Algorithms for Text Categorization: A Comparative Study

S. Ramasundaram and S.P. Victor

Department of Computer Science, Madurai Kamaraj University College, Madurai - 625 002, India
Department of Computer Science St.Xavier College, Tirunelveli - 627 002, India

Abstract: Text Categorization’s significance is on a continuous acceleration due to the present mammoth escalation in textual data thrusting the importance of analysing and examining the methods for handling textual data. This paper discusses and compares six algorithms for Text categorization, such as Naïve Bayes, Support Vector Machine, N-Grams, K-Nearest Neighbourhood, Back Propagation Network and Genetic Algorithm. Furthermore, the performances of all these algorithms are compared and tabulated.

Key words: Inductive Learning algorithms · Neural network · Text classifiers · Features · Irrelevant Features · Population · Chromosomes · Crossover · Mutation

INTRODUCTION

Text Categorization (TC) focuses on automatically assigning each document to a predefined category like Politics, Sports, Religion, Entertainment and etc [1]. This process of assigning becomes acutely pertinent and significant due to the colossal texts available on the Internet. Texts are in the form of online journals, mail messages, written contracts, internal memos, facsimiles, online magazines and etc. These texts have to be managed and classified accurately due to the necessity of preserving and transferring texts and only an effective text categorization algorithm can afford this. Hence such an algorithm has to be identified. This paper is produced with this intent.

Text Categorization can be stated like a set \( x = (x_1, x_2, \ldots, x_n) \) where \( x_i \) is the \( i \)th document to be categorized and let set \( c = (c_1, c_2, \ldots, c_m) \) where \( c_i \) is the predefined category to which text document \( x_i \) will be mapped for a function \( f \). Here \( n \) refers to the total number of documents that has to be categorized and \( m \) denotes the total number of predefined categories. It is represented like this:

\[
f: x_i \rightarrow c_j
\]

Now the function \( f \) has to map the document in the set \( x \) to the appropriate category specified in the set \( c \). Mathematically, TC can be represented like this. All the algorithms that deal with TC try to implement this mapping process of documents to the predefined category.

The algorithms considered here are Naïve Bayes, Support Vector Machine, N-Grams, K-Nearest Neighbourhood, Back Propagation Network and Genetic Algorithm. These algorithms are chosen in view that they represent different categories like Induction Learning, Statistical and Neural Networks. These algorithms are implemented and compared on the basis of CPU time and precision. All these results are tabulated and provided at the end of this article; furthermore, to support the table and to give a better understanding, graphical charts are also provided.

Definitions

Classifier: A classifier is a function that maps an input attribute vector, \( x = (x_1, x_2, \ldots, x_n) \), to a confidence that the input belongs to a class - that is, \( f(x) = \text{confidence} \text{(class)} \).

In the case of text classification, the attributes are words in the document and the classes correspond to text categories (e.g. typical categories include acquisitions, earnings, interest).

Examples of classifiers for the category interest include:

- If (interest AND rate) OR (quarterly), then confidence(interest category) = 0.9
Confidence(interest category) = 0.3*interest + 0.4*rate + 0.7*quarterly

Some of the classifiers that are now considered are probabilistic in the sense that confidence (class) is a probability distribution.

Feature Selection: Every text document contains a collection of words formed in an order. Among all these words only a fraction of the words reflect the prime concept or the topic. For example in a document which concentrates on sports not all the words are related to sports. Instead only a few words in that document will be related to sports, therefore, irrelevant words should be eliminated and the related words should be retained. The unrelated words are called as irrelevant features and the related words are called as relevant features.

A document without irrelevant features will improve the accuracy and efficiency in TC process. This process of feature selection also involves removal of stop words and finding the stem words[2]. Stop words are the words which have very low discriminatory value (e.g. “the”, “a”, “and”, “that”). Hence these words are removed. As far as the stem words are concerned a word which may exist in different morphology should be converted to single canonical form. For example the word “see” may also exist in other morphological variants such as “seeing”, “seen”, “saw”. Here “see” will be the canonical form. Likewise conversion of the words to the canonical form is carried out for the entire document. This process is also called as stemming. Only after this pre-processing steps of stop words removal and stemming the relevant features in the target documents are identified.

Categorization Methods: In this section the details of Naïve Bayes, Support Vector Machine, N-Grams, K-Nearest Neighbourhood, Back Propagation Network and Genetic Algorithms will be discussed.

Naive Bayes Algorithm: In a simple manner it can be stated that a Naive Bayes (hereafter referred as NB) classification assumes that the presence of a particular feature of a class is unrelated to the presence of any other feature, given class variable, to cite an instance, a moving object can be considered a bicycle if it has two wheels, two pedals and a dynamo light. Even if these features depend on each other or upon the existence of the other features, a NB classifier considers all of these properties to independently contribute to the probability that this object is a bicycle.

A Naïve-Bayes classifier is constructed by using the training data to estimate the probability of each category, given the document feature values of a new instance[3]. We use Bayes theorem to estimate the probabilities:

\[ P(C = C_k | x^x) = \frac{P(x^x | C = C_k)P(C = C_k)}{P(x^x)} \]

The quantity \( P(x^x | C = C_k) \) is often impractical to compute without simplifying assumptions. For the Naïve Bayes classifier [4], it is assumed that the features \( X_1,...X_n \) are conditionally independent, given the category variable \( C \). This simplifies the computations yielding:

\[ P(x^x | C = C_k) = \prod_i P(x_i | C = C_k) \]

In theory, Bayesian classifiers have the minimum error rate in comparison to all other classifiers [5]. But in practical situations it is not the actual case due to the faulty assumptions on class conditional independence and the lack of available probability data. Moreover, empirical studies of this classifier in comparison to neural network classifiers have been found to be comparable in some domain.

Support Vector Machine Algorithm: Support Vector Machine (SVM) is a concept in statistics and computer science for a set of supervised learning methods that analyze data and recognize patterns, used for classification and regression analysis. The standard SVM takes a set of input data and predicts, for each given input, which of two possible classes forms the input, making the SVM a non-probabilistic binary linear classifier[6]. Given a set of training examples, each marked as belonging to one of the two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other. An SVM model is a representation of the examples as points in space, mapped so that the examples of separate categories are divided by a clear gap that is as wide as possible [7].

Vapnik proposed Support Vector Machines [8], but they have only recently been gaining popularity in the learning community. In its simplest linear form, an SVM is a hyperplane that separates a set of positive examples from a set of negative examples with maximum margin.

The formula for the output of a linear SVM is \( y = w*x - b \), where is the normal vector to the hyperplane and is the input vector [9]. In the linear case, the margin is
defined by the distance of the hyperplane to the nearest of the positive and negative examples. Maximizing the margin can be expressed as an optimization problem: minimize \( \frac{1}{2} \| \mathbf{w} \|^2 \) subject to \( y_i ( \mathbf{w}^T \mathbf{x}_i + b) \geq 1 \), where \( x_i \) is the \( i \)th training example and \( y_i \) is the correct output of the SVM for the \( i \)th training example. It should be noted that all the problems are linearly separable. Some examples fall on the wrong side of the boundary. SVMs have shown good results in handwritten character recognition [10], face detection [11] and most recently text categorization comparison initially consists in weighting each term (N-grams) in each profile (categories and new document), then calculating the distance between the profile of the document and profile of each category. The document will be assigned to the category to which its profile is closest.

The \( \chi^2 \) statistic measures the degree of association between a term and the category. For the purpose of dimensionality reduction, terms with small values are discarded. The principle consists in extracting \( K \) better features characterizing best category compared to others for each category.

Classification Phase: Next the classification phase starts. In this step the profile of the document to be categorized is matched with the profiles of the categories that were already calculated in the training phase. This comparison initially consists in weighting each term (N-grams) in each profile (categories and new document), then calculating the distance between the profile of the document and profile of each category. The document will be assigned to the category to which its profile is closest.

The weights are assigned to each term to represent the importance of the term (N-grams) in the category concerned. Here the standard \( \text{tfidf} \) function is used. It is defined as:

\[
\text{tfidf}(t_k, c_i) = \text{tf}(t_k, c_i) \times \log \left( \frac{|C|}{df(t_k)} \right)
\]

where:

- \( \text{tf}(t_c, c_i) \) denotes the number of times feature \( t_c \) occurs in category \( c_i \);
- \( df(t_k) \) denotes the number of categories in which feature \( t_k \) occurs.
- \( |C| \) denotes the number of categories.

After having weighted the terms (N-grams), it is necessary to calculate the distance between the categories profiles and the profile of the document to be categorized. For this purpose Cosine measurement and Kullback & Liebler are used to study the influence of the similarity measure in the performance of categorization.

- bi-grams: _T,TE,EX,XT,T_
- tri-grams: _TE,TEX,EXT,XT,T_
- quad-grams: _TEX,TEXT,EXT,XT,T_

N Grams Algorithm: An N-gram is a sequence of N consecutive characters [13]. In a text, all the n-grams present are located and then their frequencies are counted. Space character is replaced by character "_" to facilitate detection. This technique is purely technical and does not require any knowledge of document language. Another advantage of the N-grams is the automatic capture of the most frequent roots. For example the common root of flourish, flourished, flourishing...etc is found by this technique. This technique does not need to eliminate the stop words or to proceed to stemming.

Generally N-gram term can include the notion of any co-occurring set characters in a string (e.g. an N-gram made up of the first and third character of a word), in this paper we use the term contiguous slices only. Typically one slices the string into a set of overlapping N-grams. In our system, we use N-grams of several different lengths simultaneously. We also append blanks to the beginning and ending of the string in order to help with matching beginning-of-word and ending-of-word situations. Thus, the word “TEXT” would be composed of the following N-grams:
Geometrically, the cosine similarity evaluates the cosine of the angle between two vectors \(d_i\) and \(d_j\) and is based on the angular distance. This allows us to abstract from varying vector length. The cosine similarity can be calculated as the normalized:

\[
d_{ij} = \frac{\sum_{w \in i \cap j} \text{TFIDF}_{w,i} \times \text{TFIDF}_{w,j}}{\sqrt{\sum_{w \in i} \text{TFIDF}^2_{w,i} \times \sum_{w \in j} \text{TFIDF}^2_{w,j}}}
\]

with:

- \(w\): a feature, \(I\) and \(J\): the two vectors (profiles) to be compared.
- \(\text{TFIDF}_{w,I}\): the weight of the term \(w\) in \(I\) and \(\text{TFIDF}_{w,j}\): the weight of the term \(w\) in \(J\). This can also be put as follows:

"When the common features are more, these features have strong weightings and the similarity will be close to 1 and vice versa."

Discussing about the Kullback & Liebler measurement, it is a statistical measurement of information called function of discrimination by taking into account two probability distributions. Kullback & Liebler distance is defined as follows:

\[
D(P, Q) = \sum_{x \in X} ((P(x) - Q(x)) \times \log(P(x)/Q(x)))
\]

\(P\) and \(Q\) are the two probabilities on a finished set \(X\).

N-gram-based approach to text categorization is tolerant of textual errors. Also the system is small, fast and robust.

**K-Nearest Neighbor and Simulated Annealing:**
K-Nearest Neighbor (KNN) algorithm is a very popular for TC. K-nearest neighbor algorithm assumes all instances correspond to points in the \(n\)-dimensional space \(\mathbb{R}^n\) [14]. The nearest neighbors of an instance are defined in terms of the standard Euclidean or Cosine distance. More precisely, let an arbitrary instance \(x\) be described by the feature vector

\((a_1(x), a_2(x), \ldots, a_r(x))\)

where \(a_r(x)\) denotes the value of the \(r\)th attribute of instance \(x\). Then the distance between two instances \(x_i\) and \(x_j\) is defined to be \(d(x_i, x_j)\), where

\[
d(x_i, x_j) = \frac{\sum_{t=1}^{n} x_{it} \times x_{jt}}{\sqrt{\sum_{t=1}^{n} x_{it}^2 \times \sum_{t=1}^{n} x_{jt}^2}}
\]

Classification algorithm is defined as: Given a query instance \(x_q\) to be classified, let \(x_1, \ldots, x_k\) denotes the \(k\) instances from training that are nearest to \(x_q\).

\[
f(x_q) \leftarrow \arg\max_{i=1}^{k} y(x_j, x_k)
\]

where \(x_q\) is a test instance, \(x_j\) is one of the neighbours in the training set, \(y(x_j, c_k) \in \{0, 1\}\) indicates whether \(x_j\) belongs to class \(c_k\). Equation (2) means that the prediction will be the class that has the largest number of members in the \(k\) nearest neighbours.

The aim of \(k\)-nearest neighbors is to get the \(k\) highest values of all distances between the test instance and the other training instances. We can also define it more specifically and concretely.

\[
Max_k d(x_i, x_j) = \frac{\sum_{t=1}^{n} x_{it} \times x_{jt}}{\sqrt{\sum_{t=1}^{n} x_{it}^2 \times \sum_{t=1}^{n} x_{jt}^2}}
\]

It can be changed into:

\[
Min_k d(x_i, x_j) = \frac{-\sum_{t=1}^{n} x_{it} \times x_{jt}}{\sqrt{\sum_{t=1}^{n} x_{it}^2 \times \sum_{t=1}^{n} x_{jt}^2}}
\]

From the above formula (4), we can see that in fact, this is a problem of combinatorial optimization and what we have to compute instantaneously is \(\sum_{t=1}^{n} x_{it} \times x_{jt}\).

Simulated annealing is a solution method in the field of combinatorial optimization based on analogy with the physical process of annealing, so we can borrow the idea of simulated annealing to improve the computational efficiency.
Simulated Annealing: Simulated Annealing (SA) is a random-search technique to find a good solution to an optimization problem by trying random variations of the current solution. The concept is based on the manner in which liquids freeze or metals recrystallize in the process of annealing. The fundamental idea is to allow moves resulting in solutions of worse quality than the current solution (uphill moves) in order to escape from local minima. The probability of doing such a move is decreased during the search. SA forms the basis of an optimization technique for combinatorial and other problems.

One of the drawbacks of KNN algorithm is its efficiency. KNN is a lazy categorization and instead of estimating the target function once for the entire instance, they delay processing until a new instance must be classified and it needs to compare a test instance or document with all samples in the training set. In addition, the performance of this algorithm greatly depends on two factors, that is, a suitable similarity function and an appropriate value for the parameter k.

Backpropagation Network: Among the various algorithms provided by Artificial Neural Network System (ANS), Backpropagation is a very popular algorithm. Backpropagation as an ANS is very useful in recognizing complex patterns and performing nontrivial mapping functions. Following figure represents a simple Backpropagation diagram. The rounded objects represent the neurons or processing elements of a neural network. The directed lines that are connecting the neurons are called weights. Also every line of processing elements is a layer of a network. Thus in this figure there are three layers available. Though there can be multiple layers present in ANS generally there will be three layers present in the Backpropagation network (BPN).

The next aspect which deserves interest before understanding the Backpropagation algorithm is the network topology of the text classifier.

As shown in the figure, the neural network is a three layer fully connected feed-forward network which consists of an input layer, a hidden layer and an output layer. All neurons in the neural network are non-linear units with sigmoid function as the activation function. In the input layer, the number of input units (r) is equal to the dimensionality of the reduced feature space. In the output layer, the number of output units (m) is equal to the number of pre-defined categories in the particular text categorization task. The number of hidden units in the neural network affects the generalization performance.

The choice depends on the size of the training set and the complexity of the classification task, the network is trying to learn and can be found empirically based on the categorization performance.

For classification of the documents, reduced feature vectors representing the documents are fed to the input layer of the neural network classifier as input signal. These input signals are then propagated forward through the neural network so that the output of the neural network is computed in the output layer. As the sigmoid function is used as the activation function in the output units, the output of the neural network classifier is a real-valued classification vector with the component values in the range [0, 1]. This real-valued classification vector represents a graded classification decision, in which the ith vector component indicates the probability that the input document belongs to the ith category. If binary classification is desired, a threshold can be set such that a document is considered to be belonging to the ith category only if the ith component of the classification vector greater than the threshold.

There are certain basic assumptions incorporated into this algorithm. First, the output function on all hidden and output layer units is assumed to be the sigmoid function. Moreover we have included the momentum term in the weight update calculations.

The BPN algorithm is produced in two phases here. In the first phase forward signal propagation occurs in the network. In the second phase the error terms are then fed back to all other input units. In this case they are feature vectors.
Genetic Algorithm: Genetic Algorithms (GA’s) are stochastic search methods inspired to the biological evolution [15]. Their capability to provide good solutions for classical optimization tasks has been demonstrated by various applications, including TSP and Knapsack. Genetic algorithms (GAs) are a set of artificial intelligence search algorithms based on evolutionary theory. They represent the solution to a problem as an individual (also called chromosome) in a population pool. They evolve the population of individuals (chromosomes) generation by generation following the genetic transformation operations — such as reproduction, crossover and mutation — with the aim of discovering chromosomes with better fitness values. A fitness function is available to assign the fitness value for each individual.

A Genetic Algorithm Involves the Following Steps:

- Generate a random population of n chromosomes.
- Evaluate the fitness function \( f(x) \) of each chromosome \( x \) in the population with WTSD fitness function.

Create a new population by repeating following steps until the new population is complete:

- Select two parent chromosomes from a population according to their fitness.
- With a crossover probability crossover the parents to form new offspring.
- With a mutation probability mutate new offspring at each locus.
- Place new offspring in the new population for further run of the algorithm.
- Use new generated population for a further run of an algorithm.
- If the end condition is satisfied, stop and return the best solution in the current population, otherwise go to step 2.

Certain terms used in the GA require some explanation. The term chromosomes refer the set of words with their weights that have to be classified. The fitness function is a function created to evaluate the appropriateness of the solution at the end of each iteration. Here, weighted topic standard deviation is taken as the fitness function. Crossover is the process for selecting the new chromosomes. Mutation is a genetic operator used for changing the gene values in a chromosome. If the chromosomes are words then genes are letters.

Experimental Result: All the above mentioned algorithms were implemented in a Pentium IV system which had a clock speed of 2.4GHz. The software environment in which these algorithms were tested is as follows: Windows XP Professional is the operating system used. VB.NET and SQL Server2005 were the front-end and Back-end respectively.

For the comparison of the algorithms two aspects were involved. One is the execution time of the algorithms and another one is the precision value of the algorithm. Precision is calculated based on the following formula:

- Precision = Categories-matched-correctly/total-categories found

Based on these two aspects the performance of these algorithms were compared for the twenty document files of various English dailies published in India. The categories for TC include Politics, Movies, Sports and Education.

Table 1 represents the comparative study of various algorithms on the basis of the execution time. It can also be seen that maximum execution time for a document is recorded in the KNN Algorithm. Also it can be noted that N-Grams algorithm has a minimum execution for many documents. The Naive Bayes and SVM produces very close performances compared to GA and BPN. At the same time GA and BPN follow N-Grams very closely. This is very closely represented in Chart 1 also.

The comparison of algorithms on the basis of Precision is depicted in Table 2 and Chart 2. It can be seen that Table 1 and Table 2 matches in many cases. It indicates that precision values also represent the same type of results. Here also N-Grams has better precision in many cases. It is closely followed by GA and BPN. In a few cases SVM too has a better precision. The better results provided by NGrams can be attributed to the elimination of preprocessing steps like stemming reduces execution time. Regarding the GA and BPN the role of fitness function and training data should be fine tuned to achieve better results. The KNN suffers due to similarity function and finding the parameter K value. In SVM the problem of linear separability becomes a major constraint. Availability of probability data affects the working of Naive Bayes algorithm.
Chart 1: Comparison of execution time of various algorithms X-axis: Documents, Y-axis: Execution Time

Chart 2: Comparison of execution time of various algorithms X-axis: Documents, Y-axis: Precision Value

Table 1: Execution time for the various algorithms are represented in seconds

<table>
<thead>
<tr>
<th>Documents</th>
<th>T. Words</th>
<th>After stopwords removal</th>
<th>E. time in Naïve Bayes</th>
<th>E. time in SVM</th>
<th>E. Time in Ngrams</th>
<th>E. Time in KNN</th>
<th>E. Time in GA</th>
<th>E. Time in BPN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document 1</td>
<td>77</td>
<td>40</td>
<td>0.8896</td>
<td>0.8892</td>
<td>0.6288</td>
<td>4.5624</td>
<td>0.9764</td>
<td>0.848</td>
</tr>
<tr>
<td>Document 2</td>
<td>89</td>
<td>45</td>
<td>1.00062</td>
<td>1.00041</td>
<td>0.7074</td>
<td>5.1327</td>
<td>1.09845</td>
<td>0.954</td>
</tr>
<tr>
<td>Document 3</td>
<td>98</td>
<td>41</td>
<td>0.91184</td>
<td>0.91143</td>
<td>0.64452</td>
<td>4.67646</td>
<td>1.00062</td>
<td>0.8692</td>
</tr>
<tr>
<td>Document 4</td>
<td>116</td>
<td>50</td>
<td>1.112</td>
<td>1.1115</td>
<td>0.786</td>
<td>5.7031</td>
<td>1.2205</td>
<td>1.06</td>
</tr>
<tr>
<td>Document 5</td>
<td>139</td>
<td>69</td>
<td>1.53456</td>
<td>1.53387</td>
<td>1.08468</td>
<td>3.39062</td>
<td>1.68429</td>
<td>1.46487</td>
</tr>
<tr>
<td>Document 6</td>
<td>233</td>
<td>95</td>
<td>2.1128</td>
<td>2.11185</td>
<td>1.4934</td>
<td>10.8357</td>
<td>2.31895</td>
<td>2.01685</td>
</tr>
<tr>
<td>Document 14</td>
<td>569</td>
<td>325</td>
<td>13.51025</td>
<td>11.62525</td>
<td>5.64152</td>
<td>17.03975</td>
<td>7.54975</td>
<td>8.16075</td>
</tr>
<tr>
<td>Document 17</td>
<td>750</td>
<td>329</td>
<td>13.67653</td>
<td>11.76833</td>
<td>5.71144</td>
<td>17.24947</td>
<td>7.64152</td>
<td>8.26119</td>
</tr>
</tbody>
</table>
Table 2: Precision Value for the various algorithms

<table>
<thead>
<tr>
<th>Documents</th>
<th>NB Precision</th>
<th>SVM Precision</th>
<th>Ngrams Precision</th>
<th>KNN Precision</th>
<th>GA Precision</th>
<th>BPN Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document 1</td>
<td>0.145633</td>
<td>0.4420119</td>
<td>0.76</td>
<td>0.657</td>
<td>0.69</td>
<td>0.59</td>
</tr>
<tr>
<td>Document 2</td>
<td>0.438273</td>
<td>0.4245176</td>
<td>0.69</td>
<td>0.66</td>
<td>0.66</td>
<td>0.71</td>
</tr>
<tr>
<td>Document 3</td>
<td>0.231339</td>
<td>0.1458004</td>
<td>0.71</td>
<td>0.68</td>
<td>0.69</td>
<td>0.76</td>
</tr>
<tr>
<td>Document 4</td>
<td>0.808612</td>
<td>0.8018402</td>
<td>0.72</td>
<td>0.408411</td>
<td>0.72</td>
<td>0.65</td>
</tr>
<tr>
<td>Document 5</td>
<td>0.970417</td>
<td>0.0227901</td>
<td>0.81</td>
<td>0.68</td>
<td>0.8</td>
<td>0.69</td>
</tr>
<tr>
<td>Document 6</td>
<td>0.939275</td>
<td>0.5693175</td>
<td>0.68</td>
<td>0.67</td>
<td>0.71</td>
<td>0.81</td>
</tr>
<tr>
<td>Document 7</td>
<td>0.151648</td>
<td>0.9890791</td>
<td>0.85</td>
<td>0.6</td>
<td>0.69</td>
<td>0.77</td>
</tr>
<tr>
<td>Document 8</td>
<td>0.760024</td>
<td>0.5878237</td>
<td>0.77</td>
<td>0.65</td>
<td>0.72</td>
<td>0.68</td>
</tr>
<tr>
<td>Document 9</td>
<td>0.812871</td>
<td>0.0437858</td>
<td>0.79</td>
<td>0.72</td>
<td>0.73</td>
<td>0.66</td>
</tr>
<tr>
<td>Document 10</td>
<td>0.627609</td>
<td>0.7727432</td>
<td>0.7</td>
<td>0.61</td>
<td>0.65</td>
<td>0.62</td>
</tr>
<tr>
<td>Document 11</td>
<td>0.46163</td>
<td>0.5097816</td>
<td>0.7</td>
<td>0.61</td>
<td>0.67</td>
<td>0.6</td>
</tr>
<tr>
<td>Document 12</td>
<td>0.350079</td>
<td>0.3184212</td>
<td>0.65</td>
<td>0.59</td>
<td>0.7</td>
<td>0.68</td>
</tr>
<tr>
<td>Document 13</td>
<td>0.909816</td>
<td>0.9737966</td>
<td>0.72</td>
<td>0.66</td>
<td>0.73</td>
<td>0.66</td>
</tr>
<tr>
<td>Document 14</td>
<td>0.849321</td>
<td>0.2340496</td>
<td>0.69</td>
<td>0.58</td>
<td>0.8</td>
<td>0.69</td>
</tr>
<tr>
<td>Document 15</td>
<td>0.165843</td>
<td>0.3528359</td>
<td>0.86</td>
<td>0.71</td>
<td>0.81</td>
<td>0.7</td>
</tr>
<tr>
<td>Document 16</td>
<td>0.015779</td>
<td>0.3720599</td>
<td>0.8</td>
<td>0.648003</td>
<td>0.76</td>
<td>0.72</td>
</tr>
<tr>
<td>Document 17</td>
<td>0.220662</td>
<td>0.2745313</td>
<td>0.79</td>
<td>0.6</td>
<td>0.77</td>
<td>0.73</td>
</tr>
<tr>
<td>Document 18</td>
<td>0.473773</td>
<td>0.041502</td>
<td>0.73</td>
<td>0.66</td>
<td>0.69</td>
<td>0.75</td>
</tr>
<tr>
<td>Document 19</td>
<td>0.544628</td>
<td>0.3414689</td>
<td>0.84</td>
<td>0.77</td>
<td>0.68</td>
<td>0.61</td>
</tr>
<tr>
<td>Document 20</td>
<td>0.313244</td>
<td>0.0697869</td>
<td>0.82</td>
<td>0.69</td>
<td>0.77</td>
<td>0.59</td>
</tr>
</tbody>
</table>

**CONCLUSION**

In this article, the six algorithms for Text Categorization have been implemented and compared with each other on their performance. The six algorithms considered are Naive Bayes, Support Vector Machine, N-Grams, K-Nearest Neighbor, Genetic Algorithm, and Backpropagation Network. After the implementation and comparison of these algorithms following observations were made:

- Based on the execution time, N-Grams algorithm is found to consistently produce lesser time than the others. GA and BPN closely follow N-Grams pattern.
- Considering the Precision value, N-Grams algorithm has more precision on most cases. SVM and Naive Bayes produce more precision on few cases.

**REFERENCES**

9. Susan Dumais, John Platt, David Heckerman and Mehran Sahami, 1998. Inductive Learning Algorithms and Representations for Text Categorization. 7th International Conference on Knowledge Management,

