

Classifying the Sentiment of Movie Review Data

CS 224N Final Project Report

Cheng-Tao Chu, Ryohei Takahashi, Pei-Chin Wang

1: Introduction

With the rapidly increasing amount of text available on the internet, organizing this vast amount of information has become increasingly important. Many researchers in natural language processing have studied the problem of automatically assigning documents to different categories. One type of categorization that has been studied is classifying the sentiment found in documents. Although many documents, such as movie reviews, already have some measure of sentiment, such as star ratings, many do not, and classifying documents as having positive or negative sentiment automatically would be useful for these unlabelled documents.

In our final project, we investigate the effectiveness of applying different feature extraction heuristics and feature selection methods and make an in-depth comparison among three very popular classifiers in the task of classifying movie reviews as having positive or negative sentiment. The three classifiers we examine are Maximum Entropy (MaxEnt) Classifier, Support Vector Machine (SVM), and Decision Tree (DT). The MaxEnt classifier models the probability distribution that maximizes the entropy under the constraints imposed by the training data. The SVM classifies data by maximizing the margin between the support vectors, which are the boundary for the classification. Finally, the DT classifies data into different categories by recursively partitioning the feature space into two parts and assigning different categories based upon which region in the divided space a document is, based on its features.

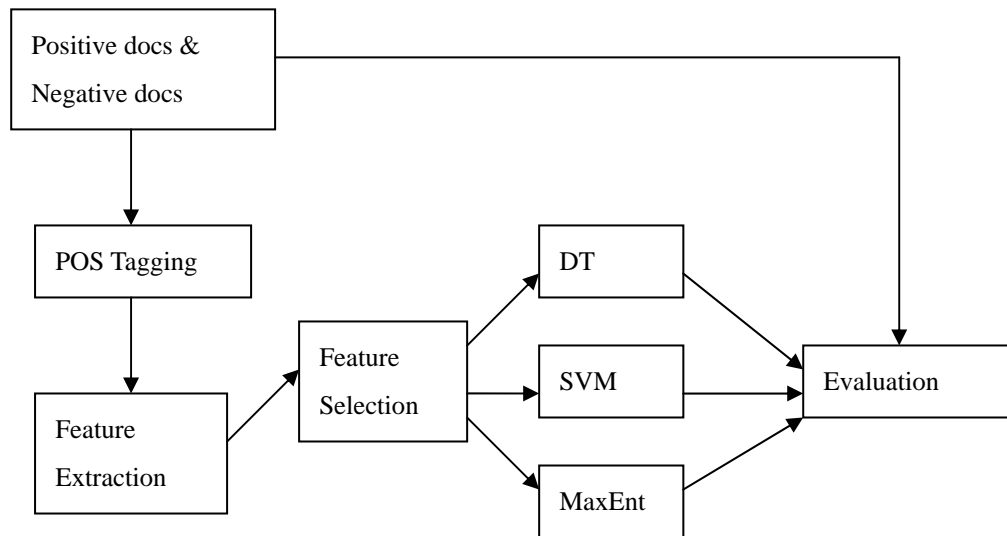
The training and test data consists of 1000 positive reviews and 1000 negative reviews, found at <http://www.cs.cornell.edu/people/pabo/movie-review-data/> and used by Pang et al (2002). Since we do not know how to divide the data, we use cross-validation to evaluate the performance and decrease the variance of the resulting estimate.

2: Methods

The procedures we apply in the system are as follows. First, we perform part-of-speech tagging to the reviews, using the Stanford Tagger provided by the Stanford NLP Group (2004). We then perform further processing on the data. We replace all stop words with a special token, so that our feature counts are not dominated by many commonly-occurring words. We also perform stemming using the Porter Stemming Algorithm (Porter 2002), so that words with the same stem are counted as the same feature, reducing the number of features and the sparsity of the data. In addition, we use an idea from Pang et al (2002): the tag “NOT_” is added to words after a negation word, such as “not” or contractions ending in “n’t.” However, we modify their idea by only appending the tag to adjectives instead of to all words as they do, as it does not make sense to negate words such as prepositions or pronouns. Also, instead of adding the tag to every word between a negative word and the first punctuation mark following the word, we only consider words within a three-word window after the negative word. We do this because although a negative word will tend to negate adjectives that follow it, the “range” of a negative word generally does not extend very far, only a few words (e.g. “not very good,” “not very likely”).

Next, we use the feature extraction heuristics to build up the feature matrix. In the matrix, we treat each document as a row and each feature as a column, and the value is the heuristic value for the feature in the document. After the feature extraction, we optionally apply some feature selection method to select the most significant features to reduce the dimension of the feature matrix. We then pass the reduced feature matrix to each classifier and evaluate their performance.

A diagram showing the control flow in our system is shown below.



2.1: Feature Extraction

Feature Set	Number of Features	Description
Unigram	14408	The presence of each unigram as a feature
Unigram+POS	21921	The presence of each unigram concatenated with its POS tag as a feature
Unigram+Bigram	96725	The presence of unigram and bigram as a feature
Unigram+Length	14508	The presence of each unigram and the percentages of sentences in each document having a particular length

In the Unigram+Length feature set, the length features are calculated as the percentage of sentences in a document having a particular length. For example, if 5% of the sentences in a document have length 7, then the LENGTH-7 feature has a value of 0.05 for that document.

As shown in the above table, we have four sets of features which can be used in the classifiers. In this table, the “Number” shows the number of features extracted in a subset of the whole data set, consisting of 150 positive and 150 negative reviews, which we used for some parameter tuning experiments. As we can see, the number of features increases significantly in the Unigram+Bigram case. Essentially, the bigrams to some extent reflect the context of a word, although it dramatically increases the dimension and thus the training and the testing time.

In keeping with the results of Pang et al (2002), we only consider presence features for unigrams, unigrams with POS tags, and bigrams, instead of using the frequency. This also allows us to use the MaxEnt classifier with the same features, as the MaxEnt classifier must have features that are either present or absent.

2.2: Feature Selection

As shown in the previous section, there are many more features than documents. This can create many problems. With such a large number of features, the feature matrix is very large, and the number of features is simply too big for most of the classifiers to train within a reasonable amount of time and without running out of memory. In addition, with a large number of features, the matrix is very sparse, leading to inaccurate estimates of the probability of certain features being present. Therefore we wish to select important features and discard the uninformative ones in order to speed up computation and to

improve performance. Also, by selecting out the most important features, we may eliminate noise in the data as well, also potentially improving the performance of the classifiers.

In our system, there are two feature selection algorithms. We describe them in the following two subsections.

2.2.1:Fisher Score

The following formula is used to calculate the Fisher score (Wu and Li 2003):

$$Score(feature_i) = \frac{|\mu_{i+} - \mu_{i-}|}{\sigma_{i+} + \sigma_{i-}}$$

The formula is quite self-explanatory. If the absolute difference between the mean score for the positive documents and the negative document is small, then the feature is not that discriminative, and if the standard deviation is large, the feature's values have a wide variation, and so the feature is not discriminative. From the formula, we can calculate the score for each feature and simply select the k features with the k highest scores.

2.2.2:Singular Value Decomposition (SVD)

We also use the SVD to reduce the dimensionality of the features. The SVD of a matrix A is

$$A = U\Sigma V^T,$$

where U and V are orthogonal matrices and Σ is a diagonal matrix containing the singular values of A . By taking only the first k columns of U and using the transpose of this matrix as a projection matrix that projects the feature matrix A into a k -dimensional subspace, we obtain an optimal rank k approximation to A (Manning and Raghavan 2004). We compute the SVD using the matrix package JAMA, developed by the National Institute of Standards and Technology (NIST 1999).

Although this reduction to k dimensions using the singular value decomposition gives an error with the lowest Frobenius norm (Manning and Raghavan 2004), the SVD method has the disadvantage of losing the interpretability of the features. Using features with the highest Fisher score has the advantage that the k features that are selected are actually features, and we can interpret these as being the most important and distinguishing features of the document classes. The Fisher score method, however, could select features that are highly correlated, which will reduce the usefulness of the features, as a high correlation between two features means that one of them can be discarded without much loss of predictive power. The singular value decomposition, on the other hand, projects the feature matrix along the vectors of U , which are orthogonal, and so this approximation is the best approximation out of all those that reduce the dimension of the matrix to k .

2.3:Classifiers

As mentioned in the introduction, there are three classifiers, MaxEnt, SVM, and DT, available in the system. In the following subsections, we will describe the features and the high-level overview for each classifier.

2.3.1:MaxEnt

As the name suggests, the MaxEnt classifiers maximize the entropy while maintaining the constraints imposed by the training data. The main assumption of the MaxEnt classifier is that the probability distribution of classes given its heuristic values and its associated weight is given by

$$p(c | d, \vec{\lambda}) = \frac{\exp(\sum_i \lambda_i f_i(c, d))}{\sum_{c'} \exp(\sum_i \lambda_i f_i(c', d))}.$$

Because the maximum entropy classifier take the dependence of features into account, it seems likely that it will perform better than a classifier such as Naïve Bayes, which assumes that features are independent, since features are seldom independent in natural language processing applications (Manning 2005).

Each feature in the MaxEnt classifier is an indicator function of some property of the document. For example, if the heuristic value of feature j for document i is h_{ij} , we will treat it as a feature named j which is present. This is in contrast to SVMs and decision trees, which use counts of features.

2.3.2:SVM

Support Vector Machines (SVM) (Cortes and Vapnik 1995), as a discriminative classifier, maximize the margin between different classes,

$$\min_{\omega, \beta, \xi} \frac{1}{2} \omega^T \omega + C \sum_{i=1}^l \xi_i$$

$$\text{subject to } y_i (\omega^T \phi(x_i) + b) \geq 1 - \xi_i,$$

$$\xi_i \geq 0$$

where (x_i, y_i) is the training instance pair, $\phi(\cdot)$ is the kernel function, and C is the penalty parameter for the error term ξ_i . There are several choices for the kernel function $\phi(\cdot)$, such as polynomial kernel, Radial Basis Function (RBF), and sigmoid function. SVM has been shown to be powerful in classification, but at the same time quite sensitive to parameters of kernel functions. Therefore we need to conduct parameter selection on SVM for experiments.

2.3.3:Decision Tree

For the decision tree, we use Weka, a publicly available software package containing various machine learning algorithms (Witten and Frank 2000). We use the J48 classifier in the Weka package, which is an implementation of the C4.5 algorithm for learning decision trees (Quinlan 1993).

The decision tree is built by recursively partitioning the feature space into two parts. At each step, the split that improves the error of the tree on the training data is used, and this greedy strategy continues until a tree of desired size is produced. For new data, a label is predicted by following the branches of the tree from the root node according to the values of the features of the new data. Decision trees have the advantage of being very easily interpretable, as the binary tree structure can be represented in a drawing, and a human being can follow the branches down the tree according to the input variables. They are also relatively fast to train (Friedman 2005).

3:Experiments

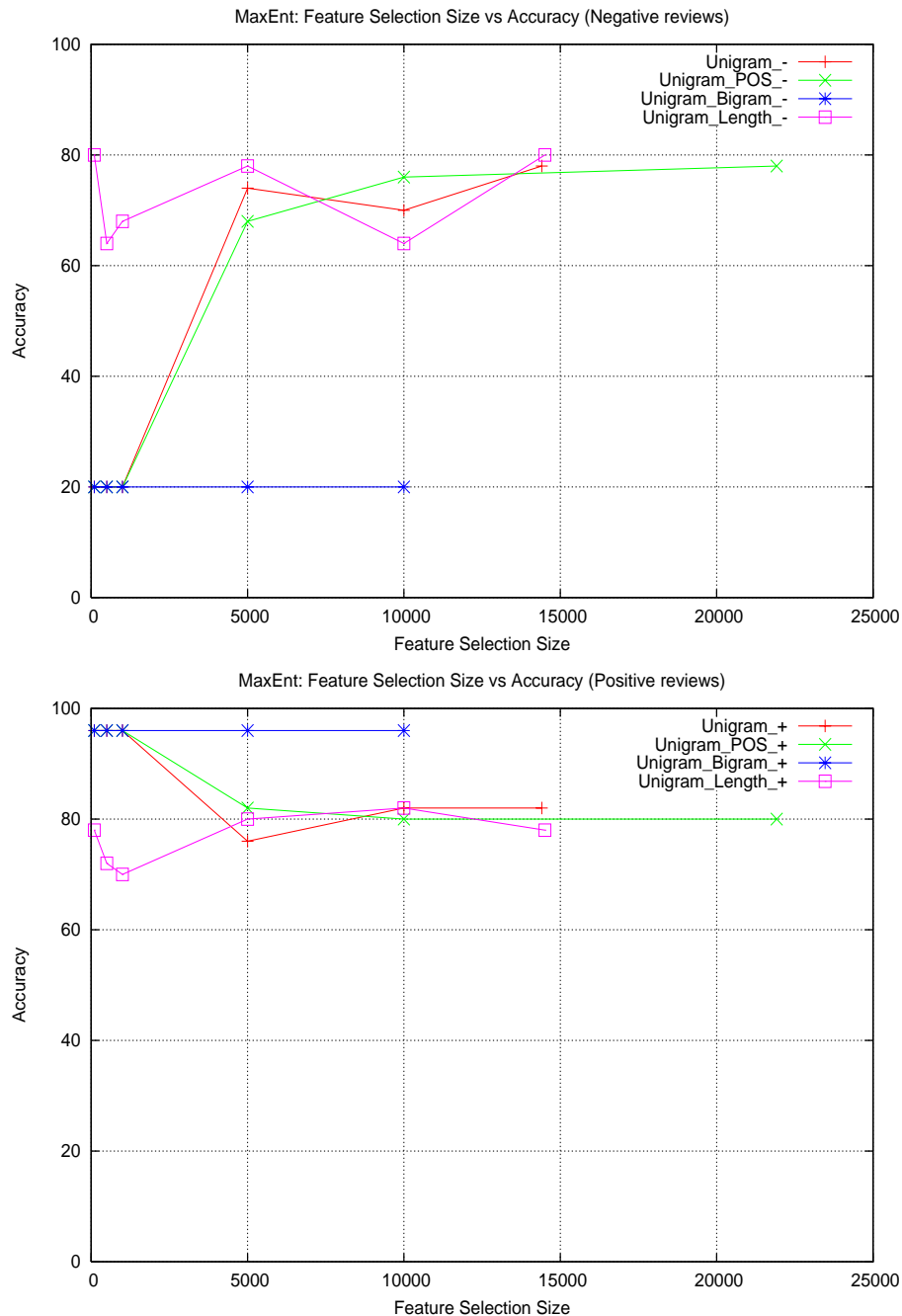
In the project, we have conducted the experiments with respect to different combinations of the feature selection methods and classifiers. The following subsections will discuss the experimental results for each classifier. We measure the performance of our system by the accuracy, which is defined as the number of correctly labeled movie reviews in the test set, divided by the total number of movie reviews in the test set.

Because the feature matrices are quite large, and the feature selection methods, especially the SVD method, take a prohibitively long time to compute, we use a smaller set of documents, consisting of 150 positive and 150 negative reviews, to determine the best set of features, the best feature selector to use, and the value of k , since this smaller set requires much less time to compute the feature matrices and train the classifiers. We use 100 documents of each type as the training set and the remaining 50 documents of each type as the test set. We assume that this set of 300 documents is a representative sample of the entire set, so that the set of parameters that result in the best performance on this smaller training and test set will result in the best performance on a bigger set of documents.

Due to limited access to computing resources, we found it impossible to use all 2000 documents at once, so we chose to use half of this set, consisting of 500 positive and 500 negative reviews, as the set on which we evaluate the performance of our classifiers. To get an accurate estimate of the misclassification errors of the classifiers, we use three-fold cross-validation on this set of 1000 documents.

3.1:MaxEnt

For the MaxEnt classifier, since the SVD projects the data to the principal components and the projected data does not reflect any heuristic for the probability distribution anymore, we decide to only include the Fisher Score feature selection scheme in this experiment. The results of the experiments are shown in the following two figures.



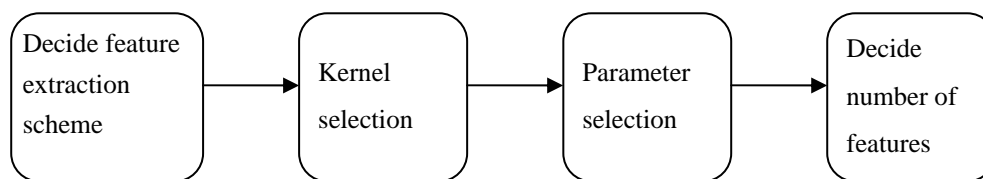
We obtained the highest accuracy on the smaller data set when we used only the unigram features with no feature selection. Using this set of parameters on the larger data set with three-fold cross-validation, we obtained an accuracy of 82.6%.

For this experiment, we found two interesting observations. The first one is that the classification accuracy for the positive reviews is always higher than that for the negative reviews. The second observation is that when we select fewer features, we get a better result for the positive reviews, but we get a worse result for the negative reviews.

As we further analyze the reviews which we classify incorrectly, we found some common scenarios in which MaxEnt will be easily misled. For positive reviews, it is not uncommon that we see that the reviewer spending almost all the text criticizing the film but saying “but I love it” or “just go to see the movie” in the end. It is easy for humans to determine the sense of the reviews, but because of the lack of context encoded in the heuristics, the MaxEnt classifier will very likely be misled. For negative reviews, there are two cases in which the classifier will be likely to give the wrong classification. The first one is that some reviewers tend to express their negative opinions by saying “XXX is good, but XXX.” For humans, we can tell that this is expressing a negative opinion, but for the classifier, it will be very vague. In a second case, reviewers sometimes simply express the negative idea without any negative words. For example, “the best part of this film is the end credits” expresses a very negative opinion of the movie, yet it does not contain any negative words that allow the classifier to determine that it is expressing a negative sentiment. It would be extremely hard for a classifier to detect the negative opinion simply by an n -gram model, even for higher values of n , so there would need to be some semantic analysis of the text for a machine learning algorithm to correctly detect a negative opinion expressed in this manner.

3.2:SVM

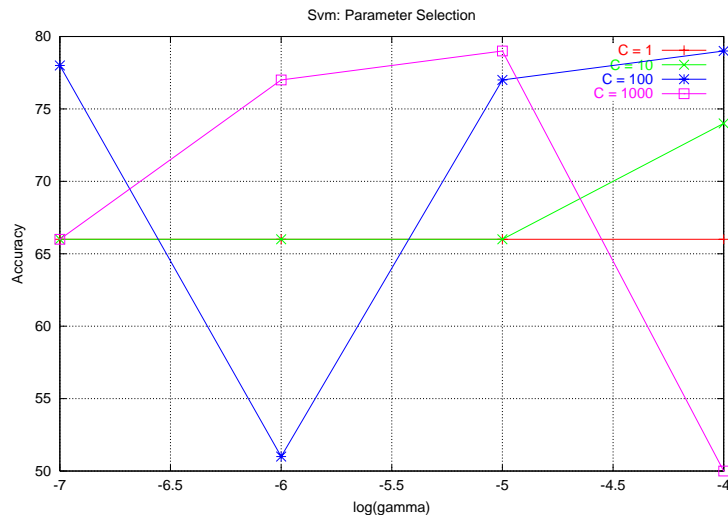
For SVM, we first need to choose the best way for feature extraction as in other classifiers. Using that feature extraction scheme, we then want to decide which kernel to use and tune the corresponding kernel parameters. At last, we use Fisher score to select most significant features. Following is the procedure we use for experiments,



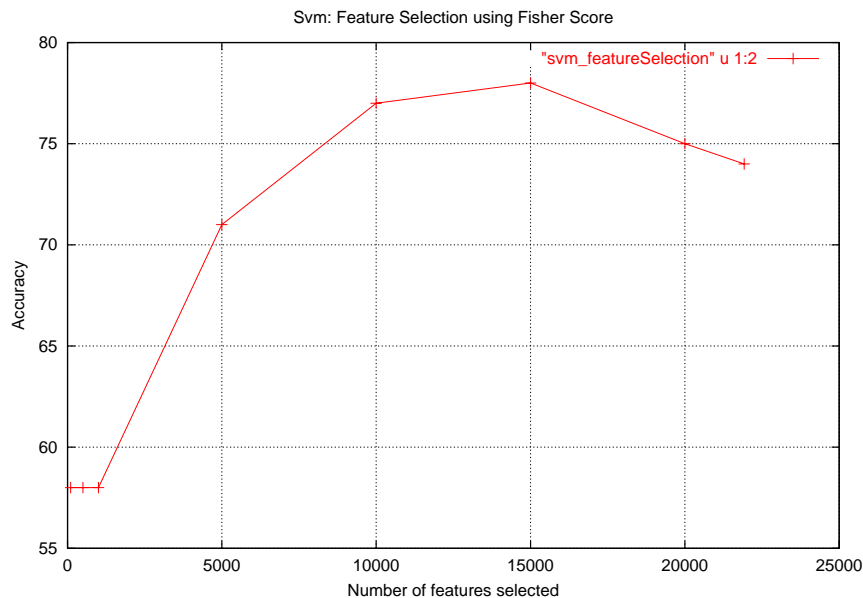
In the interest of time, we use a subset of data to do these experiments. We then apply the setting that gets best performance among these experiments to the whole data set. The SVM implementation we use here is LIBSVM (Chang and Lin 2005). For the first step, we use RBF kernel and polynomial kernel. RBF kernel is known to be quite powerful in that it could project data into unlimited dimensions. Polynomial degree, though less powerful, may be appropriate for this task in that the inner product evaluation in polynomial kernel is quite similar to the vector model in information retrieval. Therefore we think we should try the polynomial kernel as well. For RBF kernel, we use default parameters ($C=1$, $\gamma = 1/\text{num of features}$), and for polynomial kernel, we use $\text{degree} = 3$. The following table shows the result of each feature selection scheme,

Feature extraction / Kernel type	Unigram	Unigram + POS	Unigram + Bigram	Unigram + Length
Polynomial	0.50	0.51	0.50	0.50
RBF	0.75	0.77	0.76	0.75

RBF outperforms polynomial kernel a lot, and specifically polynomial kernel will classify all the reviews as positive documents. Therefore, we use RBF as the kernel function. We then conduct parameter selection over kernel parameters for RBF kernel, namely C and γ .



When $(C, \gamma) = (100, 10^{-4})$, $(1000, 10^{-5})$, the accuracy is the highest. To avoid overfitting, we choose the one with smaller C. The last step is to choose number of features used for feature selection. Using $(C, \gamma) = (100, 10^{-4})$, we conduct experiment on number of features selected and its corresponding accuracy.



When using 15,000 features selected by Fisher score, we get the best accuracy, 78%. After getting all the parameters we need, we then apply it to the bigger data set. However, considering some parameter might need to be changed as the data set grows, such as C in RBF kernel. Thus, we apply some other settings for the big data set,

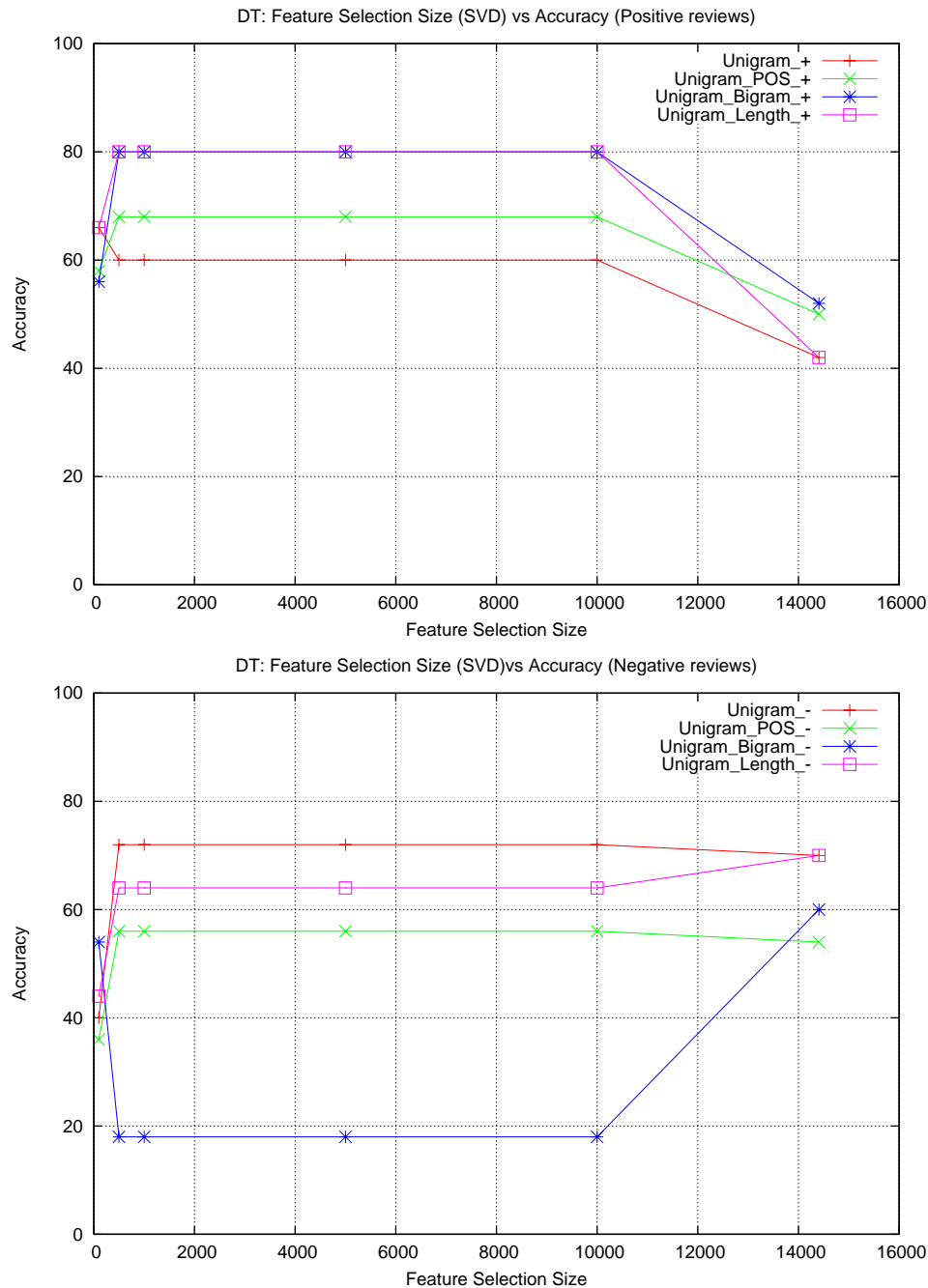
C	γ	Num of features	Accuracy
100	10^{-4}	41914 (all features)	0.821
10	10^{-4}	41914(all features)	0.824
100	10^{-4}	28680	0.826
10	10^{-4}	28680	0.806
100	10^{-4}	15000	0.767
10	10^{-4}	15000	0.787

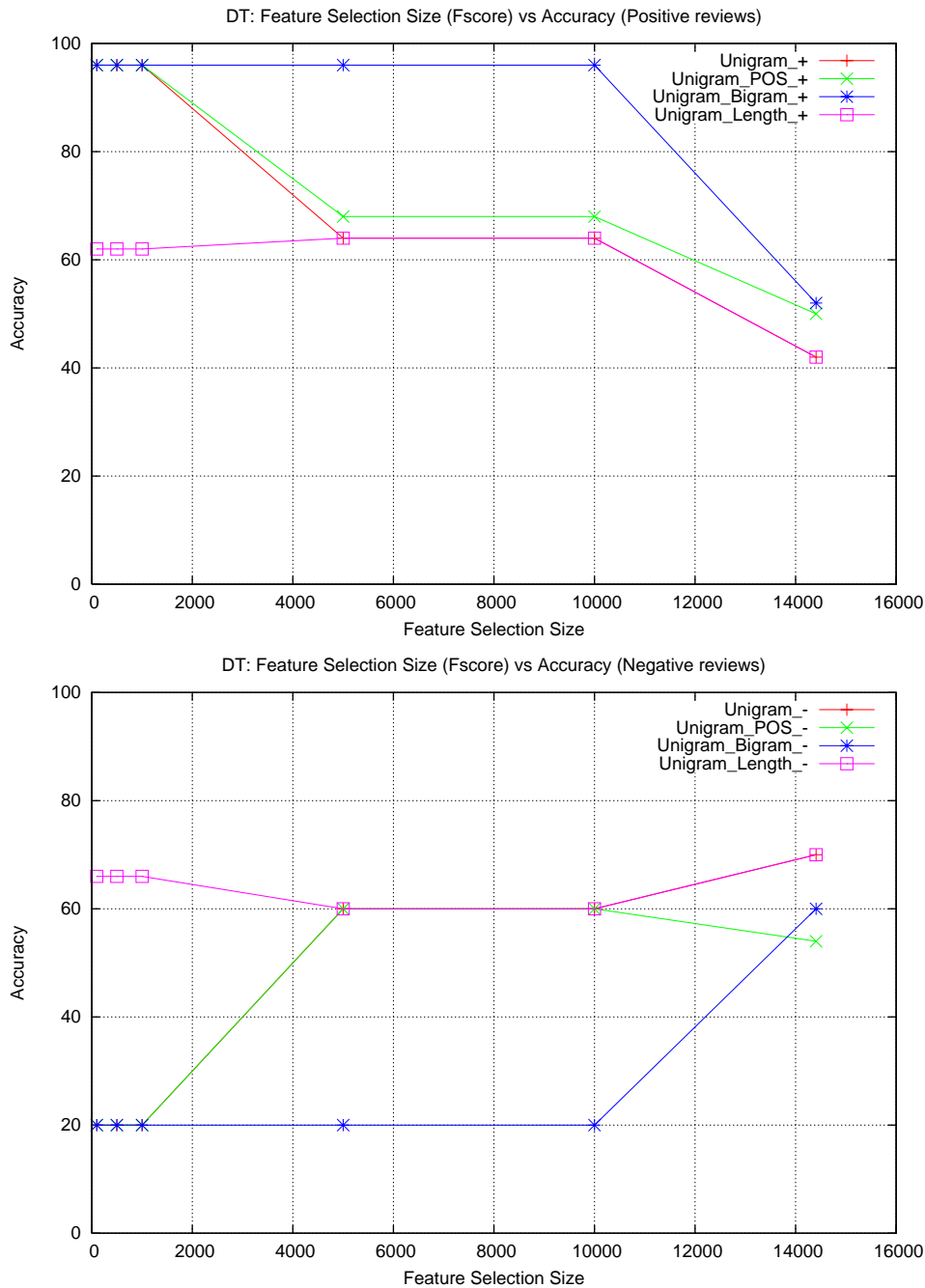
We performed experiments 28,680 features in feature selection because we found that using 15,000 features in the smaller data set produced the best results, and these were selected out of 21,921 features total for the Unigram+POS feature set, so using 68.43% of all features in the smaller data set produced the best results. Hence we tried also using this same proportion of the total number of features for the larger data set. We obtained the best performance by using 28,680 features in feature selection, C = 100,

and $\gamma = 10^{-4}$, resulting in 82.6% accuracy.

3.3: Decision Trees

Compared to the MaxEnt classifier and SVM, using the decision tree resulted in fairly poor performance, often resulting in accuracies that are only marginally better than chance (50%). We tried both feature selection methods with varying k and the different sets of features described in 2.1 on the small data set of 150 positive and 150 negative movie reviews to get an idea of which combination works best.





The results show that the best accuracy on the small data set was achieved by using SVD to reduce the dimensionality, with $k = 500$, and using both the unigram presence features and the sentence length distribution features. With 3-fold cross-validation, the results are as follows:

Accuracy	Accuracy on Positive Reviews	Accuracy on Negative Reviews
60.49%	63.78%	57.19%

The J48 package of Weka has some tunable parameters, so we experimented with different values of these. The `-C` option allows the user to specify the “confidence threshold for pruning” (Witten and Frank, Weka API), but changing this value did not seem to affect the results. The results of setting $C = 0.1, 0.3, 0.5,$ and 0.75 were all identical to when $C = 0.25$, the default value.

The `-M` option sets the “minimum number of instances per leaf” (Witten and Frank, Weka API), so that in the training, if any leaf node of the tree has less than the minimum number of instances that fall into the region specified by that leaf node, the leaf node is merged with its sibling. Setting `M` higher than the default value of 2 resulted in different accuracy values:

<i>M</i>	Accuracy	Accuracy on Positive Reviews	Accuracy on Negative Reviews
2	60.49%	63.78%	57.19%
5	61.09%	65.19%	57.54%
10	59.09%	60.79%	57.40%

The accuracy increased slightly in going from the default `M = 2` to `M = 5`, but using `M = 10` resulted in a lower accuracy than the default settings. The trees get smaller as `M` increases, since higher values of `M` force more leaf nodes to merge, causing the tree to shrink. The increased accuracy in going from `M = 2` to `M = 5` indicates that there were probably quite a few leaf nodes with less than 5 instances, and so the tree may have been overfitting the training set, causing the accuracy on the test set to decrease. Unfortunately, the highest accuracy we were able to achieve using decision trees was only 61.09%, which is poor compared to the other two classifiers. The small number of documents compared to the number of features probably contributes to the poor performance of the decision trees.

It is difficult to do meaningful error analysis on the reduced dimensions resulting from performing dimensionality reduction using the SVD, as the dimensions do not correspond to actual features. Accordingly, we instead examine the results for the full data set of using the same feature set of the unigram presence features and the sentence length distribution features and $k = 500$ as in the optimal settings above, but instead we use the Fisher score to perform feature selection.

The tree resulting from the first fold of the three-fold cross-validation, with `M = 5` since that demonstrated improved performance on the test set when SVD was used, is below.

```

UNI-laughabl <= 0
  UNI-delight <= 0
    UNI-nowher <= 0
      UNI-dull <= 0
        UNI-contrast <= 0
          UNI-robin <= 0
            UNI-brilliant <= 0
              UNI-noth <= 0
                UNI-greatest <= 0
                  UNI-overal <= 0
                    UNI-denni <= 0
                      UNI-ridicul <= 0
                        UNI-sometim <= 0
                          UNI-clich <= 0
                            UNI-world <= 0
                              UNI-blame <= 0
                                UNI-cinematographi <= 0
                                  UNI-typic <= 0
                                    UNI-hill <= 0
                                      UNI-frank <= 0
                                        UNI-silli <= 0
                                          UNI-subtl <= 0
                                            UNI-bore <= 0
                                              UNI-both <= 0
                                                UNI-seen <= 0
                                                  LENGTH-20
<= 0.075
UNI-look <= 0
UNI-plot <= 0
UNI-perform <= 0
  UNI-well <= 0: NEG (6.0/1.0)
  UNI-well > 0: POS (7.0)
UNI-perform > 0: POS (9.0/1.0)
UNI-plot > 0: NEG (15.0/3.0)
UNI-look > 0: NEG (16.0/1.0)
0.075: POS (7.0)
POS (16.0/5.0)
(15.0/3.0)
(11.0/1.0)
UNI-subtl > 0: POS (5.0)
UNI-silli > 0: NEG (10.0/1.0)
UNI-frank > 0: POS (6.0)
UNI-hill > 0: NEG (7.0/1.0)
UNI-typic > 0: POS (14.0/3.0)
UNI-cinematographi > 0: POS (8.0/1.0)
UNI-blame > 0: NEG (8.0/1.0)
UNI-world > 0
UNI-suppos <= 0: POS (48.0/8.0)
UNI-suppos > 0: NEG (6.0/2.0)

```

```

UNI-click > 0: NEG (19.0/5.0)
UNI-sometim > 0: POS (20.0/3.0)
UNI-ridicul > 0
UNI-differ <= 0: NEG (16.0/1.0)
UNI-differ > 0: POS (5.0/1.0)
UNI-denni > 0
UNI-father <= 0: NEG (10.0)
UNI-father > 0: POS (5.0/2.0)
UNI-overal > 0: POS (24.0/2.0)
UNI-greatest > 0: POS (20.0/1.0)
UNI-noth > 0
UNI-keep <= 0
UNI-true <= 0: NEG (73.0/9.0)
UNI-true > 0
UNI-look <= 0: NEG (6.0/1.0)
UNI-look > 0: POS (5.0/1.0)
UNI-keep > 0
UNI-car <= 0: POS (20.0/4.0)
UNI-car > 0: NEG (8.0/2.0)
UNI-brilliant > 0: POS (41.0/6.0)
UNI-robin > 0: NEG (29.0/5.0)
UNI-contrast > 0
LENGTH-3 <= 0: POS (21.0)
LENGTH-3 > 0: NEG (5.0/2.0)
UNI-dull > 0
UNI-histori <= 0: NEG (27.0/2.0)
UNI-histori > 0: POS (6.0/2.0)
UNI-nowher > 0: NEG (25.0/3.0)
UNI-delight > 0: POS (33.0/5.0)
UNI-laughabl > 0: NEG (34.0/5.0)

```

When running the decision tree on the test data, Weka outputs its confidence for the prediction of the label of each class. Looking through the results, it seems the confidence value is fairly low, sometimes close to 0.5, for the misclassified documents.

For example, the second document in the test set was misclassified; the decision tree classified it as negative, whereas the true class was positive, and the confidence value was only 0.6. Looking at the actual contents, the review starts, “Every now and then a movie comes along from a suspect studio, with every indication that it will be a stinker, and to everybody's surprise (perhaps even the studio) the film becomes a critical darling.” In the middle of the review, the author writes, “*Election*, a good film, does not live up to its hype.” These sentences have negative words and phrases (“stinker,” “does not live up to”), so it is understandable that the decision tree would misclassify this, but the general sentiment of the document is positive.

In another example of a misclassification, the decision tree marked a review as positive when its true class was negative. The author of the review writes, “Why is it that whenever a TV-star makes a movie it's always a romantic comedy, and then they say on *Entertainment Tonight* or something, that they were ‘attracted to the characters. They were really original.’” The word “original” here is not meant to be the author’s opinion but rather a quote from a TV star, which the author seems to be using sarcastically, as he actually does not find the characters original at all. The decision tree, and most likely other learning algorithms, would have difficulty with this, since the sarcasm is not indicated by any words the author uses. The best clue to the sarcasm is the “Why is it that” phrase at the beginning, but it is very far from the word “original” and so it would be difficult to detect that the word “original” should not be interpreted as part of the author’s opinion. In addition, this document may be hard to classify according to the above decision tree because it does not use the negative words that the training process found to be most discriminating, such as “laughable,” “dull,” “nowhere,” and “ridiculous.” This author seems to rely heavily on sarcasm to get his point across, rarely using actual negative words. Sentences like “He, of course, goes along with it. Gee, I wonder if they get together in the end” and “I've seen more original stuff on the WB. And better stuff too” do not actually contain negative sentiment words, and yet humans are easily able to detect the author’s disdain for the movie.

Using more context will most likely result in better performance. Although Pang et al (2002) report that using bigrams resulted in worse performance than using unigrams alone, this may be due to the fact that the bigrams do not provide enough context. Using trigrams has been shown to produce much better performance (Manning and Schütze 1999), but at the cost of a vastly increased vocabulary size. Because we had problems even running our classifiers with bigrams using the limited memory and computing power available to us, we were not able to investigate these higher order n -grams and other features that encapsulate more of the context.

3.4:Fisher Score Feature Selection

The feature selection was not a separate classifier, but looking at the features selected using the Fisher scores is informative. The following are the features with the highest 10 Fisher scores from each of the three folds in the cross-validation in our experiments for the SVM, where we used the (stemmed) unigrams concatenated with their POS tags.

- outstand__JJ
- stupid__JJ
- bore__NN
- whatsoever__RB
- wors__JJR
- fortun__RB
- ludicr__JJ
- hollywood'__NNS
- brilliant__JJ
- crisp__JJ
- bad__JJ
- worst__JJS
- mess__NN
- stupid__JJ
- lousi__JJ
- wast__VBN
- lame__JJ
- terribl__JJ
- flotsam__NN
- exceptionli__RB
- snap-brim__JJ
- beleiv__VB
- tangenti__JJ
- performac__NN
- sexiast__JJS
- sewell'__NNS
- twitchi__JJ

From the above list of words, we can see that words that clearly discriminate positive or negative sentiment, such as “outstand(ing)” and “terribl(e)” also have high Fisher scores, so the Fisher score statistic seems well-justified. Interestingly, words like “flotsam” and “tangenti(al),” which one would not think are commonly used in reviews, are very indicative of negative reviews. Also, the word “Sewell” was used when referring to actor Rufus Sewell, and apparently he is mentioned in negative reviews more often than positive ones, so the presence of his name in a review is an indication that it is a negative one.

4:Discussion

In the experiments using the three types of classifiers, we found that the maximum entropy classifier and SVMs both gave the best performance with 82.6% accuracy. However, decision trees were significantly worse, only achieving 61.1% accuracy. The poor performance of the decision trees was probably due to the fact that we had a very small number of documents for a very large number of features. Given more examples, the decision tree would have probably given better performance.

In general, the features we used incorporated little context since they primarily consisted of unigrams. Using higher order n -grams would most likely increase the accuracy, as they incorporate more of the context necessary to distinguish sentiments in text, but this would also drastically increase the number of features and the sparsity of the data.

However, even incorporating more context most likely would not enable the classifiers to correctly classify reviews where the author makes many critiques of the movie but in the end recommends the movie, or those where the author uses sarcasm to criticize a movie, since determining the true sentiment of these types of reviews requires more sophisticated semantic analysis over the entire review. From examining some of the movie reviews, we also found that many reviews have the true sentiment expressed in either the first or last paragraph, and the text in the middle of the review may be contrary to the reviewer’s actual opinion. Giving higher weight to features found near the beginning or the end of the document may also improve accuracy by taking advantage of this property of reviews.

Classifying the sentiment of text is a very difficult problem in general because the sentiment in many documents cannot be easily captured by using statistical counting of features such as words or lengths of sentences, as just one sentence can change the sentiment of the entire document. Because the sentiment can be so easily changed using few words, it is difficult to automatically distinguish between positive and negative sentiment without looking at the document as a whole and analyzing the discourse, which cannot be encapsulated in simple word- or sentence-level features.

5:References

- Chang, Chih-Chung and Chih-Jen Lin. “LIBSVM.” <http://www.csie.ntu.edu.tw/~cjlin/libsvm>, 2005.
- Cortes, Corinna and Vladimir Vapnik. “Support-Vector Networks.” *Machine Learning*, 20(3):273-297, September 1995.
- Friedman, Jerome H. “Decision Tree Induction.” Statistics 315B lecture notes, Spring 2005.
- Manning, Christopher D. “Maxent Models and Discriminative Estimation.” CS 224N lecture notes, Spring 2005.

- Manning, Christopher D. and Prabhakar Raghavan. CS 276A Lecture 15 notes, Fall 2004.
- Manning, Christopher D. and Hinrich Schütze, *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.
- National Institute of Standards and Technology. "JAMA: A Java Matrix Package." <http://math.nist.gov/javanumerics/jama>, 1999.
- Pang, B., L. Lee, and S. Vaithyanathan. "Thumbs Up? Sentiment Classification Using Machine Learning Techniques." In *EMNLP 2002*, 79-86.
- Porter, Martin. "The Porter Stemming Algorithm." <http://www.tartarus.org/~martin/PorterStemmer>, 2002.
- Quinlan, J. R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.
- Stanford Natural Language Processing Group. "Stanford Tagger." <http://nlp.stanford.edu/software/tagger.shtml>, 2004.
- Turney, P.D. "Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews." In *ACL 2002*, 417-424.
- Witten, Ian H. and Eibe Frank. *Data Mining: Practical Machine Learning Tools with Java Implementations*. San Francisco: Morgan Kaufmann, 2000.
- Wu, Zhi-li and Chung-hung Li. "Feature Selection using Transductive Support Vector Machine." In NIPS 2003 Workshop on Feature Extraction.