized Bayesian spam filters. Please follow the links below for precise information regarding this data and research.

- Source data: http://www.aueb.gr/users/ion/data/enron-spam/ (http://www.aueb.gr/users/ion/data/enron-spam/)
- Source publication: http://www.aueb.gr/users/ion/docs/ceas2006_paper.pdf (http://www.aueb.gr/users/ion/docs/ceas2006_paper.pdf)

Processing

For their work, Metsis et al. (the authors) appeared to have pre-processed the data, not only collapsing all text to lower-case, but additionally separating "words" by spaces, where "words" unfortunately include punctuation. As a concrete example, the sentence:

```
"Hey Jon, I hope you don't get lost out there this weekend!"
```

would have been reduced by Metsis et al. to the form:

```
"hey jon , i hope you don ' t get lost out there this weekend ! "
```

Upon seeing this we have reverted the data back toward its original state, removing spaces so that our sample sentence would now look like:

```
"hey jon, i hope you don't get lost out there this weekend!"
```

so that we have at least preserved contractions and other higher-order lexical forms. However, one must be aware that this reversion is not complete, and that some object (specifically web sites) will be ill-formatted, and that all text is still lower-cased.

Format of the Enron SPAM data

All messages are collated to a tab-delimited format:

```
ID \t SPAM \t SUBJECT \t CONTENT \n
```

where:

```
ID = string; unique message identifier
SPAM = binary; with 1 indicating a spam message
SUBJECT = string; title of the message
CONTENT = string; content of the message
```

Note that either of SUBJECT or CONTENT may be "NA", and that all tab (\t) and newline (\n)

The ENRON SPAM dataset (has only 100 records)

Save the data in the next cell to file byt executing it.

In [3]:
```
!mkdir NaiveBayes
```

```
mkdir: cannot create directory `NaiveBayes': File exists
```

In [4]:
```
%%writefile NaiveBayes/enronemail_1h.txt
0001.1999-12-10.farmer  0        christmas tree farm pictures
NA
0001.1999-12-10.kaminski        0        re: rankings    thank y
ou.
0001.2000-01-17.beck    0        leadership development pilot
" sally:  what timing, ask and you shall receive. as per our dis
cussion, listed below  is an update on the leadership pilot. you
r vendor selection team will  receive an update and even more in
formation later in the week.  on the lunch & learn for energy op
erations, the audience and focus will be  your group. we are rea
dy to start up when appropriate.  thank you for your time today.
please call me if you have any questions at  x 33597.  ---------
-------------forwarded by julie armstrong/corp/enron on 01/17/20
00  06:44 pm--------------------------  from: susan runkel @ ec
t 01/17/2000 03:22 pm  to: cindy skinner/hou/ect @ ect, brad mcs
herry/hou/ect @ ect, norma  villarreal/hou/ect @ ect, kimberly r
izzi/hou/ect @ ect, fran l mayes/hou/ect @ ect,  gary buck/hou/e
ct @ ect, robert jones/corp/enron @ enron, sheila  walton/hou/ec
t @ ect, philip conn/corp/enron @ enron, mary overgaard/pdx/ect
@ ect,  kim melodick/hou/ect @ ect, valeria a hope/hou/ect @ ect
cc: david oxley/hou/ect @ ect, susan carrera/hou/ect @ ect, jane
allen/hou/ect @ ect, christine shenkman/enron_development @ enro
n_development,  kathryn mclean/hou/ect @ ect, gracie s presas/ho
u/ect @ ect, janice  riedel/hou/ect @ ect, julie armstrong/corp/
enron @ enron  subject: leadership development pilot  good news
regarding the ena leadership curriculum! through the help of a
vendor selection team from eops, we've chosen southwest performa
nce group and  wilson learning products as one of our primary ve
ndors for the leadership  curriculum and programs. we are ready
to conduct a pilot on february 8-10 of  six modules. the purpose
of the pilot is to evaluate for fine-tuning the  wilson learning
materials and facilitators and to present just a portion of  the
leadership curriculum.  in order to evaluate the materials thoro
ughly, it would be great to get a  cross-section of ena to atten
d. we are asking that you invite several  supervisors from your
client groups to participate in any of the courses  listed below
. the sessions will be held in room 560 and times are listed  be
low. also attached is a description of the modules. all are desi
gned for  supervisors only, with the exception being "" communic
ating effectively "". this  is open to any employee. as a benefi
t in attending the pilot, i will pick up  the cost., so there wi
ll be no charge back for their attendance.  we are currently com
pleting the curriculum design and will have information  on the
full curriculum available in february. this will include options
other than "" classrom setting "" for development.  please respo
nd back to gracie presas by february 1 with your names. if you
have further questions, please contact me at 3-7394. we are real
ly excited  that we have this available and hope that your clien
ts will find it to be  valuable.  the following are half-day ses
sions. supervisors may sign up for any or all  depending on thei
r need. it would be helpful if supervisors attend a minimum  of
two modules.  date module time target audience  feb. 8 meeting l
eadership challenges 8-12 am supervisors with less  than 6 month
s experience  working styles 1-5 pm any supervisor  feb. 9 coach
ing to performance 8-12 am any supervisor  motivating for result
s 1-5 pm any supervisor  feb. 10 communicating effectively 8-12
am any employee  delegating and directing 1-5 pm any supervisor"
```

```
Overwriting NaiveBayes/enronemail_1h.txt
```

Simple EDA of ENRON Dataset

```
wc -l enronemail_1h.txt  #100 email records
     100 enronemail_1h.txt
cut -f2 -d$'\t' enronemail_1h.txt|wc  #extract second field which is
SPAM flag
     101     394    3999
JAMES-SHANAHANs-Desktop-Pro-2:HW1-Questions jshanahan$ cut -f2 -d$'\
t' enronemail_1h.txt|head

0
0
0
0
0
0
0
0
1
1
> # Display an example SPAM email record
> head -n 100 enronemail_1h.txt|tail -1|less

018.2001-07-13.SA_and_HP       1        [ilug] we need your assistan
ce to invest in your country        dear sir/madam,  i am well confi
dent of your capability to assist me in  a transaction for mutual be
nefit of both parties, ie  (me and you) i am also believing that you
will not  expose or betray the trust and confidence i am about  to e
stablish with you. i have decided to contact you  with greatest deli
ght and personal respect.  well, i am victor sankoh, son to mr. foda
y  sankoh  who was arrested by the ecomog peace keeping force  month
s ago in my country sierra leone.
```

In [5]:
```
!wc -l NaiveBayes/enronemail_1h.txt
!cut -f2 -d$'\t' NaiveBayes/enronemail_1h.txt|wc
```

```
99 NaiveBayes/enronemail_1h.txt
    100     100     200
```

HW Problems

HW2.0 Functional Programming

HW2.0.0

- What is a race condition in the context of parallel computation? Give an example.
- What is MapReduce?
- How does it differ from Hadoop?

Answer: A race condition is an undesirable situation that occurs whe
n a device or system attempts to perform two or more operations at t
he same time, but because of the nature of the device or system, the
operations must be done in the proper sequence to be done correctly.
for example in Java instruction N= N- 1 though it looks atomic , it'
s not . It gets executed into two statements i.e. temp = N -1 and N
= temp. Result gets stored in temporary value and then it gets assig
ned to original memory.

Now lets say main program initiates two threas T1 and T2
Main ---> T1 and T2
 n = 5
T1 {
n = n -1
}

T2
{
n = n +1
}

Both thredas are working on same shared memory and executing at same
time. At machine level we have four instrctions to execute

 INT1 : temp1 = n - 1 # 5 - 1 = 4
 INT2: temp2 = n + 1 # 5+ 1 = 6
 INT3: n = temp1
 INT4: n =temp2

 here final output really depends on whether INT3 or INT4 gets exect
ued first. And this is race condition. It is undesirable and unpred
ctiable situation.

What is MapReduce?

Provides framework for developers to write scalable code.
MapReduce is programming model not a programming language.
Always processes records in key/value format.
Allows data distribution between nodes. Each program consists of Map
and Reduce.Its functional programming paradigm i.e. allows developer
s to focus on business logic.Map does filtering and sorting while Re
duce performes summary operations.
This also provides automatic parallelization of data processing.

How it is different from Hadoop?

Hadoop is an open source, Java-based programming framework that supp

HW2.0.1

Here is an example of functional programming in basic python in terms of mappers and reducers (by way of example):

In [6]:
```
#EXAMPLE Mapper functions in Python
def fahrenheit(T):
    return ((float(9)/5)*T + 32)

def celsius(T):
    return (float(5)/9)*(T-32)

temperatures = (36.5, 37, 37.5, 38, 39)
F = map(fahrenheit, temperatures)
#returns  97.7  98.6  99.5 100.4 102.2

C = map(celsius, F)

#EXAMPLE Reducer function in Python
import functools
#returns 113

print "Average temp is: %.2fF" % ( functools.reduce(lambda x,y:
x+y, F)/len(F) )
#returns Average temp is 99.68F
```

```
Average temp is: 99.68F
```

Which programming paradigm is Hadoop based on? Explain and give a simple example of functional programming in raw python code and show the code running. E.g., in raw python find the average length of a string in collection of strings using a python "map-reduce" (functional programming) job (similar in style to the above). Alternatively, you can do this in python Hadoop Streaming.

```
strings = ["str1", "string2", "w261", "MAchine learning at SCALE"]
.......


import functools as reduce
temperatures = (36.5, 37, 37.5, 38, 39)
F = map(fahrenheit, temperatures)
print "Average temp is %fF" % (reduce(lambda x,y: x+y, F)/len(F) )
#returns Average temp is 99.68F



map(sqr, items)
```

In [7]:
```
strings = ["str1", "string2", "w261", "MAchine learning at SCALE
"]

print "Average length of string  is %fF" %(  reduce(lambda x, y
: x + y, list(map(lambda x: len(x), strings))) / len(strings))
```

```
Average length of string  is 10.000000F
```

Set up your directories on your local (VM) machine and on HDFS

In [8]:
```
!mkdir WordCount
```

```
mkdir: cannot create directory `WordCount': File exists
```

WordCount: A full example in Hadoop Stream to practice with

In [9]:
```
%%writefile WordCount/mapper.py
#!/usr/bin/env python

import sys
#sys.stderr.write("reporter:counter:Tokens,Total,1") # NOTE miss
ing the carriage return so wont work
# Set up counters to monitor/understand the number of times a ma
pper task is run
sys.stderr.write("reporter:counter:HW2.0.1 Mapper Counters,Calls
,1\n")
sys.stderr.write("reporter:status:processing my message...how ar
e  you\n")

for line in sys.stdin:
    for word in line.split():
        print '%s\t%s' % (word, 1)
```

```
Overwriting WordCount/mapper.py
```

In [10]:
```
%%writefile WordCount/reducer.py
#!/usr/bin/env python

import sys

cur_key = None
cur_count = 0
# Set up counters to monitor/understand the number of times a re
ducer task is run
sys.stderr.write("reporter:counter:HW2.0.1 Reducer Counters,Call
s,1\n")
for line in sys.stdin:
    key, value = line.split()
    if key == cur_key:
        cur_count += int(value)
    else:
        if cur_key:
            print '%s\t%s' % (cur_key, cur_count)
        cur_key = key
        cur_count = int(value)

print '%s\t%s' % (cur_key, cur_count)
```

```
Overwriting WordCount/reducer.py
```

In [11]:
```
!chmod a+x WordCount/mapper.py
!chmod a+x WordCount/reducer.py
```

In [12]:
```
#Unit test the mapper
!echo "foo foo quux labs foo bar quux" | WordCount/mapper.py
```

```
reporter:counter:HW2.0.1 Mapper Counters,Calls,1
reporter:status:processing my message...how are  you
foo     1
foo     1
quux    1
labs    1
foo     1
bar     1
quux    1
```

In [13]:
```
#Unit test the mapper
!echo "foo foo quux labs foo bar quux" | WordCount/mapper.py |so
rt -k1,1
```

```
reporter:counter:HW2.0.1 Mapper Counters,Calls,1
reporter:status:processing my message...how are  you
bar     1
foo     1
foo     1
foo     1
labs    1
quux    1
quux    1
```

In [14]:
```
#Systems test the mapper and reducer
!echo "foo foo quux labs foo bar quux" | WordCount/mapper.py | s
ort -k1,1 | WordCount/reducer.py| sort -k2,2nr
```

```
reporter:counter:HW2.0.1 Reducer Counters,Calls,1
reporter:counter:HW2.0.1 Mapper Counters,Calls,1
reporter:status:processing my message...how are  you
foo     3
quux    2
bar     1
labs    1
```

In [15]:
```
%%writefile testWordCountInput.txt
hello this is Jimi
jimi who Jimi three Jimi
Hello
hello
```

```
Overwriting testWordCountInput.txt
```

In [16]:
```
!hdfs dfs -rm testWordCountInput.txt
!hdfs dfs -copyFromLocal testWordCountInput.txt
!hdfs dfs -rm -r wordcount-output

##################### IMPORTANT ##################################
########################################
# Make sure you have the correct paths to the jar file as wel as
the input and output files!!
# make sure to include the -files option. Do ***** NOT ****** pu
t spaces between the file paths!
##################################################################
########################################

!hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
  -files WordCount/reducer.py,WordCount/mapper.py \
  -mapper mapper.py \
  -reducer reducer.py \
  -input testWordCountInput.txt \
  -output wordcount-output  \
  -numReduceTasks 3
```

```
Deleted testWordCountInput.txt
Deleted wordcount-output
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.ja
r] /tmp/streamjob5696563709594859662.jar tmpDir=null
17/05/23 02:18:35 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:18:35 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:18:36 INFO mapred.FileInputFormat: Total input path
s to process : 1
17/05/23 02:18:37 INFO mapreduce.JobSubmitter: number of splits
:2
17/05/23 02:18:37 INFO mapreduce.JobSubmitter: Submitting token
s for job: job_1495464826300_0088
17/05/23 02:18:37 INFO impl.YarnClientImpl: Submitted applicati
on application_1495464826300_0088
17/05/23 02:18:38 INFO mapreduce.Job: The url to track the job:
http://quickstart.cloudera:8088/proxy/application_1495464826300
_0088/
17/05/23 02:18:38 INFO mapreduce.Job: Running job: job_14954648
26300_0088
17/05/23 02:18:50 INFO mapreduce.Job: Job job_1495464826300_008
8 running in uber mode : false
17/05/23 02:18:50 INFO mapreduce.Job:  map 0% reduce 0%
17/05/23 02:19:02 INFO mapreduce.Job:  map 50% reduce 0%
17/05/23 02:19:03 INFO mapreduce.Job:  map 100% reduce 0%
17/05/23 02:19:16 INFO mapreduce.Job:  map 100% reduce 33%
17/05/23 02:19:18 INFO mapreduce.Job:  map 100% reduce 100%
17/05/23 02:19:19 INFO mapreduce.Job: Job job_1495464826300_008
8 completed successfully
17/05/23 02:19:19 INFO mapreduce.Job: Counters: 51
        File System Counters
                FILE: Number of bytes read=118
                FILE: Number of bytes written=585710
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=316
                HDFS: Number of bytes written=56
                HDFS: Number of read operations=15
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=6
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=3
                Data-local map tasks=2
                Total time spent by all maps in occupied slots
(ms)=18187
                Total time spent by all reduces in occupied slo
ts (ms)=34266
                Total time spent by all map tasks (ms)=18187
                Total time spent by all reduce tasks (ms)=34266
                Total vcore-seconds taken by all map tasks=1818
7
                Total vcore-seconds taken by all reduce tasks=3
4266
                Total megabyte-seconds taken by all map tasks=1
8623488
                Total megabyte-seconds taken by all reduce task
```

In [17]:
```
#have a look at the input
!echo  "\n--------------------------\n"
!hdfs dfs -cat testWordCountInput.txt
!echo  "\n--------------------------\n"
# Wordcount output
!hdfs dfs -cat wordcount-output/part-0000*
```

```
\n--------------------------\n
hello this is Jimi
jimi who Jimi three Jimi
Hello
hello\n--------------------------\n
Hello   1
jimi    1
this    1
three   1
Jimi    3
hello   2
is      1
who     1
```

HW2.1. Sort in Hadoop MapReduce (Partial sort, total sort) - List in alphabetical order

In this problem use Alice's Adventures in Wonderland. (You can obtain a free plain text version of the book, along with many others, from here (http://www.gutenberg.org.)

Using Hadoop, please change the mapper.py/reducer.py combination so that you get only the number of words starting with an uppercase letter, and the number of words starting with a lowercase letter. In other words, you need an output file with only 2 lines, one giving you the number of words staring with a lowercase ('a' to 'z'), and the other line indicating the number of words starting with an uppercase letter ('A' to 'Z').

Some background on Sorting in Hadoop Hadoop will always give a total sort on the key (i.e., key part of the key-value pairs produced by the mappers) when using just one reducer. When using multiple reducers Hadoop will by default give you a partial sort (i.e., all records within a partition will be sorted by the key (i.e., key part of the key-value pairs produced by the mappers) . To achieve a total sort one needs to write a custom mapper to to prepend a partition key to each record, partition on that prepended key, and then do a secondary sort on a composite key that is made up of the prepended key and the original key. This can be done with one map-reduce job. This will be covered during Live Session of Week 3.

In [18]:
```
!curl 'http://www.gutenberg.org/files/11/11-0.txt' -o alicesTExt
Filename.txt
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time
Time  Current
                                 Dload  Upload   Total   Spent
Left  Speed
100  169k  100  169k    0     0   295k      0 --:--:-- --:--:--
--:--:--  333k
```

In [19]:
```
#display the first few lines
!head alicesTExtFilename.txt
```

In [20]:
```
!mkdir UpperLower
```

```
mkdir: cannot create directory `UpperLower': File exists
```

In [21]:
```
%%writefile UpperLower/mapper.py
#!/usr/bin/env python

import sys
#sys.stderr.write("reporter:counter:Tokens,Total,1") # NOTE miss
ing the carriage return so wont work
# Set up counters to monitor/understand the number of times a ma
pper task is run
sys.stderr.write("reporter:counter:HW2.1 Mapper Counters,Calls,1
\n")
sys.stderr.write("reporter:status:processing my message...how ar
e  you\n")

# START STUDENT CODE HW21MAPPER

import re
from collections import defaultdict
 #for each document create dictionary of words

# START STUDENT CODE HW13MAPPER
for line in sys.stdin:


    #Upper case letters
    for word in re.findall(r'\b[A-Z][a-z]*\b',line):

        print '%s\t%s' % ('upper', 1)
    #lower case letters
    for word in re.findall(r'\b[a-z][a-z]*\b',line):

        print '%s\t%s' % ('lower', 1)

# END STUDENT CODE HW21MAPPER
```

```
Overwriting UpperLower/mapper.py
```

In [22]:
```
%%writefile UpperLower/reducer.py
#!/usr/bin/env python

import sys
# Set up counters to monitor/understand the number of times a re
ducer task is run
sys.stderr.write("reporter:counter:HW2.1 Reducer Counters,Calls,
1\n")


# START STUDENT CODE HW21REDUCER



cur_key = None
cur_count = 0

for line in sys.stdin:
    key, value = line.split()
    if key == cur_key:
        cur_count += int(value)
    else:
        if cur_key:
            print '%s\t%s' % (cur_key, cur_count)
        cur_key = key
        cur_count = int(value)

print '%s\t%s' % (cur_key, cur_count)


# END STUDENT CODE HW21REDUCER
```

```
Overwriting UpperLower/reducer.py
```

In [23]:
```
# INSTRUCTIONS: make mapper and reducer py files executable
# START STUDENT CODE HW21EXECUTABLE

!chmod a+x UpperLower/mapper.py
!chmod a+x UpperLower/reducer.py

# END STUDENT CODE HW21EXECUTABLE
```

In [24]:
```
#Systems test the mapper and reducer
!echo "foo Foo Quux labs foo bar quux" |UpperLower/mapper.py | s
ort -k1,1 | UpperLower/reducer.py| sort -k2,2nr
```

```
reporter:counter:HW2.1 Reducer Counters,Calls,1
reporter:counter:HW2.1 Mapper Counters,Calls,1
reporter:status:processing my message...how are  you
lower   5
upper   2
```

In [25]:
```
# INSTRUCTIONS: call hadoop with one reducer. see example above.

##################### IMPORTANT ########################
# Make sure you have the correct paths to the jar file
# as wel as the input and output files!!
# make sure to include the -files option. Do NOT put
# spaces between the file paths!
########################################################



# START STUDENT CODE HW21HADOOP

!hdfs dfs -rm alicesTExtFilename.txt
!hdfs dfs -copyFromLocal alicesTExtFilename.txt
!hdfs dfs -rm -r upperlower-output

##################### IMPORTANT ##################################
########################################
# Make sure you have the correct paths to the jar file as wel as
the input and output files!!
# make sure to include the -files option. Do ***** NOT ****** pu
t spaces between the file paths!
##################################################################
########################################

!hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
  -files UpperLower/reducer.py,UpperLower/mapper.py\
  -mapper mapper.py \
  -reducer reducer.py \
  -input alicesTExtFilename.txt \
  -output upperlower-output  \
  -numReduceTasks 1


# END STUDENT CODE HW21HADOOP
```

```
Deleted alicesTExtFilename.txt
Deleted upperlower-output
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.ja
r] /tmp/streamjob3573564961503683392.jar tmpDir=null
17/05/23 02:19:47 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:19:48 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:19:49 INFO mapred.FileInputFormat: Total input path
s to process : 1
17/05/23 02:19:49 INFO mapreduce.JobSubmitter: number of splits
:2
17/05/23 02:19:49 INFO mapreduce.JobSubmitter: Submitting token
s for job: job_1495464826300_0089
17/05/23 02:19:50 INFO impl.YarnClientImpl: Submitted applicati
on application_1495464826300_0089
17/05/23 02:19:50 INFO mapreduce.Job: The url to track the job:
http://quickstart.cloudera:8088/proxy/application_1495464826300
_0089/
17/05/23 02:19:50 INFO mapreduce.Job: Running job: job_14954648
26300_0089
17/05/23 02:19:58 INFO mapreduce.Job: Job job_1495464826300_008
9 running in uber mode : false
17/05/23 02:19:58 INFO mapreduce.Job:  map 0% reduce 0%
17/05/23 02:20:07 INFO mapreduce.Job:  map 50% reduce 0%
17/05/23 02:20:08 INFO mapreduce.Job:  map 100% reduce 0%
17/05/23 02:20:14 INFO mapreduce.Job:  map 100% reduce 100%
17/05/23 02:20:15 INFO mapreduce.Job: Job job_1495464826300_008
9 completed successfully
17/05/23 02:20:15 INFO mapreduce.Job: Counters: 51
        File System Counters
                FILE: Number of bytes read=299416
                FILE: Number of bytes written=950107
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=177923
                HDFS: Number of bytes written=23
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots
(ms)=13512
                Total time spent by all reduces in occupied slo
ts (ms)=5026
                Total time spent by all map tasks (ms)=13512
                Total time spent by all reduce tasks (ms)=5026
                Total vcore-seconds taken by all map tasks=1351
2
                Total vcore-seconds taken by all reduce tasks=5
026
                Total megabyte-seconds taken by all map tasks=1
3836288
                Total megabyte-seconds taken by all reduce task
s=5146624
```

In [26]:
```
# Wordcount output
!hdfs dfs -cat upperlower-output/part-0000* > upperlower_counts.
txt
```

In [27]:
```
!cat upperlower_counts.txt
```

```
lower   26181
upper   3760
```

HW2.1.1 Calculate the vocabulary size (number of unique words in the Alice book)

To solve this problem, a single reducer will suffice. One could use multiple reducers but then you would need a post processing step to aggregate the counts in the PART-000XX files.

Write a map/reduce job to count the number of unique words in the Alice book.
Please verify your code with straight python code.
Do you get the same answer?

In [28]:
```
!mkdir Vocab
```

```
mkdir: cannot create directory `Vocab': File exists
```

In [29]:
```
%%writefile Vocab/mapper.py
#!/usr/bin/env python
# START STUDENT CODE HW211MAPPER
import sys
import re

for line in sys.stdin:
  line = line.strip()
  for word in re.findall(r'[a-z]+', line.lower()):
    print word,"\t",1


# END STUDENT CODE HW211MAPPER
```

```
Overwriting Vocab/mapper.py
```

In [30]:
```
%%writefile Vocab/reducer.py
#!/usr/bin/env python
# START STUDENT CODE HW211REDUCER
import sys
import re
cur_key = None
total_count = 0

for line in sys.stdin:
    key, value = line.split()
    if key!= cur_key:
        cur_key = key
        total_count = total_count + 1


print '%s\t%s' % ('vocab_size',  total_count)


# END STUDENT CODE HW211REDUCER
```

```
Overwriting Vocab/reducer.py
```

In [31]:
```
!chmod a+x Vocab/mapper.py
!chmod a+x Vocab/reducer.py
#Systems test the mapper and reducer
!echo "foo Foo quux labs foo bar quux" |Vocab/mapper.py| sort -k
1,1 | Vocab/reducer.py| sort -k2,2nr
```

```
vocab_size      4
```

In [32]:
```
# START STUDENT CODE HW211HADOOP


!hdfs dfs -rm alicesTExtFilename.txt
!hdfs dfs -copyFromLocal alicesTExtFilename.txt
!hdfs dfs -rm -r vocab-output

###################### IMPORTANT ##################################
#########################################
# Make sure you have the correct paths to the jar file as wel as
the input and output files!!
# make sure to include the -files option. Do ***** NOT ****** pu
t spaces between the file paths!
#########################################################################
#########################################

!hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
  -files Vocab/reducer.py,Vocab/mapper.py\
  -mapper mapper.py \
  -reducer reducer.py \
  -input alicesTExtFilename.txt \
  -output vocab-output  \
  -numReduceTasks 1

# END STUDENT CODE HW211HADOOP
```

```
Deleted alicesTExtFilename.txt
Deleted vocab-output
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.ja
r] /tmp/streamjob1025936747395085188.jar tmpDir=null
17/05/23 02:20:35 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:20:36 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:20:37 INFO mapred.FileInputFormat: Total input path
s to process : 1
17/05/23 02:20:37 INFO mapreduce.JobSubmitter: number of splits
:2
17/05/23 02:20:38 INFO mapreduce.JobSubmitter: Submitting token
s for job: job_1495464826300_0090
17/05/23 02:20:38 INFO impl.YarnClientImpl: Submitted applicati
on application_1495464826300_0090
17/05/23 02:20:38 INFO mapreduce.Job: The url to track the job:
http://quickstart.cloudera:8088/proxy/application_1495464826300
_0090/
17/05/23 02:20:38 INFO mapreduce.Job: Running job: job_14954648
26300_0090
17/05/23 02:20:50 INFO mapreduce.Job: Job job_1495464826300_009
0 running in uber mode : false
17/05/23 02:20:50 INFO mapreduce.Job:  map 0% reduce 0%
17/05/23 02:21:00 INFO mapreduce.Job:  map 100% reduce 0%
17/05/23 02:21:08 INFO mapreduce.Job:  map 100% reduce 100%
17/05/23 02:21:09 INFO mapreduce.Job: Job job_1495464826300_009
0 completed successfully
17/05/23 02:21:09 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=305555
                FILE: Number of bytes written=962340
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=177923
                HDFS: Number of bytes written=16
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots
(ms)=14453
                Total time spent by all reduces in occupied slo
ts (ms)=5733
                Total time spent by all map tasks (ms)=14453
                Total time spent by all reduce tasks (ms)=5733
                Total vcore-seconds taken by all map tasks=1445
3
                Total vcore-seconds taken by all reduce tasks=5
733
                Total megabyte-seconds taken by all map tasks=1
4799872
                Total megabyte-seconds taken by all reduce task
s=5870592
        Map-Reduce Framework
```

In [33]:
```
!hdfs dfs -cat vocab-output/part-0000* > vocab_output.txt
```

In [34]:
```
!cat vocab_output.txt
```

```
vocab_size      3009
```

In [ ]:

In [ ]:

HW2.1.2 TOTAL SORT using a single reducer

Write a MapReduce job that creates a text file named **alice_words.txt** containing an alphabetical listing of all the words, and the number of times each occurs, in the text version of Alice's Adventures in Wonderland. (You can obtain a free plain text version of the book, along with many others, from here (http://www.gutenberg.org/cache/epub/11/pg11.txt)

Solve this TOTAL SORT problem in mapReduce using a single reducer.

The first 10 lines of your output file should look something like this (the counts are not totally precise):

```
Word              Count
======================
a                 631
a-piece           1
abide             1
able              1
about             94
above             3
absence           1
absurd            2
```

In [35]:
```
!mkdir Total_sort
```

```
mkdir: cannot create directory `Total_sort': File exists
```

In [36]:
```
%%writefile Total_sort/mapper.py
#!/usr/bin/env python
# START STUDENT CODE HW212MAPPER
import sys
import re

for line in sys.stdin:
  line = line.strip()
  for word in re.findall(r'[a-z]+', line.lower()):
    print word,"\t",1


# END STUDENT CODE HW212MAPPER
```

```
Overwriting Total_sort/mapper.py
```

In [37]:
```
%%writefile Total_sort/reducer.py
#!/usr/bin/env python
# START STUDENT CODE HW212REDUCER
import sys
import re
cur_key = None
cur_count = 0

for line in sys.stdin:
    key, value = line.split()
    if key == cur_key:
        cur_count += int(value)
    else:
        if cur_key:
            print '%s\t%s' % (cur_key, cur_count)
        cur_key = key
        cur_count = int(value)

print '%s\t%s' % (cur_key, cur_count)

# END STUDENT CODE HW212REDUCER
```

```
Overwriting Total_sort/reducer.py
```

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

In [39]:
```
#Systems test the mapper and reducer
!echo "foo Foo quux labs foo bar quux" |Total_sort/mapper.py| so
rt -k1,1 | Total_sort/reducer.py| sort -k2,2nr
```

```
foo     3
quux    2
bar     1
labs    1
```

In [40]:
```
# START STUDENT CODE HW212HADOOP
!hdfs dfs -rm alicesTExtFilename.txt
!hdfs dfs -copyFromLocal alicesTExtFilename.txt
!hdfs dfs -rm -r sorted-output

##################### IMPORTANT ##############################
########################################
# Make sure you have the correct paths to the jar file as wel as
the input and output files!!
# make sure to include the -files option. Do ***** NOT ****** pu
t spaces between the file paths!
##############################################################
########################################

!hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
  -D mapreduce.job.output.key.comparator.class=org.apache.hadoop
.mapred.lib.KeyFieldBasedComparator \
  -D mapreduce.partition.keycomparator.options="-k1,1"  \
  -files Total_sort/reducer.py,Total_sort/mapper.py\
  -mapper mapper.py \
  -reducer reducer.py \
  -input alicesTExtFilename.txt \
  -output sorted-output \
  -numReduceTasks 1

# END STUDENT CODE HW212HADOOP
```

```
Deleted alicesTExtFilename.txt
Deleted sorted-output
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.ja
r] /tmp/streamjob2414305510994011639.jar tmpDir=null
17/05/23 02:21:29 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:21:30 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:21:31 INFO mapred.FileInputFormat: Total input path
s to process : 1
17/05/23 02:21:31 INFO mapreduce.JobSubmitter: number of splits
:2
17/05/23 02:21:31 INFO mapreduce.JobSubmitter: Submitting token
s for job: job_1495464826300_0091
17/05/23 02:21:31 INFO impl.YarnClientImpl: Submitted applicati
on application_1495464826300_0091
17/05/23 02:21:31 INFO mapreduce.Job: The url to track the job:
http://quickstart.cloudera:8088/proxy/application_1495464826300
_0091/
17/05/23 02:21:31 INFO mapreduce.Job: Running job: job_14954648
26300_0091
17/05/23 02:21:42 INFO mapreduce.Job: Job job_1495464826300_009
1 running in uber mode : false
17/05/23 02:21:42 INFO mapreduce.Job:  map 0% reduce 0%
17/05/23 02:21:54 INFO mapreduce.Job:  map 50% reduce 0%
17/05/23 02:21:55 INFO mapreduce.Job:  map 100% reduce 0%
17/05/23 02:22:02 INFO mapreduce.Job:  map 100% reduce 100%
17/05/23 02:22:04 INFO mapreduce.Job: Job job_1495464826300_009
1 completed successfully
17/05/23 02:22:04 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=305555
                FILE: Number of bytes written=963450
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=177923
                HDFS: Number of bytes written=28506
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots
(ms)=19942
                Total time spent by all reduces in occupied slo
ts (ms)=6896
                Total time spent by all map tasks (ms)=19942
                Total time spent by all reduce tasks (ms)=6896
                Total vcore-seconds taken by all map tasks=1994
2
                Total vcore-seconds taken by all reduce tasks=6
896
                Total megabyte-seconds taken by all map tasks=2
0420608
                Total megabyte-seconds taken by all reduce task
s=7061504
```

In [41]:
```
!hdfs dfs -cat sorted-output/part-0000* > sorted_output.txt
```

In [42]:
```
!head sorted_output.txt
```

```
a       690
abide   2
able    1
about   102
above   3
absence 1
absurd  2
accept  1
acceptance      1
accepted        2
```

HW2.1.2.b TOTAL SORT using multiple reducers [OPTITIONAL for this week; will be covered in next live session]

Change the mapper.py/reducer.py combination from the the above WordCount example so that you get the longest word present in the text version of Alice's Adventures in Wonderland. (You can obtain a free plain text version of the book, along with many others, from here (http://www.gutenberg.org/cache/epub/11/pg11.txt).

- First use one reducer and report your result. HINT: from emit records of the form: "longestWord\theLongWordEver\t15".
- Run you Hadoop streaming job with 3 reducers? Anything change with respect to your solution.

In [43]:
```
!mkdir Total_sort_multi
```

```
mkdir: cannot create directory `Total_sort_multi': File exists
```

In [44]:
```
%%writefile Total_sort_multi/mapper.py
# START STUDENT CODE HW212MAPPER_MULTI
from __future__ import division
import math
import os
import sys
import re


count = 0
numReducers = int(os.environ.get('NUM_PARTITIONS', '4')) # defau
lt to 4

################################### PARTITION THE DATA INTO N
BUCKETS ###################################
# lowercase chars range from 97 -> 122.
# there are 26 chars. So let's partion by spliting at 26/n
# ord('a') -> 97
# chr(97) -> 'a'
# ord(word[0])-96  --> get the number of the first letter betwee
n 1 and 26, such that a -> 1; z -> 26
################################################################
##########################################

def makeKey(word,n):
  divisor = 26/n
  return int(math.ceil((ord(word[0])-96)/divisor))


for line in sys.stdin:
  line = line.strip()
  for word in re.findall(r'[a-z]+', line.lower()):
    # prepend a key based on the number of reducers
    key = makeKey(word,numReducers)
    print key,"\t",word,"\t",1

# END STUDENT CODE HW212MAPPER_MULTI
```

```
Overwriting Total_sort_multi/mapper.py
```

In [45]:
```
%%writefile  Total_sort_multi/reducer.py
# START STUDENT CODE HW212REDUCER_MULTI

import sys
import re

current_word = None
current_count = 0
word = None

for line in sys.stdin:

    key, word, count = line.split('\t')
    count = int(count)

    if current_word == word:
        current_count += count
    else:
        if current_word:
            # write result to STDOUT
            print '%s\t%s\t%s' % (key,current_word, current_coun
t)
        current_count = count
        current_word = word

# do not forget to output the last word if needed!
if current_word == word:
    print '%s\t%s\t%s' % (key,current_word, current_count)

# END STUDENT CODE HW212REDUCER_MULTI
```

```
Overwriting Total_sort_multi/reducer.py
```

In [46]:
```
!chmod a+x Total_sort_multi/mapper.py
!chmod a+x Total_sort_multi/reducer.py
```

In [47]:
```
!echo "hi hello sir Apple"|python Total_sort_multi/mapper.py| so
rt -k1,1 | python Total_sort_multi/reducer.py
```

```
2       apple   1
2       hello   1
3       hi      1
3       sir     1
```

HW2.1.3 How many times does the word alice occur in the book?

Write a MapReduce job to determine this. Please pay attention to what you use for a key and value as output from your mapper.

START STUDENT CODE HW212HADOOP_MULTI

!hdfs dfs -rm alicesTExtFilename.txt !hdfs dfs -copyFromLocal alicesTExtFilename.txt !hdfs dfs -rm -r sorted-multi

############### *IMPORTANT*

Make sure you have the correct paths to the jar file as wel as the input and output files!!

make sure to include the -files option. Do * NOT ** put spaces between the file paths!

#

!hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \ -D stream.num.map.output.key.fields=3 \ -D stream.map.output.field.separator="\t" \ -D mapreduce.partition.keypartitioner.options=-k1,1 \ -D mapreduce.job.output.key.comparator.class=org.apache.hadoop.mapred.lib.KeyFieldBasedComparatc \ -D mapreduce.partition.keycomparator.options="-k1,1 -k2,2 -k3,3" \ -files Total_sort_multi\ -mapper "python Total_sort_multi/mapper.py" \ -reducer "python Total_sort_multi/reducer.py" \ -input alicesTExtFilename.txt \ -output sorted-multi \ -numReduceTasks 1 \ -partitioner org.apache.hadoop.mapred.lib.KeyFieldBasedPartitioner \ -cmdenv NUM_PARTITIONS=4

END STUDENT CODE HW212HADOOP_MULTI

In [48]:
```
!mkdir Alice
```

```
mkdir: cannot create directory `Alice': File exists
```

In [49]:
```
%%writefile Alice/mapper.py
#!/usr/bin/env python
# START STUDENT CODE HW213MAPPER

import sys
import re

for line in sys.stdin:
  line = line.strip()
  for word in re.findall(r'[a-z]+', line.lower()):
    if word  == 'alice':
        print word,"\t",1

# END STUDENT CODE HW213MAPPER
```

```
Overwriting Alice/mapper.py
```

In [50]:
```
%%writefile Alice/reducer.py
#!/usr/bin/env python
# START STUDENT CODE HW213REDUCER
import sys
import re
cur_key = None
cur_count = 0

for line in sys.stdin:
    key, value = line.split()
    if key == cur_key:
        cur_count += int(value)
    else:
        if cur_key:
            print '%s\t%s' % (cur_key, cur_count)
        cur_key = key
        cur_count = int(value)

print '%s\t%s' % (cur_key, cur_count)

# END STUDENT CODE HW213REDUCER
```

```
Overwriting Alice/reducer.py
```

In [51]:
```
!chmod a+x Alice/mapper.py
!chmod a+x Alice/reducer.py
```

In [52]:
```
# START STUDENT CODE HW213HADOOP
!hdfs dfs -rm alicesTExtFilename.txt
!hdfs dfs -copyFromLocal alicesTExtFilename.txt
!hdfs dfs -rm -r alice-output

##################### IMPORTANT ##################################
########################################
# Make sure you have the correct paths to the jar file as wel as
the input and output files!!
# make sure to include the -files option. Do ***** NOT ****** pu
t spaces between the file paths!
################################################################
########################################

!hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
  -D mapreduce.job.output.key.comparator.class=org.apache.hadoop
.mapred.lib.KeyFieldBasedComparator \
  -D mapreduce.partition.keycomparator.options="-k1,1"  \
  -files Alice/reducer.py,Alice/mapper.py\
  -mapper mapper.py \
  -reducer reducer.py \
  -input alicesTExtFilename.txt \
  -output alice-output\
  -numReduceTasks 1

# END STUDENT CODE HW213HADOOP
```

```
Deleted alicesTExtFilename.txt
Deleted alice-output
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.ja
r] /tmp/streamjob2661230592418033778.jar tmpDir=null
17/05/23 02:22:28 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:22:29 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:22:30 INFO mapred.FileInputFormat: Total input path
s to process : 1
17/05/23 02:22:31 INFO mapreduce.JobSubmitter: number of splits
:2
17/05/23 02:22:31 INFO mapreduce.JobSubmitter: Submitting token
s for job: job_1495464826300_0092
17/05/23 02:22:31 INFO impl.YarnClientImpl: Submitted applicati
on application_1495464826300_0092
17/05/23 02:22:31 INFO mapreduce.Job: The url to track the job:
http://quickstart.cloudera:8088/proxy/application_1495464826300
_0092/
17/05/23 02:22:31 INFO mapreduce.Job: Running job: job_14954648
26300_0092
17/05/23 02:22:42 INFO mapreduce.Job: Job job_1495464826300_009
2 running in uber mode : false
17/05/23 02:22:42 INFO mapreduce.Job:  map 0% reduce 0%
17/05/23 02:22:52 INFO mapreduce.Job:  map 50% reduce 0%
17/05/23 02:22:53 INFO mapreduce.Job:  map 100% reduce 0%
17/05/23 02:23:01 INFO mapreduce.Job:  map 100% reduce 100%
17/05/23 02:23:02 INFO mapreduce.Job: Job job_1495464826300_009
2 completed successfully
17/05/23 02:23:03 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=4439
                FILE: Number of bytes written=361185
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=177923
                HDFS: Number of bytes written=10
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots
(ms)=17345
                Total time spent by all reduces in occupied slo
ts (ms)=6173
                Total time spent by all map tasks (ms)=17345
                Total time spent by all reduce tasks (ms)=6173
                Total vcore-seconds taken by all map tasks=1734
5
                Total vcore-seconds taken by all reduce tasks=6
173
                Total megabyte-seconds taken by all map tasks=1
7761280
                Total megabyte-seconds taken by all reduce task
s=6321152
```

In [53]:
```
!hdfs dfs -cat alice-output/part-0000* > alice_count.txt
```

In [54]:
```
!cat alice_count.txt
```

```
alice   403
```

HW2.2 EDA using WORDCOUNT in Hadoop - Top 10 Words

Example of word splitting

A tokenizer divides text into a sequence of tokens, which roughly correspond to "words". We provide a some code here for tokenization of English language strings. For detecting words in HW2.2 and later problems please use the following code to generate word tokens (otherwise, you will probably get a different answer to expected).

In [55]:
```
import sys, re, string
# define regex for punctuation removal


line = """ 0017.2000-01-17.beck 0        global risk management
operations      " congratulations, sally!!!  kk  ---------------
-------forwarded by kathy kokas/corp/enron on 01/17/2000  08:08
pm---------------------------  from: rick causey 01/17/2000 06:0
4 pm  sent by: enron announcements  to: all enron worldwide  cc:
subject: global risk management operations  recognizing enron ,
s increasing worldwide presence in the wholesale energy  busines
s and the need to insure outstanding internal controls for all o
f our  risk management activities, regardless of location, a glo
bal risk management  operations function has been created under
the direction of sally w. beck,  vice president. in this role, s
ally will report to rick causey, executive  vice president and c
hief accounting officer.  sally , s responsibilities with regard
to global risk management operations  will mirror those of other
recently created enron global functions. in this  role, sally wi
ll work closely with all enron geographic regions and wholesale
companies to insure that each entity receives individualized reg
ional support  while also focusing on the following global respo
nsibilities:  1. enhance communication among risk management ope
rations professionals.  2. assure the proliferation of best oper
ational practices around the globe.  3. facilitate the allocatio
n of human resources.  4. provide training for risk management o
perations personnel.  5. coordinate user requirements for shared
operational systems.  6. oversee the creation of a global intern
al control audit plan for risk  management activities.  7. estab
lish procedures for opening new risk management operations offic
es  and create key benchmarks for measuring on-going risk contro
ls.  each regional operations team will continue its direct repo
rting relationship  within its business unit, and will collabora
te with sally in the delivery of  these critical items. the hous
ton-based risk management operations team under  sue frusco , s
leadership, which currently supports risk management activities
for south america and australia, will also report directly to sa
lly.  sally retains her role as vice president of energy operati
ons for enron  north america, reporting to the ena office of the
chairman. she has been in  her current role over energy operatio
ns since 1997, where she manages risk  consolidation and reporti
ng, risk management administration, physical product  delivery,
confirmations and cash management for ena , s physical commodity
trading, energy derivatives trading and financial products tradi
ng.  sally has been with enron since 1992, when she joined the c
ompany as a  manager in global credit. prior to joining enron, s
ally had four years  experience as a commercial banker and spent
seven years as a registered  securities principal with a regiona
l investment banking firm. she also owned  and managed a retail
business for several years.  please join me in supporting sally
in this additional coordination role for  global risk management
operations."
"""
docID, docClass,title,body = line.split("\t",3)
regex = re.compile('[%s]' % re.escape(string.punctuation))
emailStr = regex.sub(' ', title + " " +body.lower())
emailStr = re.sub( '\s+', ' ', emailStr )
# split the line into words
```

```
global  1
risk    1
management      1
operations      1
congratulations         1
sally   1
kk      1
forwarded       1
by      1
kathy   1
kokas   1
corp    1
enron   1
on      1
01      1
17      1
2000    1
08      1
08      1
pm      1
from    1
rick    1
causey  1
01      1
17      1
2000    1
06      1
04      1
pm      1
sent    1
by      1
enron   1
announcements   1
to      1
all     1
enron   1
worldwide       1
cc      1
subject         1
global  1
risk    1
management      1
operations      1
recognizing     1
enron   1
        1
s       1
increasing      1
worldwide       1
presence        1
in      1
the     1
wholesale       1
energy  1
business        1
and     1
the     1
need    1
to      1
```

HW2.2.1 WORDCOUNT

Using the Enron data from and Hadoop MapReduce streaming, write the mapper/reducer job that will determine the word count (number of occurrences) of each white-space delimitted token (assume spaces, fullstops, comma as delimiters). Examine the word “assistance” and report its word count in both SPAM and HAM classes.

CROSSCHECK the frequency using Unic commands (e.g., use multiple grep to get the frequency in each class):

```
grep assistance enronemail_1h.txt|cut -d$'\t' -f4| grep assistance|w
c -l
       8
```

NOTE: "assistance" occurs on 8 lines but how many times does the token occur? 10 times! This is the number we are looking for!

OUTPUT:
The expected output for 2.2.1 is given as:
`assistance 10`
(assistance followed by a tab and the number 10).

In [56]:
```
!mkdir Enron
```

```
mkdir: cannot create directory `Enron': File exists
```

In [57]:
```
!grep assistance NaiveBayes/enronemail_1h.txt|cut -d$'\t' -f4| g
rep assistance|wc -l
```

```
8
```

In [58]:
```
%%writefile Enron/mapper2.2.1.py
#!/usr/bin/env python
import sys, re, string

# START STUDENT CODE HW221MAPPER


# define regex for punctuation removal
#regex = re.compile('[%s]' % re.escape(string.punctuation))
# input comes from STDIN (standard input)

# use subject and body

# remove punctuations, only have white-space as delimiter

# write the results to STDOUT (standard output);
# what we output here will be the input for the
# Reduce step, i.e. the input for reducer.py
#
# tab-delimited; the trivial word count is 1
for line in sys.stdin:
    docID, docClass,title,body = line.split("\t",3)
    regex = re.compile('[%s]' % re.escape(string.punctuation))
    emailStr = regex.sub(' ', title + " " +body.lower())
    emailStr = re.sub( '\s+', ' ', emailStr )
# split the line into words
    words = emailStr.split()
    for w in words:

            print w, "\t", 1 #or yield(w, 1)

# END STUDENT CODE HW221MAPPER
```

```
Overwriting Enron/mapper2.2.1.py
```

In [59]:
```
%%writefile Enron/reducer2.2.1.py
#!/usr/bin/env python
from operator import itemgetter
import sys

# START STUDENT CODE HW221REDUCER
import sys
import re
cur_key = None
cur_count = 0

for line in sys.stdin:
    key, value = line.split()
    if key == cur_key:
        cur_count += int(value)
    else:
        if cur_key:
            print '%s\t%s' % (cur_key, cur_count)
        cur_key = key
        cur_count = int(value)

print '%s\t%s' % (cur_key, cur_count)


# END STUDENT CODE HW221REDUCER
```

```
Overwriting Enron/reducer2.2.1.py
```

In [60]:
```
!chmod a+x Enron/mapper2.2.1.py
!chmod a+x Enron/reducer2.2.1.py
```

In [61]:
```
# START STUDENT CODE HW221HADOOP
!hdfs dfs -rm enronemail_1h.txt
!hdfs dfs -copyFromLocal NaiveBayes/enronemail_1h.txt
!hdfs dfs -rm -r HW2.2.1/results/

##################### IMPORTANT ##################################
########################################
# Make sure you have the correct paths to the jar file as wel as
the input and output files!!
# make sure to include the -files option. Do ***** NOT ****** pu
t spaces between the file paths!
################################################################
########################################

!hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
  -D mapreduce.job.output.key.comparator.class=org.apache.hadoop
.mapred.lib.KeyFieldBasedComparator \
  -D mapreduce.partition.keycomparator.options="-k1,1"  \
  -files Enron/mapper2.2.1.py,Enron/reducer2.2.1.py\
  -mapper mapper2.2.1.py \
  -reducer reducer2.2.1.py \
  -input enronemail_1h.txt \
  -output HW2.2.1/results/\
  -numReduceTasks 1


# END STUDENT CODE HW221HADOOP
```

```
Deleted enronemail_1h.txt
Deleted HW2.2.1/results
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.ja
r] /tmp/streamjob8708841748129486093.jar tmpDir=null
17/05/23 02:23:32 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:23:33 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:23:34 INFO mapred.FileInputFormat: Total input path
s to process : 1
17/05/23 02:23:35 INFO mapreduce.JobSubmitter: number of splits
:2
17/05/23 02:23:35 INFO mapreduce.JobSubmitter: Submitting token
s for job: job_1495464826300_0093
17/05/23 02:23:35 INFO impl.YarnClientImpl: Submitted applicati
on application_1495464826300_0093
17/05/23 02:23:35 INFO mapreduce.Job: The url to track the job:
http://quickstart.cloudera:8088/proxy/application_1495464826300
_0093/
17/05/23 02:23:35 INFO mapreduce.Job: Running job: job_14954648
26300_0093
17/05/23 02:23:47 INFO mapreduce.Job: Job job_1495464826300_009
3 running in uber mode : false
17/05/23 02:23:47 INFO mapreduce.Job:  map 0% reduce 0%
17/05/23 02:24:00 INFO mapreduce.Job:  map 50% reduce 0%
17/05/23 02:24:01 INFO mapreduce.Job:  map 100% reduce 0%
17/05/23 02:24:09 INFO mapreduce.Job:  map 100% reduce 100%
17/05/23 02:24:10 INFO mapreduce.Job: Job job_1495464826300_009
3 completed successfully
17/05/23 02:24:10 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=351817
                FILE: Number of bytes written=1056085
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=217089
                HDFS: Number of bytes written=52015
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots
(ms)=21192
                Total time spent by all reduces in occupied slo
ts (ms)=6647
                Total time spent by all map tasks (ms)=21192
                Total time spent by all reduce tasks (ms)=6647
                Total vcore-seconds taken by all map tasks=2119
2
                Total vcore-seconds taken by all reduce tasks=6
647
                Total megabyte-seconds taken by all map tasks=2
1700608
                Total megabyte-seconds taken by all reduce task
s=6806528
```

In [62]:
```
!hdfs dfs -cat HW2.2.1/results/part-0000* > enron_counts.txt
```

In [63]:
```
!grep -i assistance enron_counts.txt
```

```
assistance      10
```

HW2.2.2

Using Hadoop MapReduce and your wordcount job (from HW2.2.1) determine the top-10 occurring tokens (most frequent tokens) using a single reducer for the SPAM class and for the HAM class.

The expected output for 2.2.2 is in terms of three tab-separated columns:
`CLASS\tWORD\tCOUNT`
with the HAM top 10 coming first followed by the SPAM top 10.

In [64]:
```
%%writefile Enron/mapper2.2.2.py
#!/usr/bin/env python
import sys, re, string

# START STUDENT CODE HW221MAPPER


# define regex for punctuation removal
#regex = re.compile('[%s]' % re.escape(string.punctuation))
# input comes from STDIN (standard input)

# use subject and body

# remove punctuations, only have white-space as delimiter

# write the results to STDOUT (standard output);
# what we output here will be the input for the
# Reduce step, i.e. the input for reducer.py
#
# tab-delimited; the trivial word count is 1
for line in sys.stdin:
    docID, docClass,title,body = line.split("\t",3)
    regex = re.compile('[%s]' % re.escape(string.punctuation))
    emailStr = regex.sub(' ', title + " " +body.lower())
    emailStr = re.sub( '\s+', ' ', emailStr )
# split the line into words
    words = emailStr.split()
    for w in words:

            print "%s\t%d\t%s"%(w,1,docClass)# w, "\t", 1,"\t",d
ocClass #or yield(w, 1)

# END STUDENT CODE HW222MAPPER
```

```
Overwriting Enron/mapper2.2.2.py
```

In [65]:
```
%%writefile Enron/reducer2.2.2.py
#!/usr/bin/env python
from operator import itemgetter
import sys

# START STUDENT CODE HW221REDUCER
import sys
import re
cur_key = None

docClass = 0

wordcount = {}

for line in sys.stdin:
    key, value ,docClass = line.split("\t")
    docClass = docClass.rstrip()
    if key == cur_key:
        wordcount[(docClass,cur_key)]+= int(value)
    else:
        #if cur_key:
        #    print '%s\t%s' % (cur_key, wordcount[(docClass,cur_
key)])
        cur_key = key

        if docClass == '0':
            wordcount[('0',cur_key)] = int(value)
            wordcount[('1',cur_key)] = 0
        else:
            wordcount[('1',cur_key)] = int(value)
            wordcount[('0',cur_key)] = 0
        #cur_count = int(value)

count  = 0
#print wordcount
for w in sorted(wordcount, key=wordcount.get, reverse=True):
     if w[0] == "0" and count <11:
        print '%s\t%s\t%s' %(w[0],w[1],wordcount[w])
        count = count +1
count  = 0
for w in sorted(wordcount, key=wordcount.get, reverse=True):
     if w[0] == "1" and count <11:
        print '%s\t%s\t%s' % (w[0],w[1],wordcount[w])
        count = count +1
#print '%s\t%s' % (cur_key, cur_count)


# END STUDENT CODE HW222REDUCER
```

```
Overwriting Enron/reducer2.2.2.py
```

In [66]:
```
!chmod a+x Enron/mapper2.2.2.py
!chmod a+x Enron/reducer2.2.2.py
```

In [67]:
```
!echo "0001.1999-12-10.kaminski 1        re: rankings    thank y
ou."|Enron/mapper2.2.2.py|Enron/reducer2.2.2.py
```

```
0       thank   0
0       rankings        0
0       you     0
0       re      0
1       re      1
1       thank   1
1       rankings        1
1       you     1
```

In [68]:
```
# START STUDENT CODE HW222HADOOP

!hdfs dfs -rm enronemail_1h.txt
!hdfs dfs -copyFromLocal NaiveBayes/enronemail_1h.txt
!hdfs dfs -rm -r HW2.2.2/results/

##################### IMPORTANT ##################################
########################################
# Make sure you have the correct paths to the jar file as wel as
the input and output files!!
# make sure to include the -files option. Do ***** NOT ****** pu
t spaces between the file paths!
################################################################
########################################

!hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
  -D mapreduce.job.output.key.comparator.class=org.apache.hadoop
.mapred.lib.KeyFieldBasedComparator \
  -D mapreduce.partition.keycomparator.options="-k1,1"  \
  -files Enron/mapper2.2.2.py,Enron/reducer2.2.2.py\
  -mapper mapper2.2.2.py \
  -reducer reducer2.2.2.py \
  -input enronemail_1h.txt \
  -output HW2.2.2/results/\
  -numReduceTasks 1


# END STUDENT CODE HW222HADOOP
```

```
Deleted enronemail_1h.txt
Deleted HW2.2.2/results
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.ja
r] /tmp/streamjob4876128199393379846.jar tmpDir=null
17/05/23 02:24:36 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:24:37 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:24:38 INFO mapred.FileInputFormat: Total input path
s to process : 1
17/05/23 02:24:39 INFO mapreduce.JobSubmitter: number of splits
:2
17/05/23 02:24:39 INFO mapreduce.JobSubmitter: Submitting token
s for job: job_1495464826300_0094
17/05/23 02:24:39 INFO impl.YarnClientImpl: Submitted applicati
on application_1495464826300_0094
17/05/23 02:24:39 INFO mapreduce.Job: The url to track the job:
http://quickstart.cloudera:8088/proxy/application_1495464826300
_0094/
17/05/23 02:24:39 INFO mapreduce.Job: Running job: job_14954648
26300_0094
17/05/23 02:24:49 INFO mapreduce.Job: Job job_1495464826300_009
4 running in uber mode : false
17/05/23 02:24:49 INFO mapreduce.Job:  map 0% reduce 0%
17/05/23 02:25:01 INFO mapreduce.Job:  map 50% reduce 0%
17/05/23 02:25:02 INFO mapreduce.Job:  map 100% reduce 0%
17/05/23 02:25:10 INFO mapreduce.Job:  map 100% reduce 100%
17/05/23 02:25:11 INFO mapreduce.Job: Job job_1495464826300_009
4 completed successfully
17/05/23 02:25:11 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=385020
                FILE: Number of bytes written=1122494
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=217089
                HDFS: Number of bytes written=210
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots
(ms)=19435
                Total time spent by all reduces in occupied slo
ts (ms)=7100
                Total time spent by all map tasks (ms)=19435
                Total time spent by all reduce tasks (ms)=7100
                Total vcore-seconds taken by all map tasks=1943
5
                Total vcore-seconds taken by all reduce tasks=7
100
                Total megabyte-seconds taken by all map tasks=1
9901440
                Total megabyte-seconds taken by all reduce task
s=7270400
```

In [69]:
```
!hdfs dfs -cat HW2.2.2/results/part-0000* > sorted_enron_counts.
txt
```

In [70]:
```
# display top from each class in the sorted_enron_counts.txt fil
e
!cat sorted_enron_counts.txt
```

```
0	the	549
0	to	398
0	ect	382
0	and	278
0	of	230
0	hou	206
0	a	196
0	in	182
0	for	170
0	on	135
0	will	132
1	the	698
1	to	566
1	and	392
1	your	357
1	a	347
1	you	345
1	of	336
1	in	236
1	for	204
1	com	153
1	it	152
```

HW2.2.3 (Optional)

Using Hadoop MapReduce and your wordcount job (from HW2.2.1) determine the top-10 occurring tokens (most frequent tokens) using multiple reducers.

To achieve a total sort one needs to write a custom mapper to to prepend a partition key to each record. The shuffle phase will need a custom partitioner based upon the prepended key, while the sort is based upon a composite key which is made up of the partition key and the word count (i.e., we will do a secondary sort on a composite key that is made up of the prepended key and the word count. This all can be done with one map-reduce job.

In [ ]:
```
```

HW2.3 Multinomial NAIVE BAYES with NO Smoothing using a single reducer

Multinomial NAIVE BAYES model with NO Smoothing using a single reducer

In this assignment you will produce a spam filter based upon a multinomial naive Bayes classifier. For a quick reference on the construction of the Multinomial NAIVE BAYES classifier that you will code, please consult the following:

- A nice textbook introduction to the different flavors of Naive Bayes is provide in chapter 13 (http://nlp.stanford.edu/IR-book/pdf/13bayes.pdf) of the IRBook. Nice worked out examples are also provdied
- The "Document Classification" section of the wikipedia page on Naive Bayes (https://en.wikipedia.org/wiki/Naive_Bayes_classifier#Document_classification)
- OR the original paper (http://www.aueb.gr/users/ion/docs/ceas2006_paper.pdf) by the curators of the Enron email data.

For the sake of this assignment we will focus on the basic construction of the parallelized classifier, and not consider its validation or calibration, and so you will have the classifier operate on its own training data (unlike a field application where one would use non-overlapping subsets for training, validation and testing). NOTE: please use the subject field and the body field for all your Naive Bayes modeling.

For all tasks in this HW problem, please use one (1) reducer.

HW2.3.1 Learn a Multinomial Naive Bayes model on a small dataset (Chinese dataset: 5 documents)

Using Hadoop streaming MapReduce, write a mapper/reducer job(s) that will learn a Naive Bayes classifier Use all white-space delimitted tokens as independent input variables (assume spaces, fullstops, commas as delimiters). Note: for multinomial Naive Bayes, the class conditional probability for a word such as "assistance" given the class is SPAM, Pr(X="assistance"|Y=SPAM), is calculated as follows:

$$\frac{\textit{the number of times "assistance" occurs in SPAM labeled documents}}{\textit{the number of words in documents labeled SPAM}}$$

E.g., consider that "assistance" occurs 5 times in all of the documents Labeled SPAM, and the length in terms of the number of words in all documents labeled as SPAM (when concatenated) is 1,000. Then $Pr(X = \text{"assistance"} \mid Y = SPAM) = 5/1000$. Note this is a multinomial estimation of the class conditional for a Naive Bayes Classifier. No smoothing is needed in this HW problem. Please represent you model as a record where the key is the first field (TAB separated), and the value, the remaining part, is composed of two values corresponding the class conditional counts or probabilities depending on what phase of learning we are in. A typical record whether in a file or in memory will have the following KEY-VALUE structure:

- `Word\tCount(of Word in documents corresponding to HAM)\tCount(of Word in documents corresponding to SPAM)`
- In memory this TSV-type data can be stored as a dictionary or defaultDict to record the learnt model or intermediate versions of the model

Multinomial Naive bayes Classification

Sketch of mathematics:

In Naive Bayes, the probability of a document Doc has classification C is calculated as follows:

$$P\Big(C \mid \bigwedge_{w_i \in Doc} w_i\Big) \approx \frac{P(C)\prod_{w_i \in Doc} P(w_i \mid C)}{P(\wedge w_i)}$$

where the $\approx$ is due to the naive approximation of joint probability in the LHS numerator.

To avoid floating point errors in the prompt we can hence use the equivilent statement that:

$$log\Big(P\Big(C \mid \bigwedge_{w_i \in Doc} w_i\Big) log\Big) \approx log(P(C)) + \sum_{w_i \in Doc} log\Big(P(w_i \mid C)\Big) - log(P(\wedge w_i))$$

So for fixed document $log(P(\wedge w_i))$ we can optimize the RHS, by taking the following:

$$\hat{y} \approx argmax_{C \in \{\text{spam}, \text{ham}\}} \Big(log(P(C)) + \sum_{w_i \in Doc} log\Big(P(w_i \mid C)\Big)\Big)$$

and the bag of words assumption and without smoothing:

$$P(w_i \mid C) \approx \frac{Count(w_i, C)}{\sum_j Count(w_j, C)}$$

where $Count(w_j, C)$ is the count of instances of term w_j in the class C (viz, spam or ham).

Hence we need to generat the following calculations to build our NB classifier:

1. $\forall w_j \in Vocab, C \in \{\text{spam}, \text{ham}\})) : Count(w_j, C)$
    - We can then calculate the conditional probabilities:
        - $P(w_i \mid C)$
2. For one of the values $C \in \{\text{spam}, \text{ham}\})) : P(C) = Count(C) \ / \ |Emails|$
    - This only needs to be calculated a single time and then can be summed with the conditional log probabilities to get our $\hat{y}$ equation arguments

In [ ]:

Divide the task into 2 map-reduce tasks

- a modeling phase where the output is a MODEL file
- A classification phase where we use the model

For phase 1: Build a multionmial Naive Bayes Model

- MAPPER_3a:
  - As with the mapper in hw2.2, generates raw word tokens with an added field for the spam value associated with that instance.
- REDUCER_3a: For each word w_i
  - Agragates from mapper: $\forall w_j \in Vocab, C \in \{\text{spam, ham}\} : Count(w_j, C)$
  - And calculates $P(w_i \mid C) \approx \frac{Count(w_i, C)}{\sum_j Count(w_j, C)}$

For phase 2: Classify each example using the learnt multionmial Naive Bayes Model

- MAPPER_3b: for each Doc, calculates
  - $\forall w_j \in Vocab : argmax_{C \in \{spam, ham\}} \left(log(P(C)) + \sum_{w_i \in Doc} log\left(P(w_i \mid C)\right)\right)$
- REDUCER_3b: calculates
  - $Err(Model, DF) = \frac{count_{DF}(\hat{y} \neq label)}{|DF|}$

Suggested Record format that can be used of all Mappers and Reducers

Here is the learnt binary multinomial Naive Bayes model for the Chinese Dateset which is described below. Note this is a small dataset with 4 records (i.e., four documnents). Each record in the model file has the following format:
`Word\t freq(word in class0) ,freq(word in class1), (Pr(Word|Class =0), Pr(Word|Class =1))`

In [71]:
```
%mkdir NaiveBayes
```

```
mkdir: cannot create directory `NaiveBayes': File exists
```

In [72]:
```
%%writefile NaiveBayes/model1.txt
Beijing 0.0,1.0,0.111111111111,0.142857142857
Chinese 1.0,5.0,0.222222222222,0.428571428571
Tokyo   1.0,0.0,0.222222222222,0.0714285714286
Shanghai        0.0,1.0,0.111111111111,0.142857142857
ClassPriors     1.0,3.0,0.25,0.75
Japan   1.0,0.0,0.222222222222,0.0714285714286
Macao   0.0,1.0,0.111111111111,0.142857142857
```

```
Overwriting NaiveBayes/model1.txt
```

Load model file and play with it before writing the classifier job

In [73]:
```
# load model file as strings (it is not tokenized; that will com
e next)
modelStats = {}
recordStrs = [s.split('\n')[0].split('\t') for s in open("NaiveB
ayes/model1.txt").readlines()]
for word, statsStr in recordStrs:
    print word, "-->", statsStr
```

```
Beijing --> 0.0,1.0,0.111111111111,0.142857142857
Chinese --> 1.0,5.0,0.222222222222,0.428571428571
Tokyo --> 1.0,0.0,0.222222222222,0.0714285714286
Shanghai --> 0.0,1.0,0.111111111111,0.142857142857
ClassPriors --> 1.0,3.0,0.25,0.75
Japan --> 1.0,0.0,0.222222222222,0.0714285714286
Macao --> 0.0,1.0,0.111111111111,0.142857142857
```

In [74]:
```
# load model file
# notice the string quotes around the frequncy and probabilits
# we need to fix that next
modelStats = {}
recordStrs = [s.split('\n')[0].split('\t') for s in open("NaiveB
ayes/model1.txt").readlines()]
for word, statsStr in recordStrs:
    modelStats[word] =  statsStr.split(",")
modelStats
```

Out[74]:
```
{'Beijing': ['0.0', '1.0', '0.111111111111', '0.142857142857'],
 'Chinese': ['1.0', '5.0', '0.222222222222', '0.428571428571'],
 'ClassPriors': ['1.0', '3.0', '0.25', '0.75'],
 'Japan': ['1.0', '0.0', '0.222222222222', '0.0714285714286'],
 'Macao': ['0.0', '1.0', '0.111111111111', '0.142857142857'],
 'Shanghai': ['0.0', '1.0', '0.111111111111', '0.142857142857']
,
 'Tokyo': ['1.0', '0.0', '0.222222222222', '0.0714285714286']}
```

In [75]:
```
# load model file
# convert strings to floats using Pythons map function
# the map iterates over each element in the list apply the float
() function
# which converts a string to float
# RESULT: modelStats now contains our multinomial Naive Bayes mo
del
modelStats = {}
recordStrs = [s.split('\n')[0].split('\t') for s in open("NaiveB
ayes/model1.txt").readlines()]
for word, statsStr in recordStrs:
    modelStats[word] = map(float, statsStr.split(","))
modelStats
```

Out[75]:
```
{'Beijing': [0.0, 1.0, 0.111111111111, 0.142857142857],
 'Chinese': [1.0, 5.0, 0.222222222222, 0.428571428571],
 'ClassPriors': [1.0, 3.0, 0.25, 0.75],
 'Japan': [1.0, 0.0, 0.222222222222, 0.0714285714286],
 'Macao': [0.0, 1.0, 0.111111111111, 0.142857142857],
 'Shanghai': [0.0, 1.0, 0.111111111111, 0.142857142857],
 'Tokyo': [1.0, 0.0, 0.222222222222, 0.0714285714286]}
```

In [76]:
```
#classify a sample document using the model
# Use the following record to test
#   D5  0       Chinese Chinese Chinese Tokyo Japan
# FIRST : print Pr(Word|Class) in an unextracted form
line = "D5      0       Chinese Chinese Chinese Tokyo Japan"
docID, docClass,text = line.split("\t",2)
words = text.split()
#print docID,docClass, words
for word in words:
    print "Pr(",word, "| Class)", modelStats[word]  #Pr(Class=0|
Doc)  all stats

# STILL under constuction
```

```
Pr( Chinese | Class) [1.0, 5.0, 0.222222222222, 0.428571428571]
Pr( Chinese | Class) [1.0, 5.0, 0.222222222222, 0.428571428571]
Pr( Chinese | Class) [1.0, 5.0, 0.222222222222, 0.428571428571]
Pr( Tokyo | Class) [1.0, 0.0, 0.222222222222, 0.0714285714286]
Pr( Japan | Class) [1.0, 0.0, 0.222222222222, 0.0714285714286]
```

In [77]:
```
# classify a sample document using the model
# Use the following record to test
# D5    0       Chinese Chinese Chinese Tokyo Japan
# ------------------------------------------------------------
------
# Posterior Probabilities Pr(Class=0| Doc) and Pr(Class=1| Doc)
# Naive Bayes inference Pr(Class=0| Doc)  ~ Pr(Class=0) * Pr(Cla
ss=0| word1) * Pr(Class=0| word2)......

line = "D5      0       Chinese Chinese Chinese Tokyo Japan"
docID, docClass,text = line.split("\t",2)
words = text.split()
#Class priors (Pr(Class =0) and Pr(Class =1))
c0, c1, prClass0, prClass1 = map(float, modelStats["ClassPriors"
])
print "prClass0=%04.3f, prClass1=%04.3f" % (prClass0, prClass1)
# Posterior Probabilities Pr(Class=0| Doc) and Pr(Class=1| Doc)
# Naive Bayes inference Pr(Class=0| Doc)  ~ Pr(Class=0) * Pr(Cla
ss=0| word1) * Pr(Class=0| word2)......
PrClass0GivenDoc = prClass0
PrClass1GivenDoc = prClass1
for word in words:
    #print word, modelStats[word]  #Pr(word|class = 0) all stats
    print 'Pr(%s|class = 0) = %04.3f' %(word, modelStats[word][2
])  #Pr(word|class = 0)
    print 'Pr(%s|class = 1) = %04.3f' %(word, modelStats[word][3
])  #Pr(word|class = 1)
    PrClass0GivenDoc *= modelStats[word][2]
    PrClass1GivenDoc *= modelStats[word][3]

print "Pr(Class=0| Doc=D5) is %6.5f" % (PrClass0GivenDoc)
print "Pr(Class=1| Doc=D5) is %6.5f" % (PrClass1GivenDoc)
```

```
prClass0=0.250, prClass1=0.750
Pr(Chinese|class = 0) = 0.222
Pr(Chinese|class = 1) = 0.429
Pr(Chinese|class = 0) = 0.222
Pr(Chinese|class = 1) = 0.429
Pr(Chinese|class = 0) = 0.222
Pr(Chinese|class = 1) = 0.429
Pr(Tokyo|class = 0) = 0.222
Pr(Tokyo|class = 1) = 0.071
Pr(Japan|class = 0) = 0.222
Pr(Japan|class = 1) = 0.071
Pr(Class=0| Doc=D5) is 0.00014
Pr(Class=1| Doc=D5) is 0.00030
```

Use logs to avoid precision problems

In [78]:
```
# classify a sample document using the model
# Use the following record to test
# D5    0       Chinese Chinese Chinese Tokyo Japan
# ------------------------------------------------------------
------
# Posterior Probabilities Pr(Class=0| Doc) and Pr(Class=1| Doc)
# Naive Bayes inference Pr(Class=0| Doc)  ~ Pr(Class=0) * Pr(Cla
ss=0| word1) * Pr(Class=0| word2)......

from math import log
from math import exp

line = "D5      0       Chinese Chinese Chinese Tokyo Japan"
docID, docClass,text = line.split("\t",2)
words = text.split()
#Class priors (Pr(Class =0) and Pr(Class =1))
c0, c1, prClass0, prClass1 = map(float, modelStats["ClassPriors"
])
print "prClass0=%04.3f, prClass1=%04.3f" % (prClass0, prClass1)
# Posterior Probabilities Pr(Class=0| Doc) and Pr(Class=1| Doc)
# Naive Bayes inference Pr(Class=0| Doc)  ~ log(Pr(Class=0)) + l
og(Pr(Class=0| word1)) + log(Pr(Class=0| word2))......
PrClass0GivenDoc = log(prClass0)
PrClass1GivenDoc = log(prClass1)
for word in words:
    #print word, modelStats[word]  #Pr(word|class = 0) all stats
    print 'Pr(%s|class = 0) = %04.3f' %(word, modelStats[word][2
])  #Pr(word|class = 0)
    print 'Pr(%s|class = 1) = %04.3f' %(word, modelStats[word][3
])  #Pr(word|class = 1)
    PrClass0GivenDoc += log(modelStats[word][2])
    PrClass1GivenDoc += log(modelStats[word][3])

print "Pr(Class=0| Doc=D5) = %6.5f, log(Pr(Class=0| Doc=D5)) = %
f" % (exp(PrClass0GivenDoc), PrClass0GivenDoc)
print "Pr(Class=1| Doc=D5) = %6.5f, log(Pr(Class=1| Doc=D5)) = %
f" % (exp(PrClass1GivenDoc), PrClass0GivenDoc)
```

```
prClass0=0.250, prClass1=0.750
Pr(Chinese|class = 0) = 0.222
Pr(Chinese|class = 1) = 0.429
Pr(Chinese|class = 0) = 0.222
Pr(Chinese|class = 1) = 0.429
Pr(Chinese|class = 0) = 0.222
Pr(Chinese|class = 1) = 0.429
Pr(Tokyo|class = 0) = 0.222
Pr(Tokyo|class = 1) = 0.071
Pr(Japan|class = 0) = 0.222
Pr(Japan|class = 1) = 0.071
Pr(Class=0| Doc=D5) = 0.00014, log(Pr(Class=0| Doc=D5)) = -8.90
6681
Pr(Class=1| Doc=D5) = 0.00030, log(Pr(Class=1| Doc=D5)) = -8.90
6681
```

Create a Naive Bayes Model class

In [79]:
```
!mkdir NaiveBayes
```

```
mkdir: cannot create directory `NaiveBayes': File exists
```

In [80]:
```
%%writefile NaiveBayes/NaiveBayesModel.py
#!/usr/bin/env python
from math import log
from math import exp

class NaiveBayesModel(object):

    def __init__(self, modelFile):
        self.model = {}
        recordStrs = [s.split('\n')[0].split('\t') for s in open
(modelFile).readlines()]
        for word, statsStr in recordStrs:
            self.model[word] = map(float, statsStr.split(","))
        #Class priors: counts and probs (Pr(Class =0) and Pr(Cla
ss =1))
        self.c0, self.c1, self.prClass0, self.prClass1 = map(flo
at, self.model["ClassPriors"])



    def classify(self, doc):
        # Posterior Probabilities Pr(Class=0| Doc) and Pr(Class=
1| Doc)
        # Naive Bayes inference Pr(Class=0| Doc)  ~ Pr(Class=0)
* Pr(Class=0| word1) * Pr(Class=0| word2)......
        PrClass0GivenDoc = self.prClass0
        PrClass1GivenDoc = self.prClass1
        for word in doc:
            PrClass0GivenDoc *= self.model[word][2]
            PrClass1GivenDoc *= self.model[word][3]
        return([PrClass0GivenDoc, PrClass1GivenDoc])

    # the natural log based version of this
    # helps avoid underflow issues
    def classifyInLogs(self, doc):
        # Posterior Probabilities Pr(Class=0| Doc) and Pr(Class=
1| Doc)
        # Naive Bayes inference Pr(Class=0| Doc)  ~ Pr(Class=0)
* Pr(Class=0| word1) * Pr(Class=0| word2)......
        PrClass0GivenDoc = log(self.prClass0)
        PrClass1GivenDoc = log(self.prClass1)
        for word in doc:  #NOTE: Improvement: on loading one sho
uld convert probs to log probs!
            c0 = self.model[word][2]
            c1 = self.model[word][3]
            if c0 != 0:
                PrClass0GivenDoc += log(c0)
            else:
                PrClass0GivenDoc = float("-inf")
            if c1 != 0:
                PrClass1GivenDoc += log(c1)
            else:
                PrClass1GivenDoc = float("-inf")

        return([PrClass0GivenDoc, PrClass1GivenDoc])


    def printModel(self):
```

```
Overwriting NaiveBayes/NaiveBayesModel.py
```

In [81]:
```
%run NaiveBayes/NaiveBayesModel.py
```

Test Driver for Multinomial Naive Bayes Classifier

In [82]:
```
#classify a sample document using the model
# Use the following record to test
#   D5  0       Chinese Chinese Chinese Tokyo Japan
# ----------------------------------------------------------------
------
# Posterior Probabilities Pr(Class=0| Doc) and Pr(Class=1| Doc)
# Naive Bayes inference Pr(Class=0| Doc)  ~ Pr(Class=0) * Pr(Cla
ss=0| word1) * Pr(Class=0| word2)......

NBModel = NaiveBayesModel("NaiveBayes/model1.txt")
NBModel.printModel()
line = "D5      0       Chinese Chinese Chinese Tokyo Japan"
docID, docClass,text = line.split("\t",2)
words = text.split()
PrClass0GivenDoc, PrClass1GivenDoc = NBModel.classify(words)

print "Pr(Class=0| Doc=%s) is %6.5f" % (docID, PrClass0GivenDoc)
print "Pr(Class=1| Doc=%s) is %6.5f" % (docID, PrClass1GivenDoc)

PrClass0GivenDoc, PrClass1GivenDoc = NBModel.classifyInLogs(word
s)

print "Pr(Class=0| Doc=D5) = %6.5f, log(Pr(Class=0| Doc=D5)) = %
f" % (exp(PrClass0GivenDoc), PrClass0GivenDoc)
print "Pr(Class=1| Doc=D5) = %6.5f, log(Pr(Class=1| Doc=D5)) = %
f" % (exp(PrClass1GivenDoc), PrClass1GivenDoc)
```

```
NaiveBayes Model starts here
----------------
PRIORS: prClass0=0.250, prClass1=0.750
Pr( Beijing | Class) [0.0, 1.0, 0.111111111111, 0.142857142857]
Pr( Chinese | Class) [1.0, 5.0, 0.222222222222, 0.428571428571]
Pr( Tokyo | Class) [1.0, 0.0, 0.222222222222, 0.0714285714286]
Pr( Shanghai | Class) [0.0, 1.0, 0.111111111111, 0.142857142857
]
Pr( ClassPriors | Class) [1.0, 3.0, 0.25, 0.75]
Pr( Japan | Class) [1.0, 0.0, 0.222222222222, 0.0714285714286]
Pr( Macao | Class) [0.0, 1.0, 0.111111111111, 0.142857142857]
NaiveBayes Model ENDS here
----------------
Pr(Class=0| Doc=D5) is 0.00014
Pr(Class=1| Doc=D5) is 0.00030
Pr(Class=0| Doc=D5) = 0.00014, log(Pr(Class=0| Doc=D5)) = -8.90
6681
Pr(Class=1| Doc=D5) = 0.00030, log(Pr(Class=1| Doc=D5)) = -8.10
7690
```

Modelling Phase

In [83]:
```
%%writefile NaiveBayes/mapper_model.py
#!/usr/bin/env python
import sys, re, string

# Init mapper phase
# define regex for punctuation removal
regex = re.compile('[%s]' % re.escape(string.punctuation))

# inner loop mapper phase: process each record
# input comes from STDIN (standard input)

for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    # use subject and body

    parts = line.split("\t")
    docID, docClass, title = parts[0:3]
    if len(parts) == 4:
        body = parts[3]
    else:
        body = ""

    # remove punctuations, only have white-space as delimiter
    regex = re.compile('[%s]' % re.escape(string.punctuation))
    emailStr = regex.sub(' ', title.lower() + " " +body.lower())
#replace each punctuation with a space
    emailStr = re.sub( '\s+', ' ', emailStr )            # repla
ce multiple spaces with a space
    # split the line into words
    words = emailStr.split()

# START STUDENT CODE HW231MAPPER_MODEL

# define regex for punctuation removal

# increase counters
# write the results to STDOUT (standard output);
# what we output here will be the input for the
# Reduce step, i.e. the input for reducer.py
#
# tab-delimited; the trivial word count is 1

    for w in words:
        print "%s\t%d\t%d\t%s"%(w,1,int(docClass),docID)# w, "\

# END STUDENT CODE HW231MAPPER_MODEL
```

```
Overwriting NaiveBayes/mapper_model.py
```

In [84]:
```
%%writefile NaiveBayes/reducer_model.py
#!/usr/bin/env python
from operator import itemgetter
import sys, operator
import numpy as np
from  collections import OrderedDict

# START STUDENT CODE HW231REDUCER_MODEL

current_word = None
smooth_factor = 0 # no smoothing
current_count = [smooth_factor, smooth_factor]
msgIDs = {}
word = None
wordcount = {}

# input comes from STDIN
# remove leading and trailing whitespace

# parse the input we got from mapper.py

# convert count and spam flag (currently a string) to int


# handle msgID - store all IDs as we don't have too much
# not the best way to get prior, a two-level MapReduce jobs (ID
- word) would be optimal

# calculate NB parameters, and write the dictionary to a file fo
r the classification job
# prior probabilities

# conditional probability
# input comes from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # parse the input we got from mapper.py
    word, count, isSpam, msgID = line.split('\t', 3)


    # convert count and spam flag (currently a string) to int
    try:
        count = int(count)
        isSpam = int(isSpam)
    except ValueError:
        # count was not a number, so silently
        # ignore/discard this line
        continue

    # handle msgID - store all IDs as we don't have too much
    # not the best way to get prior, a two-level MapReduce jobs
(ID - word) would be optimal
    if msgID not in msgIDs:
        msgIDs[msgID] = isSpam

    # this IF-switch only works because Hadoop sorts map output
```

```
Overwriting NaiveBayes/reducer_model.py
```

In [85]:
```
!chmod a+x NaiveBayes/mapper_model.py
!chmod a+x NaiveBayes/reducer_model.py
```

Classification Phase

In [86]:
```
%%writefile NaiveBayes/mapper_classify.py
#!/usr/bin/env python
from NaiveBayesModel import NaiveBayesModel
import sys, re, string, subprocess
import sys, operator, math
import numpy as np
from math import log
from math import exp
# Init mapper phase

# read the MODEL into memory
# The model file resides the local disk (make sure to ship it ho
me from HDFS).
# In the Hadoop command linke be sure to add the follow the -fil
es commmand line option
NBModel = NaiveBayesModel("NaiveBayes/NaiveBayes.txt")
#NBModel.printModel()
# define regex for punctuation removal
regex = re.compile('[%s]' % re.escape(string.punctuation))

# inner loop mapper phase: process each record
# input comes from STDIN (standard input)
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    parts = line.split("\t")
    docID, docClass, title = parts[0:3]
    if len(parts) == 4:
        body = parts[3]
    else:
        body = ""
    # use subject and body
    # remove punctuations, only have white-space as delimiter
    emailStr = regex.sub(' ', title.lower() + " " +body.lower())
#replace each punctuation with a space
    emailStr = re.sub( '\s+', ' ', emailStr )            # repla
ce multiple spaces with a space
    # split the line into words
    words = emailStr.split()

# START STUDENT CODE HW231MAPPER_CLASSIFY



    #PrClass0GivenDoc, PrClass1GivenDoc = NBModel.classify(words
)

    #print "Pr(Class=0| Doc=%s) is %6.5f" % (docID, PrClass0Give
nDoc)
    #print "Pr(Class=1| Doc=%s) is %6.5f" % (docID, PrClass1Give
nDoc)

    PrClass0GivenDoc, PrClass1GivenDoc = NBModel.classify(words)
    if PrClass0GivenDoc > PrClass1GivenDoc:
        print "%s\t%s\t%s"%(docID,0,docClass)
    else:
        print "%s\t%s\t%s"%(docID,1,docClass)
```

```
Overwriting NaiveBayes/mapper_classify.py
```

In [87]:
```
%%writefile NaiveBayes/reducer_classify.py
#!/usr/bin/env python
from operator import itemgetter
import sys, operator, math
import numpy as np

numberOfRecords = 0
NumberOfMisclassifications=0
classificationAccurary = 0

# START STUDENT CODE HW231REDUCER_CLASSIFY

# input comes from STDIN
print "%s\t%s\t%s"%("PREDICTION","LABEL","DOCID")
for line in sys.stdin:
    # remove leading and trailing whitespace
    #line = line.strip()
    # split the line into words
    parts = line.split("\t")
    docID, preds, label = parts[0:3]
    print "%s\t%s\t%s"%( preds, label.rstrip(),docID.rstrip())
    if int(preds) != int(label):
        NumberOfMisclassifications +=1
    numberOfRecords+=1

classificationAccurary = float(float(numberOfRecords-NumberOfMis
classifications) / float(numberOfRecords)) * float(100)
print 'Multinomial Naive Bayes Classifier Results are Total Reco
rds: %d, Misclassifications: %d, Accuracy :%2.5f' % (numberOfRec
ords, NumberOfMisclassifications, classificationAccurary)
# END STUDENT CODE HW231REDUCER_CLASSIFY
```

```
Overwriting NaiveBayes/reducer_classify.py
```

In [88]:
```
!chmod a+x NaiveBayes/mapper_classify.py
!chmod a+x NaiveBayes/reducer_classify.py
!chmod a+x NaiveBayes/NaiveBayesModel.py
```

Run Map Reduce Job to learn a multinomical Naive Model from data

In [89]:
```
# START STUDENT CODE HW231HADOOP_MODEL
# STEP 1: make input directory on HDFS
#!hdfs dfs -mkdir -p /user/Xxxxxxx

# STEP2:  upload data to HDFS
!hdfs dfs -rm chineseExample_train.txt
!rm NaiveBayes/chineseExample_train.txt
!head -n 4 NaiveBayes/chineseExample.txt  > NaiveBayes/chineseEx
ample_train.txt
!hdfs dfs -copyFromLocal NaiveBayes/chineseExample_train.txt
!hdfs dfs -rm -r HW231HADOOP_CHINESE_UNIT_TEST/model/

##################### IMPORTANT ##################################
########################################
# Make sure you have the correct paths to the jar file as wel as
the input and output files!!
# make sure to include the -files option. Do ***** NOT ****** pu
t spaces between the file paths!
######################################################################
########################################

!hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
  -D mapreduce.job.output.key.comparator.class=org.apache.hadoop
.mapred.lib.KeyFieldBasedComparator \
  -D mapreduce.partition.keycomparator.options="-k1,1"  \
  -files NaiveBayes/mapper_model.py,NaiveBayes/reducer_model.py
\
  -mapper mapper_model.py\
  -reducer reducer_model.py \
  -input chineseExample_train.txt\
  -output HW231HADOOP_CHINESE_UNIT_TEST/model/\
  -numReduceTasks 1\
  -cmdenv PATH=/opt/anaconda/bin:$PATH




 # END STUDENT CODE HW231HADOOP_MODEL
```

```
Deleted chineseExample_train.txt
Deleted HW231HADOOP_CHINESE_UNIT_TEST/model
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.ja
r] /tmp/streamjob8963259309953632384.jar tmpDir=null
17/05/23 02:25:40 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:25:40 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:25:42 INFO mapred.FileInputFormat: Total input path
s to process : 1
17/05/23 02:25:42 INFO mapreduce.JobSubmitter: number of splits
:2
17/05/23 02:25:42 INFO mapreduce.JobSubmitter: Submitting token
s for job: job_1495464826300_0095
17/05/23 02:25:43 INFO impl.YarnClientImpl: Submitted applicati
on application_1495464826300_0095
17/05/23 02:25:43 INFO mapreduce.Job: The url to track the job:
http://quickstart.cloudera:8088/proxy/application_1495464826300
_0095/
17/05/23 02:25:43 INFO mapreduce.Job: Running job: job_14954648
26300_0095
17/05/23 02:25:54 INFO mapreduce.Job: Job job_1495464826300_009
5 running in uber mode : false
17/05/23 02:25:54 INFO mapreduce.Job:  map 0% reduce 0%
17/05/23 02:26:04 INFO mapreduce.Job:  map 50% reduce 0%
17/05/23 02:26:05 INFO mapreduce.Job:  map 100% reduce 0%
17/05/23 02:26:14 INFO mapreduce.Job:  map 100% reduce 100%
17/05/23 02:26:15 INFO mapreduce.Job: Job job_1495464826300_009
5 completed successfully
17/05/23 02:26:16 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=188
                FILE: Number of bytes written=353388
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=391
                HDFS: Number of bytes written=182
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots
(ms)=17803
                Total time spent by all reduces in occupied slo
ts (ms)=6829
                Total time spent by all map tasks (ms)=17803
                Total time spent by all reduce tasks (ms)=6829
                Total vcore-seconds taken by all map tasks=1780
3
                Total vcore-seconds taken by all reduce tasks=6
829
                Total megabyte-seconds taken by all map tasks=1
8230272
                Total megabyte-seconds taken by all reduce task
s=6992896
```

In [90]:
```
#put results file into local directories

!hdfs dfs -ls HW231HADOOP_CHINESE_UNIT_TEST/model/
!rm NaiveBayes/NaiveBayes.txt
!hdfs dfs -get HW231HADOOP_CHINESE_UNIT_TEST/model/part-00000 Na
iveBayes/NaiveBayes.txt
```

```
Found 2 items
-rw-r--r--   1 root supergroup          0 2017-05-23 02:26 HW23
1HADOOP_CHINESE_UNIT_TEST/model/_SUCCESS
-rw-r--r--   1 root supergroup        182 2017-05-23 02:26 HW23
1HADOOP_CHINESE_UNIT_TEST/model/part-00000
```

Run Map Reduce Job to classify data

In [91]:
```
# START STUDENT CODE HW231HADOOP_CLASSIFY

!hdfs dfs -rm chineseExample_test.txt
!rm NaiveBayes/chineseExample_test.txt
!tail -n 1 NaiveBayes/chineseExample.txt  > NaiveBayes/chineseEx
ample_test.txt
!hdfs dfs -copyFromLocal NaiveBayes/chineseExample_test.txt
!hdfs dfs -rm -r HW231HADOOP_CHINESE_UNIT_TEST/classifications/

##################### IMPORTANT ##################################
########################################
# Make sure you have the correct paths to the jar file as wel as
the input and output files!!
# make sure to include the -files option. Do ***** NOT ****** pu
t spaces between the file paths!
################################################################
########################################

!hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
  -D mapreduce.job.output.key.comparator.class=org.apache.hadoop
.mapred.lib.KeyFieldBasedComparator \
  -D mapreduce.partition.keycomparator.options="-k1,1"  \
  -files NaiveBayes\
  -mapper NaiveBayes/mapper_classify.py\
  -reducer NaiveBayes/reducer_classify.py  \
  -input chineseExample_test.txt \
  -output HW231HADOOP_CHINESE_UNIT_TEST/classifications/\
  -numReduceTasks 1\
  -cmdenv PATH=/opt/anaconda/bin:$PATH




# END STUDENT CODE HW231HADOOP_CLASSIFY
```

```
Deleted chineseExample_test.txt
Deleted HW231HADOOP_CHINESE_UNIT_TEST/classifications
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.ja
r] /tmp/streamjob5920961933740473233.jar tmpDir=null
17/05/23 02:26:43 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:26:44 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:26:46 INFO mapred.FileInputFormat: Total input path
s to process : 1
17/05/23 02:26:46 INFO mapreduce.JobSubmitter: number of splits
:2
17/05/23 02:26:47 INFO mapreduce.JobSubmitter: Submitting token
s for job: job_1495464826300_0096
17/05/23 02:26:47 INFO impl.YarnClientImpl: Submitted applicati
on application_1495464826300_0096
17/05/23 02:26:47 INFO mapreduce.Job: The url to track the job:
http://quickstart.cloudera:8088/proxy/application_1495464826300
_0096/
17/05/23 02:26:47 INFO mapreduce.Job: Running job: job_14954648
26300_0096
17/05/23 02:26:58 INFO mapreduce.Job: Job job_1495464826300_009
6 running in uber mode : false
17/05/23 02:26:58 INFO mapreduce.Job:  map 0% reduce 0%
17/05/23 02:27:09 INFO mapreduce.Job:  map 50% reduce 0%
17/05/23 02:27:10 INFO mapreduce.Job:  map 100% reduce 0%
17/05/23 02:27:18 INFO mapreduce.Job:  map 100% reduce 100%
17/05/23 02:27:18 INFO mapreduce.Job: Job job_1495464826300_009
6 completed successfully
17/05/23 02:27:18 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=15
                FILE: Number of bytes written=351962
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=294
                HDFS: Number of bytes written=139
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots
(ms)=18440
                Total time spent by all reduces in occupied slo
ts (ms)=6233
                Total time spent by all map tasks (ms)=18440
                Total time spent by all reduce tasks (ms)=6233
                Total vcore-seconds taken by all map tasks=1844
0
                Total vcore-seconds taken by all reduce tasks=6
233
                Total megabyte-seconds taken by all map tasks=1
8882560
                Total megabyte-seconds taken by all reduce task
s=6382592
```

Display the accuracy measure

In [92]:
```
# START STUDENT CODE HW231HADOOP_CLASSIFY_RESULTS
# (you may or may not need to add anything else in this block de
pending on your implementation)

!hdfs dfs -cat HW231HADOOP_CHINESE_UNIT_TEST/classifications/par
t-00000

# END STUDENT CODE HW231HADOOP_CLASSIFY_RESULTS
```

```
PREDICTION      LABEL   DOCID
0       0       D5
Multinomial Naive Bayes Classifier Results are Total Records: 1
, Misclassifications: 0, Accuracy :100.00000
```

In [ ]:
```

```

Write a systems test to regression test your map reduce job

Write a systems test to test your learning algorithm implementation using the following "Chinese" dataset. Please reserve document D5 as an independent test document (i.e., dont use it for training. Just use it for testing) Use the Chinese dataset to unit test your Mapper, reducer and final output.

Chinese dataset

In [93]:
```
%%writefile NaiveBayes/chineseExample.txt
D1      1       Chinese Beijing Chinese
D2      1       Chinese Chinese Shanghai
D3      1       Chinese Macao
D4      0       Tokyo Japan     Chinese
D5      0       Chinese Chinese Chinese Tokyo Japan
```

```
Overwriting NaiveBayes/chineseExample.txt
```

In [94]:
```
!rm NaiveBayes/NaiveBayes.txt
!head -n 4 NaiveBayes/chineseExample.txt  |NaiveBayes/mapper_mod
el.py| sort -k1,1 |NaiveBayes/reducer_model.py>NaiveBayes/NaiveB
ayes.txt
```

In [95]:
```
!echo 'D5       0       Chinese Chinese Chinese Tokyo Japan' |Na
iveBayes/mapper_classify.py
```

```
D5      0       0
```

In [96]:
```
# START STUDENT CODE HW231HADOOP_CHINESE_UNIT_TEST
!head -n 4 NaiveBayes/chineseExample.txt  |NaiveBayes/mapper_mod
el.py| sort -k1,1 |NaiveBayes/reducer_model.py>NaiveBayes/NaiveB
ayes.txt
import unittest
import re
class Testchinese(unittest.TestCase):

    def test_model(self):
        NBModel = NaiveBayesModel("NaiveBayes/NaiveBayes.txt")
        self.assertEqual([1,3,0.25,0.75], map(float, NBModel.mod
el["ClassPriors"]))
        self.assertAlmostEqual(0.333,  float(NBModel.model["chin
ese"][2]),2)
        self.assertAlmostEqual(0.125,  float(NBModel.model["maca
o"][3]),2)

    def test_classification(self):
        NBModel = NaiveBayesModel("NaiveBayes/NaiveBayes.txt")
        line = 'D5      0       Chinese Chinese Chinese Tokyo Ja
pan'.strip()
        # split the line into words
        parts = line.split("\t")
        docID, docClass, title = parts[0:3]
        if len(parts) == 4:
            body = parts[3]
        else:
            body = ""
        emailStr = regex.sub(' ', title.lower() + " " +body.lowe
r()) #replace each punctuation with a space
        emailStr = re.sub( '\s+', ' ', emailStr )            # r
eplace multiple spaces with a space
        words = emailStr.split()
        PrClass0GivenDoc, PrClass1GivenDoc = NBModel.classify(wo
rds)
        self.assertAlmostEqual(0.0010282922839403295,  PrClass0G
ivenDoc,2)

suite = unittest.TestLoader().loadTestsFromTestCase(Testchinese)
unittest.TextTestRunner(verbosity=2).run(suite)
# END STUDENT CODE HW231HADOOP_CHINESE_UNIT_TEST
```

```
test_classification (__main__.Testchinese) ... ok
test_model (__main__.Testchinese) ... ok

----------------------------------------------------------------
-------
Ran 2 tests in 0.012s

OK
```

Out[96]:
```
<unittest.runner.TextTestResult run=2 errors=0 failures=0>
```

HW2.3.2 Learn a multinomial naive Bayes model (with no smoothing) by hand

Learn the multinomial naive Bayes by hand and show the formulas, and your calculations in a nice tabular form.

Compare your hand calculations for the following:

- the learnt multinomial naive Bayes with NO smoothing
- the classification of the D5 test document

with textbook calculation listed here:

- Note the worked example here (https://www.dropbox.com/s/f17c4mvmm5fuwav/chineseTestCaseFullyWorkedOut.png) is with smoothing "https://www.dropbox.com/s/f17c4mvmm5fuwav/chineseTestCaseFullyWorkedOut.png (https://www.dropbox.com/s/f17c4mvmm5fuwav/chineseTestCaseFullyWorkedOut.png)". It is taken from the IRBook chapter (http://nlp.stanford.edu/IR-book/pdf/13bayes.pdf) on Naive Bayes.

In [97]:
```
# download the image with worked solution and render it in the next cell below.
!curl "https://www.dropbox.com/s/f17c4mvmm5fuwav/chineseTestCaseFullyWorkedOut.png"
```

NB Example

In [98]:
```
%%HTML
<img src="chineseTestCaseFullyWorkedOut.png">
```

▶ **Table 13.1** Data for parameter estimation examples.

	docID	words in document	in $c =$ *China*?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

▶ **Table 13.2** Training and test times for NB.

mode	time complexity
training	$\Theta(\|\mathbb{D}\|L_{ave} + \|\mathbb{C}\||V|)$
testing	$\Theta(L_a + \|\mathbb{C}\|M_a) = \Theta(\|\mathbb{C}\|M_a)$

We have now introduced all the elements we need for training and applying an NB classifier. The complete algorithm is described in Figure 13.2.

Example 13.1: For the example in Table 13.1, the multinomial parameters we need to classify the test document are the priors $\hat{P}(c) = 3/4$ and $\hat{P}(\overline{c}) = 1/4$ and the following conditional probabilities:

$$\begin{aligned}
\hat{P}(\text{Chinese}|c) &= (5+1)/(8+6) = 6/14 = 3/7 \\
\hat{P}(\text{Tokyo}|c) = \hat{P}(\text{Japan}|c) &= (0+1)/(8+6) = 1/14 \\
\hat{P}(\text{Chinese}|\overline{c}) &= (1+1)/(3+6) = 2/9 \\
\hat{P}(\text{Tokyo}|\overline{c}) = \hat{P}(\text{Japan}|\overline{c}) &= (1+1)/(3+6) = 2/9
\end{aligned}$$

The denominators are $(8+6)$ and $(3+6)$ because the lengths of $text_c$ and $text_{\overline{c}}$ are 8 and 3, respectively, and because the constant B in Equation (13.7) is 6 as the vocabulary consists of six terms.

We then get:

$$\begin{aligned}
\hat{P}(c|d_5) &\propto 3/4 \cdot (3/7)^3 \cdot 1/14 \cdot 1/14 \approx 0.0003. \\
\hat{P}(\overline{c}|d_5) &\propto 1/4 \cdot (2/9)^3 \cdot 2/9 \cdot 2/9 \approx 0.0001.
\end{aligned}$$

Thus, the classifier assigns the test document to $c =$ *China*. The reason for this classification decision is that the three occurrences of the positive indicator Chinese in d_5 outweigh the occurrences of the two negative indicators Japan and Tokyo.

Hand calculations for Multinomial naive Bayes (Learning and classifiction)

- Insert hand calculations for learning a multinomial naive Bayes Classifier from the Chinese dataset
- Insert hand calculations for classifying the test document "D5" using the learnt multinomial naive Bayes Classifier from the Chinese dataset

HINTS:

Because Markdown is a superset of HTML you can even add things like HTML tables

For more background on notebook formatting see: here (https://athena.brynmawr.edu/jupyter/hub/dblank/public/Jupyter%20Notebook%20Users%20Manual.ipynb, notebook formating)

Example table in MD

This	**is**
a	table

Learnt multinomial naive Bayes Model with Smoothing

Word	**Word Class Conditional counts and probs. \n dsdsd**
Beijing	[0.0, 1.0, 0.111, 0.142]
Chinese	[1.0, 5.0, 0.222, 0.428]
Tokyo	[1.0, 0.0, 0.222, 0.0714]
Shanghai	[0.0, 1.0, 0.111, 0.142]
ClassPrior	[1.0, 3.0, 0.25, 0.75]
Japan	[1.0, 0.0, 0.222, 0.071]
Macao	[0.0, 1.0, 0.111, 0.142]

D1 1 Chinese Beijing Chinese D2 1 Chinese Chinese Shanghai D3 1 Chinese Macao D4 0 Tokyo Japan Chinese Testing record D5 0 Chinese Chinese Chinese Tokyo Japan

We are not using smoothing so math for each conditional probability becomes:-

P(word|Class) = number of times "word" occurs in particular class / total number of words in that class

No of words in class -SPAM = 8 No of words in class -HAM = 3

```
P(chinese|Class = SPAM) = 5/ 8 = 0.625
P(chinese|class =HAM) = 1/3= 0.33333
```

Similarly,

```
P(Beijing|Class = SPAM)  = 1/8 = 0.125
P(Beijing|Class = HAM) = 0/3 = 0
```

Word	Pr(HAM)	Pr(SPAM)
chinese	0.3333	0.625
beijing	0	0.125
tokyo	0.3333	0.0
shanghai	0.0	0.125
japan	0.3333	0.0
macao	0.0	0.125

Additionaly we need prior probabilities of classes here we have 3 SPAM and 1 HAM example so P(class= SPAM) = 3/4 = 0.75 P(class = HAM) = 1/4 = 0.25

Now for statement lets calculate probabilitis for each class . D5 0 Chinese Chinese Chinese Tokyo Japan

```
P(class = HAM) = 0.25 *0.3333 * 0.3333 * 0.3333 * 0.3333 * 0.3333 =
0.0010282922839403295
P(class=SPAM) = 0.75 * 0.625 * 0.625 * 0.625 *0*0 = 0
```

So we take the arg max for last calcualtion and we will classify this as HAM i.e. class 0

HW2.3.3 Learn a multinomial naive Bayes model (with no smoothing) for SPAM filtering

Systems test your code first with the Chinese Example and show the resulting model.

Learn a SPAM filtering model from the ENRON dataset provided above. Save the model to file SPAM_Model_MNB.tsv.

Show the top 10 terms alphabetically sortig the words increasing and their corresponding model entries. Write a mapreduce job to accomplish this. Show the bottom 10 terms also.

Run Map Reduce Job to learn a multinomical Naive Model from data

In [99]:
```
# START STUDENT CODE HW231HADOOP_MODEL_SPAM
# STEP 1: make input directory on HDFS
# !hdfs dfs -mkdir -p /user/Xxxxxxx

# STEP2:  upload data to HDFS
!hdfs dfs -rm enronemail_1h.txt
!hdfs dfs -copyFromLocal NaiveBayes/enronemail_1h.txt
# STEP3: Make sure to remove the results directiry
!hdfs dfs -rm -r HW233HADOOP_MODEL_SPAM/model/
# STEP4: Run job

# STEP5: display model (first 10 lines only)



##################### IMPORTANT ##################################
########################################
# Make sure you have the correct paths to the jar file as wel as
the input and output files!!
# make sure to include the -files option. Do ***** NOT ****** pu
t spaces between the file paths!
################################################################
########################################

!hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
  -D mapreduce.job.output.key.comparator.class=org.apache.hadoop
.mapred.lib.KeyFieldBasedComparator \
  -D mapreduce.partition.keycomparator.options="-k1,1"  \
  -files NaiveBayes/mapper_model.py,NaiveBayes/reducer_model.py
\
  -mapper mapper_model.py\
  -reducer reducer_model.py \
  -input enronemail_1h.txt \
  -output HW233HADOOP_MODEL_SPAM/model/\
  -numReduceTasks 1\
  -cmdenv PATH=/opt/anaconda/bin:$PATH

!hdfs dfs -ls HW233HADOOP_MODEL_SPAM/model/
!rm NaiveBayes/SPAM_Model_MNB.tsv
!hdfs dfs -get HW233HADOOP_MODEL_SPAM/model//part-00000 NaiveBay
es/SPAM_Model_MNB.tsv
print "*****First top 10 lines******"
!head -n 10 NaiveBayes/SPAM_Model_MNB.tsv
print "***** bottom 10 lines******"
!tail -n 10 NaiveBayes/SPAM_Model_MNB.tsv
# END STUDENT CODE HW231HADOOP_MODEL_SPAM
```

```
Deleted enronemail_1h.txt
Deleted HW233HADOOP_MODEL_SPAM/model
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.ja
r] /tmp/streamjob3629235698483604612.jar tmpDir=null
17/05/23 02:27:40 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:27:40 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:27:42 INFO mapred.FileInputFormat: Total input path
s to process : 1
17/05/23 02:27:42 INFO mapreduce.JobSubmitter: number of splits
:2
17/05/23 02:27:42 INFO mapreduce.JobSubmitter: Submitting token
s for job: job_1495464826300_0097
17/05/23 02:27:43 INFO impl.YarnClientImpl: Submitted applicati
on application_1495464826300_0097
17/05/23 02:27:43 INFO mapreduce.Job: The url to track the job:
http://quickstart.cloudera:8088/proxy/application_1495464826300
_0097/
17/05/23 02:27:43 INFO mapreduce.Job: Running job: job_14954648
26300_0097
17/05/23 02:27:51 INFO mapreduce.Job: Job job_1495464826300_009
7 running in uber mode : false
17/05/23 02:27:51 INFO mapreduce.Job:  map 0% reduce 0%
17/05/23 02:28:02 INFO mapreduce.Job:  map 50% reduce 0%
17/05/23 02:28:03 INFO mapreduce.Job:  map 100% reduce 0%
17/05/23 02:28:13 INFO mapreduce.Job:  map 100% reduce 100%
17/05/23 02:28:13 INFO mapreduce.Job: Job job_1495464826300_009
7 completed successfully
17/05/23 02:28:13 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=1172006
                FILE: Number of bytes written=2696982
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=217089
                HDFS: Number of bytes written=196151
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots
(ms)=17957
                Total time spent by all reduces in occupied slo
ts (ms)=7057
                Total time spent by all map tasks (ms)=17957
                Total time spent by all reduce tasks (ms)=7057
                Total vcore-seconds taken by all map tasks=1795
7
                Total vcore-seconds taken by all reduce tasks=7
057
                Total megabyte-seconds taken by all map tasks=1
8387968
                Total megabyte-seconds taken by all reduce task
s=7226368
```

HW 2.3.4 Classify Documents using the learnt Multinomial Naive Bayes model using Hadoop Streaming

Classify each Enron email messages using the learnt Naive Bayes classifier (Testing on the training set is bad practice but we will allow that here to simplify the work here).

Write a separate map-reduce job to classify a corpus of documents using a provided/learnt Multinomial Naive Bayes model. A model file consisting of the triples *word\tPr(Word|HAM) \tPr(Word|SPAM)* should be broadcast to the worker nodes using the *-file* command line option when running a Hadoop streaming job. Please write the corresponding mapper and reducer portions of this classifier job.

Note: Map Tasks and map lifecycles

Note that for each chunk in the input data a mapper task is executed. Each mapper task has three phases: a init phase (to initialize variables used down stream in the mapper task or read in data from disk that might be used downstream in the map task); a loop to process each record in the input stream; and a final phase that is executed prior to the map task finishing. A Reduce task goes through a similar lifecycle.

NOTE: on small multiplying small numbers

Multiplying lots of probabilities, which are between 0 and 1, can result in floating-point underflow. Since log(xy) = log(x) + log(y), it is better to perform all computations by summing logs of probabilities rather than multiplying probabilities. Please pay attention to probabilites that are zero! They will need special attention. Count up how many times clasification of a document results in a zero class posterior probabilty for each class and report when using the Enron training set for evaluation.

- Report the performance of your learnt classifier in terms of misclassifcation error rate of your multinomial Naive Bayes Classifier.
  - Error Rate = misclassification rate with respect to a provided set (say training set in this case). It is more formally defined here:
- Let DF represent the evalution set in the following:
  - Err(Model, DF) = |{(X, c(X)) ∈ DF : c(X) != Model(x)}| / |DF|

Where || denotes set cardinality; c(X) denotes the class of the tuple X in DF; and Model(X) denotes the class inferred by the Model “Model”

In this exercise, please complete the following tasks:

- Once again unit test your classifier map reduce job using the Chinese example. Please show a trace of your prediction and classification steps.
- Once you are happy with the results from the Chinese dataset, repeat the process for the Enron dataset.

In [100]:
```
%%writefile NaiveBayes/mapper_classify.py
#!/usr/bin/env python
from NaiveBayesModel import NaiveBayesModel
import sys, re, string, subprocess
import sys, operator, math
import numpy as np
from math import log
from math import exp
# Init mapper phase

# read the MODEL into memory
# The model file resides the local disk (make sure to ship it ho
me from HDFS).
# In the Hadoop command linke be sure to add the follow the -fil
es commmand line option
NBModel = NaiveBayesModel("NaiveBayes/SPAM_Model_MNB.tsv")
#NBModel.printModel()
# define regex for punctuation removal
regex = re.compile('[%s]' % re.escape(string.punctuation))

# inner loop mapper phase: process each record
# input comes from STDIN (standard input)
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    parts = line.split("\t")
    docID, docClass, title = parts[0:3]
    if len(parts) == 4:
        body = parts[3]
    else:
        body = ""
    # use subject and body
    # remove punctuations, only have white-space as delimiter
    emailStr = regex.sub(' ', title.lower() + " " +body.lower())
#replace each punctuation with a space
    emailStr = re.sub( '\s+', ' ', emailStr )            # repla
ce multiple spaces with a space
    # split the line into words
    words = emailStr.split()

# START STUDENT CODE HW231MAPPER_CLASSIFY



    #PrClass0GivenDoc, PrClass1GivenDoc = NBModel.classify(words
)

    #print "Pr(Class=0| Doc=%s) is %6.5f" % (docID, PrClass0Give
nDoc)
    #print "Pr(Class=1| Doc=%s) is %6.5f" % (docID, PrClass1Give
nDoc)

    PrClass0GivenDoc, PrClass1GivenDoc = NBModel.classifyInLogs(
words)
    if PrClass0GivenDoc > PrClass1GivenDoc:
        print "%s\t%s\t%s"%(docID,0,docClass)
    else:
```

```
Overwriting NaiveBayes/mapper_classify.py
```

Run Map Reduce Job to classify data

In [101]:
```
# START STUDENT CODE HW234HADOOP_CLASSIFY_SPAM

#run classifier job
!hdfs dfs -rm enronemail_1h.txt
!hdfs dfs -copyFromLocal NaiveBayes/enronemail_1h.txt
!hdfs dfs -rm -r HW234HADOOP_CLASSIFY_SPAM/classifications/

##################### IMPORTANT ##############################
########################################
# Make sure you have the correct paths to the jar file as wel as
the input and output files!!
# make sure to include the -files option. Do ***** NOT ****** pu
t spaces between the file paths!
##############################################################
########################################

!hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
  -D mapreduce.job.output.key.comparator.class=org.apache.hadoop
.mapred.lib.KeyFieldBasedComparator \
  -D mapreduce.partition.keycomparator.options="-k1,1"  \
  -files NaiveBayes\
  -mapper NaiveBayes/mapper_classify.py\
  -reducer NaiveBayes/reducer_classify.py  \
  -input enronemail_1h.txt \
  -output HW234HADOOP_CLASSIFY_SPAM/classifications/\
  -numReduceTasks 1\
  -cmdenv PATH=/opt/anaconda/bin:$PATH


#Print accuracy

# END STUDENT CODE HW234HADOOP_CLASSIFY_SPAM
```

```
Deleted enronemail_1h.txt
Deleted HW234HADOOP_CLASSIFY_SPAM/classifications
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.ja
r] /tmp/streamjob1864329690012134656.jar tmpDir=null
17/05/23 02:28:39 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:28:40 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:28:42 INFO mapred.FileInputFormat: Total input path
s to process : 1
17/05/23 02:28:43 INFO mapreduce.JobSubmitter: number of splits
:2
17/05/23 02:28:43 INFO mapreduce.JobSubmitter: Submitting token
s for job: job_1495464826300_0098
17/05/23 02:28:43 INFO impl.YarnClientImpl: Submitted applicati
on application_1495464826300_0098
17/05/23 02:28:43 INFO mapreduce.Job: The url to track the job:
http://quickstart.cloudera:8088/proxy/application_1495464826300
_0098/
17/05/23 02:28:43 INFO mapreduce.Job: Running job: job_14954648
26300_0098
17/05/23 02:28:54 INFO mapreduce.Job: Job job_1495464826300_009
8 running in uber mode : false
17/05/23 02:28:54 INFO mapreduce.Job:  map 0% reduce 0%
17/05/23 02:29:05 INFO mapreduce.Job:  map 50% reduce 0%
17/05/23 02:29:06 INFO mapreduce.Job:  map 100% reduce 0%
17/05/23 02:29:14 INFO mapreduce.Job:  map 100% reduce 100%
17/05/23 02:29:14 INFO mapreduce.Job: Job job_1495464826300_009
8 completed successfully
17/05/23 02:29:15 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=2878
                FILE: Number of bytes written=357658
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=217089
                HDFS: Number of bytes written=2806
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots
(ms)=18934
                Total time spent by all reduces in occupied slo
ts (ms)=5865
                Total time spent by all map tasks (ms)=18934
                Total time spent by all reduce tasks (ms)=5865
                Total vcore-seconds taken by all map tasks=1893
4
                Total vcore-seconds taken by all reduce tasks=5
865
                Total megabyte-seconds taken by all map tasks=1
9388416
                Total megabyte-seconds taken by all reduce task
s=6005760
```

Display the accuracy measure

In [102]:
```
# START STUDENT CODE HW234HADOOP_CLASSIFY_RESULTS
!hdfs dfs -cat HW234HADOOP_CLASSIFY_SPAM/classifications/part-00
000

# END STUDENT CODE HW234HADOOP_CLASSIFY_RESULTS
```

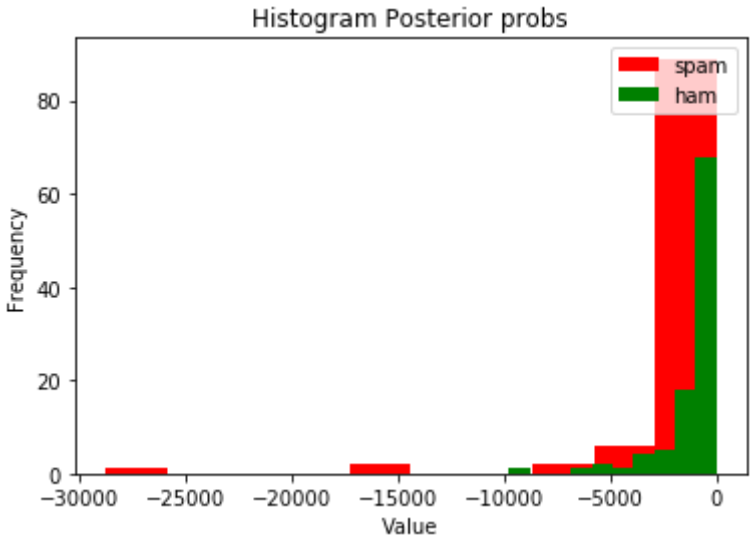
PREDICTION	LABEL	DOCID
0	0	0001.1999-12-10.farmer
0	0	0001.1999-12-10.kaminski
0	0	0001.2000-01-17.beck
0	0	0001.2000-06-06.lokay
0	0	0001.2001-02-07.kitchen
0	0	0001.2001-04-02.williams
0	0	0002.1999-12-13.farmer
0	0	0002.2001-02-07.kitchen
1	1	0002.2001-05-25.SA_and_HP
1	1	0002.2003-12-18.GP
1	1	0002.2004-08-01.BG
0	0	0003.1999-12-10.kaminski
0	0	0003.1999-12-14.farmer
0	0	0003.2000-01-17.beck
0	0	0003.2001-02-08.kitchen
1	1	0003.2003-12-18.GP
1	1	0003.2004-08-01.BG
0	0	0004.1999-12-10.kaminski
0	0	0004.1999-12-14.farmer
0	0	0004.2001-04-02.williams
1	1	0004.2001-06-12.SA_and_HP
1	1	0004.2004-08-01.BG
0	0	0005.1999-12-12.kaminski
0	0	0005.1999-12-14.farmer
0	0	0005.2000-06-06.lokay
0	0	0005.2001-02-08.kitchen
1	1	0005.2001-06-23.SA_and_HP
1	1	0005.2003-12-18.GP
0	0	0006.1999-12-13.kaminski
0	0	0006.2001-02-08.kitchen
0	0	0006.2001-04-03.williams
1	1	0006.2001-06-25.SA_and_HP
1	1	0006.2003-12-18.GP
1	1	0006.2004-08-01.BG
0	0	0007.1999-12-13.kaminski
0	0	0007.1999-12-14.farmer
0	0	0007.2000-01-17.beck
0	0	0007.2001-02-09.kitchen
1	1	0007.2003-12-18.GP
1	1	0007.2004-08-01.BG
0	0	0008.2001-02-09.kitchen
1	1	0008.2001-06-12.SA_and_HP
1	1	0008.2001-06-25.SA_and_HP
1	1	0008.2003-12-18.GP
1	1	0008.2004-08-01.BG
0	0	0009.1999-12-13.kaminski
0	0	0009.1999-12-14.farmer
0	0	0009.2000-06-07.lokay
0	0	0009.2001-02-09.kitchen
1	1	0009.2001-06-26.SA_and_HP
1	1	0009.2003-12-18.GP
0	0	0010.1999-12-14.farmer
0	0	0010.1999-12-14.kaminski
0	0	0010.2001-02-09.kitchen
1	1	0010.2001-06-28.SA_and_HP
1	1	0010.2003-12-18.GP
1	1	0010.2004-08-01.BG
0	0	0011.1999-12-14.farmer

Plot a histogram of the posterior probabilities

Plot a histogram of the posterior probabilities (i.e., Pr(Class|Doc)) for each class over the ENRON training set. Summarize what you see.

In [103]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import re
import string
import numpy as np
# START STUDENT CODE HW234PLOT
from NaiveBayesModel import NaiveBayesModel
NBModel = NaiveBayesModel("NaiveBayes/SPAM_Model_MNB.tsv")
regex = re.compile('[%s]' % re.escape(string.punctuation))
class0 =[]
class1 = []
with open("NaiveBayes/enronemail_1h.txt") as f:

  for line in f.readlines():

    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    parts = line.split("\t")
    docID, docClass, title = parts[0:3]
    if len(parts) == 4:
        body = parts[3]
    else:
        body = ""
    # use subject and body
    # remove punctuations, only have white-space as delimiter
    emailStr = regex.sub(' ', title.lower() + " " +body.lower())
#replace each punctuation with a space
    emailStr = re.sub( '\s+', ' ', emailStr )            # repla
ce multiple spaces with a space
    # split the line into words
    words = emailStr.split()

    PrClass0GivenDoc, PrClass1GivenDoc = NBModel.classifyInLogs(
words)

    class0.append( PrClass0GivenDoc)
    class1.append(PrClass1GivenDoc)
cls0 = np.asarray(class0)
cls0[np.where( cls0 == float("-inf"))] = 0
cls1 = np.asarray(class1)
cls1[np.where( cls1 == float("-inf"))] = 0
plt.hist(cls1,color ='red',label ='spam')
plt.hist(cls0,color ='green',label='ham')
plt.title("Histogram Posterior probs")
plt.legend(loc='upper right')
plt.xlabel("Value")
plt.ylabel("Frequency")

print "No of SPAM with 0 posterior probs" ,len(np.where(cls1 ==
0)[0])
print "No of HAM with 0 posterior probs",len(np.where(cls0 == 0)
[0])
# END STUDENT CODE HW234PLOT
```

```
No of SPAM with 0 posterior probs 56
No of HAM with 0 posterior probs 44
```



- the NB posterior probability is:

$$P(Y = y_k \mid X_1, X_2, \ldots, X_n) = \frac{P(Y = y_k)\prod_i P(X_i \mid Y = y_k)}{\sum_j P(Y = y_j)\prod_i P(X_i \mid Y = y_j)}$$

if denominator becomes 0 then posteriar probability is negative infinity or even 0 for practicle purpose.

Now training data has 56 HAM and 44 SPAM emails. IF we look at the above result, posterior proabilities for SPAM class then 56 of them have -inf or 0 probability. Similary posterior probability for HAM class 44 of them have -inf or 0 probability. This can happen only if denominator becomes 0. So there are words which belong to strictly one class.

Also accuracy is 100% this could be mainly due to using same data for traning and validations. NB classifier is overconfident i.e. proabilities of training set are very high but errors on validation will be high too.

Add -1 or Laplace smoothing helps to reduce this issue. But considering the results it make sense to call this algorithm "Naive"

HW2.4 Use Laplace plus-one smoothing

Repeat HW2.3 with the following modification: use Laplace plus-one smoothing. **Please replace code tags like HW231HADOOP_CLASSIFY with HW241HADOOP_CLASSIFY through out for your submission.** In addition, compare the misclassifcation error rates for 2.3 versus 2.4 and explain the differences.

cut and paste MULTIPLE CELLS

NOTE for **Jupyter lovers**, you can now cut and paste MULTIPLE CELLS: press the ESC key and the press the SHIFT key in conjunction with UP or DOWN arrow key to select the cells you wish to copy (or delete). Then press ESC-C (to copy). Then select the cell where you wish to copy the cells (in the buffer) and press ESC-V to insert the cells in the buffer below the selected cell.

HW2.4.1 Learn a Multinomial Naive Bayes model on a small dataset (Chinese dataset: 5 documents)

Modelling Phase

In [104]:
```
%%writefile NaiveBayes/mapper_model.py
#!/usr/bin/env python
import sys, re, string

# Init mapper phase
# define regex for punctuation removal
regex = re.compile('[%s]' % re.escape(string.punctuation))

# inner loop mapper phase: process each record
# input comes from STDIN (standard input)

for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    # use subject and body

    parts = line.split("\t")
    docID, docClass, title = parts[0:3]
    if len(parts) == 4:
        body = parts[3]
    else:
        body = ""

    # remove punctuations, only have white-space as delimiter
    regex = re.compile('[%s]' % re.escape(string.punctuation))
    emailStr = regex.sub(' ', title.lower() + " " +body.lower())
#replace each punctuation with a space
    emailStr = re.sub( '\s+', ' ', emailStr )            # repla
ce multiple spaces with a space
    # split the line into words
    words = emailStr.split()

# START STUDENT CODE HW241MAPPER_MODEL

# define regex for punctuation removal

# increase counters
# write the results to STDOUT (standard output);
# what we output here will be the input for the
# Reduce step, i.e. the input for reducer.py
#
# tab-delimited; the trivial word count is 1

    for w in words:
        print "%s\t%d\t%d\t%s"%(w,1,int(docClass),docID)# w, "\

# END STUDENT CODE HW241MAPPER_MODEL
```

```
Overwriting NaiveBayes/mapper_model.py
```

In [105]:
```
%%writefile NaiveBayes/reducer_model.py
#!/usr/bin/env python
from operator import itemgetter
import sys, operator
import numpy as np
from  collections import OrderedDict

# START STUDENT CODE HW241REDUCER_MODEL

current_word = None
smooth_factor = 1 # add 1 smoothing
current_count = [smooth_factor, smooth_factor]
msgIDs = {}
word = None
wordcount = {}

# input comes from STDIN
# remove leading and trailing whitespace

# parse the input we got from mapper.py

# convert count and spam flag (currently a string) to int


# handle msgID - store all IDs as we don't have too much
# not the best way to get prior, a two-level MapReduce jobs (ID
- word) would be optimal

# calculate NB parameters, and write the dictionary to a file fo
r the classification job
# prior probabilities

# conditional probability
# input comes from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # parse the input we got from mapper.py
    word, count, isSpam, msgID = line.split('\t', 3)


    # convert count and spam flag (currently a string) to int
    try:
        count = int(count)
        isSpam = int(isSpam)
    except ValueError:
        # count was not a number, so silently
        # ignore/discard this line
        continue

    # handle msgID - store all IDs as we don't have too much
    # not the best way to get prior, a two-level MapReduce jobs
(ID - word) would be optimal
    if msgID not in msgIDs:
        msgIDs[msgID] = isSpam

    # this IF-switch only works because Hadoop sorts map output
```

```
Overwriting NaiveBayes/reducer_model.py
```

In [106]:
```
%%writefile NaiveBayes/mapper_classify.py
#!/usr/bin/env python
from NaiveBayesModel import NaiveBayesModel
import sys, re, string, subprocess
import sys, operator, math
import numpy as np
from math import log
from math import exp
# Init mapper phase

# read the MODEL into memory
# The model file resides the local disk (make sure to ship it ho
me from HDFS).
# In the Hadoop command linke be sure to add the follow the -fil
es commmand line option
NBModel = NaiveBayesModel("NaiveBayes/SPAM_Model_MNB.tsv")
#NBModel.printModel()
# define regex for punctuation removal
regex = re.compile('[%s]' % re.escape(string.punctuation))

# inner loop mapper phase: process each record
# input comes from STDIN (standard input)
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    parts = line.split("\t")
    docID, docClass, title = parts[0:3]
    if len(parts) == 4:
        body = parts[3]
    else:
        body = ""
    # use subject and body
    # remove punctuations, only have white-space as delimiter
    emailStr = regex.sub(' ', title.lower() + " " +body.lower())
#replace each punctuation with a space
    emailStr = re.sub( '\s+', ' ', emailStr )            # repla
ce multiple spaces with a space
    # split the line into words
    words = emailStr.split()

# START STUDENT CODE HW241MAPPER_CLASSIFY



    #PrClass0GivenDoc, PrClass1GivenDoc = NBModel.classify(words
)

    #print "Pr(Class=0| Doc=%s) is %6.5f" % (docID, PrClass0Give
nDoc)
    #print "Pr(Class=1| Doc=%s) is %6.5f" % (docID, PrClass1Give
nDoc)

    PrClass0GivenDoc, PrClass1GivenDoc = NBModel.classifyInLogs(
words)
    if PrClass0GivenDoc > PrClass1GivenDoc:
        print "%s\t%s\t%s"%(docID,0,docClass)
    else:
```

```
Overwriting NaiveBayes/mapper_classify.py
```

In [107]:
```
%%writefile NaiveBayes/reducer_classify.py
#!/usr/bin/env python
from operator import itemgetter
import sys, operator, math
import numpy as np

numberOfRecords = 0
NumberOfMisclassifications=0
classificationAccurary = 0

# START STUDENT CODE HW231REDUCER_CLASSIFY

# input comes from STDIN
print "%s\t%s\t%s"%("PREDICTION","LABEL","DOCID")
for line in sys.stdin:
    # remove leading and trailing whitespace
    #line = line.strip()
    # split the line into words
    parts = line.split("\t")
    docID, preds, label = parts[0:3]
    print "%s\t%s\t%s"%( preds, label,docID)
    if int(preds) != int(label):
        NumberOfMisclassifications +=1
    numberOfRecords+=1
classificationAccurary = float(float(numberOfRecords-NumberOfMis
classifications) / float(numberOfRecords)) * float(100)
print 'Multinomial Naive Bayes Classifier Results are Total Reco
rds: %d,Mis-classes: %d,Accuracy: %2.5f' % (numberOfRecords, Num
berOfMisclassifications, classificationAccurary)
# END STUDENT CODE HW231REDUCER_CLASSIFY
```

```
Overwriting NaiveBayes/reducer_classify.py
```

In [108]:
```
!chmod a+x NaiveBayes/mapper_model.py
!chmod a+x NaiveBayes/reducer_model.py
!chmod a+x NaiveBayes/mapper_classify.py
!chmod a+x NaiveBayes/reducer_classify.py
!chmod a+x NaiveBayes/NaiveBayesModel.py
```

Chinese dataset

In [109]:
```
# START STUDENT CODE HW231HADOOP_MODEL
# STEP 1: make input directory on HDFS
#!hdfs dfs -mkdir -p /user/Xxxxxxx

# STEP2:  upload data to HDFS
!hdfs dfs -rm chineseExample_train.txt
!rm NaiveBayes/chineseExample_train.txt
!head -n 4 NaiveBayes/chineseExample.txt  > NaiveBayes/chineseEx
ample_train.txt
!hdfs dfs -copyFromLocal NaiveBayes/chineseExample_train.txt
!hdfs dfs -rm -r HW241HADOOP_CHINESE_UNIT_TEST/model/

###################### IMPORTANT ##################################
########################################
# Make sure you have the correct paths to the jar file as wel as
the input and output files!!
# make sure to include the -files option. Do ***** NOT ****** pu
t spaces between the file paths!
##########################################################################
########################################

!hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
  -D mapreduce.job.output.key.comparator.class=org.apache.hadoop
.mapred.lib.KeyFieldBasedComparator \
  -D mapreduce.partition.keycomparator.options="-k1,1"  \
  -files NaiveBayes/mapper_model.py,NaiveBayes/reducer_model.py
\
  -mapper mapper_model.py\
  -reducer reducer_model.py \
  -input chineseExample_train.txt\
  -output HW241HADOOP_CHINESE_UNIT_TEST/model/\
  -numReduceTasks 1\
  -cmdenv PATH=/opt/anaconda/bin:$PATH
```

```
Deleted chineseExample_train.txt
Deleted HW241HADOOP_CHINESE_UNIT_TEST/model
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.ja
r] /tmp/streamjob9010610586522889029.jar tmpDir=null
17/05/23 02:29:41 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:29:42 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:29:43 INFO mapred.FileInputFormat: Total input path
s to process : 1
17/05/23 02:29:43 INFO mapreduce.JobSubmitter: number of splits
:2
17/05/23 02:29:44 INFO mapreduce.JobSubmitter: Submitting token
s for job: job_1495464826300_0099
17/05/23 02:29:44 INFO impl.YarnClientImpl: Submitted applicati
on application_1495464826300_0099
17/05/23 02:29:44 INFO mapreduce.Job: The url to track the job:
http://quickstart.cloudera:8088/proxy/application_1495464826300
_0099/
17/05/23 02:29:44 INFO mapreduce.Job: Running job: job_14954648
26300_0099
17/05/23 02:29:55 INFO mapreduce.Job: Job job_1495464826300_009
9 running in uber mode : false
17/05/23 02:29:55 INFO mapreduce.Job:  map 0% reduce 0%
17/05/23 02:30:05 INFO mapreduce.Job:  map 50% reduce 0%
17/05/23 02:30:06 INFO mapreduce.Job:  map 100% reduce 0%
17/05/23 02:30:14 INFO mapreduce.Job:  map 100% reduce 100%
17/05/23 02:30:14 INFO mapreduce.Job: Job job_1495464826300_009
9 completed successfully
17/05/23 02:30:14 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=188
                FILE: Number of bytes written=353388
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=391
                HDFS: Number of bytes written=275
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots
(ms)=16182
                Total time spent by all reduces in occupied slo
ts (ms)=5489
                Total time spent by all map tasks (ms)=16182
                Total time spent by all reduce tasks (ms)=5489
                Total vcore-seconds taken by all map tasks=1618
2
                Total vcore-seconds taken by all reduce tasks=5
489
                Total megabyte-seconds taken by all map tasks=1
6570368
                Total megabyte-seconds taken by all reduce task
s=5620736
```

In [110]:
```
#put results file into local directories

!hdfs dfs -ls HW241HADOOP_CHINESE_UNIT_TEST/model/
!rm NaiveBayes/SPAM_Model_MNB.tsv
!hdfs dfs -get HW241HADOOP_CHINESE_UNIT_TEST/model/part-00000 Na
iveBayes/SPAM_Model_MNB.tsv
```

```
Found 2 items
-rw-r--r--   1 root supergroup          0 2017-05-23 02:30 HW24
1HADOOP_CHINESE_UNIT_TEST/model/_SUCCESS
-rw-r--r--   1 root supergroup        275 2017-05-23 02:30 HW24
1HADOOP_CHINESE_UNIT_TEST/model/part-00000
```

In [111]:
```
 # START STUDENT CODE HW241HADOOP_CLASSIFY
!hdfs dfs -rm chineseExample_test.txt
!rm NaiveBayes/chineseExample_test.txt
!tail -n 1 NaiveBayes/chineseExample.txt  > NaiveBayes/chineseEx
ample_test.txt
!hdfs dfs -copyFromLocal NaiveBayes/chineseExample_test.txt
!hdfs dfs -rm -r HW241HADOOP_CHINESE_UNIT_TEST/classifications/

!hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
    -D mapreduce.job.output.key.comparator.class=org.apache.hado
op.mapred.lib.KeyFieldBasedComparator \
    -D mapreduce.partition.keycomparator.options="-k1,1"  \
    -files NaiveBayes\
    -mapper NaiveBayes/mapper_classify.py\
    -reducer NaiveBayes/reducer_classify.py  \
    -input chineseExample_test.txt \
    -output HW241HADOOP_CHINESE_UNIT_TEST/classifications/\
    -numReduceTasks 1\
    -cmdenv PATH=/opt/anaconda/bin:$PATH




  # END STUDENT CODE HW231HADOOP_CLASSIFY
```

```
Deleted chineseExample_test.txt
Deleted HW241HADOOP_CHINESE_UNIT_TEST/classifications
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.ja
r] /tmp/streamjob5374346438083772412.jar tmpDir=null
17/05/23 02:30:41 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:30:41 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:30:44 INFO mapred.FileInputFormat: Total input path
s to process : 1
17/05/23 02:30:44 INFO mapreduce.JobSubmitter: number of splits
:2
17/05/23 02:30:44 INFO mapreduce.JobSubmitter: Submitting token
s for job: job_1495464826300_0100
17/05/23 02:30:45 INFO impl.YarnClientImpl: Submitted applicati
on application_1495464826300_0100
17/05/23 02:30:45 INFO mapreduce.Job: The url to track the job:
http://quickstart.cloudera:8088/proxy/application_1495464826300
_0100/
17/05/23 02:30:45 INFO mapreduce.Job: Running job: job_14954648
26300_0100
17/05/23 02:30:55 INFO mapreduce.Job: Job job_1495464826300_010
0 running in uber mode : false
17/05/23 02:30:55 INFO mapreduce.Job:  map 0% reduce 0%
17/05/23 02:31:07 INFO mapreduce.Job:  map 50% reduce 0%
17/05/23 02:31:08 INFO mapreduce.Job:  map 100% reduce 0%
17/05/23 02:31:15 INFO mapreduce.Job:  map 100% reduce 100%
17/05/23 02:31:16 INFO mapreduce.Job: Job job_1495464826300_010
0 completed successfully
17/05/23 02:31:17 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=15
                FILE: Number of bytes written=351962
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=294
                HDFS: Number of bytes written=129
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots
(ms)=19047
                Total time spent by all reduces in occupied slo
ts (ms)=6129
                Total time spent by all map tasks (ms)=19047
                Total time spent by all reduce tasks (ms)=6129
                Total vcore-seconds taken by all map tasks=1904
7
                Total vcore-seconds taken by all reduce tasks=6
129
                Total megabyte-seconds taken by all map tasks=1
9504128
                Total megabyte-seconds taken by all reduce task
s=6276096
```

In [112]:
```
# START STUDENT CODE HW231HADOOP_CLASSIFY_RESULTS
# (you may or may not need to add anything else in this block de
pending on your implementation)

!hdfs dfs -cat HW241HADOOP_CHINESE_UNIT_TEST/classifications/par
t-00000

 # END STUDENT CODE HW231HADOOP_CLASSIFY_RESULTS
```

```
PREDICTION      LABEL   DOCID
1       0
        D5
Multinomial Naive Bayes Classifier Results are Total Records: 1
,Mis-classes: 1,Accuracy: 0.00000
```

In [113]:
```
!head -n 4 NaiveBayes/chineseExample.txt  |NaiveBayes/mapper_mod
el.py| sort -k1,1 |NaiveBayes/reducer_model.py>NaiveBayes/NaiveB
ayes.txt
```

In [114]:
```
# START STUDENT CODE HW231HADOOP_CHINESE_UNIT_TEST
!head -n 4 NaiveBayes/chineseExample.txt  |NaiveBayes/mapper_mod
el.py| sort -k1,1 |NaiveBayes/reducer_model.py>NaiveBayes/NaiveB
ayes.txt
import unittest
import re
class Testchinese(unittest.TestCase):
    def test_model(self):
        NBModel = NaiveBayesModel("NaiveBayes/NaiveBayes.txt")
        self.assertEqual([1,3,0.25,0.75], map(float, NBModel.mod
el["ClassPriors"]))
        self.assertAlmostEqual(0.222,  float(NBModel.model["chin
ese"][2]),3)
        self.assertAlmostEqual(0.14285,  float(NBModel.model["ma
cao"][3]),3)

    def test_classification(self):
        NBModel = NaiveBayesModel("NaiveBayes/NaiveBayes.txt")
        line = 'D5      0       Chinese Chinese Chinese Tokyo Ja
pan'.strip()
         # split the line into words
        parts = line.split("\t")
        docID, docClass, title = parts[0:3]
        if len(parts) == 4:
              body = parts[3]
        else:
              body = ""
        emailStr = regex.sub(' ', title.lower() + " " +body.lowe
r()) #replace each punctuation with a space
        emailStr = re.sub( '\s+', ' ', emailStr )            # r
eplace multiple spaces with a space
        words = emailStr.split()
        PrClass0GivenDoc, PrClass1GivenDoc = NBModel.classify(wo
rds)
        self.assertAlmostEqual(0.0001354,  PrClass0GivenDoc,2)
        self.assertAlmostEqual(0.00291341,  PrClass0GivenDoc,2)

suite = unittest.TestLoader().loadTestsFromTestCase(Testchinese)
unittest.TextTestRunner(verbosity=2).run(suite)
 # END STUDENT CODE HW231HADOOP_CHINESE_UNIT_TEST
```

```
test_classification (__main__.Testchinese) ... ok
test_model (__main__.Testchinese) ... ok

----------------------------------------------------------------
-------
Ran 2 tests in 0.010s

OK
```

Out[114]: <unittest.runner.TextTestResult run=2 errors=0 failures=0>

HW2.4.2 Learn a multinomial naive Bayes model (with smoothing) by hand

```
Answer

D1    1    Chinese Beijing    Chinese
D2    1    Chinese Chinese    Shanghai
D3    1    Chinese    Macao
D4    0    Tokyo Japan    Chinese
```

Testing record D5 0 Chinese Chinese Chinese Tokyo Japan

We are using smoothing so math for each conditional probability becomes:-

P(word|Class) = number of times "word" occurs in particular class + 1 / total number of words in that class + Vocabularize size

No of words in class -SPAM = 8 No of words in class -HAM = 3 dictionary size = 6

```
P(chinese|Class = SPAM) = 5 + 1/ (8 + 6) = 6/14
P(chinese|class =HAM) = 1 +1 /(3 + 6)= 2/9
```

Similarly,

```
P(Beijing|Class = SPAM)  = 1+1/8 + 6= 2/9 = 0.2222
P(Beijing|Class = HAM) = 0 +1 /3 +6 = 1/9 = 0.1111
```

Word	Pr(HAM)	Pr(SPAM)
chinese	0.2222	0.4285
beijing	0.1111	0.142857
tokyo	0.222222	0.071428
shanghai	0.1111	0.14285
japan	0.22222	0.07142
macao	0.1111	0.142857

Additionaly we need prior probabilities of classes here we have 3 SPAM and 1 HAM example so P(class= SPAM) = 3/4 = 0.75 P(class = HAM) = 1/4 = 0.25

Now for statement lets calculate probabilitis for each class . D5 0 Chinese Chinese Chinese Tokyo Japan

```
P(class = HAM) = 0.25 *0.2222 * 0.2222 * 0.2222 * 0.2222 * 0.2222 =
0.0001354129756629241
P(class=SPAM) = 0.75 * 0.4285 * 0.4285 *  0.4285 *0.2222*0.2222 = 0.
00291341
```

So we take the arg max for last calcualtion and we will classify this as HAM i.e. class is SPAM.

In [ ]:

HW2.4.3 Learn a multinomial naive Bayes model (with no smoothing) for SPAM filtering

Run Map Reduce Job to learn a multinomical Naive Model from data

In [115]:
```
# START STUDENT CODE HW241HADOOP_MODEL_SPAM
# STEP 1: make input directory on HDFS
# !hdfs dfs -mkdir -p /user/Xxxxxxx

# STEP2:  upload data to HDFS
!hdfs dfs -rm enronemail_1h.txt
!hdfs dfs -copyFromLocal NaiveBayes/enronemail_1h.txt
# STEP3: Make sure to remove the results directiry
!hdfs dfs -rm -r HW243HADOOP_MODEL_SPAM/model/
# STEP4: Run job

# STEP5: display model (first 10 lines only)



##################### IMPORTANT ##################################
########################################
# Make sure you have the correct paths to the jar file as wel as
the input and output files!!
# make sure to include the -files option. Do ***** NOT ****** pu
t spaces between the file paths!
##################################################################
########################################

!hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
  -D mapreduce.job.output.key.comparator.class=org.apache.hadoop
.mapred.lib.KeyFieldBasedComparator \
  -D mapreduce.partition.keycomparator.options="-k1,1"  \
  -files NaiveBayes/mapper_model.py,NaiveBayes/reducer_model.py
\
  -mapper mapper_model.py\
  -reducer reducer_model.py \
  -input enronemail_1h.txt \
  -output HW243HADOOP_MODEL_SPAM/model/\
  -numReduceTasks 1\
  -cmdenv PATH=/opt/anaconda/bin:$PATH

!hdfs dfs -ls HW243HADOOP_MODEL_SPAM/model/
!rm NaiveBayes/SPAM_Model_MNB.tsv
!hdfs dfs -get HW243HADOOP_MODEL_SPAM/model//part-00000 NaiveBay
es/SPAM_Model_MNB.tsv
print "*****First top 10 lines******"
!head -n 10 NaiveBayes/SPAM_Model_MNB.tsv
print "***** bottom 10 lines******"
!tail -n 10 NaiveBayes/SPAM_Model_MNB.tsv
# END STUDENT CODE HW241HADOOP_MODEL_SPAM
```

```
Deleted enronemail_1h.txt
Deleted HW243HADOOP_MODEL_SPAM/model
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.ja
r] /tmp/streamjob2627401025089332207.jar tmpDir=null
17/05/23 02:31:39 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:31:40 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:31:41 INFO mapred.FileInputFormat: Total input path
s to process : 1
17/05/23 02:31:41 INFO mapreduce.JobSubmitter: number of splits
:2
17/05/23 02:31:41 INFO mapreduce.JobSubmitter: Submitting token
s for job: job_1495464826300_0101
17/05/23 02:31:42 INFO impl.YarnClientImpl: Submitted applicati
on application_1495464826300_0101
17/05/23 02:31:42 INFO mapreduce.Job: The url to track the job:
http://quickstart.cloudera:8088/proxy/application_1495464826300
_0101/
17/05/23 02:31:42 INFO mapreduce.Job: Running job: job_14954648
26300_0101
17/05/23 02:31:52 INFO mapreduce.Job: Job job_1495464826300_010
1 running in uber mode : false
17/05/23 02:31:52 INFO mapreduce.Job:  map 0% reduce 0%
17/05/23 02:32:04 INFO mapreduce.Job:  map 50% reduce 0%
17/05/23 02:32:05 INFO mapreduce.Job:  map 100% reduce 0%
17/05/23 02:32:14 INFO mapreduce.Job:  map 100% reduce 100%
17/05/23 02:32:15 INFO mapreduce.Job: Job job_1495464826300_010
1 completed successfully
17/05/23 02:32:15 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=1172006
                FILE: Number of bytes written=2696982
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=217089
                HDFS: Number of bytes written=255636
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots
(ms)=18561
                Total time spent by all reduces in occupied slo
ts (ms)=7215
                Total time spent by all map tasks (ms)=18561
                Total time spent by all reduce tasks (ms)=7215
                Total vcore-seconds taken by all map tasks=1856
1
                Total vcore-seconds taken by all reduce tasks=7
215
                Total megabyte-seconds taken by all map tasks=1
9006464
                Total megabyte-seconds taken by all reduce task
s=7388160
```

HW 2.3.4 Classify Documents using the learnt Multinomial Naive Bayes model using Hadoop Streaming

Run Map Reduce Job to classify data

In [116]:
```
# START STUDENT CODE HW244HADOOP_CLASSIFY_SPAM

#run classifier job
!hdfs dfs -rm enronemail_1h.txt
!hdfs dfs -copyFromLocal NaiveBayes/enronemail_1h.txt
!hdfs dfs -rm -r HW244HADOOP_CLASSIFY_SPAM/classifications/

##################### IMPORTANT ##############################
########################################
# Make sure you have the correct paths to the jar file as wel as
the input and output files!!
# make sure to include the -files option. Do ***** NOT ****** pu
t spaces between the file paths!
##############################################################
########################################

!hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
  -D mapreduce.job.output.key.comparator.class=org.apache.hadoop
.mapred.lib.KeyFieldBasedComparator \
  -D mapreduce.partition.keycomparator.options="-k1,1"  \
  -files NaiveBayes\
  -mapper NaiveBayes/mapper_classify.py\
  -reducer NaiveBayes/reducer_classify.py  \
  -input enronemail_1h.txt \
  -output HW244HADOOP_CLASSIFY_SPAM/classifications/\
  -numReduceTasks 1\
  -cmdenv PATH=/opt/anaconda/bin:$PATH


#Print accuracy

# END STUDENT CODE HW234HADOOP_CLASSIFY_SPAM
```

```
Deleted enronemail_1h.txt
rm: `HW244HADOOP_CLASSIFY_SPAM/classifications/': No such file
or directory
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.ja
r] /tmp/streamjob5069501834573162601.jar tmpDir=null
17/05/23 02:32:43 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:32:43 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:32:48 INFO mapred.FileInputFormat: Total input path
s to process : 1
17/05/23 02:32:48 INFO mapreduce.JobSubmitter: number of splits
:2
17/05/23 02:32:49 INFO mapreduce.JobSubmitter: Submitting token
s for job: job_1495464826300_0102
17/05/23 02:32:49 INFO impl.YarnClientImpl: Submitted applicati
on application_1495464826300_0102
17/05/23 02:32:49 INFO mapreduce.Job: The url to track the job:
http://quickstart.cloudera:8088/proxy/application_1495464826300
_0102/
17/05/23 02:32:49 INFO mapreduce.Job: Running job: job_14954648
26300_0102
17/05/23 02:33:00 INFO mapreduce.Job: Job job_1495464826300_010
2 running in uber mode : false
17/05/23 02:33:00 INFO mapreduce.Job:  map 0% reduce 0%
17/05/23 02:33:12 INFO mapreduce.Job:  map 50% reduce 0%
17/05/23 02:33:13 INFO mapreduce.Job:  map 100% reduce 0%
17/05/23 02:33:21 INFO mapreduce.Job:  map 100% reduce 100%
17/05/23 02:33:21 INFO mapreduce.Job: Job job_1495464826300_010
2 completed successfully
17/05/23 02:33:21 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=2878
                FILE: Number of bytes written=357658
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=217089
                HDFS: Number of bytes written=2897
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots
(ms)=20427
                Total time spent by all reduces in occupied slo
ts (ms)=6406
                Total time spent by all map tasks (ms)=20427
                Total time spent by all reduce tasks (ms)=6406
                Total vcore-seconds taken by all map tasks=2042
7
                Total vcore-seconds taken by all reduce tasks=6
406
                Total megabyte-seconds taken by all map tasks=2
0917248
                Total megabyte-seconds taken by all reduce task
```

Display the accuracy measure

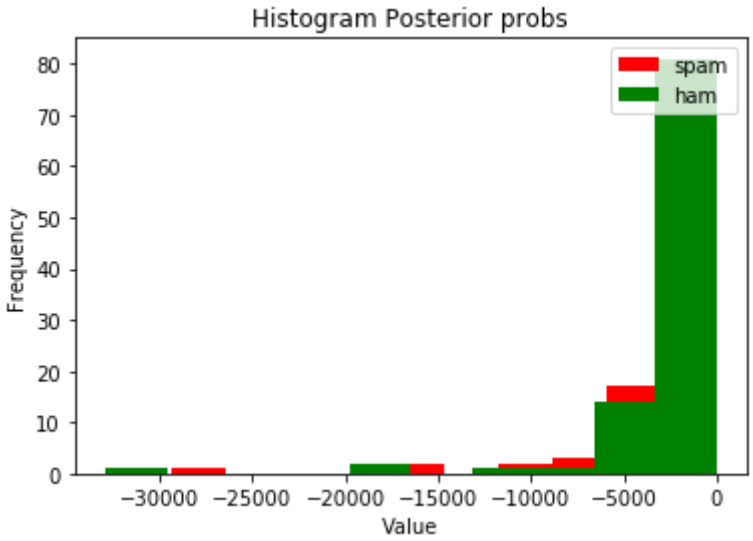
In [117]:
```
# START STUDENT CODE HW244HADOOP_CLASSIFY_RESULTS
!hdfs dfs -cat HW244HADOOP_CLASSIFY_SPAM/classifications/part-00
000

# END STUDENT CODE HW244HADOOP_CLASSIFY_RESULTS
```

PREDICTION	LABEL	DOCID
0	0	0001.1999-12-10.farmer
0	0	0001.1999-12-10.kaminski
0	0	0001.2000-01-17.beck
0	0	0001.2000-06-06.lokay
0	0	0001.2001-02-07.kitchen
0	0	0001.2001-04-02.williams
0	0	0002.1999-12-13.farmer
0	0	0002.2001-02-07.kitchen
1	1	0002.2001-05-25.SA_and_HP
1	1	0002.2003-12-18.GP
1	1	0002.2004-08-01.BG
0	0	0003.1999-12-10.kaminski
0	0	0003.1999-12-14.farmer
0	0	0003.2000-01-17.beck
0	0	0003.2001-02-08.kitchen
1	1	0003.2003-12-18.GP
1	1	0003.2004-08-01.BG
0	0	0004.1999-12-10.kaminski
0	0	0004.1999-12-14.farmer
0	0	0004.2001-04-02.williams
1	1	0004.2001-06-12.SA_and_HP
1	1	0004.2004-08-01.BG
0	0	0005.1999-12-12.kaminski
0	0	0005.1999-12-14.farmer
0	0	0005.2000-06-06.lokay
0	0	0005.2001-02-08.kitchen
1	1	0005.2001-06-23.SA_and_HP
1	1	0005.2003-12-18.GP
0	0	0006.1999-12-13.kaminski

Plot a histogram of the posterior probabilities

In [118]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import re
import string
# START STUDENT CODE HW234PLOT
from NaiveBayesModel import NaiveBayesModel
NBModel = NaiveBayesModel("NaiveBayes/SPAM_Model_MNB.tsv")
regex = re.compile('[%s]' % re.escape(string.punctuation))
class0 =[]
class1 = []
with open("NaiveBayes/enronemail_1h.txt") as f:

  for line in f.readlines():

    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    parts = line.split("\t")
    docID, docClass, title = parts[0:3]
    if len(parts) == 4:
        body = parts[3]
    else:
        body = ""
    # use subject and body
    # remove punctuations, only have white-space as delimiter
    emailStr = regex.sub(' ', title.lower() + " " +body.lower())
#replace each punctuation with a space
    emailStr = re.sub( '\s+', ' ', emailStr )            # repla
ce multiple spaces with a space
    # split the line into words
    words = emailStr.split()

    PrClass0GivenDoc, PrClass1GivenDoc = NBModel.classifyInLogs(
words)
    class0.append( float(PrClass0GivenDoc))
    class1.append(float(PrClass1GivenDoc))

cls0 = np.asarray(class0)
#cls0[np.where( cls0 == float("-inf"))] = 0
cls1 = np.asarray(class1)
#cls1[np.where( cls1 == float("-inf"))] = 0
plt.hist(cls1,color ='red',label ='spam')
plt.hist(cls0,color ='green',label='ham')
plt.title("Histogram Posterior probs")
plt.legend(loc='upper right')
plt.xlabel("Value")
plt.ylabel("Frequency")

print "No of SPAM with 0 posterior probs" ,len(np.where(cls1 ==
0)[0])
print "No of HAM with 0 posterior probs",len(np.where(cls0 == 0)
[0])
# END STUDENT CODE HW234PLOT
```

```
No of SPAM with 0 posterior probs 0
No of HAM with 0 posterior probs 0
```



Result discussion with Smoothing

So smoothing has added vocabulary in denominator so it would never be -inf hence as expected SPAM with posterior proability are 0 and same is the case for HAM.

Also probabilities are have better spread than without smoothing. This should give better results if we test on validation step.

HW2.5 Ignore rare words (Optional)

Repeat HW2.4. **Please replace code tags like HW231HADOOP_CLASSIFY with HW251HADOOP_CLASSIFY through out for your submission.** This time when modeling and classification ignore tokens with a frequency of less than three (3) in the training set. How does it affect the misclassifcation error of learnt naive multinomial Bayesian Classifier on the training dataset. Report the error and the change in error.

HINT: ignore tokens with a frequency of less than three (3). Think of this as a preprocessing step. How many mapreduce jobs do you need to solve thus homework?

Modifications to Mapper

In [119]:
```
%%writefile NaiveBayes/reducer_model.py
#!/usr/bin/env python
from operator import itemgetter
import sys, operator
import numpy as np
from  collections import OrderedDict

# START STUDENT CODE HW241REDUCER_MODEL

current_word = None
smooth_factor = 1 # add 1 smoothing
current_count = [smooth_factor, smooth_factor]
msgIDs = {}
word = None
wordcount = {}

# input comes from STDIN
# remove leading and trailing whitespace

# parse the input we got from mapper.py

# convert count and spam flag (currently a string) to int


# handle msgID - store all IDs as we don't have too much
# not the best way to get prior, a two-level MapReduce jobs (ID
- word) would be optimal

# calculate NB parameters, and write the dictionary to a file fo
r the classification job
# prior probabilities

# conditional probability
# input comes from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # parse the input we got from mapper.py
    word, count, isSpam, msgID = line.split('\t', 3)


    # convert count and spam flag (currently a string) to int
    try:
        count = int(count)
        isSpam = int(isSpam)
    except ValueError:
        # count was not a number, so silently
        # ignore/discard this line
        continue

    # handle msgID - store all IDs as we don't have too much
    # not the best way to get prior, a two-level MapReduce jobs
(ID - word) would be optimal
    if msgID not in msgIDs:
        msgIDs[msgID] = isSpam

    # this IF-switch only works because Hadoop sorts map output
```

```
Overwriting NaiveBayes/reducer_model.py
```

In [120]:
```
# START STUDENT CODE HW251HADOOP_MODEL_SPAM
# STEP 1: make input directory on HDFS
# !hdfs dfs -mkdir -p /user/Xxxxxxx

# STEP2:  upload data to HDFS
!hdfs dfs -rm enronemail_1h.txt
!hdfs dfs -copyFromLocal NaiveBayes/enronemail_1h.txt
# STEP3: Make sure to remove the results directiry
!hdfs dfs -rm -r HW243HADOOP_MODEL_SPAM/model/
# STEP4: Run job

# STEP5: display model (first 10 lines only)



##################### IMPORTANT ##################################
########################################
# Make sure you have the correct paths to the jar file as wel as
the input and output files!!
# make sure to include the -files option. Do ***** NOT ****** pu
t spaces between the file paths!
#################################################################
########################################

!hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
  -D mapreduce.job.output.key.comparator.class=org.apache.hadoop
.mapred.lib.KeyFieldBasedComparator \
  -D mapreduce.partition.keycomparator.options="-k1,1"  \
  -files NaiveBayes/mapper_model.py,NaiveBayes/reducer_model.py
\
  -mapper mapper_model.py\
  -reducer reducer_model.py \
  -input enronemail_1h.txt \
  -output HW243HADOOP_MODEL_SPAM/model/\
  -numReduceTasks 1\
  -cmdenv PATH=/opt/anaconda/bin:$PATH

!hdfs dfs -ls HW243HADOOP_MODEL_SPAM/model/
!rm NaiveBayes/SPAM_Model_MNB.tsv
!hdfs dfs -get HW243HADOOP_MODEL_SPAM/model//part-00000 NaiveBay
es/SPAM_Model_MNB.tsv
print "*****First top 10 lines******"
!head -n 10 NaiveBayes/SPAM_Model_MNB.tsv
print "***** bottom 10 lines******"
!tail -n 10 NaiveBayes/SPAM_Model_MNB.tsv
# END STUDENT CODE HW251HADOOP_MODEL_SPAM
```

```
Deleted enronemail_1h.txt
Deleted HW243HADOOP_MODEL_SPAM/model
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.ja
r] /tmp/streamjob2002589548940368776.jar tmpDir=null
17/05/23 02:33:45 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:33:45 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:33:46 INFO mapred.FileInputFormat: Total input path
s to process : 1
17/05/23 02:33:47 INFO mapreduce.JobSubmitter: number of splits
:2
17/05/23 02:33:47 INFO mapreduce.JobSubmitter: Submitting token
s for job: job_1495464826300_0103
17/05/23 02:33:48 INFO impl.YarnClientImpl: Submitted applicati
on application_1495464826300_0103
17/05/23 02:33:48 INFO mapreduce.Job: The url to track the job:
http://quickstart.cloudera:8088/proxy/application_1495464826300
_0103/
17/05/23 02:33:48 INFO mapreduce.Job: Running job: job_14954648
26300_0103
17/05/23 02:33:58 INFO mapreduce.Job: Job job_1495464826300_010
3 running in uber mode : false
17/05/23 02:33:58 INFO mapreduce.Job:  map 0% reduce 0%
17/05/23 02:34:10 INFO mapreduce.Job:  map 50% reduce 0%
17/05/23 02:34:11 INFO mapreduce.Job:  map 100% reduce 0%
17/05/23 02:34:21 INFO mapreduce.Job:  map 100% reduce 100%
17/05/23 02:34:21 INFO mapreduce.Job: Job job_1495464826300_010
3 completed successfully
17/05/23 02:34:21 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=1172006
                FILE: Number of bytes written=2696982
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=217089
                HDFS: Number of bytes written=131943
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots
(ms)=20928
                Total time spent by all reduces in occupied slo
ts (ms)=7139
                Total time spent by all map tasks (ms)=20928
                Total time spent by all reduce tasks (ms)=7139
                Total vcore-seconds taken by all map tasks=2092
8
                Total vcore-seconds taken by all reduce tasks=7
139
                Total megabyte-seconds taken by all map tasks=2
1430272
                Total megabyte-seconds taken by all reduce task
s=7310336
```

In [121]:
```
%%writefile NaiveBayes/mapper_classify.py
#!/usr/bin/env python
from NaiveBayesModel import NaiveBayesModel
import sys, re, string, subprocess
import sys, operator, math
import numpy as np
from math import log
from math import exp
# Init mapper phase

# read the MODEL into memory
# The model file resides the local disk (make sure to ship it ho
me from HDFS).
# In the Hadoop command linke be sure to add the follow the -fil
es commmand line option
NBModel = NaiveBayesModel("NaiveBayes/SPAM_Model_MNB.tsv")
worddict = NBModel.model
#NBModel.printModel()
# define regex for punctuation removal
regex = re.compile('[%s]' % re.escape(string.punctuation))

# inner loop mapper phase: process each record
# input comes from STDIN (standard input)
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    parts = line.split("\t")
    docID, docClass, title = parts[0:3]
    if len(parts) == 4:
        body = parts[3]
    else:
        body = ""
    # use subject and body
    # remove punctuations, only have white-space as delimiter
    emailStr = regex.sub(' ', title.lower() + " " +body.lower())
#replace each punctuation with a space
    emailStr = re.sub( '\s+', ' ', emailStr )            # repla
ce multiple spaces with a space
    # split the line into words
    words = emailStr.split()
    nwords = []
    for i in words:
        if i in worddict:
            nwords.append(i)
# START STUDENT CODE HW251MAPPER_CLASSIFY



    #PrClass0GivenDoc, PrClass1GivenDoc = NBModel.classify(words
)

    #print "Pr(Class=0| Doc=%s) is %6.5f" % (docID, PrClass0Give
nDoc)
    #print "Pr(Class=1| Doc=%s) is %6.5f" % (docID, PrClass1Give
nDoc)

    PrClass0GivenDoc, PrClass1GivenDoc = NBModel.classifyInLogs(
```

```
Overwriting NaiveBayes/mapper_classify.py
```

In [122]:
```
# START STUDENT CODE HW254HADOOP_CLASSIFY_SPAM

#run classifier job
!hdfs dfs -rm enronemail_1h.txt
!hdfs dfs -copyFromLocal NaiveBayes/enronemail_1h.txt
!hdfs dfs -rm -r HW254HADOOP_CLASSIFY_SPAM/classifications/

##################### IMPORTANT ##############################
########################################
# Make sure you have the correct paths to the jar file as wel as
the input and output files!!
# make sure to include the -files option. Do ***** NOT ****** pu
t spaces between the file paths!
##############################################################
########################################

!hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
  -D mapreduce.job.output.key.comparator.class=org.apache.hadoop
.mapred.lib.KeyFieldBasedComparator \
  -D mapreduce.partition.keycomparator.options="-k1,1"  \
  -files NaiveBayes\
  -mapper NaiveBayes/mapper_classify.py\
  -reducer NaiveBayes/reducer_classify.py  \
  -input enronemail_1h.txt \
  -output HW254HADOOP_CLASSIFY_SPAM/classifications/\
  -numReduceTasks 1\
  -cmdenv PATH=/opt/anaconda/bin:$PATH


#Print accuracy

# END STUDENT CODE HW254HADOOP_CLASSIFY_SPAM
```

```
Deleted enronemail_1h.txt
Deleted HW254HADOOP_CLASSIFY_SPAM/classifications
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.ja
r] /tmp/streamjob3331201059874052420.jar tmpDir=null
17/05/23 02:34:52 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:34:53 INFO client.RMProxy: Connecting to ResourceMa
nager at /0.0.0.0:8032
17/05/23 02:34:56 INFO mapred.FileInputFormat: Total input path
s to process : 1
17/05/23 02:34:56 INFO mapreduce.JobSubmitter: number of splits
:2
17/05/23 02:34:56 INFO mapreduce.JobSubmitter: Submitting token
s for job: job_1495464826300_0104
17/05/23 02:34:57 INFO impl.YarnClientImpl: Submitted applicati
on application_1495464826300_0104
17/05/23 02:34:57 INFO mapreduce.Job: The url to track the job:
http://quickstart.cloudera:8088/proxy/application_1495464826300
_0104/
17/05/23 02:34:57 INFO mapreduce.Job: Running job: job_14954648
26300_0104
17/05/23 02:35:08 INFO mapreduce.Job: Job job_1495464826300_010
4 running in uber mode : false
17/05/23 02:35:08 INFO mapreduce.Job:  map 0% reduce 0%
17/05/23 02:35:21 INFO mapreduce.Job:  map 50% reduce 0%
17/05/23 02:35:22 INFO mapreduce.Job:  map 100% reduce 0%
17/05/23 02:35:31 INFO mapreduce.Job:  map 100% reduce 100%
17/05/23 02:35:32 INFO mapreduce.Job: Job job_1495464826300_010
4 completed successfully
17/05/23 02:35:32 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=2878
                FILE: Number of bytes written=357658
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=217089
                HDFS: Number of bytes written=2897
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots
(ms)=20583
                Total time spent by all reduces in occupied slo
ts (ms)=7419
                Total time spent by all map tasks (ms)=20583
                Total time spent by all reduce tasks (ms)=7419
                Total vcore-seconds taken by all map tasks=2058
3
                Total vcore-seconds taken by all reduce tasks=7
419
                Total megabyte-seconds taken by all map tasks=2
1076992
                Total megabyte-seconds taken by all reduce task
s=7597056
```

In [123]:
```
# START STUDENT CODE HW244HADOOP_CLASSIFY_RESULTS
!hdfs dfs -cat HW254HADOOP_CLASSIFY_SPAM/classifications/part-00
000

# END STUDENT CODE HW244HADOOP_CLASSIFY_RESULTS
```

```
PREDICTION      LABEL   DOCID
0       0
        0001.1999-12-10.farmer
0       0
        0001.1999-12-10.kaminski
0       0
        0001.2000-01-17.beck
0       0
        0001.2000-06-06.lokay
0       0
        0001.2001-02-07.kitchen
0       0
        0001.2001-04-02.williams
0       0
        0002.1999-12-13.farmer
0       0
        0002.2001-02-07.kitchen
1       1
        0002.2001-05-25.SA_and_HP
1       1
        0002.2003-12-18.GP
1       1
        0002.2004-08-01.BG
0       0
        0003.1999-12-10.kaminski
0       0
        0003.1999-12-14.farmer
0       0
        0003.2000-01-17.beck
0       0
        0003.2001-02-08.kitchen
1       1
        0003.2003-12-18.GP
1       1
        0003.2004-08-01.BG
0       0
        0004.1999-12-10.kaminski
0       0
        0004.1999-12-14.farmer
0       0
        0004.2001-04-02.williams
1       1
        0004.2001-06-12.SA_and_HP
1       1
        0004.2004-08-01.BG
0       0
        0005.1999-12-12.kaminski
0       0
        0005.1999-12-14.farmer
0       0
        0005.2000-06-06.lokay
0       0
        0005.2001-02-08.kitchen
1       1
        0005.2001-06-23.SA_and_HP
1       1
        0005.2003-12-18.GP
0       0
        0006.1999-12-13.kaminski
```

HW2.6 Benchmark your code with the Python SciKit-Learn (OPTIONAL)

HW2.6 Benchmark your code with the Python SciKit-Learn implementation of the multinomial Naive Bayes algorithm

It always a good idea to benchmark your solutions against publicly available libraries/frameworks such as SciKit-Learn, the Machine Learning toolkit available in Python. In this exercise, we benchmark ourselves against the SciKit-Learn implementation of multinomial Naive Bayes. For more information on this implementation see: http://scikit-learn.org/stable/modules/naive_bayes.html (http://scikit-learn.org/stable/modules/naive_bayes.html) more

In this exercise, please complete the following tasks:

- Run the Multinomial Naive Bayes algorithm (using default settings) from SciKit-Learn over the same training data used in HW2.5 and report the misclassification error (please note some data preparation might be needed to get the Multinomial Naive Bayes algorithm from SkiKit-Learn to run over this dataset)
- Prepare a table to present your results, where rows correspond to approach used (SkiKit-Learn versus your Hadoop implementation) and the column presents the training misclassification error
- Explain/justify any differences in terms of training error rates over the dataset in HW2.5 between your Multinomial Naive Bayes implementation (in Map Reduce) versus the Multinomial Naive Bayes implementation in SciKit-Learn

HW 2.6.1 Bernoulli Naive Bayes (OPTIONAL: note this exercise is a stretch HW and optional)

- Run the Bernoulli Naive Bayes algorithm from SciKit-Learn (using default settings) over the same training data used in HW2.6 and report the misclassification error
- Discuss the performance differences in terms of misclassification error rates over the dataset in HW2.5 between the Multinomial Naive Bayes implementation in SciKit-Learn with the Bernoulli Naive Bayes implementation in SciKit-Learn. Why such big differences. Explain.

Which approach to Naive Bayes would you recommend for SPAM detection? Justify your selection.

In [ ]:

HW2.7 Preprocess the Entire Spam Dataset (OPTIONAL)

The Enron SPAM data in the following folder enron1-Training-Data-RAW (https://www.dropbox.com/sh/hemnvr0422nr36g/AAAPoK-aYxkFGxGjzaeRNEwSa?dl=0) is in raw text form (with subfolders for SPAM and HAM that contain raw email messages in the following form:

- Line 1 contains the subject
- The remaining lines contain the body of the email message.

In Python write a script to produce a TSV file called train-Enron-1.txt that has a similar format as the enronemail_1h.txt that you have been using so far. Please pay attend to funky characters and tabs. Check your resulting formated email data in Excel and in Python (e.g., count up the number of fields in each row; the number of SPAM mails and the number of HAM emails). Does each row correspond to an email record with four values? Note: use "NA" to denote empty field values.

In [ ]:

HW2.8 Build and evaluate a NB classifier on the Entire Spam Dataset (OPTIONAL)

Back to Table of Contents Using Hadoop Map-Reduce write job(s) to perform the following: -- Train a multinomial Naive Bayes Classifier with Laplace plus one smoothing using the data extracted in HW2.7 (i.e., train-Enron-1.txt). Use all white-space delimitted tokens as independent input variables (assume spaces, fullstops, commas as delimiters). Drop tokens with a frequency of less than three (3). -- Test the learnt classifier using enronemail_1h.txt and report the misclassification error rate. Remember to use all white-space delimitted tokens as independent input variables (assume spaces, fullstops, commas as delimiters). How do we treat tokens in the test set that do not appear in the training set?

In [ ]:

HW2.8.1 OPTIONAL

- Run both the Multinomial Naive Bayes and the Bernoulli Naive Bayes algorithms from SciKit-Learn (using default settings) over the same training data used in HW2.8 and report the misclassification error on both the training set and the testing set
- Prepare a table to present your results, where rows correspond to approach used (SciKit-Learn Multinomial NB; SciKit-Learn Bernouili NB; Your Hadoop implementation) and the columns presents the training misclassification error, and the misclassification error on the test data set
- Discuss the performance differences in terms of misclassification error rates over the test and training datasets by the different implementations. Which approch (Bernouili versus Multinomial) would you recommend for SPAM detection? Justify your selection.

------- END OF HOWEWORK --------