# Sentiment Classification using Machine Learning

Huzefa Dargahwala, Rahul Pasunuri, S Dinesh Balaji

December 9, 2014

## Part I
# Code implementation

## 1 Goal

The intent of this project is to do a detailed analysis of the performance of various machine learning algorithms on the task of sentiment classification on movie reviews.

The paper has been segmented in the following manner. Part I comprises of the code implementation of the discussed algorithms in which Section 2 describes the literature we studied. Section 3 describes the approach in detail, Section 4 gives the reader an overview of the datasets used and Section 5 gives the reader a wide perspective of how the algorithms perform on the task of sentiment classification task and Section 6 comprises of an insightful detailed analysis of the results. Part II of the paper focuses on WEKA. It has Section 7 which describes the goal of Part II, Section 8 gives a birds eyeview of the methodology, and Section 9 consists of the analysis of the used algorithms presented side by side the results obtained.

## 2 Literature Survey

Supervised sentiment classification is a widely researched topic[2, 3, 1]. Usage of linguistic cues is also considered[3]. An extensive survey on the sentiment classification and the sentence subjectivity classification can be found in [6]. One of the earliest work is the supervised approach to movie review classification[2], which we are using as our base model. Considering the fact that the sentiment classifiers are highly sensitive to the domain used for learning, several domain adaptation techniques have also been considered [1].

Unsupervised techniques have also been explored like in [4], where they identify the semantic orientations of a word by computing mutual information with positive words and negative words with the help of search engine results. Regression based approaches have also been considered where they categorize reviews into more than one category (E.g. ratings of 1-5) [7]. Subjectivity classification is one important factor while analyzing sentiment(not all sentences of a review might be opinions). In[8], we can see usage of simple Naive Bayes for the subjectivity classification.

## 3 Methodology

The approach in this paper is inspired by a similar sentiment classification project[2]. Once the data is acquired, it is cleaned and processed to make it suitable as input for the algorithms. This step is crucial as natural language and especially review data have certain peculiarities that make it the worst for sentiment classification[4]. Misspelling, improper punctuation, bad grammar, use of slang and abusive words have an adverse effect on the performance of the classifier being used.

### 3.1 Feature Selection Strategies

Continuous reviews are tokenized into individual words using punctuation marks and empty spaces as separators. Using punctutaion marks as separator is a simple but not an optimum strategy to tokenize words in a sentence. Words like “W.W.E” will be split into 3 separate words. This phenomenon results

in erroneous training and test vectors and is likely to cause a drop in the perfomance of the algorithms being used.

All unigrams in the cleaned dataset comprise the feature space. This is called the Bag-of-Words(BoW) approach which produces the best results[2]. However, this approach does not capture negation of words. Appending "NOT" to every word that appears after "not" till the first punctuation mark takes care of this problem.

Eg. "This is not a bad movie." => ["This", "is", "NOTa", "NOTbad", "NOTmovie"]

The feature values used are presence or absence of a word in the review. This project limits its scope to classifying a review as either positive or negative, hence the previously described method of capturing negation works well enough. If the horizon of the task was broadened to classify the reviews into a third category of neutral reviews, this approach will fail dramatically.

Due to the curse of dimensionality and the fact that many of the words used in sentences do not contribute towards the overall sentiment of the review, to reduce computational space and time complexity it is imperative that such words be excluded from the feature space. The curse of dimensionality also deterred this project from using bigrams and trigrams as features because there is not enough training data to produce dense enough feature vectors for generating a good model for the classification task. Given enough training data, bigrams and trigrams will outperfom unigrams because problems like capturing negation in the BoW will not arise in that case.

One common approach used for pruning of features is the use of a list of common stop words in the English language. This approach fails to remove domain specific stop words. In this specific task words like "movies", "film-maker" and so on. The following two strategies have been used for selecting features for one of the reasons that both strategies are asymptotically linear in time and do not add incur much runtime.

#### 3.1.1 Document Frequency Thresholding

Document Frequency Thresholding (DFT) is an intuitive feature selection strategy. Document frequency is the number of documents in which a particular word appears. It is a good indicator of how important a word is in the corpus. All words that occur in both negative and positive reviews above a predetermined threshold are filtered out. This strategy is scales to very large corpora[9].

#### 3.1.2 Information Gain

Information Gain (IG) is calculated for each unique feature and the feature is removed if the IG is below a certain threshold. IG is proven to be the best feature selection strategy among others like mutual information, chi-squared, maximum entropy and document frequency thresholding in the text classification problem[9].

The output of the feature selection process is a list of features. Each review is a sparse vector of 1's and 0's of the length of number of features. These vectors are then used to get a prediction of whether a review is positive or negative. Implementation details for each algorithm are described below.

### 3.2 Algorithms

#### 3.2.1 Logistic Regression

Inceptive implementation of this algorithm did not include optimizations for handling feature space of the order of hundreds of thousands. To reduce convergence time, certain tweaks were applied to the way the program handles data. To reduce space complexity, instead of representing each review as sparse vector of 1's and 0's, it was represented as a condensed vector of the words and its class label appended at the end. Each word in the vector is then mapped to a dictionary to get linear access time.

Eg. Test or Train Vector = ['disgusting', 'worst','kristen','stewart','NEGATIVE']

#### 3.2.2 Naïve Bayes

Similar to Logistic Regression, this algorithm required optimizations before it was run on the entire dataset.

### 3.2.3 Bagged Decision Trees

Finding an optimal decision tree is an NP complete problem[5]. Bags of several suboptimal decision trees were used in an attempt to improve accuracy of a fixed depth tree. The results obtained were quite unexpected.

# 4 Datasets

| | Set 1 | Set 2 |
|---|---|---|
| Source | IMDB | Rotten Tomatoes |
| Property | Verbose reviews | Single line Terse Reviews |
| Total Reviews | 2000 | 10662 |
| Total Size (Uncompressed) | 10.9 MB | 1.18 MB |
| Ratio of Positive to Negative Reviews | 1:1 | 1:1 |
| Ratio of Training to Test Samples | 4:1 | 4:1 |

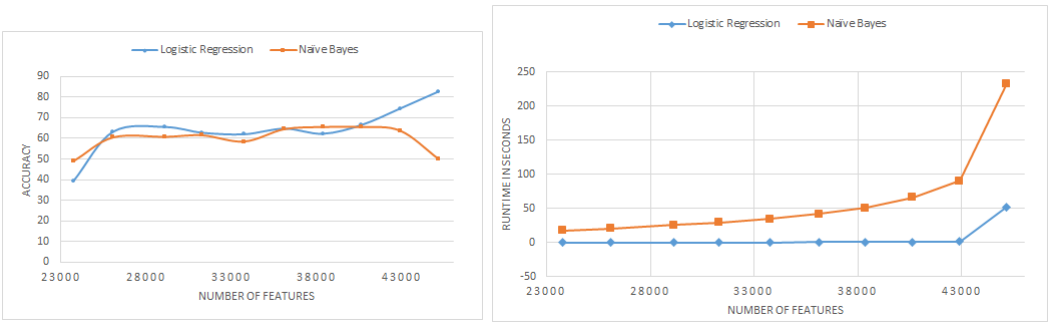
Table 1: Summary of Datasets

It has been ensured that there is no class imbalance in training and test sets.

# 5 Results

This section will compare results accumulated over a range of parameters like number of features, accuracy, and running time. Naive Bayes and Logistic Regression have been compared together as they are linear classifiers. Decision Trees have been discussed separately.

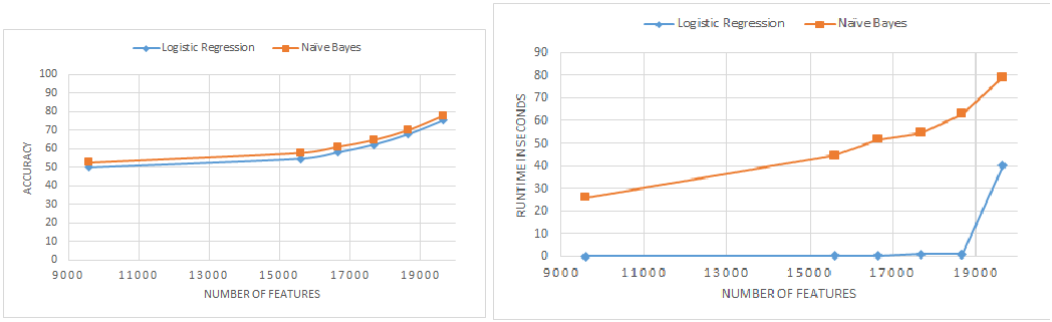
Below are the results for Logistic Regression and Naive Bayes algorithms when run on Set 1 and Set 2.




(a) Number of Features vs Accuracy

(b) Number of Features vs Running Time

Figure 1: Logistic Regression and Naive Bayes Results for Dataset 1




(a) Number of Features vs Accuracy

(b) Number of Features vs Running Time

Figure 2: Logistic Regression and Naive Bayes Results for Dataset 2

Decision Tree Results are reported below.

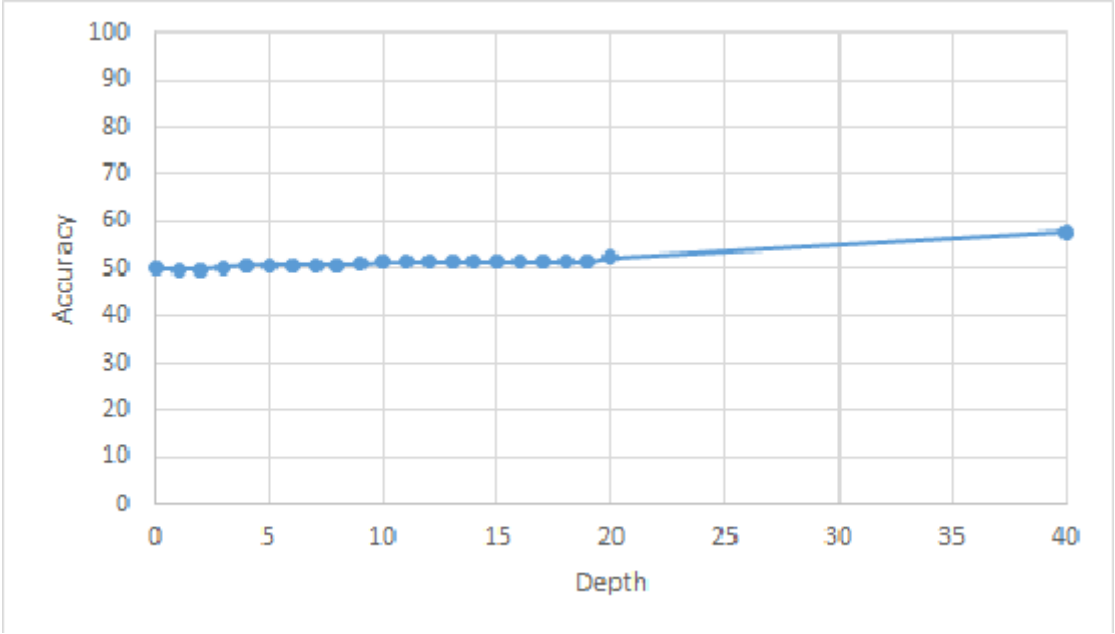
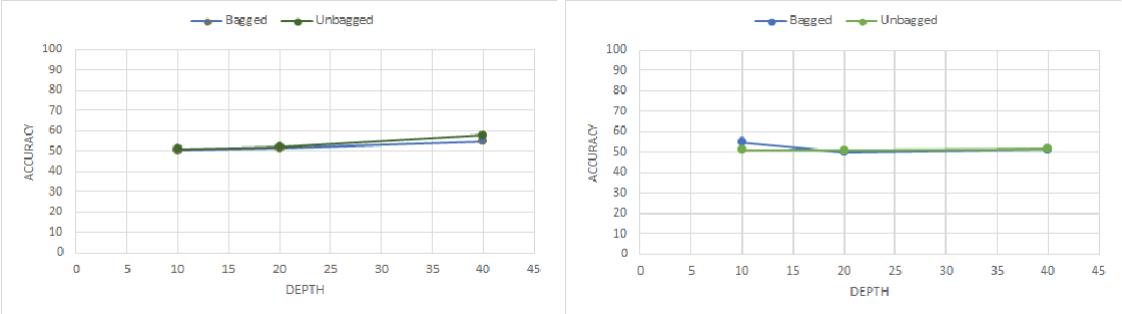



Figure 3: Depth vs Accuracy Graph for Decision Trees




(a) Dataset 1

(b) Dataset 2

Figure 4: Accuracy vs Depth for Bagged and Unbagged Decision Trees

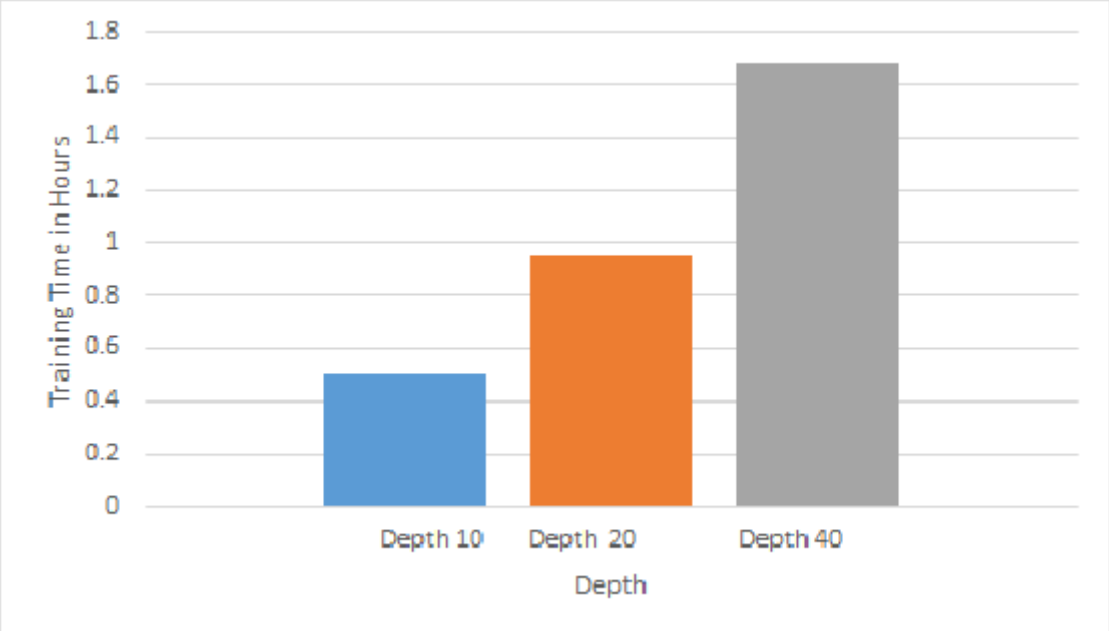



Figure 5: Training Time for Decision Trees

# 6 Analysis, Insights and Conclusion

Since the data is really informal and without strict grammar, capturing the syntactic and semantic meanings is really hard in this case. In our opinion, bag of words might be a good choice here. But in the case of different dataset, where language conforms to strict rules (E.g. newspaper articles), bag of words may not be a good choice.

From the results of Logistic Regression and Naive Bayes on Set 1 in Figure:1a, it can be observed that initially, accuracy of the both the classifiers increases with the number of features and stabilizes in a particular range. Once the number of features is beyond a certain threshold, accuracy of Naive Bayes starts decreasing but the accuarcy of Logistic Regression keeps increasing, this may be because Logistic Regression can assign importance to features. It is assigning less weight to noisy features compared to more important features. Our results show that Logistic Regression is more robust to noisy features when compared to Naive Bayes.

Set 2 has a much limited feature space than Set 1, in Figure:2a, it is observable that accuracy of both Naive Bayes and Logistic Regression increases with increase in number of features.

With the results of performance of Decision Trees, it is quite surprising that bagging decision trees actually reduces accuracy, see Figure:4. This may be because we are not using enough bags. Due to the time constraint, number of bags used for arriving at a decision is limited to 10. The base accuracy of 50% with a decision stump which increases to 52.25% at a depth of 20 and to 57.4% at the depth of 40 is not much of an increase given he very long training time for decision trees, see Figure: 5. Given the impractically large training time, it is not surprising that decision trees are not often used for sentiment classification. Logistic Regression and Naive Bayes clearly outperform Decision Trees when decision trees are not grown to a good enough depth and there are not enough bags to base the final decision on. Finding the good enough depth and the ideal number of bags for getting good accuracy is a question whose answer can be obtained by experience and trying out various depths and number of bags. This could not be done ascribing to the time constraint for the project.

Acknowledgment: We would like to thank Professor Natarajan for his guidance in this project. Majority of the computations for this project were performed on Big Red II and Hulk.

# Part II
# WEKA Analysis

## 7 Goal

This part of report concentrates on utilizing algorithms implemented in WEKA for the task of sentiment classification on movie reviews.

## 8 Methodology

The input to WEKA are features and their values in the instances along with their labels.

### 8.1 Feature Selection

The BoW approach as described in Section 3.1 has been used. All strings were preprocessed to convert all alphabets to lowercase. In the string to word vector filter, the number of words was limited to 10,000 after removing stopwords.

### 8.2 Feature Evaluation

#### 8.2.1 Information Gain

IG is preferred over other methods since it gives the best results on the task of text classification[9]. The results for which can found in Table:2a

Analysis of results: "Bad" ranks as the most discriminant feature but it and other features magnitude of importance is 10 times lower than set 2 when compared to set 1. This may be because of more reviews in set 2 compared to set 1.

#### 8.2.2 Principal Component Analysis

The idea here is to analyze which combination of features are most descriptive of the data points. The variance coverage was set to 0.95 and only the first 648 tokens in both datasets were considered due to memory constraints. The Top 3 principal components have been presented in Table:2b

Analysis of the results: The analysis here reflects on how a combination of mediocre information gain features like 'kids','movies','aliens' etc were better discriminant than any combination of features involving high information gain features like 'bad','boring','stupid', etc. One possible explanation could be that these combination of features could have high correlation value and when combined, they would have highest information gain.

| | | | | | |
|---|---|---|---|---|---|
| Set 1 | Bad(0.079) | worst(0.0465) | stupid(0.0327) | boring(0.0306) | plot(0.0237) |
| Set 2 | Bad(0.0085) | dull(0.0044) | movie(0.0039) | boring(0.0035) | performances(0.0031) |

(a) Top 5 features using Information IG

| Principal Components | Set 1 | Set 2 |
|---|---|---|
| 1 | 'scene', 'films', 'make', 'time', scene' | 'kids', 'movie', 'spy', 'adult', 'effect' |
| 2 | 'effects', 'alien', 'aliens', 'plant', 'life' | 'es','de','la','subject','opera' |
| 3 | 'comedy', 'plants', 'funny', 'effects', 'alien' | 'cast', 'direction', 'performance', 'acting', 'dialog |

(b) Top 3 Principal Components

Table 2: Results of Feature Selection in WEKA

### 8.3 Classification Task

For the classification task, 3 methods have been employed. Namely, RandomForest, ADABOOST, and SVM implementation of WEKA.

#### 8.3.1 RandomForest

Before running the RandomForest Algorithm, the number of features was set to 11 and the number of trees were varied to get different results.

#### 8.3.2 ADABOOST

A Decision stump with with meta learner ADABOOST was trained and the number of trees were varied to obtain a variety of results.

#### 8.3.3 SVM

In WEKA, SVM is implemented using Sequential minimal optimization (SMO) which is an algorithm for solving the quadratic programming (QP) problem that arises during the training of support vector machines.

# 9 Report and Analysis of WEKA Results

## 9.1 RandomForest

Below are the results obtained on running RandomForest in WEKA.

| Number of Trees | Set 1 | Set 2 |
|---|---|---|
| 10 | 70.25% | 67% |
| 100 | 83% | 76% |
| 1000 | 86.5% | 81.5% |

Table 3: Accuracy for RandomForest

Analysis of the results: Although prone for overfitting, each tree makes local choices(instances) based on local inputs(features), when aggregated, this produces much smoother decision boundary. Hence more the number of trees, more is accuracy because of aggregation of votes where all classifiers overlap to form a bigger picture

## 9.2 ADABOOST

| Number of Trees | Set 1 | Set 2 |
|---|---|---|
| 10 | 67.75% | 50.98% |
| 100 | 78.75% | 53.61% |
| 1000 | 83.25% | 60.5% |

Table 4: Accuracy for ADABOOST

Analysis of the results: Set 1 and Set 2 show different trends. While the accuracy shows an steep increase in Set 1, it does not show a significant increase in Set 2. One possible explanation could be lexical diversity. In Set 2, many of the features (words of naive reviewers) were in both classes and hence due to somewhat linearly inseparable data, the accuracy didn't improve much.

## 9.3 SVM

| | Linear Kernel | | RBF Kernel | |
|---|---|---|---|---|
| | Set 1 | Set 2 | Set 1 | Set 2 |
| Accuracy | 79.5% | 69.79% | 85% | 67.58% |
| Runtime | 10.5s | 92.78s | 32.5s | 201.58s |
| Number of Support Vectors | | | 1702 | 9660 |

Table 5: Accuracy for SVM with Linear and RBF Kernel

Analysis of the results: The relatively good performance of linear kernel over RBF kernel indicates that Set 1 has a good degree of linear separability. The Linear kernel takes almost 1/3rd the runtime of the RBF kernel.The RBF kernel is a squared exponential kernel whose model complexity is infinite (dot products in infinite space) and hence with a large amount of inseparable data (Set 2), it tries to learn a comlex model (9660 support vectors for Set 2) and eventually leads to overfitting and low accuracy.

# References


[1] A. Aue and M. Gamon. Customizing sentiment classifiers to new domains a case study. *Proceedings of Recent Advances in Natural Language Processing (RANLP)*, 2005.

[2] Pang B, Lee L, and Vaithyanathan S. Thumbs up ? sentiment classification using machine learning. *Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 79–86, 2002.

[3] P. Chesley, B. Vincent, L. Xu, and R. Srihari. Using verbs and adjectives to automatically classify blog sentiment. *AAAI Symposium on Computational Approaches to Analysing Weblogs (AAAI-CAAW)*, 2006.

[4] Turney P D. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. *Proceedings of the 40th annual meeting on association for computational linguistics*, 2002.

[5] Hyafil L and Rivest R. Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, 1976.

[6] Bing Liu. Sentiment analysis and subjectivity, handbook of natural language processing, second edition.

[7] B. Pang and L. Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *Proceedings of the Association for Computational Linguistics (ACL)*, 2005.

[8] J. Wiebe, R. F. Bruce, and OHara. Development and use of a gold standard data set for subjectivity classifications. *Proceedings of the Association for Computational Linguistics (ACL)*, 1999.

[9] Yang Y and Pederson J O. A comparitive study on feature selection in text categorization. *ICML 8*, pages 412–420, 1997.