HW2.0.0

A **Race Condition** in the context of parallel computation is the situation where the resulting value depends on the order of the computational operations.

An example of this is:

| Thread 1 | Thread 2 | | Integer value |

| | | | 0 |

| read value | |<-| 0 |

|increase value| | | 0 |

|write back | |->| 1 |

| | read value |<-| 1 |

| |increase value| | 1 |

| | write back |->| 2 |

And the result is 2; but in this case:

| Thread 1 | Thread 2 | | Integer value | | | | | 0 | | read value | |<-| 0 | | | read value |<-| 0 | |increase value| | | 0 | | |increase value| | 0 | | write back | |->| 1 | | | write back |->| 1 |

the result is 1.

(note: for some reason, latex tables don't work here)

MapReduce is a programming model for processing and generating large data sets with a parallel, distributed algorithm on a cluster. It is also a framework for processing parallelizable problems across huge data sets, using a large number of computers (nodes); clusters or grid.

The difference between MapReduce and Hadoop is that Hadoop is the Distributed File System that stores the data. The Hadoop framework is implemented in Java, and provides an API to MapReduce that allows you to write your map and reduce functions in languages other than Java.

HW 2.0.1

Hadoop is based on the MapReduce programming paradigm. Here is an example:

```
In [39]: from functools import reduce

         def length(s):
             return len(s)
         strings = ["str1", "string2", "w261", "Machine learning at Scale"]
         S = map(length, strings)

         print "Average length of a string here is", (reduce(lambda x, y: x+y, S)/len(S))
```

```
Average length of a string here is 10
```

HW 2.1 Sorting in MapReduce

First, let's generate the set of random numbers.

```
In [40]: %pwd
```

```
Out[40]: u'/home/hadoop/Notebooks/Users/Nina'
```

```
In [41]: %pwd
         %mkdir SortData
         %mkdir SortCode
```

```
mkdir: cannot create directory ‘SortData’: File exists
mkdir: cannot create directory ‘SortCode’: File exists
```

```
In [42]: %%writefile SortCode/hw2_1_generate_numbers.py
         #!/home/hadoop/anaconda2/bin/python

         import random
         random.seed(10001)
         N = 10000
         for n in range(N):
             print random.randint(0,N),"\t","NA"
```

```
Overwriting SortCode/hw2_1_generate_numbers.py
```

Now, we redirect the output of this file to a text file.

```
In [43]: !chmod +x SortCode/hw2_1_generate_numbers.py;
         !./SortCode/hw2_1_generate_numbers.py > SortData/hw2_1_NumbersDataSet.txt
         !wc -l SortData/hw2_1_NumbersDataSet.txt
```

```
10000 SortData/hw2_1_NumbersDataSet.txt
```

And then, we check the output:

In [44]:
```
!cat SortData/hw2_1_NumbersDataSet.txt | head -10
```

```
3563    NA
8248    NA
1194    NA
5810    NA
6439    NA
5566    NA
1338    NA
9680    NA
211     NA
3053    NA
cat: write error: Broken pipe
```

We create a directory for this in hdfs and put the file there:

In [7]:
```
! hdfs dfs -mkdir Sort
! hdfs dfs -put SortData/hw2_1_NumbersDataSet.txt Sort
```

```
mkdir: `Sort': File exists
put: `Sort/hw2_1_NumbersDataSet.txt': File exists
```

In [8]:
```
% mkdir SortOutput
```

```
mkdir: cannot create directory ‘SortOutput’: File exists
```

In [45]:
```
% hdfs dfs -put SortOutput
```

```
ERROR: Line magic function `%hdfs` not found.
```

In [46]:
```
! hdfs dfs -cat Sort/hw2_1_NumbersDataSet.txt | head -10
```

```
3563    NA
8248    NA
1194    NA
5810    NA
6439    NA
5566    NA
1338    NA
9680    NA
211     NA
3053    NA
cat: Unable to write to output stream.
```

Then, we write a file that reads in this file (note: this code is inspired by the posted notebook 'MIDS-W261-Partial-Total-Secondary-Sorts'):

In [47]:
```
%%writefile SortCode/identityFunction.py
#!/usr/bin/python
import sys

# input comes from STDIN (standard input)
for line in sys.stdin:
    key,value = line.split("\t", 1)
    print '%s\t%s' % (key,value)
```

```
Overwriting SortCode/identityFunction.py
```

and let it run:

In [48]:
```
!hdfs dfs -rm -r Sort/Output
!hadoop jar /usr/lib/hadoop/hadoop-streaming-2.7.2-amzn-1.jar \
   -D mapred.output.key.comparator.class=org.apache.hadoop.mapred.lib.KeyFieldBasedCo
   -D mapred.text.key.comparator.options=-nr \
   -mapper /bin/cat \
   -reducer /bin/cat \
   -input Sort/hw2_1_NumbersDataSet.txt  -output Sort/Output
```

```
16/06/03 02:19:46 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deleti
on interval = 0 minutes, Emptier interval = 0 minutes.
Deleted Sort/Output
packageJobJar: [] [/usr/lib/hadoop/hadoop-streaming-2.7.2-amzn-1.jar] /tmp/streamj
ob7752606040531101698.jar tmpDir=null
16/06/03 02:19:49 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-
7-251.us-west-1.compute.internal/172.31.7.251:8032
16/06/03 02:19:49 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-
7-251.us-west-1.compute.internal/172.31.7.251:8032
16/06/03 02:19:49 INFO metrics.MetricsSaver: MetricsConfigRecord disabledInCluster
: false instanceEngineCycleSec: 60 clusterEngineCycleSec: 60 disableClusterEngine:
 true maxMemoryMb: 3072 maxInstanceCount: 500 lastModified: 1464726748890
16/06/03 02:19:49 INFO metrics.MetricsSaver: Created MetricsSaver j-ZAC3GQDMC0E6:i
-610a91d4:RunJar:06792 period:60 /mnt/var/em/raw/i-610a91d4_20160603_RunJar_06792_
raw.bin
16/06/03 02:19:50 INFO lzo.GPLNativeCodeLoader: Loaded native gpl library
16/06/03 02:19:50 INFO lzo.LzoCodec: Successfully loaded & initialized native-lzo
library [hadoop-lzo rev 426d94a07125cf9447bb0c2b336cf10b4c254375]
16/06/03 02:19:50 INFO mapred.FileInputFormat: Total input paths to process : 1
```

Then, when we look at the output directory, we see all the sorted numbers:

In [49]:
```
!hdfs dfs -ls Sort/Output
```

```
Found 8 items
-rw-r--r--   1 hadoop hadoop          0 2016-06-03 02:20 Sort/Output/_SUCCESS
-rw-r--r--   1 hadoop hadoop      12400 2016-06-03 02:20 Sort/Output/part-00000
-rw-r--r--   1 hadoop hadoop      13112 2016-06-03 02:20 Sort/Output/part-00001
-rw-r--r--   1 hadoop hadoop      12759 2016-06-03 02:20 Sort/Output/part-00002
-rw-r--r--   1 hadoop hadoop      12832 2016-06-03 02:20 Sort/Output/part-00003
-rw-r--r--   1 hadoop hadoop      12613 2016-06-03 02:20 Sort/Output/part-00004
-rw-r--r--   1 hadoop hadoop      12729 2016-06-03 02:20 Sort/Output/part-00005
-rw-r--r--   1 hadoop hadoop      12543 2016-06-03 02:20 Sort/Output/part-00006
```

And we view the highest values are the top 10 lines in

In [50]:
```
!hdfs dfs -cat Sort/Output/part-00000 | head -10
```

```
9991    NA
9991    NA
9991    NA
9991    NA
9984    NA
9984    NA
9970    NA
9970    NA
9949    NA
9942    NA
```

and the lowest with

In [51]:
```
!hdfs dfs -cat Sort/Output/part-00004 | tail -10
```

```
79      NA
58      NA
51      NA
44      NA
44      NA
44      NA
37      NA
23      NA
16      NA
5       NA
```

HW 2.2 Wordcount

We test again:

In [52]:
```
!grep assistance enronemail_1h.txt|cut -d$'\t' -f4| grep assistance|wc -l
```

```
8
```

Now, we set the mapper and reducer for this file. In the mapper, start by printing the words and their count.

The following mapper-reducer files are borrowed from the master solution file.

In [104]:
```
import re, string
print string.punctuation
```

```
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

In [105]:
```
%%writefile mapper.py
#!/usr/bin/env python
import sys, re, string
# define regex for punctuation removal
regex = re.compile('[%s]' % re.escape(string.punctuation))
WORD_RE = re.compile(r"[\w']+")
# input comes from STDIN (standard input)
for line in sys.stdin:
    # use subject and body
    #line = line.strip().split('\t', 2)[-1]
    for c in string.punctuation:
        line= line.replace(c,"")
    line = line.strip().split('\t', 2)[-1]
    # remove punctuations, only have white-space as delimiter
    line = regex.sub(' ', line.lower())
    line = re.sub( '\s+', ' ', line )
    # split the line into words
    words = line.split()
    # increase counters
    for word in words:
        # 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
        if len(word) > 1:  #drop single character words
            print '%s\t%s' % (word, 1)
```

```
Overwriting mapper.py
```

In [106]:
```
%%writefile reducer.py
#!/usr/bin/env python
from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None
wordcount = {}

# 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 = line.split('\t', 1)

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

    # this IF-switch only works because Hadoop sorts map output
    # by key (here: word) before it is passed to the reducer
    if current_word == word:
        current_count += count
    else:
        if current_word:
            # save count
             print '%s\t%d' % (current_word, current_count)
        current_count = count
        current_word = word

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

# found count for word assistance
#findword = 'assistance'
#print '%s\t%d' %(findword, wordcount[findword] if findword in wordcount else 0)
```

```
Overwriting reducer.py
```

In [85]:
```
%%writefile WordCount/mapper.py
#!/usr/bin/env python

import sys
import re
import string
import sys
#sys.stderr.write("reporter:counter:Tokens,Total,1") # NOTE missing the carriage retu
sys.stderr.write("reporter:counter:Mapper Counters,Calls,1\n")
sys.stderr.write("reporter:status:processing my message...how are you\n")

WORD_RE = re.compile(r"[\w']+")
for line in sys.stdin:
    for word in line.split():
        word = word.lower()
        print '%s\t%s' % (word, 1)
        if word == "debt":
            sys.stderr.write("reporter:counter:EDA Counters,Calls,1\n")
```

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

In [86]:
```
%%writefile WordCount/reducer.py
#!/usr/bin/env python

import sys

cur_key = None
cur_count = 0
sys.stderr.write("reporter:counter:Reducer Counters,Calls,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)

#token_occurence = sorted(token_occurence, key=lambda key: token_occurence[val])

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

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

```
In [20]: %%writefile mapper.py
         #!/Users/ninakuklisova/miniconda2/envs/jupi/bin/python
         ## mapper.py
         ## Author: Nina Kuklisova
         ## Description: mapper code for HW2.2

         import sys
         import re
         import string
         #count = 0

         WORD_RE = re.compile(r"[\w']+")
         filename = sys.argv[2]
         findwords = sys.argv[1]
         with open (filename, "r") as myfile:

             for line in myfile.readlines():
                 count = [0]*len(findwords)
                 words = line.split()
                 for word in words:
                     word = word.lower()
                     # if the word is found in the list of words, increase its count by 1
                     if word in findwords:
                         count[findwords.index[word]] = count[findwords.index[word]] + 1
                 print count

         Overwriting mapper.py

In [107]: !chmod a+x mapper.py

In [119]: % pwd

Out[119]: u'/home/hadoop/Notebooks/Users/Nina'
```

In [22]:
```
%%writefile reducer.py
#!/Users/ninakuklisova/miniconda2/envs/jupi/bin/python

import sys
print sys.argv
sum = 0
filename = sys.argv[2] #"enronemail_1h.txt"
#findword = sys.argv[1]

with open (filename, 'rb') as myfile:
    count = 0
    # take the words in the first line
    for line in myfile:
        if count==0:
            findwords = line.split()
            word_count = len(findwords) *[0]
        else:
            counts = line.split()
            for i in range(len(findwords)):
                word_count[i]+=int(counts[i])

# find the most popular token:
token_occurence = {}
for i in range(len(findwords)):
    (key, val) = (findwords[i], word_count[i])
    token_occurence[int(key)] = val

# sort by popularity
token_occurence = sorted(token_occurence, key=lambda key: token_occurence[val])

#print top 10
top10 = {k: token_occurence[k] for k in token_occurence.keys()[:10]}


print top_10
```

```
Overwriting reducer.py
```

In [108]:
```
!chmod a+x reducer.py
```

In [114]:
```
#test mapper outside of Hadoop
!cat enronemail_1h.txt | ./mapper.py | sort -k1,1 |tail -10
```

```
zimin   1
zimin   1
zimin   1
zinc    1
zk      1
zo      1
zo      1
zolam   1
zolam   1
zxs     1
```

In [115]:
```
!cat enronemail_1h.txt | ./mapper.py | sort -k1,1 |./reducer.py |sort -k2,2nr | head
```

```
the     1246
to      961
and     662
of      560
you     427
in      415
your    391
for     373
this    260
on      258
sort: write failed: standard output: Broken pipe
sort: write error
```

In [116]:
```
#test reducer outside of Hadoop
!cat enronemail_1h.txt | ./mapper.py | sort -k1,1 |./reducer.py |grep "assistance"
```

```
assistance      10
```

HW 2.2.1 Using MapReduce on Hadoop

We let the Mapper and Reducer run on HDFS:

In [128]:
```
!hdfs dfs -rm -r HW2_2/output #Enron_email_reading_output
!hdfs dfs -rm enronemail_1h.txt
!hdfs dfs -copyFromLocal enronemail_1h.txt

!hadoop jar /usr/lib/hadoop/hadoop-streaming-2.7.2-amzn-1.jar \
    -files /home/hadoop/Notebooks/Users/Nina/mapper.py,/home/hadoop/Notebooks/Users/N
    -mapper mapper.py \
    -reducer reducer.py \
    -input enronemail_1h.txt -output HW2_2/output
```

```
rm: `HW2_2/output': No such file or directory
16/06/04 18:06:48 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deleti
on interval = 0 minutes, Emptier interval = 0 minutes.
Deleted enronemail_1h.txt
packageJobJar: [] [/usr/lib/hadoop/hadoop-streaming-2.7.2-amzn-1.jar] /tmp/streamj
ob6977620921043626238.jar tmpDir=null
16/06/04 18:06:55 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-
7-251.us-west-1.compute.internal/172.31.7.251:8032
16/06/04 18:06:55 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-
7-251.us-west-1.compute.internal/172.31.7.251:8032
16/06/04 18:06:56 INFO metrics.MetricsSaver: MetricsConfigRecord disabledInCluster
: false instanceEngineCycleSec: 60 clusterEngineCycleSec: 60 disableClusterEngine:
 true maxMemoryMb: 3072 maxInstanceCount: 500 lastModified: 1464726748890
16/06/04 18:06:56 INFO metrics.MetricsSaver: Created MetricsSaver j-ZAC3GQDMC0E6:i
-610a91d4:RunJar:27658 period:60 /mnt/var/em/raw/i-610a91d4_20160604_RunJar_27658_
raw.bin
16/06/04 18:06:56 INFO lzo.GPLNativeCodeLoader: Loaded native gpl library
16/06/04 18:06:56 INFO lzo.LzoCodec: Successfully loaded & initialized native-lzo
library [hadoop-lzo rev 426d94a07125cf9447bb0c2b336cf10b4c254375]
16/06/04 18:06:56 INFO mapred.FileInputFormat: Total input paths to process : 1
16/06/04 18:06:56 INFO mapreduce.JobSubmitter: number of splits:16
16/06/04 18:06:56 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1464
726740139_0032
16/06/04 18:06:56 INFO impl.YarnClientImpl: Submitted application application_1464
726740139_0032
16/06/04 18:06:56 INFO mapreduce.Job: The url to track the job: http://ip-172-31-7
-251.us-west-1.compute.internal:20888/proxy/application_1464726740139_0032/ (http:
//ip-172-31-7-251.us-west-1.compute.internal:20888/proxy/application_1464726740139
_0032/)
16/06/04 18:06:56 INFO mapreduce.Job: Running job: job_1464726740139_0032
16/06/04 18:07:04 INFO mapreduce.Job: Job job_1464726740139_0032 running in uber m
ode : false
16/06/04 18:07:04 INFO mapreduce.Job:  map 0% reduce 0%
16/06/04 18:07:14 INFO mapreduce.Job:  map 6% reduce 0%
16/06/04 18:07:18 INFO mapreduce.Job:  map 13% reduce 0%
16/06/04 18:07:21 INFO mapreduce.Job:  map 50% reduce 0%
16/06/04 18:07:22 INFO mapreduce.Job:  map 56% reduce 0%
16/06/04 18:07:24 INFO mapreduce.Job:  map 63% reduce 0%
16/06/04 18:07:28 INFO mapreduce.Job:  map 75% reduce 0%
16/06/04 18:07:30 INFO mapreduce.Job:  map 81% reduce 0%
16/06/04 18:07:31 INFO mapreduce.Job:  map 94% reduce 0%
16/06/04 18:07:33 INFO mapreduce.Job:  map 100% reduce 0%
16/06/04 18:07:35 INFO mapreduce.Job:  map 100% reduce 43%
16/06/04 18:07:36 INFO mapreduce.Job:  map 100% reduce 71%
16/06/04 18:07:38 INFO mapreduce.Job:  map 100% reduce 86%
16/06/04 18:07:40 INFO mapreduce.Job:  map 100% reduce 100%
16/06/04 18:07:40 INFO mapreduce.Job: Job job_1464726740139_0032 completed success
fully
16/06/04 18:07:40 INFO mapreduce.Job: Counters: 52
        File System Counters
                FILE: Number of bytes read=68240
                FILE: Number of bytes written=3158893
                FILE: Number of read operations=0
```

In [129]:
```
!hdfs dfs -ls HW2_2/output
```

```
Found 8 items
-rw-r--r--   1 hadoop hadoop          0 2016-06-04 18:07 HW2_2/output/_SUCCESS
-rw-r--r--   1 hadoop hadoop       8132 2016-06-04 18:07 HW2_2/output/part-00000
-rw-r--r--   1 hadoop hadoop       7775 2016-06-04 18:07 HW2_2/output/part-00001
-rw-r--r--   1 hadoop hadoop       7875 2016-06-04 18:07 HW2_2/output/part-00002
-rw-r--r--   1 hadoop hadoop       8295 2016-06-04 18:07 HW2_2/output/part-00003
-rw-r--r--   1 hadoop hadoop       8310 2016-06-04 18:07 HW2_2/output/part-00004
-rw-r--r--   1 hadoop hadoop       8060 2016-06-04 18:07 HW2_2/output/part-00005
-rw-r--r--   1 hadoop hadoop       8457 2016-06-04 18:07 HW2_2/output/part-00006
```

Then, we check it's output:

In [131]:
```
!hdfs dfs -cat HW2_2/output/part-00000 | head -10
```

```
000     4
0120    1
0344    1
036474336       1
0813    1
0841    1
093843  1
10000   6
100038  1
100foot 1
```

And we sort it by counts:

In [133]:
```
!hdfs dfs -cat HW2_2/output/part-*| sort -k2,2nr | head -10
```

```
the     1246
to      961
and     662
of      560
you     427
in      415
your    391
for     373
this    260
on      258
sort: write failed: standard output: Broken pipe
sort: write error
```

HW 2.3 Multinomial Naive Bayes with No Smoothing

First, we let our Mapper / Reducer learn a Multinomial Naive Bayes classifier.

The Multinomial Naive Bayes needs to go through the training set of 100 emails, evaluate the frequency of each term in each category; and based on this, calculate the HAM/SPAM probability associated with each word.

In [287]:
```
%%writefile mapper_t.py
#!/usr/bin/env python
import sys, re, string
# define regex for punctuation removal
regex = re.compile('[%s]' % re.escape(string.punctuation))
# input comes from STDIN (standard input)
for line in sys.stdin:
    # use subject and body
    msg = line.strip().split('\t', 2)
    if len(msg) < 3:
        continue
    msgID, isSpam = msg[0], msg[1]

    # remove punctuations, only have white-space as delimiter
    msgTxt = regex.sub(' ', msg[2].lower())
    msgTxt = re.sub( '\t', ' ', msgTxt )
    msgTxt = re.sub( '\s+', ' ', msgTxt )
    # split the line into words
    words = msgTxt.split()
    # increase counters
    for word in words:
        # 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
        if len(word) >1: #drop single character words
            print '%s\t%d\t%s\t%s' % (word, 1, isSpam, msgID)
```

```
Overwriting mapper_t.py
```

In [288]:
```
%%writefile reducer_t.py
#!/usr/bin/env python
from operator import itemgetter
import sys, operator
import numpy as np
#from Decimal import *

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

# 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
    if msgID not in msgIDs:
        msgIDs[msgID] = isSpam

    if word in wordcount.keys():
        if isSpam ==0:
            wordcount[word][0]+=1
        if isSpam ==1:
            wordcount[word][1]+=1
    else:
        if isSpam ==0:
            wordcount[word]=[1,0]
        if isSpam ==1:
            wordcount[word]=[0,1]


### total count of all words in ham & spam:
total_wc_ham = float(sum( wordcount[i][0] for i in wordcount.keys()))
total_wc_spam = float(sum( wordcount[i][1] for i in wordcount.keys()))

for (key,value) in zip(wordcount.keys(), wordcount.values()): #/(1.0*n_total)):
    word_total = value[0] + value[1]
    p_word = float(word_total) / (total_wc_ham + total_wc_spam)
    print key, "%.9f" % (p_word*value[0]*0.56/total_wc_ham), "%.9f" %(p_word*value[1]
```

```
Overwriting reducer_t.py
```

In [247]:
```
%%writefile mapper_c.py
#!/usr/bin/env python
import sys, re, string, subprocess
import sys, operator, math
import numpy as np
# read the probability from HDFS
prob = {}
cat = subprocess.Popen(["hadoop", "fs", "-cat", "HW2_3/nbModel/part-*"], stdout=subpr
for line in cat.stdout:
    word, p0, p1 = line.split()
    prob[word] = [p0, p1]

# get prior probability

prior = [0.56, 0.44]

# define regex for punctuation removal
regex = re.compile('[%s]' % re.escape(string.punctuation))
# input comes from STDIN (standard input)
for line in sys.stdin:
    # use subject and body
    msg = line.split('\t', 2)
    if len(msg) < 3:
        continue
    msgID, actualSPAMClass = msg[0], msg[1]

    msgTxt = regex.sub(' ', msg[2].lower())
    msgTxt = re.sub( '\t', ' ', msgTxt )
    msgTxt = re.sub( '\s+', ' ', msgTxt )
    # split the line into words
    words = msgTxt.split()
    prHAMGivenDoc = math.log(float(prior[0]))
    prSPAMGivenDoc = math.log(float(prior[1]))
    for word in words:
        if len(word) >1: #drop single letter words
            if word in prob:
                p0 = float(prob[word][0])
                p1 = float(prob[word][1])
                #print "probs", p0, p1, prHAMGivenDoc, prSPAMGivenDoc
                wordGivenHam = math.log(p0) if p0>0.0 else float('-inf')
                wordGivenSpam = math.log(p1) if p1>0.0 else float('-inf')
                if wordGivenHam != 0.0 and prHAMGivenDoc != float('-inf'):
                    prHAMGivenDoc = prHAMGivenDoc + wordGivenHam
                else:
                    prHAMGivenDoc = float('-inf')
                if wordGivenSpam != 0.0 and prSPAMGivenDoc !=float('-inf'):
                    prSPAMGivenDoc = prSPAMGivenDoc + wordGivenSpam
                else:
                    prSPAMGivenDoc=float('-inf')
            else:
                print '%s\t%s word not found in Multinomial Naive Bayes model lexicon
                sys.exit("error[", word, "]is not in the Multinomial Naive Bayes mode
    predictedClass = 1 #SPAM
    if(prHAMGivenDoc > prSPAMGivenDoc):
        predictedClass = 0 #HAM
    if int(actualSPAMClass) == predictedClass:
        print actualSPAMClass, predictedClass, prHAMGivenDoc, prSPAMGivenDoc,0  #no e
    else:
        print actualSPAMClass, predictedClass, prHAMGivenDoc, prSPAMGivenDoc,1 # erro
```

```
Overwriting mapper_c.py
```

In [280]:
```
%matplotlib inline
```

In [289]:
```
%%writefile reducer_c.py
#!/usr/bin/python
from operator import itemgetter
import sys, operator, math
import numpy as np

import matplotlib.pyplot as plt
#import plotly.plotly as py
#import plotly.graph_objs as go

numberOfRecords = 0
NumberOfMisclassifications=0
prHAMGivenDoc = []
prSPAMGivenDoc = []
# input comes from STDIN
for line in sys.stdin:
    #print line
    # remove leading and trailing whitespace
    line = line.strip()
    toks = line.split(" ")
    # calculate the probabilities of HAM or SPAM of each email
    prHAMGivenDoc.append(math.exp(float(toks[2])))
    prSPAMGivenDoc.append(math.exp(float(toks[3])))
    # account for the result
    NumberOfMisclassifications = NumberOfMisclassifications + int(toks[4])
    numberOfRecords = numberOfRecords + 1

# calculate the overall error rate
# could also calcualte  the confusion matrix
print 'Error rate: %.4f' %(1.0*NumberOfMisclassifications/float(numberOfRecords))
print 'NumberOfMisclassifications %d, numberOfRecords%d'  %(NumberOfMisclassificatio


prHAMGivenDoc = np.array(prHAMGivenDoc)
prSPAMGivenDoc = np.array(prSPAMGivenDoc)
plt.hist(prHAMGivenDoc, bins=50, color='blue')
plt.hist(prSPAMGivenDoc, bins=50, color='red')
plt.show()
```

```
Overwriting reducer_c.py
```

In [160]:
```
!cat enronemail_1h.txt|cut -f 2|grep 1 |wc -l
```

```
44
```

In [161]:
```
!cat enronemail_1h.txt|head -1
```

```
0001.1999-12-10.farmer  0        christmas tree farm pictures   NA
cat: write error: Broken pipe
```

In [296]:
```
# these work well, they just had a long output

!chmod a+x mapper_t.py
!chmod a+x reducer_t.py

#!cat enronemail_1h.txt|head -10 | ./mapper_t.py | ./reducer_t.py
```

In [282]:
```
!chmod a+x mapper_c.py
!chmod a+x reducer_c.py

!cat enronemail_1h.txt|head -15 | ./mapper_c.py | ./reducer_c.py
```

```
cat: write error: Broken pipe
Error rate: 0.0000
NumberOfMisclassifications 0, numberOfRecords15
```

In [ ]:
```
!cat enronemail_1h.txt | ./mapper.py | sort -k1,1 |./reducer.py |grep "assistance"
```

In [283]:
```
!hdfs dfs -mkdir HW2_3
!hdfs dfs -put enronemail_1h.txt HW2_3
!hdfs dfs -rm -r HW2_3/nbModel
!hdfs dfs -rm -r HW2_3/classifications

# Run MNB training job
!hadoop jar /usr/lib/hadoop/hadoop-streaming-2.7.2-amzn-1.jar  \
    -files /home/hadoop/Notebooks/Users/Nina/mapper_t.py,/home/hadoop/Notebooks/Users
    -mapper mapper_t.py \
    -reducer reducer_t.py \
    -input HW2_3/enronemail_1h.txt -output HW2_3/nbModel
```

```
mkdir: `HW2_3': File exists
put: `HW2_3/enronemail_1h.txt': File exists
16/06/07 03:14:57 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deleti
on interval = 0 minutes, Emptier interval = 0 minutes.
Deleted HW2_3/nbModel
16/06/07 03:14:59 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deleti
on interval = 0 minutes, Emptier interval = 0 minutes.
Deleted HW2_3/classifications
packageJobJar: [] [/usr/lib/hadoop/hadoop-streaming-2.7.2-amzn-1.jar] /tmp/streamj
ob2846924059786931467.jar tmpDir=null
16/06/07 03:15:02 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-
7-251.us-west-1.compute.internal/172.31.7.251:8032
16/06/07 03:15:02 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-
7-251.us-west-1.compute.internal/172.31.7.251:8032
16/06/07 03:15:03 INFO metrics.MetricsSaver: MetricsConfigRecord disabledInCluster
: false instanceEngineCycleSec: 60 clusterEngineCycleSec: 60 disableClusterEngine:
 true maxMemoryMb: 3072 maxInstanceCount: 500 lastModified: 1464726748890
16/06/07 03:15:03 INFO metrics.MetricsSaver: Created MetricsSaver j-ZAC3GQDMC0E6:i
-610a91d4:RunJar:08357 period:60 /mnt/var/em/raw/i-610a91d4_20160607_RunJar_08357_
raw.bin
16/06/07 03:15:03 INFO lzo.GPLNativeCodeLoader: Loaded native gpl library
16/06/07 03:15:03 INFO lzo.LzoCodec: Successfully loaded & initialized native-lzo
library [hadoop-lzo rev 426d94a07125cf9447bb0c2b336cf10b4c254375]
16/06/07 03:15:03 INFO mapred.FileInputFormat: Total input paths to process : 1
16/06/07 03:15:03 INFO mapreduce.JobSubmitter: number of splits:16
16/06/07 03:15:03 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1464
726740139_0038
16/06/07 03:15:04 INFO impl.YarnClientImpl: Submitted application application_1464
726740139_0038
16/06/07 03:15:04 INFO mapreduce.Job: The url to track the job: http://ip-172-31-7
-251.us-west-1.compute.internal:20888/proxy/application_1464726740139_0038/ (http:
//ip-172-31-7-251.us-west-1.compute.internal:20888/proxy/application_1464726740139
_0038/)
16/06/07 03:15:04 INFO mapreduce.Job: Running job: job_1464726740139_0038
16/06/07 03:15:12 INFO mapreduce.Job: Job job_1464726740139_0038 running in uber m
ode : false
16/06/07 03:15:12 INFO mapreduce.Job:  map 0% reduce 0%
16/06/07 03:15:22 INFO mapreduce.Job:  map 6% reduce 0%
16/06/07 03:15:25 INFO mapreduce.Job:  map 13% reduce 0%
16/06/07 03:15:29 INFO mapreduce.Job:  map 19% reduce 0%
16/06/07 03:15:33 INFO mapreduce.Job:  map 69% reduce 0%
16/06/07 03:15:34 INFO mapreduce.Job:  map 81% reduce 0%
16/06/07 03:15:37 INFO mapreduce.Job:  map 88% reduce 0%
16/06/07 03:15:38 INFO mapreduce.Job:  map 94% reduce 0%
16/06/07 03:15:39 INFO mapreduce.Job:  map 100% reduce 0%
16/06/07 03:15:42 INFO mapreduce.Job:  map 100% reduce 14%
16/06/07 03:15:43 INFO mapreduce.Job:  map 100% reduce 43%
16/06/07 03:15:44 INFO mapreduce.Job:  map 100% reduce 71%
16/06/07 03:15:45 INFO mapreduce.Job:  map 100% reduce 86%
16/06/07 03:15:47 INFO mapreduce.Job:  map 100% reduce 100%
16/06/07 03:15:47 INFO mapreduce.Job: Job job_1464726740139_0038 completed success
```

In [ ]:
```
!hdfs dfs -mkdir HW2_3
!hdfs dfs -put enronemail_1h.txt HW2_3
#!hdfs dfs -rm -r HW2_3/nbModel
!hdfs dfs -rm -r HW2_3/classifications

# Run MNB training job
!hadoop jar /usr/lib/hadoop/hadoop-streaming-2.7.2-amzn-1.jar  \
    -files /home/hadoop/Notebooks/Users/Nina/mapper_t.py,/home/hadoop/Notebooks/Users
    -mapper mapper_c.py \
    -reducer reducer_c.py \
    -input HW2_3/enronemail_1h.txt -output HW2_3/classifications
```

In [284]:
```
!hdfs dfs -cat HW2_3/nbModel/part-*|head -10
```

```
liar 0.000000000 0.000000103
chinese 0.000000000 0.000000026
saying 0.000000000 0.000000414
rob 0.000000000 0.000000233
personally 0.000000000 0.000000026
dollar 0.000000000 0.000000026
focus 0.000000182 0.000000310
krgp 0.000000182 0.000000000
existing 0.000000684 0.000000259
783518 0.000000000 0.000000026
cat: Unable to write to output stream.
cat: Unable to write to output stream.
cat: Unable to write to output stream.
cat: Unable to write to output stream.
cat: Unable to write to output stream.
cat: Unable to write to output stream.
```

(I have to admit I'm not sure why doesn't Matplotlib show the histogram; however,) We can see that this classifier seems to perform extremely well.

HW2.4 Classification with Laplace plus-one smoothing

Here, we use the same process as in the previous exercise, just slightly change the formula in the reducer that evaluates posterior probabilities.

In [297]:
```
%%writefile mapper_t.py
#!/usr/bin/env python
import sys, re, string
# define regex for punctuation removal
regex = re.compile('[%s]' % re.escape(string.punctuation))
# input comes from STDIN (standard input)
for line in sys.stdin:
    # use subject and body
    msg = line.strip().split('\t', 2)
    if len(msg) < 3:
        continue
    msgID, isSpam = msg[0], msg[1]

    # remove punctuations, only have white-space as delimiter
    msgTxt = regex.sub(' ', msg[2].lower())
    msgTxt = re.sub( '\t', ' ', msgTxt )
    msgTxt = re.sub( '\s+', ' ', msgTxt )
    # split the line into words
    words = msgTxt.split()
    # increase counters
    for word in words:
        # 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
        if len(word) >1: #drop single character words
            print '%s\t%d\t%s\t%s' % (word, 1, isSpam, msgID)
```

```
Overwriting mapper_t.py
```

In [300]:
```
%%writefile reducer_t.py
#!/usr/bin/env python
from operator import itemgetter
import sys, operator
import numpy as np
#from Decimal import *

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

# 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
    if msgID not in msgIDs:
        msgIDs[msgID] = isSpam

    if word in wordcount.keys():
        if isSpam ==0:
            wordcount[word][0]+=1
        if isSpam ==1:
            wordcount[word][1]+=1
    else:
        if isSpam ==0:
            wordcount[word]=[1,0]
        if isSpam ==1:
            wordcount[word]=[0,1]


### total count of all words in ham & spam:
total_wc_ham = float(sum( wordcount[i][0] for i in wordcount.keys()))
total_wc_spam = float(sum( wordcount[i][1] for i in wordcount.keys()))

## this section was change for Laplace Smoothing
for (key,value) in zip(wordcount.keys(), wordcount.values()): #/(1.0*n_total)):
    word_total = value[0] + value[1]
    p_word = float(word_total+1) / (total_wc_ham + total_wc_spam)
    print key, "%.9f" % (p_word*value[0]*0.56/total_wc_ham), "%.9f" %(p_word*value[1]
```

```
Overwriting reducer_t.py
```

In [301]:
```
%%writefile mapper_c.py
#!/usr/bin/env python
import sys, re, string, subprocess
import sys, operator, math
import numpy as np
# read the probability from HDFS
prob = {}
cat = subprocess.Popen(["hadoop", "fs", "-cat", "HW2_3/nbModel/part-*"], stdout=subp
for line in cat.stdout:
    word, p0, p1 = line.split()
    prob[word] = [p0, p1]

# get prior probability

prior = [0.56, 0.44]

# define regex for punctuation removal
regex = re.compile('[%s]' % re.escape(string.punctuation))
# input comes from STDIN (standard input)
for line in sys.stdin:
    # use subject and body
    msg = line.split('\t', 2)
    if len(msg) < 3:
        continue
    msgID, actualSPAMClass = msg[0], msg[1]

    msgTxt = regex.sub(' ', msg[2].lower())
    msgTxt = re.sub( '\t', ' ', msgTxt )
    msgTxt = re.sub( '\s+', ' ', msgTxt )
    # split the line into words
    words = msgTxt.split()
    prHAMGivenDoc = math.log(float(prior[0]))
    prSPAMGivenDoc = math.log(float(prior[1]))
    for word in words:
        if len(word) >1: #drop single letter words
            if word in prob:
                p0 = float(prob[word][0])
                p1 = float(prob[word][1])
                #print "probs", p0, p1, prHAMGivenDoc, prSPAMGivenDoc
                wordGivenHam = math.log(p0) if p0>0.0 else float('-inf')
                wordGivenSpam = math.log(p1) if p1>0.0 else float('-inf')
                if wordGivenHam != 0.0 and prHAMGivenDoc != float('-inf'):
                    prHAMGivenDoc = prHAMGivenDoc + wordGivenHam
                else:
                    prHAMGivenDoc = float('-inf')
                if wordGivenSpam != 0.0 and prSPAMGivenDoc !=float('-inf'):
                    prSPAMGivenDoc = prSPAMGivenDoc + wordGivenSpam
                else:
                    prSPAMGivenDoc=float('-inf')
            else:
                print '%s\t%s word not found in Multinomial Naive Bayes model lexico
                sys.exit("error[", word, "]is not in the Multinomial Naive Bayes mode
    predictedClass = 1 #SPAM
    if(prHAMGivenDoc > prSPAMGivenDoc):
        predictedClass = 0 #HAM
    if int(actualSPAMClass) == predictedClass:
        print actualSPAMClass, predictedClass, prHAMGivenDoc, prSPAMGivenDoc,0  #no e
    else:
        print actualSPAMClass, predictedClass, prHAMGivenDoc, prSPAMGivenDoc,1 # err
```

```
Overwriting mapper_c.py
```

In [302]:
```
%%writefile reducer_c.py
#!/usr/bin/python
from operator import itemgetter
import sys, operator, math
import numpy as np

import matplotlib.pyplot as plt
#import plotly.plotly as py
#import plotly.graph_objs as go

numberOfRecords = 0
NumberOfMisclassifications=0
prHAMGivenDoc = []
prSPAMGivenDoc = []
# input comes from STDIN
for line in sys.stdin:
    #print line
    # remove leading and trailing whitespace
    line = line.strip()
    toks = line.split(" ")
    # calculate the probabilities of HAM or SPAM of each email
    prHAMGivenDoc.append(math.exp(float(toks[2])))
    prSPAMGivenDoc.append(math.exp(float(toks[3])))
    # account for the result
    NumberOfMisclassifications = NumberOfMisclassifications + int(toks[4])
    numberOfRecords = numberOfRecords + 1

# calculate the overall error rate
# could also calcualte  the confusion matrix
print 'Error rate: %.4f' %(1.0*NumberOfMisclassifications/float(numberOfRecords))
print 'NumberOfMisclassifications %d, numberOfRecords%d'  %(NumberOfMisclassificatio


prHAMGivenDoc = np.array(prHAMGivenDoc)
prSPAMGivenDoc = np.array(prSPAMGivenDoc)
plt.hist(prHAMGivenDoc, bins=50, color='blue')
plt.hist(prSPAMGivenDoc, bins=50, color='red')
plt.show()
```

```
Overwriting reducer_c.py
```

In [303]:
```
# these work well, they just had a long output

!chmod a+x mapper_t.py
!chmod a+x reducer_t.py

#!cat enronemail_1h.txt|head -10 | ./mapper_t.py | ./reducer_t.py
```

In [304]:
```
!chmod a+x mapper_c.py
!chmod a+x reducer_c.py

!cat enronemail_1h.txt|head -15 | ./mapper_c.py | ./reducer_c.py
```

```
cat: write error: Broken pipe
Error rate: 0.0000
NumberOfMisclassifications 0, numberOfRecords15
```

```
In [ ]: !hdfs dfs -mkdir HW2_4
        !hdfs dfs -put enronemail_1h.txt HW2_4
        !hdfs dfs -rm -r HW2_4/nbModel
        !hdfs dfs -rm -r HW2_4/classifications

        # Run MNB training job
        !hadoop jar /usr/lib/hadoop/hadoop-streaming-2.7.2-amzn-1.jar  \
            -files /home/hadoop/Notebooks/Users/Nina/mapper_t.py,/home/hadoop/Notebooks/Users
            -mapper mapper_t.py \
            -reducer reducer_t.py \
            -input HW2_4/enronemail_1h.txt -output HW2_4/nbModel
```

```
In [ ]: !hdfs dfs -rm -r HW2_4/nbModel
        !hdfs dfs -rm -r HW2_4/classifications

        # Run MNB training job
        !hadoop jar /usr/lib/hadoop/hadoop-streaming-2.7.2-amzn-1.jar  \
            -files /home/hadoop/Notebooks/Users/Nina/mapper_t.py,/home/hadoop/Notebooks/Users
            -mapper mapper_c.py \
            -reducer reducer_c.py \
            -input HW2_3/enronemail_1h.txt -output HW2_3/classifications
```

```
In [ ]: !hdfs dfs -cat HW2_3/classifications/part-00000
```

We see that our results are consistent with teh previous exercise and with the previous homework.

HW 2.6 Benchmarking the results with scikit-learn

```
In [310]: # This tells matplotlib not to try opening a new window for each plot.
          %matplotlib inline

          # General libraries.
          import re
          import numpy as np
          from sklearn.naive_bayes import BernoulliNB
          from sklearn.naive_bayes import MultinomialNB
          from sklearn.feature_extraction.text import CountVectorizer
```

In [311]:
```
categories = ['SPAM', 'HAM']

docs = []
labels = []

with open('enronemail_1h.txt', 'r') as myfile:
    for line in myfile:
        labels.append(line.split("\t")[1])
        docs.append(line.split("\t")[2] + line.split("\t")[3])

docs=np.asarray(docs)
labels=np.asarray(labels)


#print 'labels shape:', labels.shape
#print 'docs shape:', docs.shape

vectorizer = CountVectorizer(min_df=2)
data_vec = vectorizer.fit_transform(docs)

MNB = MultinomialNB()
MNB.fit(data_vec, labels)

print ('Multinomial Naive Bayes score: ', 100 * MNB.score(data_vec,labels), '%' )
print ('Multinomial Naive Bayes error rate: ', 100*(1-MNB.score(data_vec,labels)), '%
```

```
('Multinomial Naive Bayes score: ', 98.0, '%')
('Multinomial Naive Bayes error rate: ', 2.0000000000000018, '%')
```

So, the default Multinomial Naive Bayes Algorithm from Scikit-Learn also classifies these emails with a 100% accuracy.

In [312]:
```
BNB = BernoulliNB()
BNB.fit(data_vec, labels)

print ('Bernoulli Naive Bayes score: ', 100 * BNB.score(data_vec,labels) , '%' )
print ('Bernoulli Naive Bayes error rate: ', 100*(1-BNB.score(data_vec,labels)), '%'
```

```
('Bernoulli Naive Bayes score: ', 84.0, '%')
('Bernoulli Naive Bayes error rate: ', 16.000000000000004, '%')
```

Here, we see that the Bernoulli Naive Bayes Algorithm doesn't perform as well as the Multinomial Naive Bayes.

One reason why the Multinomial Naive Bayes from scikit-learn doesn't perform as well as the one created with map-reduce is the smoothing. Without using the smoothing, even if a message has multiple features of spam, as long as it doesn't have one spam feature, its probability of being spam is evaluated as 0. However, our map-reduce-implemented classifier doesn't commit this mistake.

In [ ]:
```
```