# CSC411: Project 3
# Supervised and Unsupervised Learning for Sentiment Analysis

Due on Tuesday, March 21, 2017

**Michelle Leung and Tianyi Yu**

February 23, 2018

# Part 1

*Dataset description*

The dataset contains 2000 movie reviews categorized into positive and negative reviews, with 1000 reviews each. It is feasible to predict whether the review is positive or negative from keywords that appear in the review, since a positive review will tend to use more words of praise, while a negative review will tend to use more words to describe failures and critique the movie. Three keywords and their view statistics observed that may be useful are:

```
perfect (17.8% in positive reviews, 7.1% in negative reviews)
stupid (3.7% in positive reviews, 15.5% in negative reviews)
boring (5.1% in positive reviews, 17.8% in negative reviews)
```

# Part 2

*Implementation of Naive Bayes*

The parameter m was tuned by using a for loop on a validation set through a range of values from 0.1 to 10 for m and 0.2 to 20 for k, with the total performance, along with the respective m and k, stored in lists. Then, the following line of code was used to determine the indices of the best choices for m and k:

```
choices = np.where(np.array(T)==max(T))[0]
```

where T is the array of final performances using various combinations of m and k on the validation set, and the function `np.where()` finds the indexes of the values with the best performance. These choices for m and k were then tested again on another validation set to obtain the best values for m and k in the subset, but it was noted that the performance of the subset of best performing m's and k's were the same even with a different test set. The chosen parameters are $m = 0.1$ and $k = 0.38$
The final performances of the chosen m and k parameters are as follows:

```
Training Set
m =  0.1 ; k =  0.38
Percent right TOT:  0.999375
Percent right NEG:  0.99875
Percent right POS:  1.0

Validation Set
m =  0.1 ; k =  0.38
Percent right TOT:  0.735
Percent right NEG:  0.53
Percent right POS:  0.94

Test Set
m =  0.1 ; k =  0.38
Percent right TOT:  0.73
Percent right NEG:  0.51
Percent right POS:  0.95
```

It is plausible for the algorithm to do better on positive reviews than on negative reviews, as it is hypothesized that positive reviews will be more likely to give general words of praise that will be common to many positive reviews, whereas negative reviews will discuss the specific critical details surrounding failure of each movie, and since Naive Bayes is based on relative frequency of the words in positive and in negative reviews.

The tip $a_1 a_2 ... a_n = exp(\log a_1 + \log a_2 + ... + \log a_n)$ was used to compute:

$$\begin{aligned} P(class|a_1, ..., a_n) &= P(a_1|class) \cdot P(a_2|class) \cdot ... \cdot P(a_n|class) \\ &= \log P(a_1|class) + \log P(a_2|class) + ... + \log P(a_n|class) \end{aligned} \tag{1}$$

which was left in log-probability form, and the $argmax_{class}$ of the log sums was used to classify the reviews as positive or negative.

# Part 3

*Finding words that strongly predict Positive/Negative Reviews*

The following are the top ten words which strongly predict whether a review is positive or whether the review is negative:

```
Top 10 words that strongly predict the review is POSITIVE:
'life', 'both', 'world', 'great', 'best', 'also', 'many', 'most', 'very',
'perfect'

Top 10 words that strongly predict the review is NEGATIVE:
'bad', 'worst', 'plot', 'script', 'boring', 'stupid', 'why', 'nothing', 'least',
'should'
```

These words were obtained by computing the probability that each word appears in the positive and negative reviews of the training set (800 reviews in total), and then taking the differences. This way, words with a strong inclination towards indicating a positive or negative review would have a higher difference probability, and this also prunes out highly common words, such as "the", which appear in both sets of reviews. Taking n maximum values in the differences would indicate the highest probabilities, which can then be mapped to the respective words.

The following is the code used to obtain the top 10 words for each class of reviews, where `p_pos` and `p_neg` represent the array of probabilities for each word in the positive and negative training sets, respectively.

```
def nmax(num, T, nwords):
    """
    Takes in an array T (of probabilities) and output the top num words
    associated with the highest probabilities in T.
    """
    top_n = T.argsort()[-num:][::-1]
    for n in top_n:
        nwords.append(data['all_words'][n])
    return nwords

print('\nTop 10 words that strongly predict the review is POSITIVE:')
print(nmax(10,p_pos-p_neg,[]))
print('\nTop 10 words that strongly predict the review is NEGATIVE:')
print(nmax(10,p_neg-p_pos,[]))
```

Alternatively, one can compute just the probability of the words in the positive reviews subtracted by the probability of the words in the negative reviews, and simply pick the top ten `argmax` for words that strongly predict the review is positive, and the ten `argmin` for words that strongly predict that the review is negative.

# Part 4

*Logistic Regression*

An alternative way to train a classifier is to use logistic regression. In this part, a logistic regression model was constructed in TensorFlow using AdamOptimizer with a training step of 0.0004, using stochastic batches of size 50. L2 regularization was used and the performance on the validation set was tested using different lambdas. For each trial run, the lambda is increased by a factor of 5 and the lambda with the best performance on the validation set was chosen to train the network. The training curves using the best-performing lambda is shown below:

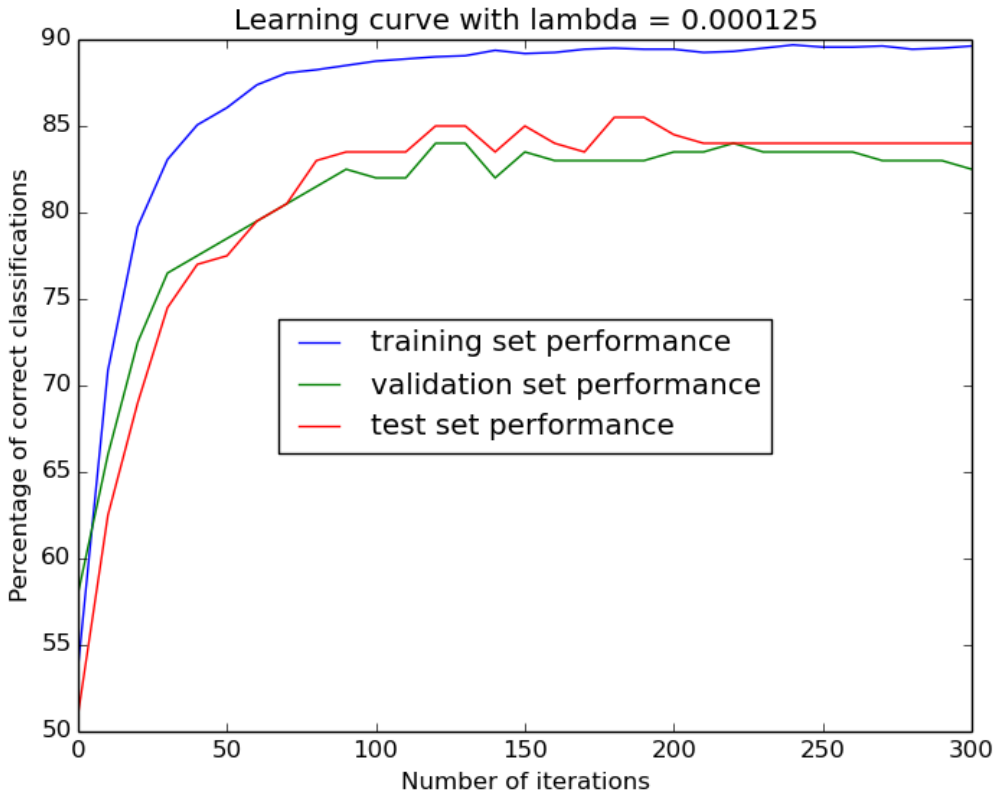


Figure 1: Learning Curves of the Logistic Regression Model

# Part 5

*θ's of Logistic Regression and Naive Bayes*

Both Logistic Regression and Naive Bayes can be computed as

$$\theta_0 + \theta_1 I_1(x) + \theta_2 I_2(x) + ... + \theta_k I_k(x) > thr \tag{2}$$

at test time.
In the case of Logistic Regression, the $\theta_i$'s represent the difference in weights and biases of each word between the positive and negative training sets in the full list of words, whereas in Naive Bayes, the $\theta_i$'s represent the difference in probabilities of each word in the positive and negative training sets. So, $\theta_0$ in Logistic Regression would be represented by

$$\theta_{i...k} = (W_{POS_{i...k}} + B_{POS}) - (W_{NEG_{i...k}} + B_{NEG}) \tag{3}$$

where $W_{POS_{i...k}}$ and $W_{NEG_{i...k}}$ is the positive and negative weights of words i...k, and $B_{POS/NEG}$ represents the positive/negative bias obtained from logistic regression.
In Naive Bayes, $\theta_0$ is zero (or m and k when computing the log probabilities), and

$$\theta_{i...k} = P(i...k|POS) - P(i...k|NEG) \tag{4}$$

where $P(i...k|POS/NEG)$ is the probability of each word i...k appearing in a review given that the review is positive or negative.
The $I_{1...k}(x)$'s for both Logistic Regression and Naive Bayes represent the presence, or lack of presence, of each word i...k in a given input review.
In both cases, $x$ will be classified as positive if it is closer to 1 (high), and negative if it is closer to 0 (low).

# Part 6

*Comparison of Naive Bayes and Logistic Regression*

The following is a list of the words (see Appendices for corresponding values of $\theta$) associated with the top 100 $\theta$'s obtained using

## Logistic Regression:

'hilarious', 'allows', 'due',
'flawless', 'wonderfully', 'understanding',
'heat ', 'transcend', 'performance',
'outstanding', 'including', 'complaints',
'throug hout', 'makes', 'chemistry',
'sings', 'matt', 'holiday', 'amazed',
'nerves', 'me morable', 'raised',
'commanding', 'great', 'enjoyable',
'7', 'subtle', 'will', ' boyle',
'unfolds', 'family', 'existence',
'supervisor', 'definitely', 'showing',
'ending', 'vulnerable', 'naturally',
'narration', 'together', 'landscape',
'pun ', 'individual', 'excellent',
'portrayal', 'bubble', 'cameron',
'assumes', 'keep s', 'mirror',
'focuses', 'touchy', 'contract',
'regardless', 'glee', 'stylishly',
"man's", 'hoot', 'ash', 'many',
'unique', 'carefully', 'delivers',
'daniel', ' fantastically', 'find',
'enjoyed', 'rushed', 'near', 'www',
'frank', 'uplifting' , 'virtually',
'subtitles', 'designer', 'greater',
'fulfill', 'museum', 'fuller' ,
'nolte', 'formulated', 'brilliant',
'horrific', 'airwaves', 'works',
'funnest' , 'trepidation', 'insightful',
'quirky', 'learning', 'kidnappers',
'interacts', 'spring', 'influenced',
'comfort', 'see', 'emotionally',
'rene', 'realistic', 'u nlike'

## Naive Bayes:

'life', 'both', 'world', 'great',
'best', 'also', 'many', 'most',
'very', 'perfect', 'performance',
'although', 'performances', 'own',
'true', 'family', 'different',
'may', 'seen', 'quite', 'hilarious',
'throughout', 'young', 'others',
'brilliant', 'american', 'perfectly',
'him', 'while', 'works', 'shows',
'between', 'especially', 'years',
'see', 'sometimes', 'yet', 'will',
'effective', 'excellent', 'strong',
'does', 'job', 'memorable', 'makes',
'well', 'subtle', 'always', 'become',
'oscar', 'people', 'wonderful',
'new', 'father', 'tale', 'gives',
'finds', 'human', 'extremely',
'overall', 'takes', 'love', 'history',
'each', 'together', 'begins',
'powerful', 'right', 'man', 'still',
'story', 'wonderfully', 'war',
'terrific', 'outstanding', 'classic',
'first', 'simple', 'often', 'times',
'allows', 'truly', 'until', 'picture',
'solid', 'realistic', 'change',
'without', 'everyone', 'other',
'wife', 'these', 'personal', 'superb',
'during', 'set', 'final', 'role',
'ever', 'beautiful'

The following sections are a summary comparisons of the words from both lists:

## Words that appear in both logistic regression and naive bayes

'excellent', 'memorable', 'makes', 'great', 'will', 'together', 'family', 'works', 'wonderfully', 'performance', 'subtle', 'allows', 'hilarious', 'many', 'throughout', 'outstanding', 'realistic', 'see', 'brilliant'

## Words that appear only in logistic regression

'raised', 'enjoyable', 'nolte', 'funnest', 'subtitles', '7', 'showing', 'transcend', 'find', 'nerves', 'rene', 'hoot', 'formulated', 'influenced', 'unlike', 'virtually', 'cameron', 'comfort', 'understanding', 'frank', 'boyle', 'unique', 'learning', 'individual', 'flawless', 'kidnappers', 'greater', 'carefully', 'emotionally', 'including', 'pun', 'rushed', 'trepidation', 'contract', 'holiday', 'due', 'delivers', 'portrayal', 'spring', 'designer', 'complaints', 'enjoyed', 'www', 'uplifting', 'airwaves', 'near', 'naturally', 'definitely', 'existence', 'touchy', 'interacts', 'heat', 'bubble', 'fuller', 'matt', 'horrific', 'commanding', 'sings', 'landscape', "man's", 'fulfill', 'vulnerable', 'supervisor', 'focuses', 'museum', 'insightful', 'regardless', 'quirky', 'fantastically', 'keeps', 'amazed', 'daniel', 'chemistry', 'ending', 'unfolds', 'glee', 'mirror', 'narration', 'ash', 'stylishly', 'assumes'

## Words that appear only in naive bayes

'finds', 'world', 'sometimes', 'human', 'does', 'him', 'different', 'others', 'terrific', 'become', 'set', 'simple', 'shows', 'without', 'father', 'ever', 'role', 'personal', 'people', 'war', 'love', 'always', 'powerful', 'change', 'also', 'beautiful', 'job', 'man', 'each', 'well', 'young', 'gives', 'these', 'first', 'perfectly', 'right', 'perfect', 'yet', 'american', 'quite', 'both', 'effective', 'final', 'seen', 'picture', 'best', 'solid', 'superb', 'history', 'although', 'true', 'wife', 'extremely', 'new', 'years', 'classic', 'often', 'everyone', 'takes', 'life', 'until', 'very', 'during', 'most', 'begins', 'tale', 'own', 'still', 'between', 'strong', 'overall', 'performances', 'other', 'while', 'especially', 'times', 'may', 'truly', 'wonderful', 'story', 'oscar'

It can be observed that there are some mutual words between in the top 100 $\theta$'s from both methods, most of which are common positive adjectives, however the majority of the words are exclusive to one method over the other. The words exclusive to logistic regression tend to be words not too commonly used as they are more specific to movie reviews, including names and adjectives, whereas the words exclusive to naive bayes tend to be common verbs and nouns that can appear in many contexts outside of specific movie reviews as well. As seen in the appendices, the numerical values of the top 100 $\theta$'s for each method are fairly similar, ranging from 0 to 0.2.

# Part 7

*word2vec investigation*

Word2vec is an interesting unsupervised learning technique that can be used to determine the likeness or similarity of two words. Given the embeddings of the vocabulary on the movie reviews, we can use the embeddings to train a logistic regression classifier to determine whether two words are likely to appear together or not. In this part, we extract all pairings of words in 100 negative reviews and 100 positive reviews as the training set.

For each word in each review, the words right before and right after are added to its context set. For example, the sentence `"he ate some pizza yesterday"` would generate a context set of {`"he", "some"`} for the word `"ate"`, {`"ate", "some"`} for the word `"pizza"`, etc. These are marked with a y-label of 1, indicating that these words do appear next to each other. For each context pair that does exist, we also add another random word that does not appear in the context, these are marked with a y-label of 0, indicating that across the entire training set, these two words do not appear together even once. With the above example, the $y = 0$ context for `"pizza"` would be {`"he", "ate"`}, assuming there's no other words in the vocabulary list. The same process is repeated with 10 negative and 10 positive reviews to form a validation set, and another 10 negative and 10 positive for the test set.

With all the context word pairs, these are converted to their embeddings and flattened, so each training word pair becomes a vector of length 256. This is used to train a logistic regression network. As with `part4`, L2 regularization was used and lambda was incremented by 0.00003 for each trial run. An interesting thing to note is the performance on the validation and test sets are substantially higher than on the training set. This is likely due to having a lower volume of total contexts and thus lower number of randomly generated word pairs for which $y = 0$. We expect with much larger training set the performance will improve substantially on the training set itself.

The learning curve of the best-performing lambda is shown below.

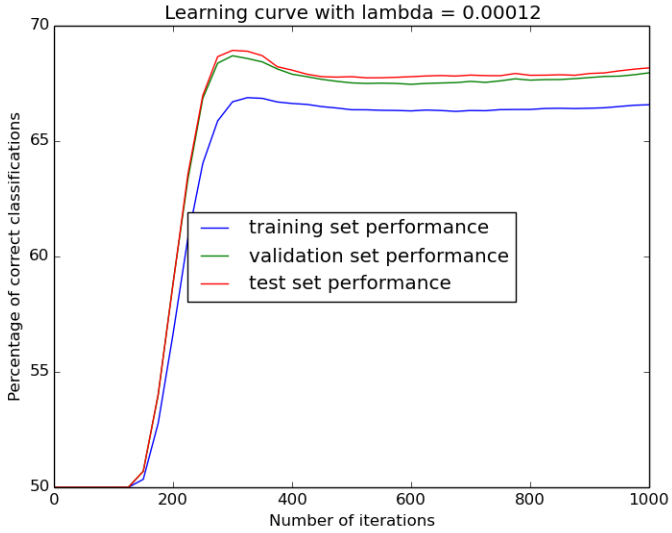


Figure 2: Learning Curves of the Best-Performing Lambda

# Part 8

*Demonstration of word2vec*

The trained embeddings on the movie reviews place words in similar sentence context in similar embeddings. Theoretically, words with very similar embeddings in terms of cosine or euclidean distance should be able to be swapped with one another in a movie review without influencing grammer - note the sentence meaning might change, but the words should be able to fill the same gap in a sentence. By looking at the embeddings, the 10 closest words to "good" and "story" are shown below. In addition, knowing the theme of movie reviews, the words similar to "movie" and "boring" are also shown. Note that some words that are similar, such as "good" and "admiral" are not very meaningful, this is likely due to local minima in the optimization process coincidentally placing these next to each other. Some of the pairings are very indicative that word2vec works, such as "story" and "plot, "movie" and "film", and "boring" and "dumb". Pairings like "good" and "bad" also appear because they fill the same niche in a sentence; the meaning changes but the result is still gramatically correct.

| CLOSE TO good | CLOSE TO story | CLOSE TO movie | CLOSE TO boring |
|---|---|---|---|
| bad | plot | film | dumb |
| great | film | picture | honor |
| wonderful | benito | movies | cuddles |
| reinforcing | simmer | enlistees | careful |
| decent | sitter | script | roxanne |
| funny | lift | thriller | priorities |
| manipulate | domineering | trapping | witnesses |
| underused | ricci | narrative | doyles |
| admiral | interviews | railway | charming |
| perplexing | acclaim | divulging | grays |

# Part 9

*Bonus*

A dataset which would work significantly better with Logistic Regression than Naive Bayes would be one in which the overall keywords for positive and negative reviews appear infrequently amongst the respective classes, but are definitive enough to distinguish between positive and negative reviews, since Naive Bayes relies on the probabilities of words determined from the training set to classify whether the review is positive or negative.
In this part, a dataset was generated with 1000 training reviews, each a vector of length 100, modelling a vocab list of 100 words. The training data is generated such that the first 98 words of each vector do not affect their y-label at all. The last two words, x[99] and x[100] give some indication as to what the y-label is. If either x[99] or x[100] = 1, then y = 1, otherwise, y is random. In the training set, the first 970 reviews do not have words x[99] and x[100] appearing, and thus their y-labels are completely random. For x[99] and x[100], they each have 15 reviews where they appear, giving a corresponding y = 1. The low amount of useful information will deter the Naive Bayes' algorithm due to low probabilities of those words appearing, and thus will not perform as well as logistic regression, where the correlation between x[99], x[100] and y is easier to pick up.
In the test set of 60, 15 reviews have x[99] appearing, 15 reviews have x[100] appearing, these all have the correct y label of 1. The other 30 reviews all have random x values with x[99] and x[100] both equal to 0, and y label of 0. Using logistic regression, an accuracy of 71.2% was achieved on the test set, only slightly worse than the optimal outcome of 75% (correctly predicting all 30 reviews with x[99]/x[100] and randomly guessing for the other 30 reviews). Note the performance is much better on the test set since 97% of the training set is random. Using Naive Bayes, only 58.3% was classified correctly.

```
Naive Bayes Test Set
nb best total performance:  0.5833333333333334
nb best neg performance:  0.16666666666666666
nb best pos performance:  1.0
```

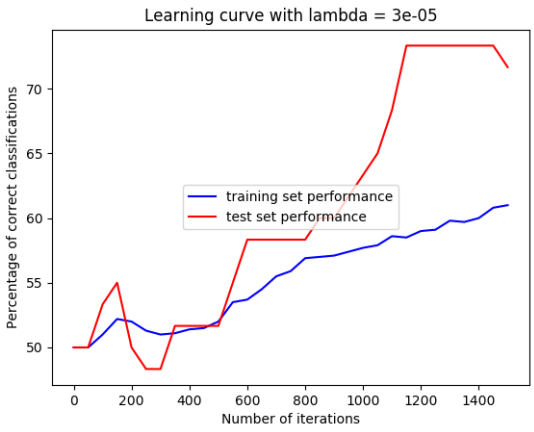


Figure 3: Logistic Regression Learning Curve

```
k = 100

## training set
# shapes of x and y
x = zeros((0, 100))
y = zeros((0, 1))

t = 2274192891
np.random.seed(t)

# generate random x, leaving x[99] and x[100] = 0, the y-label does not depend on x at all
for i in range(970):
    row = np.random.randint(2, size=98)
    row = np.append(row, array([0,0]))
    row = row.T
    x = vstack((x, row))
    if (i < 500):
        y = vstack((y, array([0])))
    else:
        y = vstack((y, array([1])))

# generate 30 reviews where x[99] or x[100] = 1, these are all positive reviews
for i in range(15):
    row = np.random.randint(2, size=98)
    row = np.append(row, array([1,0]))
    row = row.T
    x = vstack((x, row))
    y = vstack((y, array([1])))

for i in range(15):
    row = np.random.randint(2, size=98)
    row = np.append(row, array([0,1]))
    row = row.T
    x = vstack((x, row))
    y = vstack((y, array([1])))

## test set
xt = zeros((0, 100))
yt = zeros((0, 1))

# 30 negative reviews, x[99] = x[100] = 0, others all random
for i in range(30):
    row = np.random.randint(2, size=98)
    row = np.append(row, array([0,0]))
    row = row.T
    xt = vstack((xt, row))
    yt = vstack((yt, array([0])))

# 30 positive reviews, x[99] or x[100] = 1
for i in range(15):
    row = np.random.randint(2, size=98)
    row = np.append(row, array([1,0]))
    row = row.T
```

```
    xt = vstack((xt, row))
    yt = vstack((yt, array([1])))

for i in range(15):
    row = np.random.randint(2, size=98)
    row = np.append(row, array([0,1]))
    row = row.T
    xt = vstack((xt, row))
    yt = vstack((yt, array([1])))
```

# Appendices

## A. Top 100 theta's for Logistic Regression

('hilarious', 0.13200000000000001), ('allows', 0.125), ('due', 0.122), ('flawle
ss', 0.122), ('wonderfully', 0.121), ('understanding', 0.12), ('heat', 0.12), ('
transcend', 0.11899999999999999), ('performance', 0.11899999999999999), ('outsta
nding', 0.11899999999999999), ('including', 0.11700000000000001), ('complaints',
0.11600000000000001), ('throughout', 0.115), ('makes', 0.113), ('chemistry', 0.
113), ('sings', 0.113), ('matt', 0.113), ('holiday', 0.112), ('amazed', 0.111),
('nerves', 0.111), ('memorable', 0.11), ('raised', 0.11), ('commanding', 0.11),
('great', 0.109), ('enjoyable', 0.108), ('7', 0.108), ('subtle', 0.108), ('will'
, 0.107), ('boyle', 0.107), ('unfolds', 0.107), ('family', 0.106), ('existence',
0.105), ('supervisor', 0.105), ('definitely', 0.105), ('showing', 0.105), ('end
ing', 0.104), ('vulnerable', 0.104), ('naturally', 0.104), ('narration', 0.104),
('together', 0.10299999999999999), ('landscape', 0.10299999999999999), ('pun', 0.102999999
('individual', 0.10299999999999999), ('excellent', 0.10299 999999999999), ('portrayal',
0.10199999999999999), ('bubble', 0.1019999999999999 9), ('cameron', 0.10199999999999999),
('assumes', 0.10199999999999999), ('keeps' , 0.10199999999999999), ('mirror', 0.101999999
('focuses', 0.101000000 00000001), ('touchy', 0.10100000000000001), ('contract',
0.10100000000000001), ( 'regardless', 0.10100000000000001), ('glee', 0.10100000000000001),
('stylishly', 0.10000000000000001), ("man's", 0.10000000000000001), ('hoot', 0.100000000000
001), ('ash', 0.10000000000000001), ('many', 0.10000000000000001), ('unique', 0.
10000000000000001), ('carefully', 0.099000000000000005), ('delivers', 0.09900000
0000000005), ('daniel', 0.099000000000000005), ('fantastically', 0.0990000000000
00005), ('find', 0.099000000000000005), ('enjoyed', 0.099000000000000005), ('rus
hed', 0.099000000000000005), ('near', 0.099000000000000005), ('www', 0.099000000
000000005), ('frank', 0.099000000000000005), ('uplifting', 0.099000000000000005)
, ('virtually', 0.099000000000000005), ('subtitles', 0.099000000000000005), ('de
signer', 0.099000000000000005), ('greater', 0.099000000000000005), ('fulfill', 0
.099000000000000005), ('museum', 0.098000000000000004), ('fuller', 0.09800000000
0000004), ('nolte', 0.098000000000000004), ('formulated', 0.098000000000000004),
('brilliant', 0.098000000000000004), ('horrific', 0.098000000000000004), ('airw
aves', 0.098000000000000004), ('works', 0.098000000000000004), ('funnest', 0.097
000000000000000003), ('trepidation', 0.097000000000000003), ('insightful', 0.097000
000000000000003), ('quirky', 0.097000000000000003), ('learning', 0.0970000000000000
03), ('kidnappers', 0.097000000000000003), ('interacts', 0.097000000000000003),
('spring', 0.097000000000000003), ('influenced', 0.097000000000000003), ('comfor
t', 0.097000000000000003), ('see', 0.097000000000000003), ('emotionally', 0.0960
00000000000000002), ('rene', 0.096000000000000002), ('realistic', 0.096000000000000
002), ('unlike', 0.096000000000000002)

## B. Top 100 theta's for Naive Bayes

('life', 0.16500000000000001), ('both', 0.14599999999999999), ('world', 0.13200000000000001), ('great', 0.13100000000000001), ('best', 0.129), ('also', 0.128), ('many', 0.128), ('most', 0.11), ('very', 0.109), ('perfect', 0.108), ('performance', 0.106), ('although', 0.104), ('performances', 0.10299999999999999), ('own', 0.095000000000000001), ('true', 0.095000000000000001), ('family', 0.095000000000000001), ('different', 0.085999999999999993), ('may', 0.084000000000000005), ('seen', 0.084000000000000005), ('quite', 0.084000000000000005), ('hilarious', 0.082000000000000003), ('throughout', 0.082000000000000003), ('young', 0.082000000000000003), ('others', 0.081000000000000003), ('brilliant', 0.081000000000000003), ('american', 0.081000000000000003), ('perfectly', 0.081000000000000003), ('him', 0.080000000000000002), ('while', 0.079000000000000001), ('works', 0.075999999999999998), ('shows', 0.075999999999999998), ('between', 0.075999999999999998), ('especially', 0.074999999999999997), ('years', 0.074999999999999997), ('see', 0.073999999999999996), ('sometimes', 0.073999999999999996), ('yet', 0.071999999999999995), ('will', 0.071999999999999995), ('effective', 0.070999999999999994), ('excellent', 0.070999999999999994), ('strong', 0.070999999999999994), ('does', 0.070999999999999994), ('job', 0.069000000000000006), ('memorable', 0.069000000000000006), ('makes', 0.069000000000000006), ('well', 0.068000000000000005), ('subtle', 0.068000000000000005), ('always', 0.068000000000000005), ('become', 0.066000000000000003), ('oscar', 0.066000000000000003), ('people', 0.065000000000000002), ('wonderful', 0.065000000000000002), ('new', 0.065000000000000002), ('father', 0.065000000000000002), ('tale', 0.064000000000000001), ('gives', 0.064000000000000001), ('finds', 0.064000000000000001), ('human', 0.064000000000000001), ('extremely', 0.063), ('overall', 0.060999999999999999), ('takes', 0.059999999999999998), ('love', 0.058999999999999997), ('history', 0.058999999999999997), ('each', 0.058999999999999997), ('together', 0.058999999999999997), ('begins', 0.058999999999999997), ('powerful', 0.058000000000000003), ('right', 0.057000000000000002), ('man', 0.057000000000000002), ('still', 0.057000000000000002), ('story', 0.057000000000000002), ('wonderfully', 0.056000000000000001), ('war', 0.056000000000000001), ('terrific', 0.056000000000000001), ('outstanding', 0.056000000000000001), ('classic', 0.056000000000000001), ('first', 0.056000000000000001), ('simple', 0.055), ('often', 0.055), ('times', 0.055), ('allows', 0.055), ('truly', 0.052999999999999999), ('until', 0.052999999999999999), ('picture', 0.052999999999999999), ('solid', 0.052999999999999999), ('realistic', 0.051999999999999998), ('change', 0.051999999999999998), ('without', 0.051999999999999998), ('everyone', 0.051999999999999998), ('other', 0.051999999999999998), ('wife', 0.051999999999999998), ('these', 0.050999999999999997), ('personal', 0.050999999999999997), ('superb', 0.050999999999999997), ('during', 0.050999999999999997), ('set', 0.050999999999999997), ('final', 0.050999999999999997), ('role', 0.050999999999999997), ('ever', 0.050000000000000003), ('beautiful', 0.050000000000000003)