Binary Classification for Text Documents

Jihoon Kim
UCSD Division of Biomedical Informatics
j5kim@ucsd.edu

Hari Damineni
UCSD Computer Science & Engineering
hdaminen@cs.ucsd.edu

André Christoffer Andersen
NTNU Industrial Economics
andrecan@stud.ntnu.no

May 24, 2011

Abstract

Multinomial Naive Bayes (NB) is a widely used method for document classification due to its high performance and fast execution time. In this article, we apply NB to movie review data and compare its performance to Support Vector Machine under different settings of preprocessing.

1 Introduction

It is useful to classify reviews as positive or negative in *sentiment*, the overall opinion towards the subject matter. The sentiments present in product review add value to an online shopping experience. Reviews come in different formats such as, ratings, reviews, feedbacks etc. Sentiment analysis can be performed by employing supervised machine learning algorithms for classification based on sentiment analysis. Machine learning also aids in presenting a normalized view about a product for the following reasons. First, some reviews are just free flowing texts and are not accompanied by quantitative ratings. So, a simple rating to sentiment translation is not a solution. Second, although ratings are present in some instances, ratings information does not provide a consistent view about the product. For example, a rating of 2 by a critical reviewer might actually reflect a positive sentiment where as the same rating from a relatively lenient might have a negative sentiment. In some cases, ratings based on different scales (1 to 10, or 1 to 5 etc.) might result in data loss in scale translation. Machine learning techniques discussed in this article, overcome these limitations by predicting sentiments by analyzing the features present in the reviews (textual data). In this article, we apply Naive Bayes (NB) and Linear Support Vector Machine (LSVM) models for classifying movie review documents into positive and negative reviews. The documents are represented in terms of feature vectors and different NB and LSVM models are built by using different data selection, transformation, and data weighting techniques. We obtain the average accuracies 79.29% and 82.65% on test sets for NB and LSVM respectively.

2 Data Description

Data used in this article is the movie review data from http://www.cs.cornell.edu/People/pabo/movie-review-data/. The data contains 2 groups, each containing 1000 free-flowing text documents of movie reviews where each

group labels the documents belonging to the group as positive or negative. The two groups can be visualized as two bag-of-words' where the words leading to a positive sentiment come from the bag labelled positive and same is the case with words from the negative bag of words. Each document contains ASCII encoded data divided into a several sentences where each sentence occupies line. The ultimate objective of our work is to learn a supervised model based on the positive and begative bag-of-words and predict the bag to which a test example belongs.

3 Data Pre-Processing

The objective of the data pre-processing stage is to represent the documents in terms of feature vectors. In this article, feature represents a unigram feature which is a unique word present in the dataset. The pre-processing module employs the following in-order steps.

1. *Word Splitting*: Each document is parsed and converted into part-of-speech-tagger (POST) format. All the characters are converted into lower case and the text is separated into individual words by taking punctuation into consideration. This steps ensures that the free-text is converted into a collection of individual words. A modified version of the online tool at, http://l2r.cs.uiuc.edu/ cogcomp/atool.php?tkey=WS, enabled us to divide a word into sub-words which formed the initial basis for feature extraction.

2. *Stop Word Removal*: Stop words are words which do not provide any information about the substance in the document and therfore do not contribute to classification. For example, words such as *he*, *she*, *we* do not add any information and therefore are redundant. In this step, we remove the stop words from all the documents and removed redundant features and reduced the feature space size. We use the stop words from the list found at, http://www.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/stop_words.

3. *Stemming*: Words are converted to their root form. For instance, word *running* is converted to *run*. This step ensures that derived words are normalized to their stem or root form which aids in contracting different words to their root form and reducing the total number of unique words. We modify the Porter Stemmer code found at http://tartarus.org/ martin/PorterStemmer/ for this purpose.

4. *Feature Extraction*: All the documents, pre-processed by steps above, both positive and negative are scanned and a set of single words, unigrams, is created. This set of words is the set of unigram features and acts as the feature space for our learning and testing experiments. The feature set is represented as a vocabulary set, V, built from the dataset where each feature is identified by a feature identifier. The feature set is represented by the following equation.

$$V = \{f_1, f_2, ..., f_m\} \tag{1}$$

   where m is the size of the feature set.

5. *Feature Pruning*: We eliminate features with just one occurrence and only considered features with more than one occurrence. This is done because the sole presence of the word in all the documents could be because of some typographical error and not accounting for this would lead to a expansion of feature size and computing inefficiencies without any statistically significant increase in prediction power. Based on the $2,000$ documents and using all the unique features, our feature set size is $18,433$. Though there are $18,433$ unique words, each word has different frequency of occurrence (Figure 1). Most word sizes range between 4 and 8. To aid experimentation purposes, a different feature set is created based on degree of association of a word and positive label using Fisher's exact test.

6. *Document Vector Creation*: Based on the vocabulary, V mentioned in Equation 1, each document is parsed and transformed into a frequency vector representation as following

$$d = \{x_1, x_2, ..., x_m\} \tag{2}$$

where x_i is the number of times feature f_i occurred in the document. In addition to the feature frequency vectors, to aid our experimentation efforts, document vectors based on feature presence are also created. They are represented as follows.

$$d = \{I(f_1), I(f_2), ..., I(f_m)\} \tag{3}$$

where $I(.)$ is an indicator function indicating the presence of a corresponding feature in the document. The motive behind the feature presence vectors is to study the effect of just using the presence of features as opposed to the frequency of features on prediction quality.

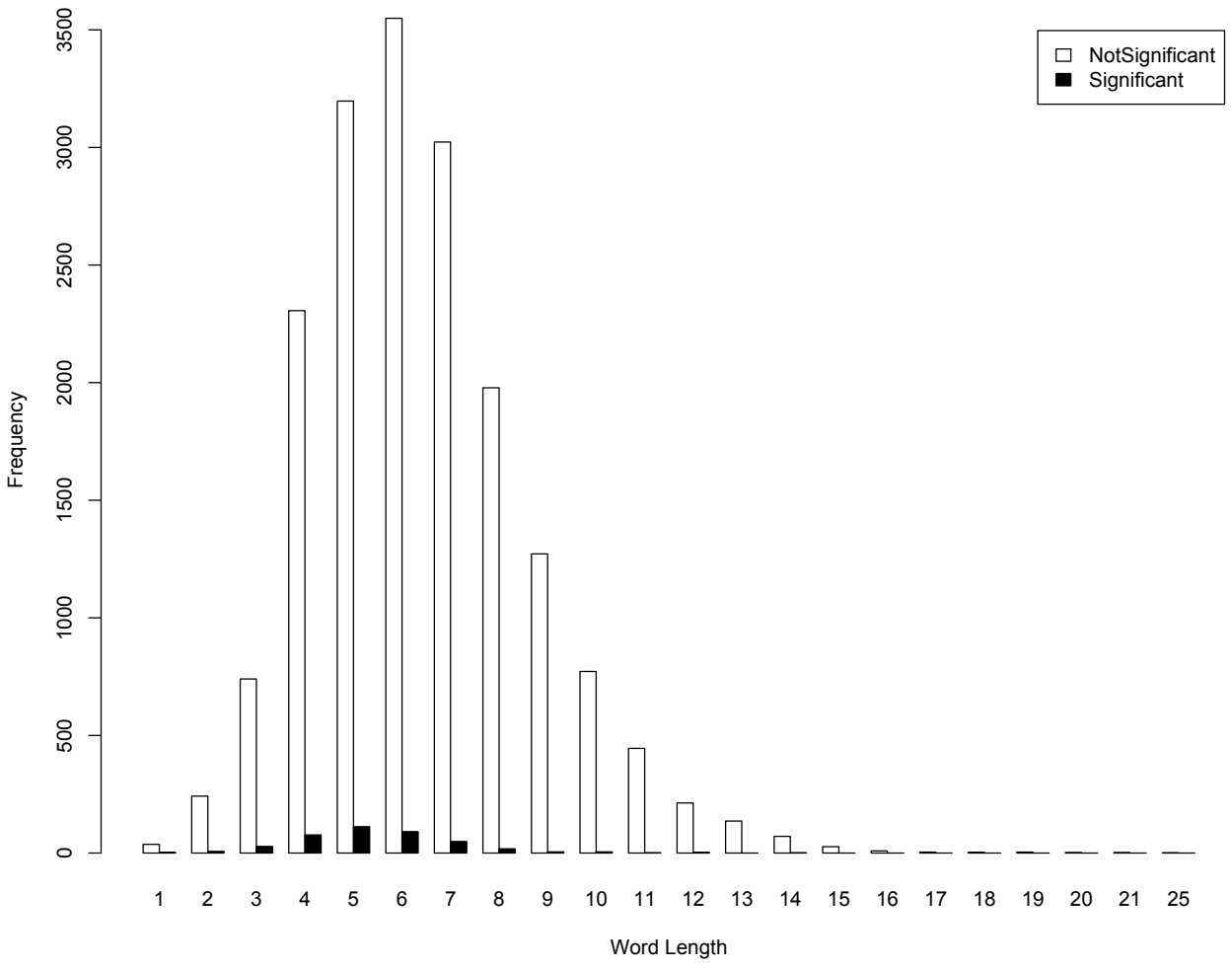

Figure 1: Histogram of the word size in the bag-of-words per group of features. Significant features ($P < 0.05$) are classified into two groups using Fisher's exact test.

The data pre-processing module is implemented in Java in the steps where no third-party tools were used.

4 Naive Bayes Modeling

According to NB method of binomial classification, classification task of document d, under certain conditions described below, reduces to

$$\Pi_{j=1}^{m} \alpha_j^{x_j} > \Pi_{j=1}^{m} \beta_j^{x_j} \tag{4}$$

where α_j is the probability of feature j in the positive training set and similarly β_j is the probability of feature j in the negative training set with size similar to that of the positive training set. NB modeling consists of computing α_j and β_j using the positive and negative training vectors respectively using the following equations.

$$\alpha_j = \frac{\sum_{xp} xp_j}{\sum_{xp} \sum_j xp_j} \tag{5}$$

$$\beta_j = \frac{\sum_{xn} xn_j}{\sum_{xn} \sum_j xn_j} \tag{6}$$

where xp and xn represent all documents from the positive and negative training examples. Equation 4 assumes equal probability of both positive and negative classes in the training example sets. This is achieved by using same number of positive and negative documents during training. Once the α_j and β_j vectors are separately computed based on positive and negative training examples, a test example can be classified into positive or negative by testing the condition in Equation 4 in linear $O(m)$ time.

Add-one Smoothing. Sometimes absence of certain features, $\sum_x x_j = 0$ from the training set of a specific class results in zero probability for that feature. Furthermore, the probability of a test document with the missing feature and belonging to this class would be zero because of the Π operation over the probability of features found in the test document. To avoid such non-realistic zero probabilities, we artificially add an integral value to the zero x_j value for feature j so that the resultant probability of feature j is non-zero. We used add-one smoothing in order to smooth by assigning minimum strength to the missing features. Smoothing is done before the probabilities are calculated so that smoothing updates the $\sum_x \sum_j x_j$ factor also.

NB implementation. The implementation of NB modeling is done using a Java program. The space and run-time complexity are in the order of $O(m)$ for each class where m is the size of the feature set. Scalability could be a issue particularly when dealing with large feature set sizes. However, as the multiplication of probabilities is order independent, feature set can be divided into clusters and distributed parallelization paradigms could be used to make the model scalable. One more issue with the computation is that, as the number of multiplications of probabilities increases, the value exponentially decreases since the probalities are small values and moreover they are raised to the powers of feature frequencies. This could lead to underflow situations where the value cannot be represnted by machine representation. Our implementation avoided this underflow issue by repeatedly using a constant factor to scale the multiplication result after each multiplication if the result goes lower than a fixed threshold. We applied the same process and amount of scaling for the negative side calculations also. Our scale factor was 100 and the threshold is 0.00001.

Leakage. We used 10-fold cross validation to build the NB model. In order to avoid the test fold influence on the model through leakage, probabilities of features in the positive and negative classes, α_j and β_j are calculated using the data in the training fold only.

5 Linear SVM Modeling

Test documents are converted into real valued vectors during the pre-processing stage. The resultant vectors are augmented with class label information, 1 for positive and -1 for negative classes. The augmented vectors of the two classes are combined into a single dataset, randomized and are used for Linear Support Vector Machine (LSVM) modeling for binomial classification. Rapid Miner package is used to model LSVM and 5-fold cross-validation is used to obtain the final accuracy measures on the test data.

Regularization. LSVM model carries the risk of overfitting. A dataset of 2,000 example vectors and 18,433 features is a candidate for over-fitting as the feature/example ratio is too high. To avoid overfitting, grid-search to find the regularization strength parameter (low values of box-constraint parameter, C) is used. Grid-search enabled us to find C (closer to optimum value) with high average accuracy on the test set.

6 Experiments and Results

We conduct three experiments to examine the effect of different classifiers and conditions on accuracy in text classification. Namely, we are interested in examining effects of feature type, feature size and classifier on accuracy. According to Analysis Of VAriance (ANOVA) results in Table 1, all three conditions significantly impact on accuracy.

Feature Type (Freq. vs Presence)	**Feature Size (5K vs .18K)**	**Classifier(NB vs. SVM)**
2.2e-16	7.4e-06	2.2e-16

Table 1: Overall effects on three conditions on accuracy (in ANOVA P-values).

Below we describe each condition.

6.1 Feature Type: Frequency vs. Presence

Frequency: Unigram features are extracted out of text documents and the documents are represented in terms of vectors of feature frequencies where each vector component represents the number of times the feature occurred in the document. NB with modeling 10-fold cross validation and LSVM modeling with 5-fold cross validation is done and average prediction accuracies on test documents is obtained.

Feature Presence: Documents are represented in terms of vectors of boolean values where each non-zero value suggests presence of the specific feature in the document while the zero value presents the absence of the same. NB with modeling 10-fold cross validation and LSVM modeling with 5-fold cross validation is done and average prediction accuracies on test documents are obtained.

6.2 Feature Selection: 5K vs. 18K

According to our pre-processing module there are 18,433 unique features in the combined class document set. he feature set is sorted in descending order according to the occurrence frequency and the top 5,000 features are selected. Feature frequency and feature presence vectors are derived based on the new feature space. Modeling, cross-validation and accuracy calculation is the same as the previous experiments.

6.3 Classifier: NB vs. SVM

Support Vector Machine(SVM) is known to be successful at document classification. We compare its performance with that of Naive Bayes(NB). Table 2 presents the results obtained for the experiments explained so far. The accuracies mentioned are the average accuracy percentages obtained on the test data.

Due to high dimensionality, LSVM needs to be regularized. A continuously decaying by power-of-2 grid search for parameter C is run till we find that the average accuracy on test set stops increasing and starts to decrease.

No. of Features	Feature Type	NB	SVM	P-value
18433	Frequency	76.42	80.90	3.73e-08
18433	Presence	**79.29**	**82.65**	2.51e-07
5000	Frequency	75.01	79.63	5.91e-09
5000	Presence	78.73	81.37	1.48e-05

Table 2: Average accuracy on test-set using NB and SVM under different conditions of feature number and measurement type. P-value is from t-test between NB and SVM under the same row-wise condition.

Table 3 below presents the grid-search results for modeling the feature frequency and feature presence data vectors.

C Parameter	Feature Frequency	Feature Presence
1.0	79.30	81.05
0.5	79.35	81.27
0.0625	79.90	81.43
0.03125	80.15	81.94
0.0078125	80.60	82.13
0.00390625	**80.90**	**82.65**
0.0001	77.45	79.90
0.00001	66.70	68.25

Table 3: Average accuracy on test-set using grid search for LSVM with regularization

7 Discussion and Conclusion

The results show that LSVM outperformed NB. LSVM's relative superiority over NB is consistent with the findings in [1] for comparable models. Our models have better accuracies than those in [1] but not by much difference. The reason for this could be the difference in the number of features. We use a total of 18,433 features in contrast to [1] ,where 16,165 features are used. The accuracy of NB model using the feature presence vectors is better than that of when feature frequency vectors are used. This means that using frequency as a feature increases either the number of false-positives, false-negatives or both. One reason for this could be, as mentioned in [1], *expectation thwarting*, where the document contains too many words which run contrary to the actual sentiment. For example, "Script was amazing, acting was superb, stunts were unbelievable. But I did not enjoy the movie". Pang *et. al* suggest the presence of such documents within the corpus. It would be difficult to classify such documents correctly, and using frequency as a feature would exacerbate the problem.

Experiments in supervised learning are conducted. Labeled documents are provided and the documents labeling process holds the key to better models. The information on the document corpus states that documents are labeled as positive or negative based on the start ratings provided by the users. However, as explained before in this document, a rating is not a uniform indicator of sentiment and therefore labeling based on a rating could be erroneous. For example, a rating of 2 by a user could come from a satisfied user also. Un-supervised learning methods such as topic Models, Latent Semantic Analysis (LSA) or Latent Dirichlet Allocation (LDA) could be used to classify documents based on prior information and topic counts (positive or negative).

References

[1] B. Pang, L. Lee, and S. Vaithyanathan. *Thumbs up? Sentiment Classication using Machine Learning Techniques.* Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Philadelphia, July 2002, pp. 79-86.