CS440 Assignment 3 Report

All the work is done together.

Xuan Wang Shuo Feng Yuchen Qian
Xwang182 sfeng15 yqian14

Part 1.1

In this part of assignment, we implemented a training program to identify of testing images.

Here is our data structure:

myArray[10][28][28]: store the point distribution of each digit
confusionMatrix[10][10]: store the accuracy of each digit
highest[10], lowest[10]: store the first 10 highest/lowest accuracy digit

Here is out pseudo-code:

Training part:

```
For each line in traing-image and traing-label
    Line = line in training-image
    Number = digit corresponding to the traing-label
    Read 28 lines to get the 28*28 squrare in traing-image
        If there is a “#” or “+”:
            We increase the occurrence:(myArray[number][x][y]) by 1
```

At last, we smooth the result and divide each occurrence of digit by the total number occurred of that digit.

Testing Part:

For each line in testing-image and testing-label

Line = line in testing-image

Number = digit corresponding to the testing-label

Read 28 lines to get the 28*28 square in testing-image

If there is a “#” or “+”:

Calculate the possibility of that print to form a certain digit.

```
for(int k=0;k < 10;k++){
    predict[k] = predict[k] + log(myArray[k][i][j]);
}
```

after calculation, we calculate the most likely digit by getting the maximum in the predict[k];

Last, we compare with the real digit in the testing-label and our estimated result and store them into our confusionMatrix.

We also get the first 10 highest and lowest accuracy digit.

Result:

Confusion matrix:

For simplicity, we omit the row number ranging from 0~9.

Row number---real digit, column number---our estimate

For index i,j, we define that class i is detected as class j. For example, matrix[2][8] means the rate for digit 2 is recognized as digit 8.

0	1	2	3	4	5	6	7	8	9
97.78%	0.00%	0.00%	0.00%	0.00%	0.00%	1.11%	0.00%	1.11%	0.00%
0.00%	87.96%	0.00%	3.70%	0.00%	0.00%	0.93%	0.00%	7.41%	0.00%
11.65%	0.97%	71.17%	1.65%	0.00%	0.00%	6.80%	1.94%	5.83%	0.00%
6.00%	0.00%	0.00%	84.00%	0.00%	0.00%	2.00%	2.00%	4.00%	2.00%
4.67%	0.00%	0.00%	0.93%	74.77%	0.00%	6.54%	0.93%	2.80%	9.35%
3.70%	0.00%	0.00%	2.61%	3.26%	71.52%	3.26%	1.09%	10.13%	1.43%
16.48%	1.10%	0.00%	0.00%	4.40%	0.00%	74.73%	0.00%	3.30%	0.00%
4.72%	2.83%	3.77%	0.94%	0.94%	0.00%	1.89%	76.42%	5.66%	2.83%
9.71%	0.00%	0.00%	12.33%	0.00%	0.00%	3.88%	2.91%	71.17%	0.00%
4.30%	0.00%	0.00%	4.70%	13.00%	0.00%	1.00%	5.00%	2.60%	71.40%

The classification rate is the diagonal value in the matrix. Matrix[0][0] is the classification rate for digit 0.

Highest test examples for each digit:

```
      +###+
     +#####+
    +#######++
    +#########
   +#####+#####+
  +#####+ ++###+
 +######+   +###+
+#######+   +####+
+####++#+    +###+
####+  +     +###+
###+         +###+
###+         +###+
###+        +####+
###         #####+
###+      ++#####+
###+    ++######+
####+#########++
++############+
  +##########++
   +++####++
```

```
        +#####+
        +#####
        +####+
        ####+
       +####+
       +###+
      +####+
      +####+
      +####+
      #####+
     +#####+
     ######+
     #####+
     #####+
     ####+
     ####+
     ####
     ####
     ##++
     #++
```

```
       ++#####++
      +##########++
     +#############+
     +###+++++######+
      ++        ++###+
                 +###+
               +####+
              ++####
             +#####+
            +#####+
          +#####+
        ++#####+
      ++######+
     +######+
   ++####++         ++
  +######+++++++##++##
  ###################+
  ################+++
  ++###########++
     +++


+      +++
 ##++++####+
 +##########++
  ############++
  +++      +###+
         +######
       +########
        ######++
        +###++
         ####+
         ++####+
           +####+
             ++##+
               +##+
       ++       +##+
        ++++++++####+
         ++#########+
          ++#########
               ++++++
```

```
     ++
     ##+
     ##+  ++
    +##+  +#+
   +###+  +##+
   +##+   +##+
  +##+    +##+
 +###     ###+
 ###+     ###
+###++++++###++++++
###################+
+###################
  +++++++###++++++++
         ###+
         ###
         ###+
         ###+
         +##+
         +##+
          ++




     +#########+
    +###########+
    ############+
    ############
    ####+++#####
    +##+     +++
    +###++++
     ########+
     #########+
     ##########+
     +##########
      +++  ++###+
             +###
 ++           ###
+###++       +###
 ######++   +###+
 ++#############
  ++###########+
    +#########+
      ++####++
```

```
         ###+
       +####+
      +####++
    ++#####
   +#####+
  +#####+
  +####+
 +####+
+########++++
+############++
+##############++
################+
####+####++++#####+
#### ++++    ++####+
####++         +###+
+#####+        +###+
 +########++++#####+
  +###############+
   +#############++
     ++++######++




             +++
          +++##+
+#+++++++#####+
+#########+###+
 ++++++++ +##+
          +##
         +##+
         +##
         +#+
        +##
        +#+
        ##+
       +##+
       +#+
       ##+
       ##+
      +##+
      +##
     +##+
```

```
  ++
 +###+ +++###+
 ####++########+
 +#############+
 +#####+++ +##+
  ####+    +##+
  +####+  +##+
   +#### +##+
   +#######+
    +######
     #####+
    +######
    #######+
   +#######+
  +###++###+
 +###+++###+
+###++####+
+########+
+#######+
+###+++
```

```
   +#####++
  ++#######+
  +########+
  +##+  ###+
 +##++  ####+
+###+  +####+
+###  +#####
+####+#####+
+###########+
+###########+
 +##+++++###+
  ++    +###+
        +###+
        +###+
        +####+
        +####+
         ####+
        +####+
        +####+
         +##+
```

Lowest test examples for each digit:

```
++
##++++
######++++
+#########++++++
 +#++############+++
 +#+   ++++#########
 +#+         +++###+
 +#+           +##+
 +#+           +#+
              +##+
              +##+
              ###+
              ###+
              ###+
              +##+
```

```
 ++
 ##
 ##
 ##
 ##
 ##
 ##
 ##
+#+
 #+
 #+
 ##
 #+
 #+
 #+
 #+
 ##
 ##
 ##
 +#
```

```
+++++
 +++#++
     ++#+
       +#+
        +#+
         #++
         +#+
          #+
          +#
           #+
          +#+
           #+
          +#
          ++
          #+
         +#
        +#+
   +++++#+
  +######+++
   +#+++++#+++++++


      +#+
      +#+
      +#+
      +##+
      +##+
      +##+
      +##+
      +##+
      +##+
       +#+
       +#+
        ##+
        ###
        +##
        +##+
         +#+
          +#
          +#
          +#+
            #+
```

```
            #+
           +#+
           +#
  ++       +#
  #        #+
 ++        #+
 #+        #+
 #         #+
++        +#+
+#+++#######
 +++++++  +#
          +#
          ++
          #+
          #+
          #+
          #
         +#
         +#
         +#
```

```
       +++##+
   +++######+
   +#####++++
  +####+
 +###+
 +##+
+##+
 ##+
 +##+
 ++###+++
   ++#####++
       ++###+
         ++##+
          ++##+
            +##
    ++      +##
    +#+   +####
    +##++#####+
     +#######+
      +++#+++
```

```
  +++
  +##+
   +#+
    +#+
    +#+
    +#+
    +#+
    +#
   +#+
   +#+
   #+
  +#
 +#+          +++++++
 ##+++++###########++
 #######++++++
 +++++++
```

```
 ##+
+####++
 ++++##+
     ++#+
       +#+
        +#+
         #+
         ##
         +#
         ##
         ##
        +#+
        ##+
       +#+
      +##
      +#+
     +#+
     +#+
     ++
     ++
```

```
    +#+
   +##+
   +##+
    +##+
     ##+
     ###
     ###+
     ###+
     +##+
     +##+
     +##+
      ###
      +##+
      +##+
      +##+
      +##+
      +###+
       +##+
       +##+
        ++
```

```
       +##
      +##+
     ++#+
    +#+#+
+++##+ #+
+##++  ++
+++    ++
       +#
        #
        #+
        +#
     ++++##+
    +###+##+
     ++  +#
          #+
          +#
          +#+
           #+
           +#+
            +
```

Most four confusion Pairs:

1st worst confused class: 6->0 16.48%

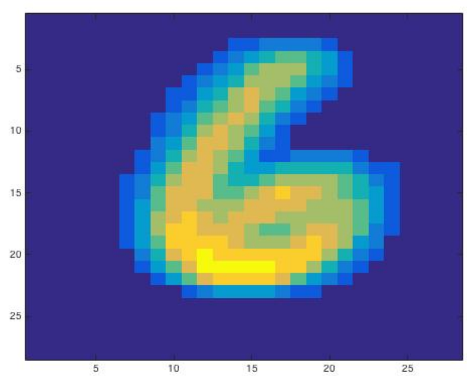
2nd worst confused class: 9->4 13.00%

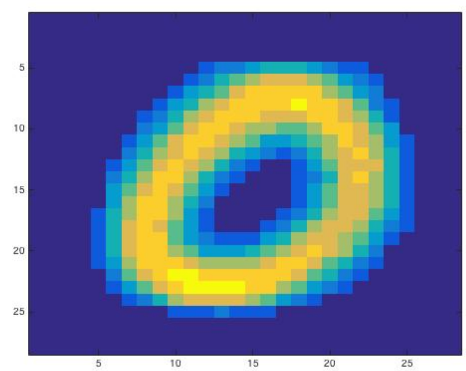
3rd worst confused class: 8->3 12.33%

4th worst confused class: 2->0 11.65%

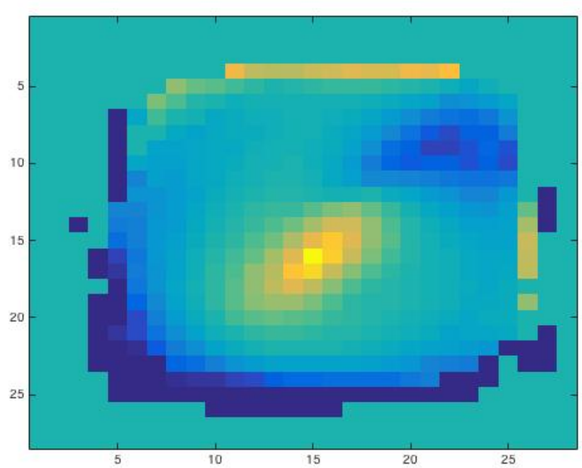
6->0:

6 & 0 feature likelihood



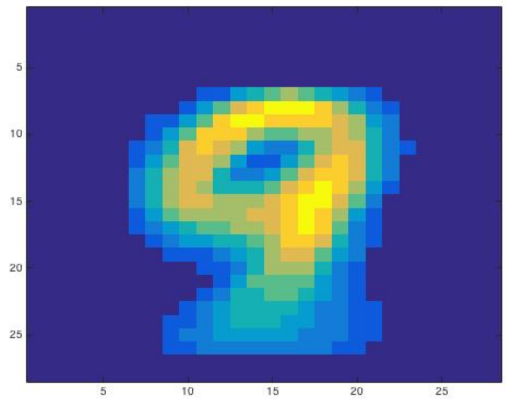


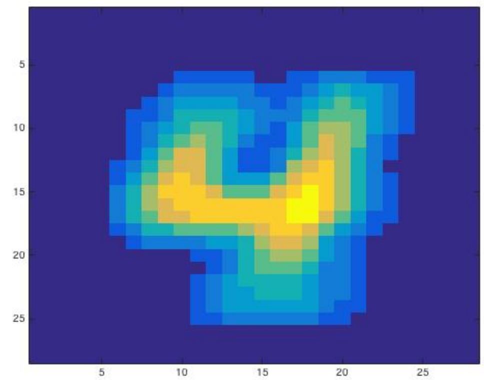
Odd ratios:



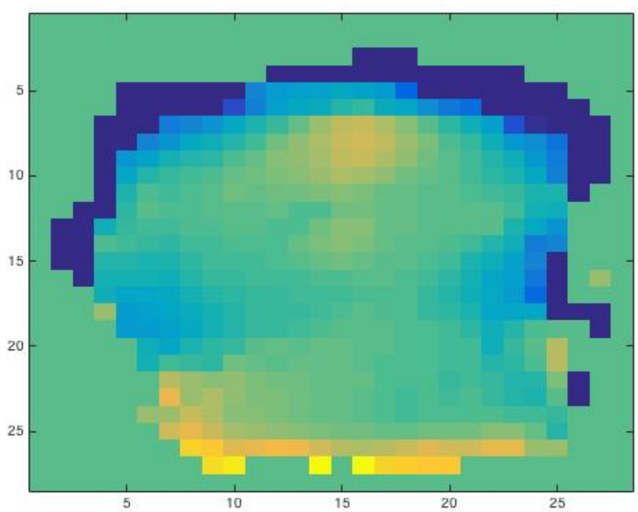
9->4:

9 & 4 feature likelihood



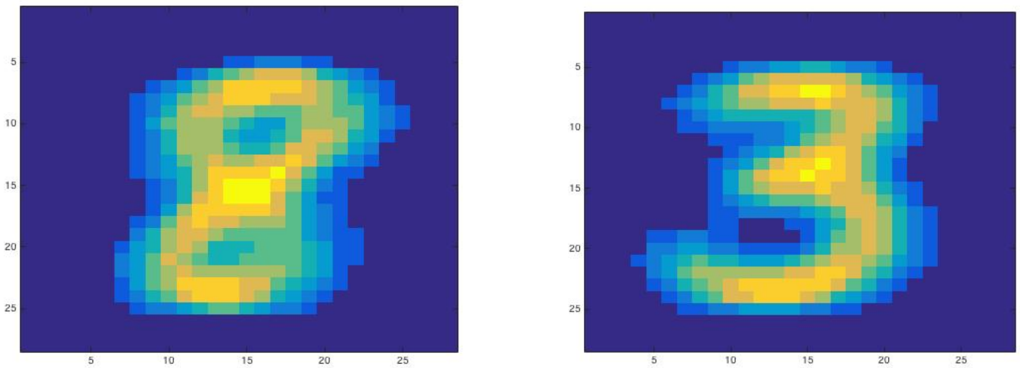


Odd ratios:

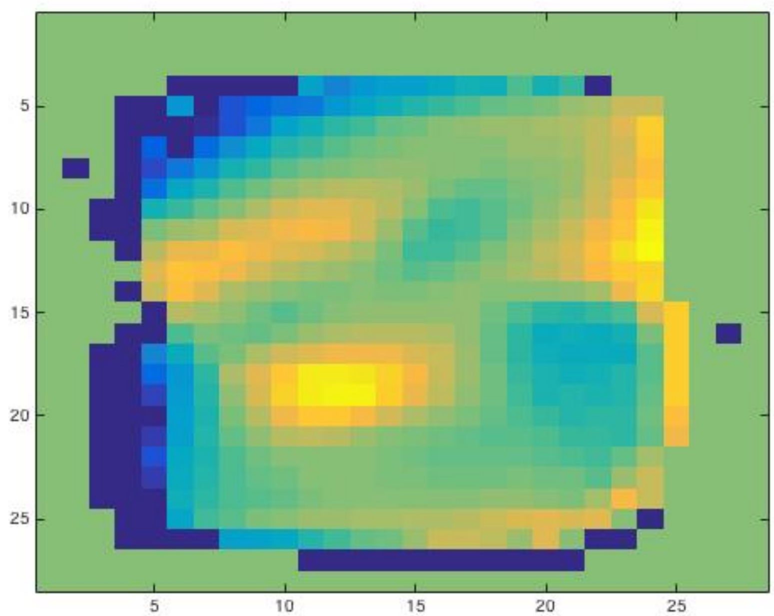


8->3:

8 & 3 feature likelihood

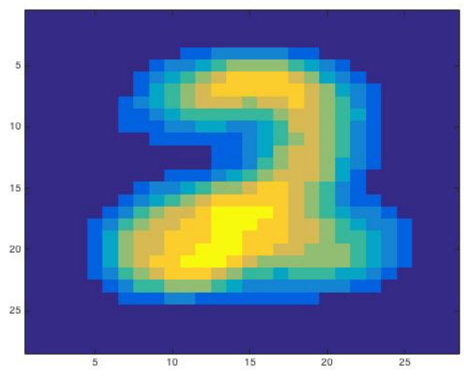


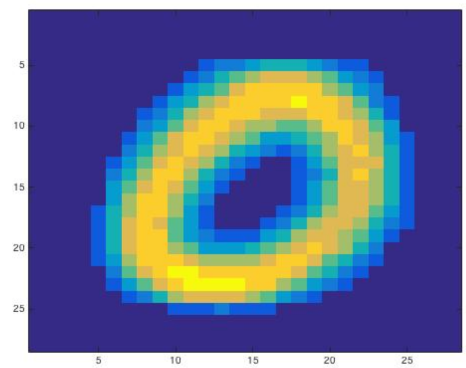
Odd ratios:

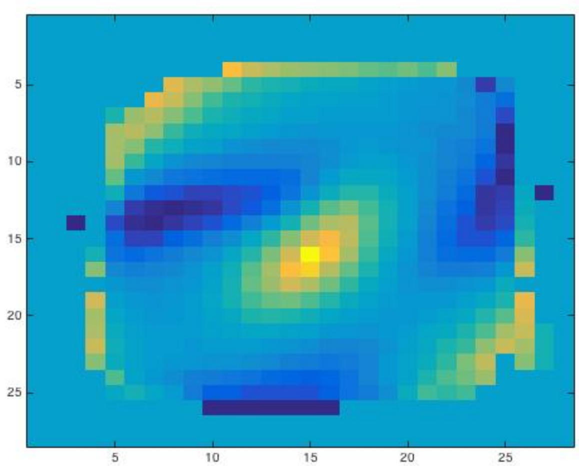


2->0:

2 & 0 feature likelihood







Part 2.1

In this part of assignment, we implement a training program to estimate the email is spam/not spam and review is positive/negative.

Here is our data structure:

confusionMatrix[2][2]: store the possibility of the paragraph in a specific class

map<string, double> goodEmail(goodReview) && map<string, double> badEmail(badReview):

Two maps which store the occurrence of a word in the whole email(review) as a spam/non-spam email (positive/negative review). We then converted the occurrence into the possibility. (the calculation of the occurrence and possibility differs in two kinds of bayes models, but the data structure is the same)

Here is our pseudo-code:

This is the code for the multinomial naïve Bayes, for the Bernoulli naïve Bayes, see the code in comment

Training Part:

```
Parse each line in the train_email/rt-train
    type = save the class (spam/notspam or positive/negative) of that line
    if type is 1:
        we parse the line and get the word as well as its occurrence:
        name = word;
```

value = occurrence;

goodEmail[name] += value;

(Bernoulli : goodEmail[name] += 1)

else:

we parse the line and get the word as well as its occurrence:

name = word;

value = occurrence;

badEmail[name] += value;

(Bernoulli : badEmail[name] += 1)

At last, we iterator through the map and convert the occurrence into the possibility of each word in that class.

(there is a tiny difference between the calculation of Bernoulli and multinomial naïve bayes.)

For Bernoulli, we divide the occurrence by the total line number in that class. But for the multinomial naïve Bayes, we divide the occurrence by the total number of words in that class.

Testing Part:

Parse each line in the test_email/rt-test

type = save the class (spam/not-spam or positive/negative) of that line

for each line:

we calculate its possibility to be a spam/not spam email (positive/negative review) based on its words possibility in goodEmail/ badEmail:

we get every word and its occurrence in test-email

name = word;

value = occurrence;

oneGood = oneGood + goodEmail[name] * value;

oneBad = oneBad + badEmail[name] * value;

then we compare oneGood and oneBad to classify the email

Last, we compare with our estimate with the real result and store into our confusionMatrix. At the same time, get the top 20 words with highest likelihood.

Result:

email-testing for multinomial naïve bayes: 0.969231

line, real, estimate

85: 1, 0

98: 1, 0

117: 1, 0

118: 1, 0

170: 0, 1

175: 0, 1

180: 0, 1

184: 0, 1

Confusion matrix:

	Estimate spam	Estimate non-spam
Real spam	126	4
Real non-spam	4	126

Highest:

Non-Spam:

```
email            0.016394
s                0.0143344
order            0.0137629
report           0.0125009
our              0.0114532
address          0.0113222
mail             0.0109532
program          0.00982213
send             0.00948877
free             0.00882206
money            0.00856014
list             0.00845298
receive          0.0078458
name             0.0074291
business         0.0072029
one              0.00654809
d                0.00640522
work             0.00625045
com              0.00620282
nt               0.0061552
internet         0.00592899
```

```
Spam:
language             0.0239365
university           0.0191789
s                    0.0139753
linguistic           0.0100673
de                   0.00938768
information          0.00936644
conference           0.00796466
```

```
workshop              0.00758235
email                 0.00675403
paper                 0.00673279
e                     0.00660536
english               0.00656288
one                   0.00588323
please                0.00584075
include               0.00581951
edu                   0.00569208
http                  0.0055434
research              0.00543721
abstract              0.00530977
address               0.00528853
papers                0.0051611
```

email-testing for bernoulli naïve bayes: 0.923077

line, real, estimate

70: 1, 0

117: 1, 0

118: 1, 0

133: 0, 1

143: 0, 1

145: 0, 1

158: 0, 1

170: 0, 1

171: 0, 1

175: 0, 1

176: 0, 1

180: 0, 1

181: 0, 1

183: 0, 1

184: 0, 1

189: 0, 1

191: 0, 1

204: 0, 1

205: 0, 1

207: 0, 1

Confusion matrix:

	Estimate spam	Estimate non-spam
Real spam	127	3
Real non-spam	17	113

Highest:

Non-Spam:

```
our            0.00386582
s              0.00379648
free           0.00343243
please         0.00325908
email          0.00320707
mail           0.00310306
one            0.00291237
address        0.0028777
list           0.0028777
com            0.00277368
receive        0.00272168
http           0.00272168
us             0.00270434
send           0.00266967
day            0.00266967
information    0.00265234
remove         0.00260033
here           0.00253099
over           0.00253099
want           0.00251365
need           0.00251365
```

```
Spam:
language           0.00436254
university         0.00363847
s                  0.00342125
```

```
information          0.00314972
linguistic           0.00307731
http                 0.00247995
email                0.00246185
please               0.00240754
e                    0.00237134
follow               0.00235324
fax                  0.00235324
include              0.00235324
one                  0.00226273
english              0.00226273
call                 0.00215412
research             0.00209981
www                  0.00209981
word                 0.00206361
address              0.00204551
interest             0.00202741
send                 0.0019731
```

Review-testing for multinomial naïve bayes: 0.704

Confusion matrix:

	Estimate positive	Estimate negative
Real positive	365	135
Real negative	161	339

Highest:

Positive:

```
film                 0.00133748
movie                0.000921373
--                   0.000704829
one                  0.00059868
like                 0.000547729
story                0.000526499
good                 0.000484039
comedy               0.000479793
way                  0.000467056
even                 0.000450072
time                 0.000437334
best                 0.000433088
much                 0.000407612
```

performances	0.000390628
funny	0.000382136
make	0.000382136
life	0.000373644
us	0.000373644
makes	0.000373644
characters	0.000365153
work	0.000352415

Negative:

movie	0.00133939
film	0.0010757
like	0.000807819
one	0.000724107
--	0.000602725
bad	0.000489714
story	0.000481343
much	0.000472972
time	0.000439487
even	0.000418559
good	0.000393445
characters	0.000393445
little	0.000385074
would	0.000368332
comedy	0.000364146
never	0.000347404
nothing	0.000343218
makes	0.000339033
plot	0.000339033
make	0.000334847
script	0.00032229

Review-testing for bernoulli naïve bayes: 0.705

Confusion matrix:

	Estimate positive	Estimate negative
Real positive	365	135
Real negative	160	340

Highest:

Positive:

film	0.00132198
movie	0.000888401

one	0.000590851
like	0.000544093
--	0.000539842
story	0.000510087
comedy	0.00047183
way	0.000459078
even	0.000450577
good	0.000437825
best	0.000429323
time	0.000425072
much	0.00040807
performances	0.000391067
funny	0.000382565
makes	0.000369813
life	0.000365562
make	0.000365562
characters	0.000361312
work	0.00035281
still	0.000348559

Negative:

movie	0.00131162
film	0.00106019
like	0.00077943
one	0.000699811
story	0.000477715
much	0.000469334
--	0.000460953
bad	0.000444191
time	0.000427429
even	0.000414858
characters	0.000393905
little	0.000385524
good	0.000381334
would	0.000368762
comedy	0.000364572
nothing	0.00034362
makes	0.000339429
plot	0.000339429
never	0.000335239
make	0.000326858
script	0.000322667