DATSCIW261 ASSIGNMENT 1

MIDS UC Berkeley, Machine Learning at Scale

AUTHOR : Rajesh Thallam
EMAIL : rajesh.thallam@ischool.berkeley.edu
WEEK : 1
DATE : 15-Sep-15

HW1.0.0

Define big data. Provide an example of a big data problem in your domain of expertise.

I perceive **Big data** as a relative term referring to set of tools, technologies, frameworks and architectures designed to process and extract value from very large volumes of varieties of data by enabling high velocity data discovery or analysis. As discussed in the async lectures the problems that can be solved by **Big Data** tools and technologies are those that a traditional database management application or a bare commercial machine cannnot handle.

In my previous experience at a large Investment Bank, we used big data tools and technologies to calculate risk exposure to the bank when making a deal between two counterparties. The risk exposure calculations required to run approximately 300-1000 Monte Carlo simulations on last 10 years of credit risk and market risk data. This is totally a big data problem considering volume and variety of data. Traditionally the bank used C++ based implementation which had a lag of 4 days to report risk exposure numbers to the traders. After mapreduce based implementation (Hadoop and GreenPlum) this has considerably gone done down to 6 hours lag time.

HW1.0.1

In 500 words (English or pseudo code or a combination) describe how to estimate the bias, the variance, the irreducible error for a test dataset T when using polynomial regression models of degree 1, 2, 3, 4, 5 are considered. How would you select a model?

Bias, Variance and Irreducible Error
Bias The error due to bias is the difference between the expected prediction of the model and the correct value model is trying to predict. Imagine repeating the model building process more than once and each time new data is gathered (by resampling or acquiring new data) and run a new analysis creating a new model. Due to randomness in the underlying data sets, the resulting models will have a range of predictions. Bias measures how far off in general these models' predictions are from the correct value. If the expected prediction values way off from the actual values, bias is high and if they are similar, bias is low.

Variance The error due to variance is the variability of a model prediction for a given data point. Imagine repeating the model building process multiple times. The variance is how much the predictions for a given point vary between different realizations of the model. If the predictions are all similar (clustered), the variance is low and if they are spread out, the variance is high.

Irreducible Error Irreducible error is the noise term that cannot fundamentally be reduced by any model. We should be able to reduce both the bias and variance terms to 0 by calibrating the model and data. However, there is a tradeoff between minimizing the bias and minimizing the variance.

Model selection based on the order of polynomial regression Models
An ideal model is one which accurately captures the differences in its training data and also generalizes well to the unseen data. In case of polynomial order regression models

- **High-variance models (higher-order regression polynomials) may represent the training set well, but are at risk of overfitting to noisy or unseen data.**
- **High-bias models (lower-order regression polynomials) may produce simpler models and may underfit training data, failing to capture the differences in the training data.**
- **Low-bias/low-variance are usually more complex (higher-order regression polynomials) representing the training set more accurately but may also represent a large noise component overfitting the training set and failing to genralize to unseen data, and thereby making predictions less accurate.**

HW1.1.

Read through the provided control script (pNaiveBayes.sh) and all of its comments. When you are comfortable with their purpose and function, respond to the remaining homework questions below. A simple cell in the notebook with a print statmement with a "done" string will suffice here.

In [12]:
```
# HW 1.1 Read through the control script pNaiveBayes.sh and all its functions,
#        purpose and comments to become comfortable with the code.

def hw1_1():
    print "done"

hw1_1()
```

```
done
```

HW1.2.

Provide a mapper/reducer pair that, when executed by pNaiveBayes.sh will determine the number of occurrences of a single, user-specified word. Examine the word "assistance" and report your results. To do so, make sure that:

- mapper.py counts all occurrences of a single word, and
- reducer.py collates the counts of the single word.

Assumptions

1. **Both email body and subject is considered for the search**
2. **Removed punctuations, special characters from email content**

Mapper

In [13]:
```
%%writefile mapper.py
#!/usr/bin/python
import traceback
import sys
import re

from collections import Counter

# read input parameters
data_file = sys.argv[1]
words = sys.argv[2].split()

try:
    with open (data_file, "r") as emails:
        for email in emails:
            # split email by tab (\t)
            mail = email.split('\t')

            # handle missing email content
            if len(mail) == 3:
                mail.append(mail[2])
                mail[2] = ""
            assert len(mail) == 4

            # email id
            email_id = mail[0]
            # email content - remove special characters and punctuations
            content = re.sub('[^A-Za-z0-9\s]+', '', mail[2] + " " +  mail[3])

            find_words = re.compile("|".join(r"\b%s\b" % w for w in words))
            hits = Counter(re.findall(find_words, content))

            hits = {k: v for k, v in hits.iteritems()}

            # emit tuple delimited by |
            # (spam ind, content word count, word hit counts)
            print "{} | {}".format(email_id, hits)
except Exception:
    traceback.print_exc()
```

```
Overwriting mapper.py
```

Reducer

In [14]:
```
%%writefile reducer.py
#!/usr/bin/python
import traceback
import sys
import ast

from collections import Counter

# read input parameters
files = sys.argv[1:]

try:
    word_counts = {}

    # read each map output
    for f in files:
        with open (f, "r") as emails:
            for email in emails:
                # parse map out
                mail = email.split(" | ")
                # read word counts
                hits = ast.literal_eval(mail[1])

                # reduce phase/fold to calculate counts
                word_counts = dict(Counter(hits) + Counter(word_counts))

    # output of reduce phase
    print "{0: <50} | {1}".format("word", "count")
    print "{0: <50}-+-{1}".format("-" * 50, "-" * 8)
    for key, value in word_counts.iteritems():
        print "{0:<50} | {1}".format(key, value)
except Exception:
    traceback.print_exc()
```

```
Overwriting reducer.py
```

Set Permissions

In [15]:
```
!chmod a+x mapper.py
!chmod a+x reducer.py
!chmod a+x pNaiveBayes.sh
```

Driver Function

In [16]:
```
# HW 1.2  Mapper/reducer pair to determine the number of occurrences
#         of a single, user-specified word

def hw1_2():
    # run pNaiveBayes.sh
    !./pNaiveBayes.sh 4 "assistance"

    # display count on the screen
    print "output from mapper/reducer to determine the number of occurrences of word assistance \n"
    with open ("enronemail_1h.txt.output", "r") as f:
        print f.read()

    # CROSSCHECK
    print "output from command line mapper/reducer \n"
    ! grep assistance enronemail_1h.txt | awk -F'\t' '{print $3, $4}' | grep -o assistance | wc -l

hw1_2()
```

```
output from mapper/reducer to determine the number of occurrences of word assistance

word                                               | count
---------------------------------------------------+---------
assistance                                         | 10

output from command line mapper/reducer

10
```

HW1.3.

Provide a mapper/reducer pair that, when executed by pNaiveBayes.sh will classify the email messages by a single, user-specified word using the Naive Bayes Formulation. Examine the word "assistance" and report your results. To do so, make sure that:

- mapper.py
- reducer.py that performs a single word Naive Bayes classification

Assumptions

1. **As per the instructions, this mapper is used across all mappers for HW1.3 - HW1.5**
2. **Based on the instructions on LMS, only email body is considered for classification**
3. **Mapper and reducer takes care of classifiation based on single user specified word, multiple words or all words**

Mapper

In [17]:
```
%%writefile mapper.py
#!/usr/bin/python
import traceback
import sys
import re

from collections import Counter

# read input parameters
data_file = sys.argv[1]
words = sys.argv[2]

try:
    search_all = 0

    if words == "*":
        search_all = 1
        word_list = []
    else:
        word_list = words.split()

    with open (data_file, "r") as emails:
        for email in emails:
            # split email by tab (\t)
            mail = email.split('\t')

            # handle missing email content
            if len(mail) == 3:
                mail.append(mail[2])
                mail[2] = ""
            assert len(mail) == 4

            # email id
            email_id = mail[0]
            # spam/ham binary indicator
            is_spam = mail[1]
            # email content - remove special characters and punctuations
            #content = re.sub('[^A-Za-z0-9\s]+', '', mail[2] + " " +  mail[3])
            content = re.sub('[^A-Za-z0-9\s]+', '', mail[3])
            # count number of words
            content_wc = len(content.split())

            # find words with counts - works for single word or list of words
            if search_all == 1:
                hits = Counter(content.split())
            else:
                find_words = re.compile("|".join(r"\b%s\b" % w for w in word_list))
                hits = Counter(re.findall(find_words, content))

            hits = {k: v for k, v in hits.iteritems()}

            # emit tuple delimited by |
            # (spam ind, content word count, word hit counts)
            print "{} | {} | {} | {} | {}".format(email_id, is_spam, content_wc, word_list, hits)
except Exception:
    traceback.print_exc()
```

```
Overwriting mapper.py
```

Reducer

In [18]:
```
%%writefile reducer.py
```

```
#!/usr/bin/python
import traceback
import math
import sys
import ast

from collections import Counter

# read input parameters
files = sys.argv[1:]

try:
    spam_count = 0
    ham_count = 0
    spam_all_wc = 0
    ham_all_wc = 0
    spam_term_wc = {}
    ham_term_wc = {}
    pr_word_given_spam = {}
    pr_word_given_ham = {}

    # read each mapper output
    for f in files:
        with open (f, "r") as emails:
            for email in emails:
                # parse mapper output
                mail = email.split(" | ")
                # read spam/ham indicator, content word count,
                is_spam = int(mail[1])
                content_wc = int(mail[2])
                vocab = ast.literal_eval(mail[3])
                hits = ast.literal_eval(mail[4])

                # capture counts required for naive bayes probabilities
                if is_spam:
                    # spam mail count
                    spam_count += 1
                    # term count when spam
                    spam_term_wc = dict(Counter(hits) + Counter(spam_term_wc))
                    # all word count when spam
                    spam_all_wc += content_wc
                else:
                    # ham email count
                    ham_count += 1
                    # term count when ham
                    ham_term_wc = dict(Counter(hits) + Counter(ham_term_wc))
                    # all word count when ham
                    ham_all_wc += content_wc

    # vocab size
    vocab = dict(Counter(vocab) + Counter(spam_term_wc) + Counter(ham_term_wc))
    V = len(vocab) * 1.0
    print "vocab size = {}".format(V)

    # calculate priors
    pr_spam_prior = (1.0 * spam_count) / (spam_count + ham_count)
    pr_ham_prior = (1.0 - pr_spam_prior)
    pr_spam_prior = math.log10(pr_spam_prior)
    pr_ham_prior = math.log10(pr_ham_prior)

    # calculate conditional probabilites with laplace smoothing = 1
    # pr_word_given_class = ( count(w, c) + 1 ) / (count(c) + 1 * |V|)
    for word in vocab:
        pr_word_given_spam[word] = math.log10((spam_term_wc.get(word, 0) + 1.0) / (spam_all_wc + V))
        pr_word_given_ham[word] = math.log10((ham_term_wc.get(word, 0) + 1.0) / (ham_all_wc + V))

    print "/*log probabilities*/"
    print "pr_spam_prior = {}".format(pr_spam_prior)
    print "pr_ham_prior = {}".format(pr_ham_prior)

    print "\n"
    print "{0: <50} | {1} | {2}".format("ID", "TRUTH", "CLASS")
    print "{0: <50}-+-{1}-+-{2}".format("-" * 50, "-" * 7, "-" * 10)

    # spam/ham prediction using Multinomial Naive Bayes priors and conditional probabilities
    accuracy = []
    for f in files:
        with open (f, "r") as emails:
            for email in emails:
                # initialize
                word_count = 0
                pred_is_spam = 0
                pr_spam = pr_spam_prior
                pr_ham = pr_ham_prior

                # parse mapper output
```

```
                mail = email.split(" | ")
                email_id = mail[0]
                is_spam = int(mail[1])
                hits = ast.literal_eval(mail[4])

                # number of search words
                word_count = sum(hits.values())

                # probability for each class for a given email
                # argmax [ log P(C) + sum( P(Wi|C) ) ]
                for word in vocab:
                    pr_spam += (pr_word_given_spam.get(word, 0) * hits.get(word, 0))
                    pr_ham += (pr_word_given_ham.get(word, 0) * hits.get(word, 0))

                # predict based on maximum likelihood
                if pr_spam > pr_ham:
                    pred_is_spam = 1

                # calculate accuracy
                accuracy.append(pred_is_spam==is_spam)

                print '{0:<50} | {1:<7} | {2:<10}'.format(email_id, is_spam, pred_is_spam)

    print "\n"
    print "/*accuracy*/"
    print "accuracy = {:.2f}".format(sum(accuracy) / float(len(accuracy)))

except Exception:
    traceback.print_exc()
```

```
Overwriting reducer.py
```

Driver Function

In [19]:
```
# HW 1.3  Mapper/reducer pair to classify the email messages by a single,
#         user-specified word using the Naive Bayes Formulation

def hw1_3(word):
    # run pNaiveBayes.sh
    !./pNaiveBayes.sh 4 "{word}"

    # reducer output on the screen
    print "Accuracy of the Naive Bayes classifier with single word '{}'\n".format(word)
    with open ("enronemail_1h.txt.output", "r") as f:
        print f.read()

hw1_3("assistance")
```

```
Accuracy of the Naive Bayes classifier with single word 'assistance'

vocab size = 1.0
/*log probabilities*/
pr_spam_prior = -0.356547323514
pr_ham_prior = -0.251811972994


ID                                                 | TRUTH | CLASS
---------------------------------------------------+---------+-----------
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                            | 0       | 0
0002.2001-05-25.SA_and_HP                          | 1       | 0
0002.2003-12-18.GP                                 | 1       | 0
0002.2004-08-01.BG                                 | 1       | 1
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                            | 0       | 0
0003.2003-12-18.GP                                 | 1       | 0
0003.2004-08-01.BG                                 | 1       | 0
0004.1999-12-10.kaminski                           | 0       | 1
0004.1999-12-14.farmer                             | 0       | 0
0004.2001-04-02.williams                           | 0       | 0
0004.2001-06-12.SA_and_HP                          | 1       | 0
0004.2004-08-01.BG                                 | 1       | 0
0005.1999-12-12.kaminski                           | 0       | 1
0005.1999-12-14.farmer                             | 0       | 0
0005.2000-06-06.lokay                              | 0       | 0
0005.2001-02-08.kitchen                            | 0       | 0
```

0005.2001-06-23.SA_and_HP	1	0
0005.2003-12-18.GP	1	0
0006.1999-12-13.kaminski	0	0
0006.2001-02-08.kitchen	0	0
0006.2001-04-03.williams	0	0
0006.2001-06-25.SA_and_HP	1	0
0006.2003-12-18.GP	1	0
0006.2004-08-01.BG	1	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	0	0
0007.2003-12-18.GP	1	0
0007.2004-08-01.BG	1	0
0008.2001-02-09.kitchen	0	0
0008.2001-06-12.SA_and_HP	1	0
0008.2001-06-25.SA_and_HP	1	0
0008.2003-12-18.GP	1	0
0008.2004-08-01.BG	1	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	0	0
0009.2001-06-26.SA_and_HP	1	0
0009.2003-12-18.GP	1	0
0010.1999-12-14.farmer	0	0
0010.1999-12-14.kaminski	0	0
0010.2001-02-09.kitchen	0	0
0010.2001-06-28.SA_and_HP	1	1
0010.2003-12-18.GP	1	0
0010.2004-08-01.BG	1	0
0011.1999-12-14.farmer	0	0
0011.2001-06-28.SA_and_HP	1	1
0011.2001-06-29.SA_and_HP	1	0
0011.2003-12-18.GP	1	0
0011.2004-08-01.BG	1	0
0012.1999-12-14.farmer	0	0
0012.1999-12-14.kaminski	0	0
0012.2000-01-17.beck	0	0
0012.2000-06-08.lokay	0	0
0012.2001-02-09.kitchen	0	0
0012.2003-12-19.GP	1	0
0013.1999-12-14.farmer	0	0
0013.1999-12-14.kaminski	0	0
0013.2001-04-03.williams	0	0
0013.2001-06-30.SA_and_HP	1	0
0013.2004-08-01.BG	1	1
0014.1999-12-14.kaminski	0	0
0014.1999-12-15.farmer	0	0
0014.2001-02-12.kitchen	0	0
0014.2001-07-04.SA_and_HP	1	0
0014.2003-12-19.GP	1	0
0014.2004-08-01.BG	1	0
0015.1999-12-14.kaminski	0	0
0015.1999-12-15.farmer	0	0
0015.2000-06-09.lokay	0	0
0015.2001-02-12.kitchen	0	0
0015.2001-07-05.SA_and_HP	1	0
0015.2003-12-19.GP	1	0
0016.1999-12-15.farmer	0	0
0016.2001-02-12.kitchen	0	0
0016.2001-07-05.SA_and_HP	1	0
0016.2001-07-06.SA_and_HP	1	0
0016.2003-12-19.GP	1	0
0016.2004-08-01.BG	1	0
0017.1999-12-14.kaminski	0	0
0017.2000-01-17.beck	0	0
0017.2001-04-03.williams	0	0
0017.2003-12-18.GP	1	0
0017.2004-08-01.BG	1	0
0017.2004-08-02.BG	1	0
0018.1999-12-14.kaminski	0	0
0018.2001-07-13.SA_and_HP	1	1
0018.2003-12-18.GP	1	1

/*accuracy*/
accuracy = 0.60

HW1.4.

Provide a mapper/reducer pair that, when executed by pNaiveBayes.sh will classify the email messages by a list of one or more user-specified words. Examine the words "assistance", "valium", and "enlargementWithATypo" and report your results. To do so, make sure that

- mapper.py counts all occurrences of a list of words, and
- reducer.py
that performs a single word Naive Bayes classification

Assumptions

1. **For this part of homework I used same mapper and reducer as HW 1.3**
2. **Based on the instructions on LMS, only email body is considered for classification**

Driver Function

In [20]:
```
# HW 1.4  Mapper/reducer pair to classify the email messages by a
#         list of one or more user-specified words using the
#         Naive Bayes Formulation

def hw1_4(words):
    # run pNaiveBayes.sh
    !./pNaiveBayes.sh 4 "{words}"

    # reducer output on the screen
    print "Accuracy of the Naive Bayes classifier with list of words '{}' is ".format(words)
    with open ("enronemail_1h.txt.output", "r") as f:
        print f.read()

hw1_4("assistance valium enlargementWithATypo")
```

```
Accuracy of the Naive Bayes classifier with list of words 'assistance valium enlargementWithATypo' is
vocab size = 3.0
/*log probabilities*/
pr_spam_prior = -0.356547323514
pr_ham_prior = -0.251811972994


ID                                                  | TRUTH | CLASS
----------------------------------------------------+-------+-----------
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                             | 0     | 0
0002.2001-05-25.SA_and_HP                           | 1     | 0
0002.2003-12-18.GP                                  | 1     | 0
0002.2004-08-01.BG                                  | 1     | 1
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                             | 0     | 0
0003.2003-12-18.GP                                  | 1     | 0
0003.2004-08-01.BG                                  | 1     | 0
0004.1999-12-10.kaminski                            | 0     | 1
0004.1999-12-14.farmer                              | 0     | 0
0004.2001-04-02.williams                            | 0     | 0
0004.2001-06-12.SA_and_HP                           | 1     | 0
0004.2004-08-01.BG                                  | 1     | 0
0005.1999-12-12.kaminski                            | 0     | 1
0005.1999-12-14.farmer                              | 0     | 0
0005.2000-06-06.lokay                               | 0     | 0
0005.2001-02-08.kitchen                             | 0     | 0
0005.2001-06-23.SA_and_HP                           | 1     | 0
0005.2003-12-18.GP                                  | 1     | 0
0006.1999-12-13.kaminski                            | 0     | 0
0006.2001-02-08.kitchen                             | 0     | 0
0006.2001-04-03.williams                            | 0     | 0
0006.2001-06-25.SA_and_HP                           | 1     | 0
0006.2003-12-18.GP                                  | 1     | 0
0006.2004-08-01.BG                                  | 1     | 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                             | 0     | 0
0007.2003-12-18.GP                                  | 1     | 0
0007.2004-08-01.BG                                  | 1     | 0
```

```
0008.2001-02-09.kitchen                 | 0       | 0
0008.2001-06-12.SA_and_HP               | 1       | 0
0008.2001-06-25.SA_and_HP               | 1       | 0
0008.2003-12-18.GP                      | 1       | 0
0008.2004-08-01.BG                      | 1       | 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                 | 0       | 0
0009.2001-06-26.SA_and_HP               | 1       | 0
0009.2003-12-18.GP                      | 1       | 1
0010.1999-12-14.farmer                  | 0       | 0
0010.1999-12-14.kaminski                | 0       | 0
0010.2001-02-09.kitchen                 | 0       | 0
0010.2001-06-28.SA_and_HP               | 1       | 1
0010.2003-12-18.GP                      | 1       | 0
0010.2004-08-01.BG                      | 1       | 0
0011.1999-12-14.farmer                  | 0       | 0
0011.2001-06-28.SA_and_HP               | 1       | 1
0011.2001-06-29.SA_and_HP               | 1       | 0
0011.2003-12-18.GP                      | 1       | 0
0011.2004-08-01.BG                      | 1       | 0
0012.1999-12-14.farmer                  | 0       | 0
0012.1999-12-14.kaminski                | 0       | 0
0012.2000-01-17.beck                    | 0       | 0
0012.2000-06-08.lokay                   | 0       | 0
0012.2001-02-09.kitchen                 | 0       | 0
0012.2003-12-19.GP                      | 1       | 0
0013.1999-12-14.farmer                  | 0       | 0
0013.1999-12-14.kaminski                | 0       | 0
0013.2001-04-03.williams                | 0       | 0
0013.2001-06-30.SA_and_HP               | 1       | 0
0013.2004-08-01.BG                      | 1       | 1
0014.1999-12-14.kaminski                | 0       | 0
0014.1999-12-15.farmer                  | 0       | 0
0014.2001-02-12.kitchen                 | 0       | 0
0014.2001-07-04.SA_and_HP               | 1       | 0
0014.2003-12-19.GP                      | 1       | 0
0014.2004-08-01.BG                      | 1       | 0
0015.1999-12-14.kaminski                | 0       | 0
0015.1999-12-15.farmer                  | 0       | 0
0015.2000-06-09.lokay                   | 0       | 0
0015.2001-02-12.kitchen                 | 0       | 0
0015.2001-07-05.SA_and_HP               | 1       | 0
0015.2003-12-19.GP                      | 1       | 0
0016.1999-12-15.farmer                  | 0       | 0
0016.2001-02-12.kitchen                 | 0       | 0
0016.2001-07-05.SA_and_HP               | 1       | 0
0016.2001-07-06.SA_and_HP               | 1       | 0
0016.2003-12-19.GP                      | 1       | 0
0016.2004-08-01.BG                      | 1       | 0
0017.1999-12-14.kaminski                | 0       | 0
0017.2000-01-17.beck                    | 0       | 0
0017.2001-04-03.williams                | 0       | 0
0017.2003-12-18.GP                      | 1       | 0
0017.2004-08-01.BG                      | 1       | 1
0017.2004-08-02.BG                      | 1       | 0
0018.1999-12-14.kaminski                | 0       | 0
0018.2001-07-13.SA_and_HP               | 1       | 1
0018.2003-12-18.GP                      | 1       | 1


/*accuracy*/
accuracy = 0.62
```

HW1.5.

Provide a mapper/reducer pair that, when executed by pNaiveBayes.sh will classify the email messages by all words present. To do so, make sure that

- mapper.py counts all occurrences of all words, and
- reducer.py performs a word-distribution-wide Naive Bayes classification

Assumptions

1. **For this part of homework I used same mapper and reducer as HW 1.3**
2. **Based on the instructions on LMS, only email body is considered for classification**

Driver Function

In [21]:
```
# HW 1.5  Mapper/reducer pair to classify the email messages by a
#         all words present to perform a word-distribution-wide Naive
#         Bayes classification

def hw1_5(words):
    # run pNaiveBayes.sh
    !./pNaiveBayes.sh 4 "{words}"

    # reducer output on the screen
    print "Accuracy of the Naive Bayes classifier with list of words '{}' is ".format(words)
    with open ("enronemail_1h.txt.output", "r") as f:
        print f.read()

hw1_5("*")
```

```
Accuracy of the Naive Bayes classifier with list of words '*' is
vocab size = 5680.0
/*log probabilities*/
pr_spam_prior = -0.356547323514
pr_ham_prior = -0.251811972994


ID                                                  | TRUTH | CLASS
----------------------------------------------------+-------+-----------
0001.1999-12-10.farmer                              | 0     | 0
0001.1999-12-10.kaminski                            | 0     | 1
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                             | 0     | 0
0002.2001-05-25.SA_and_HP                           | 1     | 1
0002.2003-12-18.GP                                  | 1     | 1
0002.2004-08-01.BG                                  | 1     | 1
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                             | 0     | 0
0003.2003-12-18.GP                                  | 1     | 1
0003.2004-08-01.BG                                  | 1     | 1
0004.1999-12-10.kaminski                            | 0     | 0
0004.1999-12-14.farmer                              | 0     | 0
0004.2001-04-02.williams                            | 0     | 0
0004.2001-06-12.SA_and_HP                           | 1     | 1
0004.2004-08-01.BG                                  | 1     | 1
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                             | 0     | 0
0005.2001-06-23.SA_and_HP                           | 1     | 1
0005.2003-12-18.GP                                  | 1     | 1
0006.1999-12-13.kaminski                            | 0     | 0
0006.2001-02-08.kitchen                             | 0     | 0
0006.2001-04-03.williams                            | 0     | 0
0006.2001-06-25.SA_and_HP                           | 1     | 1
0006.2003-12-18.GP                                  | 1     | 1
0006.2004-08-01.BG                                  | 1     | 1
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                             | 0     | 0
0007.2003-12-18.GP                                  | 1     | 1
0007.2004-08-01.BG                                  | 1     | 1
0008.2001-02-09.kitchen                             | 0     | 0
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                                  | 1     | 1
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                             | 0     | 0
0009.2001-06-26.SA_and_HP                           | 1     | 1
0009.2003-12-18.GP                                  | 1     | 1
0010.1999-12-14.farmer                              | 0     | 0
0010.1999-12-14.kaminski                            | 0     | 0
0010.2001-02-09.kitchen                             | 0     | 0
0010.2001-06-28.SA_and_HP                           | 1     | 1
0010.2003-12-18.GP                                  | 1     | 0
0010.2004-08-01.BG                                  | 1     | 1
0011.1999-12-14.farmer                              | 0     | 0
0011.2001-06-28.SA_and_HP                           | 1     | 1
0011.2001-06-29.SA_and_HP                           | 1     | 1
0011.2003-12-18.GP                                  | 1     | 1
```

```
0011.2004-08-01.BG                          | 1       | 1
0012.1999-12-14.farmer                      | 0       | 0
0012.1999-12-14.kaminski                    | 0       | 0
0012.2000-01-17.beck                        | 0       | 0
0012.2000-06-08.lokay                       | 0       | 0
0012.2001-02-09.kitchen                     | 0       | 0
0012.2003-12-19.GP                          | 1       | 1
0013.1999-12-14.farmer                      | 0       | 0
0013.1999-12-14.kaminski                    | 0       | 0
0013.2001-04-03.williams                    | 0       | 0
0013.2001-06-30.SA_and_HP                   | 1       | 1
0013.2004-08-01.BG                          | 1       | 1
0014.1999-12-14.kaminski                    | 0       | 0
0014.1999-12-15.farmer                      | 0       | 0
0014.2001-02-12.kitchen                     | 0       | 0
0014.2001-07-04.SA_and_HP                   | 1       | 1
0014.2003-12-19.GP                          | 1       | 1
0014.2004-08-01.BG                          | 1       | 1
0015.1999-12-14.kaminski                    | 0       | 0
0015.1999-12-15.farmer                      | 0       | 0
0015.2000-06-09.lokay                       | 0       | 0
0015.2001-02-12.kitchen                     | 0       | 0
0015.2001-07-05.SA_and_HP                   | 1       | 1
0015.2003-12-19.GP                          | 1       | 1
0016.1999-12-15.farmer                      | 0       | 0
0016.2001-02-12.kitchen                     | 0       | 0
0016.2001-07-05.SA_and_HP                   | 1       | 1
0016.2001-07-06.SA_and_HP                   | 1       | 1
0016.2003-12-19.GP                          | 1       | 1
0016.2004-08-01.BG                          | 1       | 1
0017.1999-12-14.kaminski                    | 0       | 0
0017.2000-01-17.beck                        | 0       | 0
0017.2001-04-03.williams                    | 0       | 0
0017.2003-12-18.GP                          | 1       | 1
0017.2004-08-01.BG                          | 1       | 1
0017.2004-08-02.BG                          | 1       | 1
0018.1999-12-14.kaminski                    | 0       | 0
0018.2001-07-13.SA_and_HP                   | 1       | 1
0018.2003-12-18.GP                          | 1       | 1


/*accuracy*/
accuracy = 0.98
```

HW1.6.

Benchmark your code with the Python SciKit-Learn implementation of Naive Bayes

- **Run the Multinomial Naive Bayes algorithm (using default settings) from SciKit-Learn over the same training data used in HW1.5 and report the Training error (please note some data preparation might be needed to get the Multinomial Naive Bayes algorithm from SkiKit-Learn to run over this dataset)**
- **Run the Bernoulli Naive Bayes algorithm from SciKit-Learn (using default settings) over the same training data used in HW1.5 and report the Training error**
- **Run the Multinomial Naive Bayes algorithm you developed for HW1.5 over the same data used HW1.5 and report the Training error**
- **Please prepare a table to present your results**
- **Explain/justify any differences in terms of training error rates over the dataset in HW1.5 between your Multinomial Naive Bayes implementation (in Map Reduce) versus the Multinomial Naive Bayes implementation in SciKit-Learn**
- **Discuss the performance differences in terms of training error rates over the dataset in HW1.5 between the Multinomial Naive Bayes implementation in SciKit-Learn with the Bernoulli Naive Bayes implementation in SciKit-Learn**

Assumptions

1. **Based on the instructions on LMS, only email body is considered for classification**

In [22]:
```
import re
import numpy as np

# import scikit learn libraries
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer

def hw1_6():
    # Structure to Load Data
    train_X = []
    train_Y = []

    with open ("enronemail_1h.txt", "r") as emails:
        for email in emails:
            # split email by tab (\t)
            mail = email.split('\t')

            # handle missing email content
            if len(mail) == 3:
                mail.append(mail[2])
                mail[2] = ""
            assert len(mail) == 4

            # email id
            email_id = mail[0]
            # spam/ham binary indicator
            is_spam = mail[1]
            # email content - remove special characters and punctuations
            #content = re.sub('[^A-Za-z0-9\s]+', '', mail[2] + " " +  mail[3])
            content = re.sub('[^A-Za-z0-9\s]+', '', mail[3])

            train_Y.append(is_spam)
            train_X.append(content)

    train_Y = np.asarray(train_Y)

    # mail tokenizer
    vector = CountVectorizer()
    tokens = vector.fit_transform(train_X)

    print "vocabulary size = {} words.".format(tokens.shape[1])
    print "non-zero words per email = {0:.2f}".format(tokens.nnz / float(tokens.shape[0]))

    # multinomial naive bayes
    print "Multinomial Naive Bayes"
    mnb = MultinomialNB()
    mnb.fit(tokens, train_Y)
    print "Accuracy = {:.2f}".format(mnb.score(tokens, train_Y))
    print "Training error = {:.2f}".format(sum(train_Y != mnb.predict(tokens)) /float(train_Y.shape[0]))

    # bernoulli naive bayes
    print "Bernoulli Naive Bayes"
    bnb = BernoulliNB()
    bnb.fit(tokens, train_Y)
    print "Accuracy = {:.2f}".format(bnb.score(tokens, train_Y))
    print "Training error = {:.2f}".format(sum(train_Y != bnb.predict(tokens)) /float(train_Y.shape[0]))

    print
    model_list = { "Models": ["scikit multinomial NB", "scikit Bernoulli NB", "MapReduce Multinomial NB"],
                   "Accuracy": ["0.98", "0.79", "0.98"],
                   "Training Error Rate": ["0.02", "0.21", "0.02"]
                 }

    print "{0: <25}-+-{1: <25}-+-{2: <25}-+-{3: <25}".format("-" * 25, "-" * 25, "-" * 25, "-" * 25)
    for k, v in model_list.iteritems():
        print "{0: <25} | {1: <25} | {2: <25} | {3: <25}".format(k, v[0], v[1], v[2])
        print "{0: <25}-+-{1: <25}-+-{2: <25}-+-{3: <25}".format("-" * 25, "-" * 25, "-" * 25, "-" * 25)

hw1_6()
```

```
vocabulary size = 5644 words.
non-zero words per email = 146.99
Multinomial Naive Bayes
Accuracy = 0.98
Training error = 0.02
Bernoulli Naive Bayes
Accuracy = 0.79
Training error = 0.21
```

Models	scikit multinomial NB	scikit Bernoulli NB	MapReduce Multinomial NB
Training Error Rate	0.02	0.21	0.02
Accuracy	0.98	0.79	0.98

Reporting Results

1. Difference in performance between MapReduce Multinomial Naive Bayes vs. SciKit-Learn Multinomial Naive Bayes

MapReduce implementation of Multinomial Naive Bayes performed same as Scikit-Learn Multinomial Naive Bayes classifier and classifies the training data at 98% accuracy. I expected mapreduce implementation would suffer from higher error rate than scikit-learn as the mapreduce implementation did not consider stemming, removing stop words in the vocabulary. Even though the vocabulary size varies between the two, I did not observe any change in the training error rate of the classifier in both the implementations.

2. Difference in performance between MapReduce Multinomial Naive Bayes vs. SciKit-Learn Bernoulli Naive Bayes

MapReduce implementation of Multinomial Naive Bayes outperforms scikit-learn Bernoulli Naive Bayes (BNB) classifier and mapreduce implementation classifies the training data at 98% accuracy whereas Bernoulli has 79% accuracy. This is something I was expecting to observe because BNB does not take frequency of occurence of words in an email into account and instead considers whether a word is present or not present (binary) in the email. Since MNB accounts for additional feature of frequency of words, I expect classifier has more power and hence better training error rate than BNB.

**** -- END OF ASSIGNMENT 1 -- ****