
PERFORMANCE ANALYSIS OF SUPERVISED MACHINE LEARNING ALGORITHMS FOR TEXT CLASSIFICATION

PUBLISHED IN 2016 19TH INTERNATIONAL CONFERENCE ON COMPUTER AND INFORMATION TECHNOLOGY (ICCIT)

Sadia Zaman Mishu

Department of Computer Science and Engineering
Rajshahi University of Engineering and Technology
Rajshahi 6204, Bangladesh
sadia_cse09@yahoo.com

S M Rafiuddin

Department of Computer Science and Engineering
Rajshahi University of Engineering and Technology
Rajshahi 6204, Bangladesh
torifat.cs@gmail.com

Published in 2016

ABSTRACT

The demand for text classification is growing significantly in web searching, data mining, web ranking, recommendation systems, and so many other fields of information and technology. This paper illustrates the text classification process on different dataset using some standard supervised machine learning techniques. Text documents can be classified through various kinds of classifiers. Labeled text documents are used to classify the text in supervised classifications. This paper applies these classifiers on different kinds of labeled documents and measures the accuracy of the classifiers. An Artificial Neural Network (ANN) model using Back Propagation Network (BPN) is used with several other models to create an independent platform for labeled and supervised text classification process. An existing benchmark approach is used to analysis the performance of classification using labeled documents. Experimental analysis on real data reveals which model works well in terms of classification accuracy.

1 Introduction

Text classification of labeled documents is becoming increasingly necessary due to the rapid growth of documents across the World Wide Web (WWW). Text classification is the task of assigning a document to a predefined category [1]. In this research, several machine learning algorithms are applied to multiple datasets to determine the accuracy of the classifiers. The machine learning algorithms explored include Naïve Bayes techniques such as Multinomial Naïve Bayes and Bernoulli Naïve Bayes; linear classifiers including Logistic Regression and Stochastic Gradient Descent; Support Vector Machine models such as Support Vector Clustering (SVC) and Linear SVC; and Artificial Neural Network models such as the Back Propagation Network. These algorithms are evaluated on experimental datasets such as the Reuters Corpus [13], the Brown Corpus [14], and the Movie Review Corpus [15].

Formally, let $D = \{d_1, d_2, \dots, d_n\}$ be a set of documents, and let $C = \{c_1, c_2, \dots, c_m\}$ be the set of classes. The text classification process aims to assign each document d_i to the appropriate class c_j [1]. This research addresses hard categorization of text, where every document must be placed into exactly one of the specific classes.

The text classification process generally involves the following steps [1]:

1. Reading documents
2. Tokenizing texts
3. Stemming
4. Stop word deletion

5. Vector representation of text
6. Feature selection
7. Applying supervised learning algorithms
8. Measuring accuracy

Different metrics can be used to measure classification performance; in this research, the F1 score is employed as the primary evaluation metric [1]. By comparing the classification methods, a voted classifier is ultimately constructed based on the documents and their features.

2 Literature Review

There are several works in the recent past on text classification. Li et al. [2] proposed a text classification technique using positive and unlabeled data. This paper defined two classes: one labeled as positive and another as unlabeled. The unlabeled data contained both positive and negative examples. The task was to identify the labeled data within the unlabeled class. The key finding was that only one class needed explicit labels, while the other did not.

Aggarwal et al. [3] surveyed various approaches to text categorization. This work discussed multiple aspects of automatic text categorization and compared techniques along with their benefits and shortcomings for different applications.

Tong et al. [4] applied support vector machines (SVMs) to text categorization. Their study demonstrated that many relevant features could be discovered when applying SVMs. In contrast, our work employs an extensive package of SVM classification techniques optimized for text classification, with a voting process to ensure the best results.

Liu et al. [5] discussed classifying text by labeling words instead of documents. While this method can be effective, it requires documents to be enriched with many relevant words for each class. In our work, we instead use Named Entity Recognition (NER) in Natural Language Processing (NLP) to simplify identifying relevant features for document labeling.

Schütze [6] estimated the conditional probability of a word given a class as its relative frequency in documents belonging to that class. However, Bernoulli Naïve Bayes can be inefficient for long documents, as it does not account for multiple occurrences of words.

Tang et al. [7] employed a Bayesian approach for automated text categorization, using information gain (IG) and maximum discrimination (MD) for feature selection. In our work, we use the document frequency metric for feature selection and apply Naïve Bayes to compare results with previous works [1].

3 Methodologies

Document frequency (DF) is used for feature selection in text classification [1]. It is defined as

$$DF(t_k) = P(t_k), \quad (1)$$

where $DF(t_k)$ is the document frequency of term t_k and $P(t_k)$ is its probability of occurrence. The classifier models used for text classification are described below.

3.1 Naïve Bayes Model

The Naïve Bayes model includes Multinomial Naïve Bayes and Bernoulli Naïve Bayes [8].

3.1.1 Multinomial Naïve Bayes

Multinomial Naïve Bayes models the count of terms in documents [8]. This variation accounts for the number of occurrences of a term t in training documents from class C , including multiple occurrences.

Fig. 1 Multinomial Naïve Bayes Training

```

1:  $V \leftarrow \text{EXTRACTVOCABULARY}(D)$ 
2:  $N \leftarrow \text{COUNTDOCS}(D)$ 
3: for each  $c \in C$  do
4:    $N_c \leftarrow \text{COUNTDOCSINCLASS}(D, c)$ 
5:    $\text{prior}[c] \leftarrow N_c/N$ 
6:    $\text{text}_c \leftarrow \text{CONCATTEXTOFALLDOCSINCLASS}(D, c)$ 
7:   for each  $t \in V$  do
8:      $T_{ct} \leftarrow \text{COUNTTOKENSOFTERM}(\text{text}_c, t)$ 
9:   for each  $t \in V$  do
10:     $\text{condprob}[t][c] \leftarrow \frac{T_{ct} + 1}{\sum_{t'} (T_{ct'} + 1)}$ 
11: return  $V, \text{prior}, \text{condprob}$ 

```

Fig. 2 Multinomial Naïve Bayes Application

```

1:  $W \leftarrow \text{EXTRACTTOKENSFROMDOC}(V, d)$ 
2: for each  $c \in C$  do
3:    $\text{score}[c] \leftarrow \log \text{prior}[c]$ 
4:   for each  $t \in W$  do
5:      $\text{score}[c] \leftarrow \text{score}[c] + \log \text{condprob}[t][c]$ 
6: return  $\arg \max_{c \in C} \text{score}[c]$ 

```

3.1.2 Bernoulli Naïve Bayes

This variant generates each token from the vocabulary and assigns 1 if the token appears in the document and 0 otherwise [8].

Fig. 3 Bernoulli Naïve Bayes Training

```

1:  $V \leftarrow \text{EXTRACTVOCABULARY}(D)$ 
2:  $N \leftarrow \text{COUNTDOCS}(D)$ 
3: for each  $c \in C$  do
4:    $N_c \leftarrow \text{COUNTDOCSINCLASS}(D, c)$ 
5:    $\text{prior}[c] \leftarrow N_c/N$ 
6:   for each  $t \in V$  do
7:      $N_{ct} \leftarrow \text{COUNTDOCSINCLASSCONTAININGTERM}(D, c, t)$ 
8:      $\text{condprob}[t][c] \leftarrow \frac{N_{ct} + 1}{N_c + 2}$ 
9: return  $V, \text{prior}, \text{condprob}$ 

```

Fig. 4 Bernoulli Naïve Bayes Application

```

1:  $V_d \leftarrow \text{EXTRACTTERMSFROMDOC}(V, d)$ 
2: for each  $c \in C$  do
3:    $\text{score}[c] \leftarrow \log \text{prior}[c]$ 
4:   for each  $t \in V$  do
5:     if  $t \in V_d$  then
6:        $\text{score}[c] \leftarrow \text{score}[c] + \log \text{condprob}[t][c]$ 
7:     else
8:        $\text{score}[c] \leftarrow \text{score}[c] + \log(1 - \text{condprob}[t][c])$ 
9: return  $\arg \max_{c \in C} \text{score}[c]$ 

```

3.2 Linear Classifier Model

The linear classifier model is based on gradient-descent methods, including Logistic Regression and Stochastic Gradient Descent (SGD).

3.2.1 Logistic Regression

The probability of assigning document $d^{(i)}$ to class $c = j$ is [9]

$$P(c^{(i)} = j \mid d^{(i)}; \theta) = \frac{\exp(\theta_j^\top d^{(i)})}{\sum_{l=1}^k \exp(\theta_l^\top d^{(i)})}, \quad (2)$$

where $\theta_1, \dots, \theta_k \in \mathbb{R}^{n+1}$ are the parameters. The hypothesis vector is

$$h_\theta(d^{(i)}) = \begin{bmatrix} P(c^{(i)}=1 \mid d^{(i)}; \theta) \\ \vdots \\ P(c^{(i)}=k \mid d^{(i)}; \theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^k \exp(\theta_j^\top d^{(i)})} \begin{bmatrix} \exp(\theta_1^\top d^{(i)}) \\ \vdots \\ \exp(\theta_k^\top d^{(i)}) \end{bmatrix}. \quad (3)$$

The (multiclass) cross-entropy cost is

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m \sum_{j=1}^k \mathbf{1}\{c^{(i)} = j\} \log \frac{\exp(\theta_j^\top d^{(i)})}{\sum_{l=1}^k \exp(\theta_l^\top d^{(i)})}, \quad (4)$$

with gradient

$$\nabla_{\theta_j} J(\theta) = -\frac{1}{m} \sum_{i=1}^m [d^{(i)} (\mathbf{1}\{c^{(i)} = j\} - P(c^{(i)} = j \mid d^{(i)}; \theta))]. \quad (5)$$

3.2.2 Stochastic Gradient Descent

For a training pair $(d^{(i)}, c^{(i)})$, a simple squared-error cost is [10]

$$\text{cost}(\theta; (d^{(i)}, c^{(i)})) = \frac{1}{2} (h_\theta(d^{(i)}) - c^{(i)})^2, \quad (6)$$

and the empirical objective is

$$J_{\text{train}}(\theta) = \frac{1}{2m} \sum_{i=1}^m \text{cost}(\theta; (d^{(i)}, c^{(i)})). \quad (7)$$

Fig. 5 Stochastic Gradient Descent (one epoch)

| | |
|--|---------------------|
| 1: Randomly shuffle the dataset | |
| 2: for $i = 1$ to m do | ▷ training examples |
| 3: for $j = 0$ to n do | ▷ features |
| 4: $\theta_j \leftarrow \theta_j - \alpha (h_\theta(d^{(i)}) - c^{(i)}) d_j^{(i)}$ | |

Here, α is the learning rate (often decayed over iterations) to promote convergence of θ .

3.3 Support Vector Machine Model

The Support Vector Machine (SVM) model includes Support Vector Clustering (SVC) and Linear Support Vector Clustering (Linear SVC) [17]. The algorithms are applied using an optimized SVM library, specifying parameters and kernel functions to measure similarity between classes and documents.

3.3.1 Support Vector Clustering (SVC)

In SVC [17], the Gaussian kernel is defined as

$$f_G = -\frac{\|D - l^{(i)}\|}{\sigma^2}, \quad (8)$$

where D is the set of documents, $l^{(i)}$ is a landmark equal to $d^{(i)}$, and σ is the kernel bandwidth. Applying this kernel is particularly useful when the number of features n is small and the number of training examples m is large.

3.3.2 Linear Support Vector Clustering (Linear SVC)

To predict whether a document d belongs to class c , Linear SVC evaluates the decision boundary

$$\theta^\top d \geq 0. \quad (9)$$

If the inequality holds, then d is assigned to class c ; otherwise, it is not. Applying a linear kernel is especially effective when the number of features n is high and the number of training examples m is small.

3.4 Artificial Neural Network Model

The Artificial Neural Network (ANN) model includes only the Backpropagation Network. The documents are given as inputs to the input layer. There are two hidden layers that adapt the weights and thresholds. The outputs correspond to the documents associated with their proper classes.

The advantage of this model is that the number of documents and classes can vary, which allows adjusting the number of input and output nodes. Sigmoid and Gaussian functions are used as transfer functions in the Backpropagation Network.

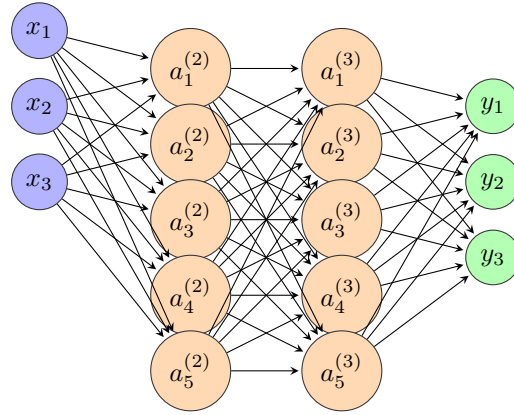


Figure 1: Fully connected feedforward artificial neural network with two hidden layers. The input layer has 3 neurons (x_1, x_2, x_3), each connected to 5 neurons in the first hidden layer ($a_1^{(2)}, \dots, a_5^{(2)}$), which in turn connect to 5 neurons in the second hidden layer ($a_1^{(3)}, \dots, a_5^{(3)}$), and finally to 3 output neurons (y_1, y_2, y_3). Every neuron in each layer connects to every neuron in the next, illustrating the dense topology of a multilayer perceptron.

3.4.1 Backpropagation Network

The Backpropagation algorithm is described as follows [18]:

Fig. 6 Backpropagation Algorithm

-
- 1: Initialize the training set:

$$\{(d^{(1)}, c^{(1)}), (d^{(2)}, c^{(2)}), \dots, (d^{(m)}, c^{(m)})\}$$
 - 2: Set $\Delta_{ij}^{(l)} := 0$ for all layers l , nodes i and j .
 - 3: **for** $i \leftarrow 1$ to m **do**
 - 4: Set $a^{(1)} = d^{(i)}$.
 - 5: Perform forward propagation to compute $a^{(l)}$ for $l = 1, 2, \dots, L$.
 - 6: Using $c^{(i)}$, compute $\delta^L = a^L - c^{(i)}$.
 - 7: Compute $\delta^{(L-1)}, \delta^{(L-2)}, \dots, \delta^{(2)}$.
 - 8: Update: $\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$.
-

Here, m is the number of training examples, L is the total number of layers, $\delta_j^{(l)}$ is the error of node j in layer l , and $a_j^{(l)}$ is the activation value of node j in layer l .

The weight update term is defined as:

$$D_{ij}^{(l)} := \begin{cases} \frac{1}{m} \Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)}, & j \neq 0, \\ \frac{1}{m} \Delta_{ij}^{(l)}, & j = 0. \end{cases}$$

The cost function is given by:

$$J(\Theta) = -\frac{1}{m} \left[\sum_{i=1}^m \left(c^{(i)} \log(h_{\Theta}(d^{(i)})) + (1 - c^{(i)}) \log(1 - h_{\Theta}(d^{(i)})) \right) \right] + \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{S_l} \sum_{j=1}^{S_{l+1}} (\Theta_{ji}^{(l)})^2 \quad (10)$$

The gradient of the cost function is:

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)} \quad (11)$$

4 Results

The metric of accuracy depends on four features [1]:

- **True positives (TP)**: relevant documents correctly identified as the proper class.
- **True negatives (TN)**: irrelevant documents correctly identified as an improper class.
- **False positives (FP)**: irrelevant documents incorrectly identified as proper class (type I error).
- **False negatives (FN)**: relevant documents incorrectly identified as improper class (type II error).

From these, two parameters are used to calculate the F1 score.

Precision

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (12)$$

Recall

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (13)$$

F1 Score

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (14)$$

Using the F1 score metric, the accuracy percentage of text classifiers on different datasets is summarized in Table 1.

Table 1: Accuracy (%) of Text Classification by F1 Score

| Classifier Model | Reuters Corpus [13] | Brown Corpus [14] | Movie Review Corpus [15] |
|--|---------------------|-------------------|--------------------------|
| Naïve Bayes Model | | | |
| Multinomial Naïve Bayes | 72.0 | 72.5 | 76.0 |
| Bernoulli Naïve Bayes | 75.0 | 78.0 | 79.0 |
| Linear Classifier Model | | | |
| Logistic Regression | 73.5 | 79.5 | 74.5 |
| Stochastic Gradient Descent | 76.0 | 83.5 | 81.5 |
| SVM Model | | | |
| SVC | 78.0 | 78.0 | 79.5 |
| Linear SVC | 83.0 | 77.0 | 80.5 |
| Artificial Neural Network Model | | | |
| Back Propagation Network | 89.0 | 93.0 | 94.5 |
| Voted Classifier | 89.0 | 93.0 | 94.5 |

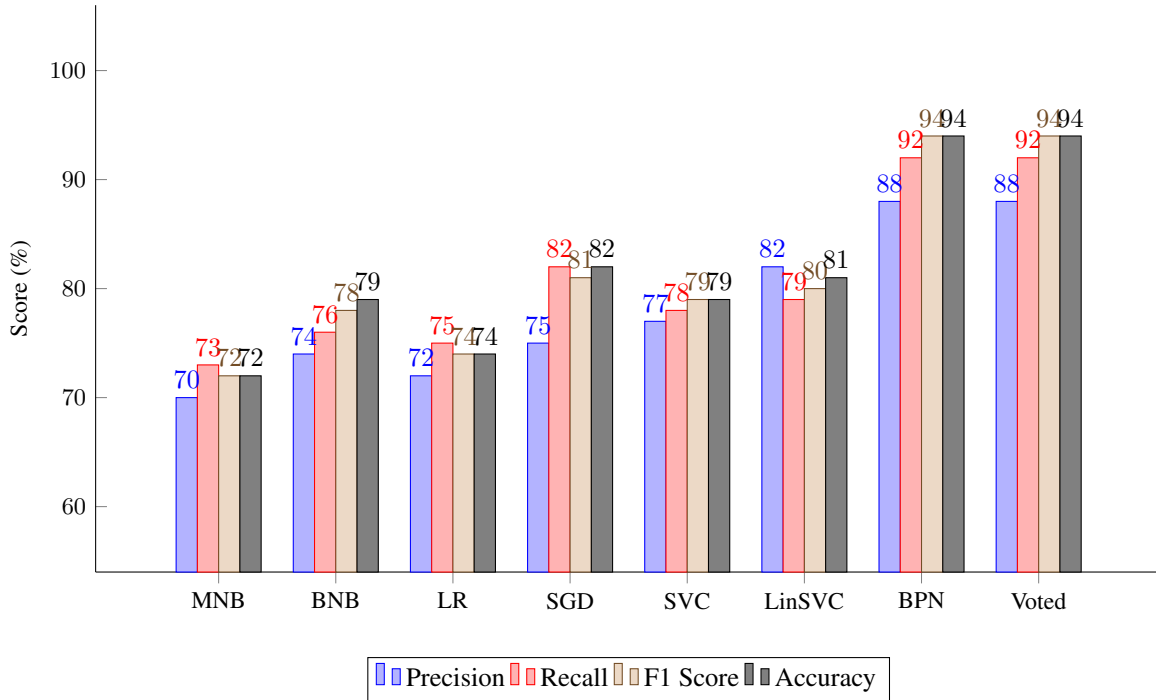


Figure 2: Performance comparison of classifiers on text classification tasks (Precision, Recall, F1 Score, and Accuracy).

5 Conclusion

This research aimed to identify the best classification method and to select the voted classifier, i.e., the classifier that achieves the highest accuracy. It also demonstrates that the performance of text document classification depends not only on the classification method but also on how well the associated corpus is organized.

Each corpus dataset was divided into three categories: 60% of the data was used for training, 20% for cross-validation, and the remaining 20% for testing. From the results, it can be observed that classification performance may vary slightly across time and datasets, but the accuracy approximation remains consistent.

It was also found that the Artificial Neural Network (ANN) achieved higher accuracy compared to the other classifier models. This is because, until the classification process becomes satisfactory, the ANN continues to iterate and optimize. Although it requires more iterations and execution time, it provides the most accurate classification of text documents.

This research combines methods from Machine Learning and fundamental Natural Language Processing techniques. As text documents continue to grow rapidly worldwide, processing and categorizing them into proper classes will remain a major concern in future research.

Acknowledgment

The authors would like to thank Biprodip Pal, Assistant Professor, Department of Computer Science and Engineering, Rajshahi University of Engineering and Technology, for his support and enthusiasm. The authors are also grateful to Professor Andrew Ng of Stanford University and his online course on Machine Learning in Coursera [16].

References

- [1] M. Ikonomakis, S. Kotsiantis, and V. Tampakas, "Text classification using machine learning techniques," *WSEAS Trans. Computers*, vol. 4, no. 8, pp. 966–974, 2005.
- [2] X. Li and B. Liu, "Learning to classify texts using positive and unlabeled data," in *Proc. IJCAI*, vol. 3, 2003.
- [3] C. C. Aggarwal and C. Zhai, "A survey of text classification algorithms," in *Mining Text Data*, Springer US, 2012, pp. 163–222.
- [4] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *J. Mach. Learn. Res.*, vol. 2, Nov. 2001, pp. 45–66.
- [5] B. Liu, W. S. Lee, P. S. Yu, and X. Li, "Text classification by labeling words," in *Proc. AAAI*, vol. 4, 2004.
- [6] H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to Information Retrieval*. Cambridge University Press, 2008.
- [7] B. Tang, S. Kay, and H. He, "A Bayesian classification approach using class-specific features for text categorization," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 6, pp. 1602–1606, 2016.
- [8] A. McCallum and K. Nigam, "A comparison of event models for naive Bayes text classification," in *AAAI-98 Workshop on Learning for Text Categorization*, vol. 752, 1998.
- [9] A. Genkin, D. D. Lewis, and D. Madigan, "Large-scale Bayesian logistic regression for text categorization," *Technometrics*, vol. 49, no. 3, pp. 291–304, 2007.
- [10] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. COMPSTAT 2010*, Physica-Verlag HD, 2010, pp. 177–186.
- [11] T. Finley and T. Joachims, "Supervised clustering with support vector machines," in *Proc. ICML*, ACM, 2005, pp. 217–224.
- [12] A. J. C. Trappey *et al.*, "Development of a patent document classification and search platform using a back-propagation network," *Expert Syst. Appl.*, vol. 31, no. 4, pp. 755–765, 2006.
- [13] Reuters Corpus, <http://about.reuters.com/researchandstandards/corpus/>, accessed 2025-08-31.
- [14] Brown Corpus, http://www.essex.ac.uk/linguistics/external/clmt/w3c/corpus_ling/content/corpora/list/private/brown/brown.html, accessed 2025-08-31.
- [15] Cornell Movie Review Data, <https://www.cs.cornell.edu/people/pabo/movie-review-data/>, accessed 2025-08-31.
- [16] Coursera Machine Learning Course, <https://www.coursera.org/learn/machine-learning/>, accessed 2025-08-31.
- [17] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.