CSE628 Homework #1
03-13-2014

Sai Chaitanya Mallampati
Sravanthi Pereddy

Problem

Sentiment Analysis is a special category of Text Categorization wherein with the help of several Machine Learning techniques, we aim to classify documents not by their topic but by their overall sentiment. As suggested in the research paper "Thumbs Up ? Sentiment Classification using Machine Learning Techniques", capturing the overall sentiment of a particular document demands better understanding of the data. This report presents and analyzes the results observed by three machine learning methods (language model classifier, perceptron classifier and support vector machine) we employed on a dataset of equally distributed 2000 movie reviews.

Datsset Split

`https://www.cs.cornell.edu/people/pabo/movie-review-data/review_polarity.tar.gz` has a data set consisting of 2000 movie reviews out of which 1000 have a positive sentiment and others have a negative sentiment. To perform a evenly distributed 5-fold cross validation, we split the data into 5 sets - consisting of 200 positive and negative reviews each. With each fold as test dataset and the remaining 4 folds as training dataset, we run the algorithm 5 times and calculate the average performance of the model.

Technique

The basic steps involved in implementing any machine learning technique are :
1) Data Pre-processing
2) Applying Machine Learning Algorithm on training data to develop a prediction model.
3) Testing the prediction accuracy of the model using test data.

Data Preprocessing

During this process, we scanned through the entire dataset on a file-to-file basis and removed unwanted data like delimiters, extra spaces, numerals etc. We then ran several machine learning algorithms on this moulded data, the summary of which has been reported below.

Unigram Language Model Classifier

Firstly, we tried to classify the documents using only unigram features. Using perplexity as a measure of accuracy, we ended up the results shown in the table below.

Fold #	**Unigram Frequency Miscalculations**	**Unigram Presence Miscalculations**
1	77 out of 400 [80.75 % accuracy]	74 out of 400 [81.50 % accuracy]
2	70 out of 400 [82.50 % accuracy]	66 out of 400 [83.50 % accuracy]
3	77 out of 400 [80.75 % accuracy]	72 out of 400 [82.00 % accuracy]
4	68 out of 400 [83 % accuracy]	68 out of 400 [83.00 % accuracy]
5	83 out of 400 [79.25 % accuracy]	65 out of 400 [83.75 % accuracy]

On an average, considering unigram frequency gives us an prediction accuracy of 81.25 % (individually 80.75 %, 82.5 %, 80.75 %, 83.0 %, 79.25 %). Similarly considering unigram presence gives us an prediction accuracy of 82.75 % on an average (individually 81.5 %, 83.5 %, 82 %, 83. %, 83.75 %). From the above, it can be inferred that slightly better performance is achieved by taking into account only the unigram presence, not unigram frequency.

We tried to further enhance the model by not including basic words like a, an, the, or , and , for etc. as part of the unigram features. This didn't have much affect when considering unigram frequency giving us an prediction accuracy of 81.1 %. The more imporant observation is in the case of the unigram presence where the performance increased to **84.25 %**. The complete results can be obtained from the table below.

Fold #	Unigram Frequency Miscalculations	Unigram Presence Miscalculations
1	82 out of 400 [79.5 % accuracy]	65 out of 400 [83.75 % accuracy]
2	71 out of 400 [82.25 % accuracy]	66 out of 400 [83.5 % accuracy]
3	79 out of 400 [80.25 % accuracy]	61 out of 400 [84.75 % accuracy]
4	65 out of 400 [83.75 % accuracy]	52 out of 400 [87 % accuracy]
5	81 out of 400 [79.75 % accuracy]	71 out of 400 [82.25 % accuracy]

Bigram Language Model Classifier

By taking into consideration, the dominance of unigram presence features over unigram frequency features, we ran the data set through a language model classifier with bigram presence features. Just like in the case of unigram, considering perplexity as the measure, we categorized the documents with a success rate of **73.25 %** on an average. Detailed fold-wise prediction accuracy can be seen in the table below.

Fold #	Bigram Presence Miscalculations
1	89 out of 400 [77.75 % accuracy]
2	110 out of 400 [72.5 % accuracy]
3	111 out of 400 [72.25 % accuracy]
4	116 out of 400 [71.00 % accuracy]
5	109 out of 400 [72.75 % accuracy]

Perceptron Classifier

Since there is no hard and fast rule for feature selection in binary perceptron classifier, we took unigrams as the features.We feature engineered and calculated the weight for each feature by iterating until the weights converged. For each test document, an activation sum is calculated. Based on this value, the document is classified either to have positive setiment or negative sentiment. Considering unigram presence features, we were able to implement a model with an accuracy of nearly **85 %** . Refer to the table below for fold-wise accuracies.

Fold #	Perceptron Frequency Miscalculations	Perceptron Presence Miscalculations
1	82 out of 400 [79.5 % accuracy]	54 out of 400 [86.5 % accuracy]
2	67 out of 400 [83.25 % accuracy]	60 out of 400 [85 % accuracy]
3	73 out of 400 [81.75 % accuracy]	66 out of 400 [83.5 % accuracy]
4	64 out of 400 [84 % accuracy]	59 out of 400 [85.25 % accuracy]
5	70 out of 400 [82.5 % accuracy]	66 out of 400 [83.5 % accuracy]

Also, in the case of frequency the number of iterations taken for weight vectors to converge is around 330 but in the case of unigram presence it is just around 40.

Similarily, we also used bigrams as the feature vectors instead of unigrams. We got an average prediction accuracy of 82.90 %. Complete accuracies can be found in the table below.

Fold #	Bigram Presence Accuracy
1	84.75 %
2	81.25 %
3	83.50 %
4	80.00 %
5	85.00 %

Support Vector Machine Classifier

For support vector machine classifier we make use of the already existing LibLinear toolkit. We pass the unigram features to the toolkit as in the format of feature nodes. The model is then trained using the 1600 training reviews and is later used to predict the sentiment of the 400 testing reviews. This is repeated for 5 times and an average prediction accuracy is calculated.

We used the value of C parameter as 1.0 and eps parameter as 0.01 and ended up with the accuracies shown in the table below.

Fold #	**Unigram Frequency Accuracy**	**Unigram Presence Accuracy**
1	81.00 %	86.25 %
2	93.25 %	92.25 %
3	84.5 %	86.50 %
4	86.5 %	90.00 %
5	83.5 %	86.75 %

The average prediction accuracy in the case of unigram presence is obatained as **88.35 %**. L2R_LR_DUAL is the solver type used.

For the same SVM parameters when bigrams are used as feature vectors instead of unigrams, the accuracies are slightly lower. On an average we get a predictive accuracy of 83.05 %. Complete accuracies can be found in the table below.

Fold #	**Bigram Presence Accuracy**
1	84.50 %
2	81.50 %
3	81.25 %
4	83.00 %
5	85.00 %

Summary

To summarize,

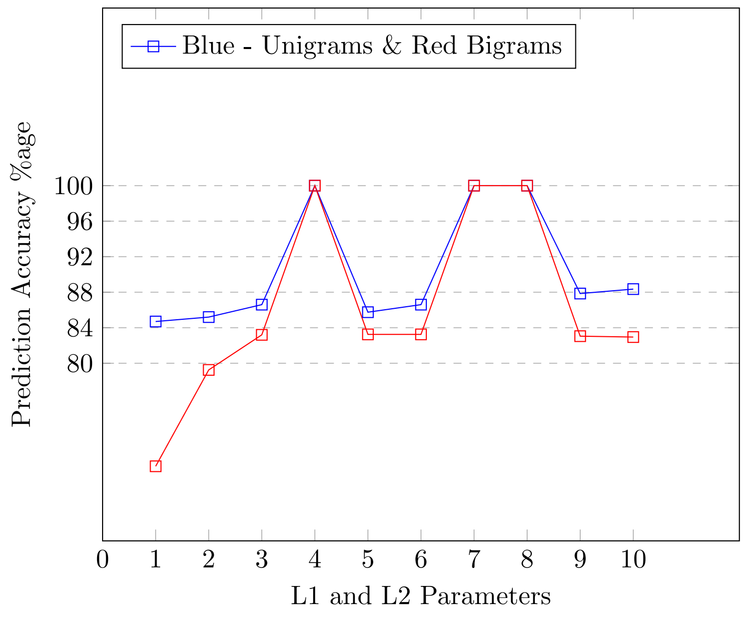
Classifier Type	**Average Prediction Accuracy**
All Unigram frequency	81.25 %
All Unigram presence	82.75 %
Unigram frequency [Length ¿ 3]	81.1 %
Unigram presence [Length ¿ 3]	84.25 %
Bigram presence	73.25 %
Perceptron unigram frequency	82.2 %
Perceptron unigram presence	84.75 %
Perceptron bigram presence	82.90%
LibLinear SVM unigram frequency	85.75 %
LibLinear SVM unigram presence	**88.35 %**
LibLinear SVM bigram presence	83.05 %

Observations

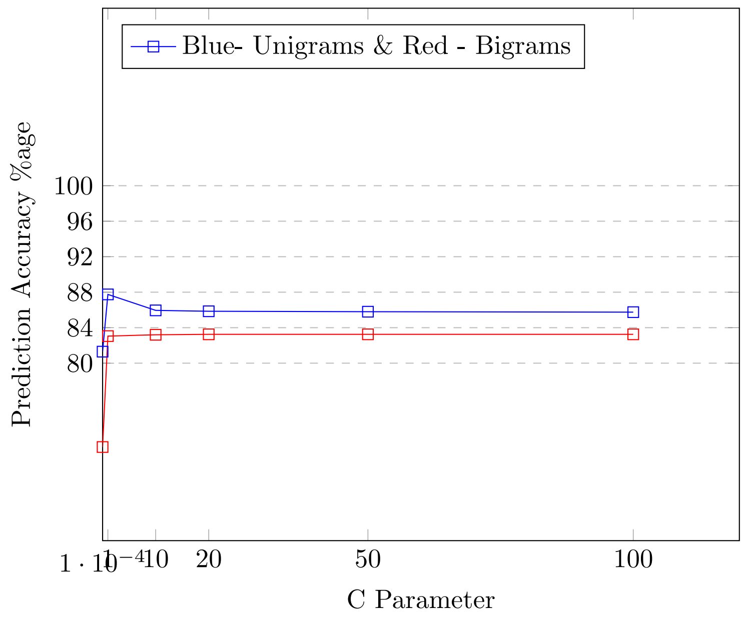
During this study we have come across many interesting observations. We listed some of them below -

Regularization Parameters : In the case of SVM, we analyzed how the prediction accuracy changed with respect to the choice of L1 and L2 parameters for both unigrams and bigrams. Apart from the SVR_DUAL solver types, L2R_LR_DUAL and L2R_L1R solver types give high prediction accuracies. Below is the graphical view -

On x-axis : 1 - L1R_L2LOSS_SVC ; 2 - L1R_L2 ; 3 - L2R_L1LOSS_SVC_DUAL ; 4 - L2R_L1LOSS_SVR_DUAL ; 5 - L2R_L2LOSS_SVC ; 6 - L2R_L2LOSS_SVC_DUAL ; 7 - L2R_L2LOSS_SVR ; 8 - L2R_L2LOSS_SVR_DUAL ; 9 - L2R_LR ; 10 - L2R_LR_DUAL

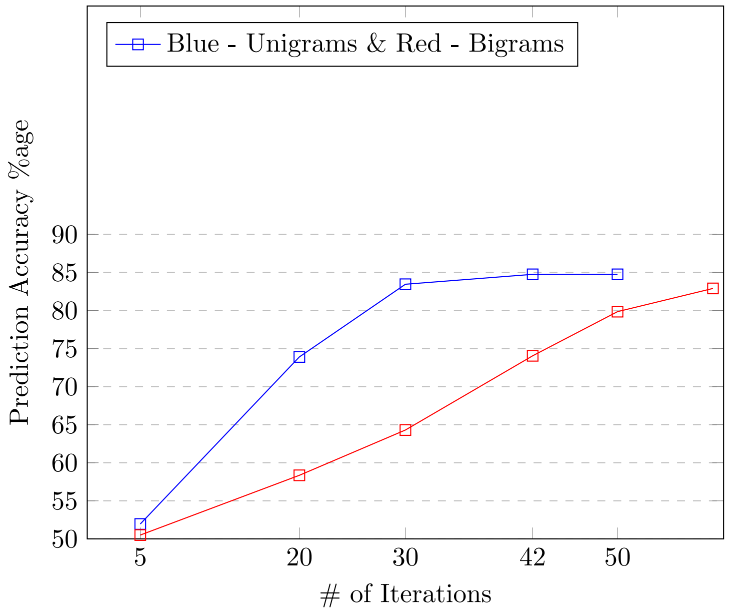


Similarly, we studied the affect of C on the prediction accuracy of SVM model. As the value of C approaches 1, the prediction accuracy increases. For any value greater than 1, the prediction accuracy keeps decreasing slightly. Below is the graphical view -



In the case of perceptron classifier, we also studied theaffect of number of iterations on the prediction accuracy. What we noted is quite obvious - as the number of iterations increased initially the prediction percentage also increases. As the iteration count reaches a particular value, the prediction accuracy doesn't change much as the weight vectors stop changing.

From a graphical point of view -



Different feature vectors : For perceptron and SVM, we implemented the model with features as unigram presence , unigram frequency and bigram presence. In both the cases, unigram presence has a higher prediction accuracy than unigram frequency.

One more striking feature is that unigrams have a better prediction accuracy than bigrams. Theoritically, bigrams are considered to have better contextual information than unigrams as they take into consideration the impact of the previous word on the occurance of the presence word. Also, it is believed that sentimental analysis requires more contextual understanding of the data than traditional text classification problems. If that is the case, bigrams should have higher prediction accuracy compared to unigrams. But, from the reports above we can clearly see that unigrams have a considerably better prediction accuracy than bigrams (in the case of frequency as well as presence). This shows that in our setting of predicting sentiment of movie reviews, bigrams are not as effective at capturing the context.

In the case of SVM, to classify based on more impactful words, we ignored trival words like a, an, the, for, of, if etc. as part of the features. But this decreased the prediction accuracy by 2.4 % from the initial value.

Pros and cons of individual classifiers :

i. Perceptron :

If the proper solution exists, Perceptron Classifier converges very quickly and gives a highly accurate and efficient model. In case of lack of proper solutions, it might take a while to converge. For example, when unigrams presence was considered as feature vector the number of iterations taken to arrive at appropriate weight vectors was around 40 and in the case of unigrams frequency it was around 330.

On a negative note, perceptron classifier doesn't classify accurately when the data samples are not separable.

ii. Support Vector Machine :

Support Vector Machine being a kernel based framework has high flexibility. It also produces a highly accurate prediction model for practical purposes even when the training sample is minimal. SVMs, especially the LibLinear is very fast and robust.

On the negative side, SVM requires high quota of memory and processing time. And, it produces a efficient model only when the training data has equal distribution of negative and positive data samples.

iii. N-gram language model classifier

The major pros of n-gram language model classifiers is its relative simplicity and ability to scale up. We can incorporate more and more context by simply increasing the value of n thereby making the application more and more scalable.

Error Analysis

For error analysis, we took a look at the movie reviews for which all the classifiers get confused about. We gathered a set of reviews which all/most of the classifiers implemented couldn't classify correctly and manually looked at where things could have gone wrong.

One of the major pitfall for most of the wrongly classified review was 'twarted expectations'. For example, consider the following part of a negative review

"best remembered for his understated performance ..... brian cox brings something special to every movie he works on.....he's only occasionally given something meaty and substantial to do........if you want to see some brilliant acting , check out his work as a dogged police inspector opposite frances mcdormand in ken loach's hidden agenda........he's an even-tempered , funny , robust old man who actually listens to the kids' problems.......charming them with temptations from the grown-up world"

Although the reviewer didn't like the movie as a whole, he/she liked a character and how the actor potrayed it. Thus the review has many words which indicate the opposite sentiment to that of the entire review. Such instances are difficult to comprehend and confusing for classifiers and better understood by humans. This can be overcomed to a certain extent by peforming subjectivity extraction (considering only subjective sentences and leaving out objective sentences) during data pre-processing.

Also, some of the users used sarcasm and a bit of sense of humour in their reviews. Sarcasm and sense of humour detection are hard problems for machines and often result in misclassification.

Reference

[1] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan, Thumbs Up ? Sentiment Classification using Machine Learning Techniques. EMNLP, 2012.