Representation and Classification of Text Documents: A Brief Review

B S Harish
Department of Studies in Computer Science
Manasagangotri, University of Mysore
Mysore – 570 006, India.

D S Guru
Department of Studies in Computer Science
Manasagangotri, University of Mysore
Mysore – 570 006, India.

S Manjunath
Department of Studies in Computer Science
Manasagangotri, University of Mysore
Mysore – 570 006, India.

ABSTRACT

Text classification is one of the important research issues in the field of text mining, where the documents are classified with supervised knowledge. In literature we can find many text representation schemes and classifiers/learning algorithms used to classify text documents to the predefined categories. In this paper, we present various text representation schemes and compare different classifiers used to classify text documents to the predefined classes. The existing methods are compared and contrasted based on qualitative parameters viz., criteria used for classification, algorithms adopted and classification time complexities.

General Terms

Pattern Recognition, Text Mining, Algorithms

Keywords

Text classification, Documents, Text Representation, Classifiers

1. INTRODUCTION

Over the past two decades, the automatic management of electronic documents has been a major research field in computer science. Text documents have become the most common type of information repositories especially with the increased popularity of the internet and the World Wide Web (WWW). Internet and web documents like web pages, emails, newsgroup messages, internet news feed etc., contain million or even billion of text documents. In the last decades content-based document management tasks have gained a prominent status in the information systems field, due to the increased availability of documents in digital form [1] [2].

From several decades, automatic document management tasks have gained a prominent status in the field of information retrieval. Until late 80's, text classification task was based on Knowledge Engineering (KE), where a set of rules were defined manually to encode the expert knowledge on how to classify the documents under the given categories [3]. Since there is a requirement of human intervention in knowledge engineering, researchers in 90's have proposed many machine learning techniques to automatically manage the text documents [3]. The advantages of a machine learning based approach are that the accuracy is comparable to that achieved by human experts and no intervention from either knowledge engineers or domain experts needed for the construction of a document management tool [4]. Many text mining methods like document retrieval, clustering, classification, routing, filtering are often used for effective management of text documents. However, in this paper we concentrate only on classification of text documents.

The task of text classification is to assign a boolean value to each pair $(d_j, k_i) \in D \times K$, where 'D' is the domain of documents and 'K' is a set of predefined categories. The task is to approximate the true function $\phi : D \times K \rightarrow 1,0$ by means of a function $\hat{\phi} : D \times K \rightarrow \{1,0\}$ such that $\phi \, and \, \hat{\phi}$ coincide as much as possible. The function $\hat{\phi}$ is called a classifier. A classifier can be built by training it systematically using a set of training documents D, where all of the documents belonging to D are labeled according to K [4, 5]. Text classification presents many challenges and difficulties. First, it is difficult to capture high-level semantics and abstract concepts of natural languages just from a few key words. Furthermore, semantic analysis, a major step in designing an information retrieval system, is not well understood, although there are some techniques that have been successfully applied to limited domains. Second, high dimensionality (thousands of features) and variable length, content and quality are the characteristics of a huge number of documents on the Web. These place both efficiency and accuracy demands on classification systems [4]. Since any classifier is unable to understand a document in its raw format, a document has to be converted into a standard representation. Extensive work is carried out to propose various text representation techniques and text classification methods in the literature. But, it is essential for researchers/practitioners to have a complete knowledge on all existing representation schemes and classifiers in order to select an appropriate representation scheme and classifier which best suits their purpose/application. In this context, we reiterate that our focus is only on the widely accepted representation schemes and classifiers. Thus, this paper aims at providing an overview about various text representation schemes and classifiers. In addition, we also present a comparative study on different representation schemes and classifiers used to classify text documents to the predefined classes. The existing methods are compared and contrasted based on qualitative parameters viz., criteria used for classification, algorithms adopted and classification time complexities.

This paper is organized as follows: In section 2, a brief introduction to widely used text representation schemes are discussed. Overview of various text classifiers is given section 3. The outcome of quantitative and qualitative analysis of various text classifiers is reported in section 4. The paper concludes in section 5.

2. TEXT REPRESENTATION

In automatic text classification, it has been proved that the term is the best unit for text representation and classification [6]. Though a text document expresses vast range of information, unfortunately, it lacks the imposed structure of traditional database. Therefore, unstructured data, particularly free running text data has to be transformed into a structured data. To do this, many preprocessing techniques are proposed in literature [7, 8]. After converting an unstructured data into a structured data, we need to have an effective document representation model to build an efficient classification system. Bag of Word (BoW) is one of the basic methods of representing a document. The BoW is used to form a vector representing a document using the frequency count of each term in the document. This method of document representation is called as a Vector Space Model (VSM) [9]. Unfortunately, BoW/VSM representation scheme has its own limitations. Some of them are: high dimensionality of the representation, loss of correlation with adjacent words and loss of semantic relationship that exist among the terms in a document [10]. To overcome these problems, term weighting methods are used to assign appropriate weights to the term to improve the performance of text classification [11, 12]. Jain and Li in [13] used binary representation for given document. The major drawback of this model is that it results in a huge sparse matrix, which raises a problem of high dimensionality. Hotho et al., in [14] proposed an ontology representation for a document to keep the semantic relationship between the terms in a document. This ontology model preserves the domain knowledge of a term present in a document. However, automatic ontology construction is a difficult task due to the lack of structured knowledge base. Cavanar., (1994) in [15] used a sequence of symbols (byte, a character or a word) called N-Grams, that are extracted from a long string in a document. In a N-Gram scheme, it is very difficult to decide the number of grams to be considered for effective document representation. Another approach in [16] uses multi-word terms as vector components to represent a document. But this method requires a sophisticated automatic term extraction algorithms to extract the terms automatically from a document. Wei et al., (2008) in [17] proposed an approach called Latent Semantic Indexing (LSI) which preserves the representative features for a document. The LSI preserves the most representative features rather than discriminating features. Thus to overcome this problem, Locality Preserving Indexing (LPI) [18] was proposed for document representation. The LPI discovers the local semantic structure of a document. Unfortunately LPI is not efficient in time and memory [19]. Choudhary and Bhattacharyya (2002) in [20] used Universal Networking Language (UNL) to represent a document. The UNL represents the document in the form of a graph with words as nodes and relation between them as links. This method requires the construction of a graph for every document and hence it is unwieldy to use for an application where large numbers of documents are present. Craven et al., (1998) in [21] developed a Web-KB project for constructing and maintaining large knowledge bases. An ontology is constructed manually and a seed knowledge base comprising a set of labeled web pages learns to instantiate knowledge-base objects and relations from the web. In [22], a new representation to model the web documents is proposed. HTML tags are used to build the web document representation. They used histogram representation for frequency of terms in four sections of HTML codes: text, bold, links and titles. Each symbolic object is built after the web collection is analyzed and the most frequent terms are obtained. Isa et al., (2008) in [23] used the Bayes formula to vectorize a document according to a probability distribution reflecting the probable categories that the document may belong to. Using this probability distribution as the vectors to represent the document, the SVM is used to classify the documents. The same work has been extended by Guru et al., (2010) [21] to represent a text document by the use of interval valued symbolic features. The probability distributions of terms in a document are used to form a symbolic representation and then it is used for training and classification purposes. Dinesh et al., (2009) [1] proposed a new datastructure called status matrix which preserves the sequence of term occurrence in a document. Classification of documents is done based on this new representation.

3. OVERVIEW OF TEXT CLASSIFIERS

Naïve Bayes Classifier [5, 8, 25]

The Naïve Bayes classifier is the simplest probabilistic classifier used to classify the text documents. The classifier estimates the joint probability of a document d_i belonging to class C_k, i.e. $P(C_k|d_i)$. The output of the classifier is the probability of the document of belonging to each class and it is a vector of $|C|$ elements. However, this probability can be estimated using a simple Bayes formula and $P(C_k|d_i)$ can be rewritten as $P(C_k|d_i) = P(d_i|C_k) \times \frac{P(C_k)}{P(d_i)}$. The classifier estimates $P(d_i|C_k), P(C_k)$, where $P\ d_i|C_k$ is the probability of k for the given class C, $P\ C_k$ is the priori probability of training data d_i and $P(d_i)$ is the priori probability of training data d_i. Here, $P(C_k)$ and $P(d_i)$ are supposed to be constants. Thus, in the context of text classification, using the Bag of Words representation we can compute, $P(d_i|C_k)$ as give below:

$$P(d_i \mid C_k) = P(Bow(d_i) \mid C_k) = P(w_{1,i}, w_{2,i}, \ldots, w_{|V|,i} \mid C_k).$$

But the assumption of a Naïve Bayes classifier is that j^{th} word in a i^{th} text document w_{ji} is not correlated with other ones.

$$P(d_i \mid C_k) = P(w_{1,i}, w_{2,i}, \ldots, w_{|V|,i} C_k) = \prod_{j}^{|V|} P(w_{j,i} \mid C_k)$$

Here the problem is reduced to estimate the probability of the single word W_{ji} in the class C_k.

Nearest Neighbor classifier [8, 26]

The Nearest Neighbor classifier is used for text classification. The Nearest Neighbor classification is a non-parametric method and it can be shown that for large datasets the error rate of the 1-Nearest Neighbor classifier is never larger than twice the optimal error rate. In this classifier to decide whether the document d_i belongs to class C_k, the similarity $Sim(d_i, d_j)$ or dissimilarity $Dissim(d_i, d_j)$ to all documents d_j in the training set is determined. The k most similar training documents (neighbors) are selected. The proportion of neighbors having the same class may be taken as an estimator for the probability of that class and the class with the largest proportion is assigned to the document d_j. The algorithm has two parameters (k and similarity/dissimilarity value) which decide the performance of the classifier and are empirically determined. However, the optimal number 'k' of neighbors may be estimated from additional training data by cross validation [8, 27, 28]. The major drawback of the classifier is the computational effort during classification, as basically the similarity of a document with respect to all other documents of a training set has to be determined.

Centroid based classifier [29]

Centroid based classifier is the most popular supervised approach used to classify texts into a set of predefined classes with relatively low computation. Based on the vector space model, the performance of the classifier depends on the way to weigh the terms in documents in order to construct a representative class vector for each class and degree of spherical shape in class [30]. Based on the documents in each class the centroid based classifier selects a single representative called "centroid" and then it works like K-NN classifier with K=1. Given a set S of documents and their representation, we need to compute the summed centroid and normalized centroid of class C_i by C_i^s and C_i^N [29]. Then we have to calculate the similarity between documents d with each class C_i. Based on these similarities, assign d the class label corresponding to the most similar centroid. Also the improved batch updated centroid classifier is proposed in [29], which introduces a constant parameter called "Learn Rate", which improves the efficiency of the classifier.

Decision trees [13, 26, 31, 32]

Decision trees are the most widely used inductive learning methods. Their robustness to noisy data and their capability to learn disjunctive expressions seem suitable for document classification. One of the most well known decision tree algorithms is ID3 and its successor C4.5 and C5. It is a top-down method which recursively constructs a decision tree classifier [13]. A Decision Tree (DT) text classifier is a tree in which internal nodes are labeled by terms, branches departing from them are labeled by the weight that the term has in the test document, and leafs are labeled by categories. Such a classifier categorizes a test document d_j by recursively testing for the weights that the terms labeling the internal nodes have in vector $\vec{d}_j$, until a leaf node is reached; the label of this node is then assigned to d_j.

A possible method for learning a DT for category C_j consists in a "divide and conquer" strategy of (i) checking whether all the training examples have the same label (ii) if not, selecting a term t_k, partitioning from the pooled classes of documents that have the same value for t_k, and placing each such class in a separate subtree. The process is repeated on the subtrees until each leaf of the tree contains training examples assigned to the same class C_j, which is then chosen as the label for the leaf. The key step is the choice of the term t_k on which to operate the partition. Generally, a choice is generally made according to an information gain or entropy criterion. However, such a "fully grown" tree may be prone to overfitting, as some branches may be too specific to the training data. Most DT learning methods thus include a method for growing the tree and one for pruning it, for removing the overly specific branches [26, 31]. DT text classifiers have been used either as the main classification tools or as baseline classifiers or as members of classifier committees.

Linear Classifiers [33]

(Yang and Chute, 1994) in [33] has proposed a regression model called Linear Least Square Fit (LLSF) applied to text classification. In LLSF, each document d_j has two vectors associated to it: an input vector $I\ d_j$ of $|T_j|$ weighted terms and an output vector $O\ d_j$ of weights $|C|$ representing the categories. Thus text classification can be seen as the task of determining an output vector $O\ d_j$ for a test document d_j, given its input vector $I\ d_j$. A linear classifier [26] [34] is also used for text classification, which computes the categorization status value that corresponds to the dot product of $\vec{d}_j$ *and* $\vec{C}_j$. Basically there are two methods to learn the linear classifiers: Batch methods, which build a classifier by analyzing the training set all at once and On-line method, which builds a classifier soon after examining the first training document and incrementally refine it as they examine new ones. A classifier [32] rewards the closeness of a test document to the centroid of the positive training examples and its distance from the centroid of the negative training examples. The effectiveness of the classifier depends on the Positive (POS) and Negative (NPOS) factors and two control parameters β and γ. The role of negative examples is usually deemphasized by setting β to a high value and γ to a low value.

Support Vector Machine (SVM) [5, 8, 26, 35]

A Support Vector Machine (SVM) is a supervised classification algorithm that has been extensively and successfully used for text classification tasks. A document d_j is represented by a – possibly weighted – vector $t_{d_1}, t_{d_2}, ..., t_{d_j}$ of the counts of its words. A single SVM can only separate two classes—a positive class $L1$ (indicated by y = +1) and a negative class $L2$ (indicated by y = -

1). In the space of input vectors a hyperplane may be defined by setting $y = 0$ in the following linear equation [8].

$$y = f(\vec{t_d}) = b_0 + \sum_{j=1}^{N} b_j t_{d_j}$$

The SVM algorithm determines a hyperplane which is located between the positive and negative examples of the training set. The parameters b_j are adapted in such a way that the distance ξ – called *margin* – closest positive and negative example documents is maximized. The documents having distance ξ from the hyperplane are called *support vectors* and determine the actual location of the hyperplane. Usually only a small fraction of documents are support vectors. A new document with term vector $\vec{t_d}$ is classified in $L1$ if the value $f\ \vec{t_d} > 0$ and into $L2$ otherwise. In case that the document vectors of the two classes are not linearly separable a hyperplane is selected such that as few as possible document vectors are located on the "wrong" side [8]. The most important property of SVMs is that learning is nearly independent of the dimensionality of the feature space. It rarely requires feature selection as it inherently selects data points (the support vectors) required for a good classification. This allows good generalization even in the presence of a large number of features and makes SVM especially suitable for the classification of texts.

Neural Network classifier [26]

Neural Network based text classifier [26] are also found in the literature, where the input units represent terms, the output unit(s) represents the category or categories of interest and the weights on the edges connecting units represent dependence relations. For classifying a given test document d_j, its term weights w_{kj} are loaded into the input units; the activation of these units is propagated forward through the network, and the value of the output unit(s) determines the categorization decision(s). A typical way of training neural networks is backpropagation, whereby the term weights of a training document are loaded into the input units, and if a misclassification occurs the error is "back propagated" so as to change the parameters of the network and eliminate or minimize the error.

4. COMPARATIVE OBSERVATIONS

The overall classification accuracy of any machine learning classifier suffers from inductive bias or model misfits. The classification accuracy can be increased, whenever the nature of the data fits well the assumptions of the underlying classification strategy. Model misfit may occur for many reasons, including the choice of type of features used in the model, choice of the number of features to use, choice of scoring method, choice of similarity functions etc., these factors may also compound and the more serious the model misfit, the poorer classification performance will be [36]. However some of the observations are mentioned below:

The works in [26, 37, 38] give two different methods to compare the performances of classifiers.

Direct comparison: Two classifiers Φ and $\Phi^{''}$ may be compared by testing them on the same collection of documents, usually by the same researchers and with the same background conditions. This is the more reliable method.

Indirect comparison: Two classifiers Φ and $\Phi^{''}$ may be compared when

(1) They have been tested on different collections of documents, by different researchers and with possibly different background conditions;

(2) Here one or more "other baseline" classifier is used to evaluate the results obtained by the direct comparison method.

Thus, from the above discussion, following analysis on various classifiers can be inferred.

Naïve Bayes classifier is a very simple classifier which works very well on numerical and textual data. It is very easy to implement and computationally cheap when compared to any other classification algorithms. One of the major limitations of this classifier is that it performs very poorly when features are highly correlated. Also with respect to a text classification, it fails to consider the frequency of word occurrences in the feature vector. On contrary, Nearest Neighbor classifier is very effective and it is non-parametric in nature. Here the local characteristics of the document can be preserved. Unfortunately the classification time is very long and finding the optimum value of k is a difficult issue. Also, the performance of a KNN classifier is primarily determined by the choice of k as well as the distance metric applied. However, in case of KNN classifier when the points are not uniformly distributed, predetermining the value of 'k' becomes difficult. Generally, smaller values of k results in compact classes where there will be a lot of overlapping of documents belonging to the different classes. On the other hand, larger values of k are more immune to the noise presented, and make boundaries smoother between classes. As a result, choosing the same (optimal) k becomes almost impossible for different applications. Although centroid classification is a simple and is linear, it often suffers from the model misfit incurred by its assumption. Centroid classifier always assumes that a given document should be assigned to a particular class if the similarity of this document to the centroid of the class is largest. However, this assumption is not always true in practical cases. A Rule based classifier like a decision tree is easy to understand and reduces the problem complexity. Unfortunately more training time is required and document is represented in a tree structure. In this case, once the mistake is made at higher level than any subtree will become wrong. Also the decision tree fails to handle the continuous variable and also they suffer from over fitting problem. Even though the rocchio's algorithms are very easy to implement and fast learner, this classifier gives very low classification accuracy. Further selecting the constant β and γ values are empirical. Neural Network based classifier produces a very good result in complex domains and it has a capability to handle both discrete and continuous data. The drawback is that the training is relatively slow and the learned results are difficult for users to interpret. Also adaptive selection of a learning rate is a complex task. Compare to the other classifiers, SVM has a capability to capture the inherent characteristics of the data in a better way. It has an ability to learn independently about the dimensionality of the feature space. The complexity of the classifier lies in the parameter tuning and kernel selection. Table 1 gives the various results quoted by others using widely used text representations schemes. Also the Table 2 gives the qualitative comparison between the various classifiers discussed. The table presents the comparison between the criteria used for classification, algorithms adopted and classification time complexities.

5. CONCLUSION

This paper gives a brief introduction to the various text representation schemes and classifiers used in the field of text mining. The existing methods are compared and contrasted based on various parameters namely criteria used for classification, algorithms adopted and classification time complexities. From the above discussion it is understood that no single representation scheme and classifier can be recommended as a general model for any application. Different algorithms perform differently depending on data collections. None of them appears globally superior over the other. However, to the certain extent SVM with term weighted VSM representation scheme performs well in many text classification tasks.

6. REFERENCES

[1] Dinesh, R., Harish, B. S., Guru, D.S., and Manjunath, S. 2009. Concept of Status Matrix in Text Classification. In the Proceedings of Indian International Conference on Artificial Intelligence, Tumkur, India, pp. 2071 – 2079.

[2] Guru, D. S., Harish B. S., and Manjunath, S. 2009. Clustering of Textual Data: A Brief Survey," In the Proceedings of International Conference on Signal and Image Processing, pp. 409 – 413.

[3] Mitra, V., Wang, C.J., and Banerjee, S. 2007. Text Classification: A least square support vector machine approach. Journal of Applied Soft Computing. vol. 7, pp. 908 – 914.

[4] Fung, G.P.C., Yu, J.X., Lu. H., and Yu, P.S. 2006. Text classification without negative example revisit. IEEE Transactions on Knowledge and Data Engineering. Vol. 18, pp. 23 – 47.

[5] Rigutini, L. 2004. Automatic Text Processing: Machine Learning Techniques. Ph.D. Thesis, University of Siena.

[6] Song, F., Liu, S., and Yang, J. 2005. A comparative study on text representation schemes in text categorization," Journal of Pattern Analysis Application, Vol 8, 2005, pp 199 – 209.

[7] Porter, M.F. 1980. An algorithm for suffix stripping. Program, Vol. 14 (3), pp. 130 –137.

[8] Hotho, A., Nürnberger, A., and Paaß, G. 2005. A Brief Survey of Text Mining. Journal for Computational Linguistics and Language Technology. Vol. 20, pp. 19 – 62.

[9] Salton, G., Wang, A., and Yang, C.S.1975. A Vector Space Model for Automatic Indexing. Communications of the ACM, Vol. 18, pp. 613 – 620.

[10] Bernotas, M., Karklius, K., Laurutis, R., and Slotkiene, A. 2007. The peculiarities of the text document representation, using ontology and tagging-based clustering technique. Journal of Information Technology and Control. Vol. 36, pp. 217 – 220.

[11] Lan, M., Tan, C. L., Su. J., and Lu, Y.2009. Supervised and Traditional Term Weighting Methods for Automatic Text Categorization. IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume: 31 (4), pp. 721 – 735.

[12] Altınçay, H., and Erenel, Z. 2010. Analytical evaluation of term weighting schemes for text categorization. In Journal of Pattern Recognition Letters, vol. 31 (11), pp. 1310 – 1323.

[13] Jain, A. K., and Li, Y. H. 1998. Classification of Text Documents. The Computer Journal, Vol 41, pp. 537 – 546.

[14] Hotho, A., Maedche, A., and Staab, S. 2001. Ontology-based text clustering. In Proceedings of International Joint Conference on Artificial Intelligence, pp. 30 –37.

[15] Cavnar, W.B. 1994. Using an N-Gram based document representation with a vector processing retrieval model. In Proceedings of The Third Text Retrieval Conference (TREC-3), pp. 269 – 278.

[16] Milios, E., Zhang, Y., He, B., and Dong, L. 2003. Automatic term extraction and document similarity in special text corpora. In Proceedings of Sixth Conference of the Pacific Association for Computational Linguistics (PACLing'03), pp. 275 – 284.

[17] Wei, C. P., Yang, C. C., and Lin, C. M. 2008. A Latent Semantic Indexing-based approach to multilingual document clustering. Journal of Decision Support System. Vol. 45, pp. 606 – 620.

[18] He, X., Cai, D., Liu, H., and Ma, W.Y. 2004. Locality Preserving Indexing for document representation. In SIGIR, pp. 96—103.

[19] Cai, D., He, X., Zhang, W.V., and Han J. 2007. Regularized Locality Preserving Indexing via Spectral Regression. In ACM International Conference on Information and Knowledge Management (CIKM'07), pp. 741—750.

[20] Choudhary, B., and Bhattacharyya, P. 2003. Text clustering using Universal Networking Language representation. In Eleventh International World Wide Web Conference.

[21] Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T. M., Nigam, K., and Slattery, S. 1998. Learning to Extract Symbolic Knowledge from the World Wide Web. In Proceedings of AAAI/IAAI', pp. 509 – 516.

[22] Esteban, M., and Rodrıguez, O. R. 2006. A Symbolic Representation for Distributed Web Document Clustering. In the Proceedings of Fourth Latin American Web Congress, Cholula, Mexico.

[23] Isa, D., Lee, L. H., Kallimani, V. P., and Rajkumar, R. 2008. Text document preprocessing with the Bayes formula for classification using the support vector machine. IEEE Transactions on Knowledge and Data Engineering. Vol. 20, pp. 23 – 31.

[24] Guru D. S., Harish B. S., and Manjunath. S. 2010. Symbolic representation of text documents. In Proceedings of Third Annual ACM Bangalore Conference.

[25] Mitchell, T. M. 1997. Machine Learning. Mc Graw Hill, New York, NY.

[26] Sebastiani, F. 2002. Machine learning in automated text categorization. ACM Computing Surveys. Vol 34, pp. 1 – 47.

[27] Yang, Y., Slattery, S., and Ghani, R. 2002. A study of approaches to hypertext categorization. Journal of Intelligent Information Systems, Vol 18(2), pp. 219 – 241.

[28] Sandip, K. 2003. An experimental study of some algorithms for text categorization. M.Tech Thesis, IIT Kanpur, India.

[29] Tan, S. 2008. An improved centroid classifier for text categorization. Journal of Expert System with Applications, Vol 35, pp 279 – 285.

[30] Theeramunkong, T., and Lertnattee, V. 2001. Improving centroid-based text classification using term distribution-based weighting system and clustering. ISCIT, pp. 33–36.

[31] Lewis, D. D., and Ringuette, M. 1998. A comparison of two learning algorithms for text classification. In the Proceedings of Third annual symposium on Document Analysis and Information Retrieval, pp. 81–93.

[32] Joachims, Y. 1997. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In the Proceedings of the Fourteenth International Conference on Machine Learning, pp. 143 –151.

[33] Yang, Y., and Chute, C. G. 1994. An example based mapping method for text categorization and retrieval. ACM Transactions on Information Systems, Vol 12(3), pp 252-277.

[34] Lewis, D. D., Schapire, R. E., Callan, J. P., and Papka, R. 1996. Training algorithms for linear text classifiers. In the Proceedings of the Nineteenth International Conference on Research and Development in Information Retrieval (SIGIR'96), pp. 289–297.

[35] Joachims, T. 1998. Text categorization with support vector machines: Learning with many relevant features. In the Proceedings of European Conference on Machine Learning (ECML), Vol 1398, pp. 137-142.

[36] Songbo, T., Cheng, X., Ghanem, M. M., Wnag, B., and Xu, H. 2005. A novel refinement approach for text categorization. In the Proceedings of Fourteenth ACM International Conference on Information and Knowledge Management, pp 469 – 476.

[37] Yang, Y. 1999. An evaluation of statistical approaches to text categorization. Information Retrieval, Vol 1, pp 69 – 90.

[38] Ikonomakis, M., Kotsiantis, S., and Tampakas, V. 2005. Text Classification Using Machine Learning Techniques.2005. Wseas Transactions on Computers, Vol 4 (8), pp 966 – 974.

[39] Ko, Y. J., Park, J., and Seo, J. 2004. Improving text categorization using the importance of sentences. An International Journal Information Processing and Management, Vol. 40, pp. 65 – 79.

[40] Liang, C. Y., Guo, L., Xia, Z. H., Nie, F. G., Li, X. X., Su, L., and Yang, Z. Y. 2006. Dictionary-based text categorization of chemical web pages. An International Journal Information Processing and Management, Vol 42, pp. 1017 – 1029.

[41] Mubaid, H. A., and Umair, S. A. 2006. A New Text Categorization Technique Using Distributional Clustering and Learning Logic. IEEE Transactions on Knowledge and Data Engineering, Vol 18 (9), pp. 1156 – 1165.

[42] Hao, P. Y., Chiang, J. H., and Tu, Y. K. 2007. Hierarchically SVM classification based on support vector clustering method and its application to document categorization. An International Journal Expert Systems with Applications, Vol 33(3), pp. 627-635.

[43] Shang, W., Huang, H., Zhu, H., Lin, Y., Qu, Y., and Wang, Z. 2007. A novel feature selection algorithm for text categorization. An International Journal Expert Systems with Applications, Vol 33(1), pp. 1-5.

[44] Qian, T., Xiong, H., Wang, Y., and Chen, E. 2007. On the strength of hyperclique patterns for text categorization. An International Journal Information Sciences, Vol. 177, pp. 4040–4058.

Table 1: Comparative Results Among Different Representation Schemes and Classifiers obtained on Reuters 21578 and 20 Newsgroup Datasets.

Results reported by	**Dataset**	**Representation Scheme**	**Classifier Used**	**Micro F1**	**Macro F1**
[Ko et al., 2004] [39]	20 Newsgroup	Vector representation with different weights	Naïve Bayes	83.00	83.30
			Rocchio	79.10	78.60
			K-NN	81.04	81.20
			SVM	86.10	86.00
[Tan et al., 2005] [36]	20 Newsgroup	Vector representation	Naïve Bayes	0.835	0.835
			Centroid	0.842	0.838
			K-NN	0.848	0.846
			SVM	0.889	0.887
[Liang et al., 2005] [40]	Reuters 21578	Vector representation	K-NN	0.797	-
[Mubaid and Umair., 2006] [41]	20Newsgroup	Vector representation	L Square	86.45	83.05
			SVM	84.62	78.19
	Reuters 21578	Vector representation	L Square	-	94.57
			SVM	-	95.53
[Hao et al., 2006] [42]	Reuters 21578	Hierarchical graph structure	SVM(Polynomial)	86.20	-
			SVM(rbf)	86.50	-
			K-NN	0.788	-
			Decision Tree	0.879	-
[Shang et al., 2007] [43]	Reuters 21578	Vector representation + Gini Index	SVM	88.59	69.94
		Vector representation + Information Gain	K-NN	85.76	66.86
[Qian et al., 2007] [44]	Reuters 21578	Hyperclique patterns	Naïve Bayes	81.50	74.80
			Bayes Net	85.00	78.80
			Decision Tree	88.40	82.20
			Linear SVM	92.00	87.10
	20 Newsgroup	Hyperclique patterns	ATC-H	75.77	74.97
[Lan et al., 2009] [11]	Reuters 21578	VSM with term weighting schemes	SVM	0.921	0.900
			K-NN	0.840	0.825
	20 Newsgroup	VSM with term weighting schemes	SVM	0.808	0.808
			K-NN	0.691	0.691

Table 2: Qualitative Factors Based Comparison

Classifier Name	Time Complexity	Type of Algorithm	Classifier Criterion
Naïve Bayes Classifier [5][6][7]	Training→ $\Theta(\|D\|L_{ave} + \|C\|\|V\|)$ Testing→ $\Theta(L_a + \|C\|M_a) = \Theta(\|C\|M_a)$ Complexity of computing parameter is $\Theta(\|C\|\|V\|)$ since the set of parameters consists of $\|C\|\|V\|$ conditional probabilities and $\|C\|$ priors Where, $\|D\|$ = Number of documents L_{ave} = average length of a document L_a = Number of tokens M_a = types, in the test document	Binary	$C_{NB}^{*} = \arg\max_{c_j \in C} P(c_j) \prod_{i=1}^{d'} P(w_i \| c_j)$ Where, $P(c_j)$ → priori probability of class c_j $P(w_i \| c_j)$ → conditional probability of word w_i given in cluster c_j
Nearest Neighbor Classifier [4][6][8] [9]	Training → $O(NL_d)$ Testing→ $o(\frac{N}{V} L_V^2) + O(N)$ Where, N: The number of training documents L_d: The average document length (word count) L_V: The average number of unique words in a document V: The number of features (the vocabulary size)	M-way	The k top ranking training documents are selected and thus they are used to compute a score for every category as follows: $score(c_j \| \vec{x}) = \sum_{\vec{x} \in KNN} \sum_{\vec{x} \in c_j} sim(\vec{x} \in \vec{x}_i)$
Centroid Classifier [10][11]	If there are 'N' training documents, 'T' test documents, 'W' words in total, K classes and M iteration steps, then complexity to compute the summed centroid and normalized centroid is $O(NW + KW)$, since $K < N$ the time complexity is $O(NW)$. Overall time complexity of centroid classifier is $O(TKW)$.	M-way	Summed centroid $C_i^s = \sum_{d \in c_i} d$ Normalized Centroid $C_i^N = C_i^S / \|\|C_i^S\|\|_2$ Improved Centroid classifier

			$C_A^* = C_A + Learnrate\ X \left(\sum_{\substack{d \notin Category\ A \\ Classified\ into \\ other\ categories}} d - \sum_{\substack{d \notin Category\ A \\ Classified\ into \\ other\ categorie}} d \right)$ $C_{A,i}^{**} = \begin{matrix} C_{A,i}^{*}\ if\ C_{A,i}^{*} \geq 0 \\ 0 \quad if\ C_{A,i}^{*} < 0 \end{matrix}$ $C_A^{***} = C_A^{**} \big/ \left\lVert C_A^{**} \right\rVert_2$
Decision Tree [4][12] [13][14]	Training set D→ $O(n \times \lvert D \rvert \times \log \lvert D \rvert)$, where 'n' is the number of attributes describing the tuples in D and │D│ is the number of training tuples in D.	Generate a decision tree from the training tuples of data partition D	• Data partition, D, which is a set of training tuples and their associated class labels; • attribute_list, the set of candidate attributes; • attribute_selection_methods, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes.
Neural Network [6]	Depends upon the selection of learning rate. If the learning rate is too small, then learning will occur at a very slow pace. If the learning rate is too large, then oscillation between inadequate solutions may occur. Thumb rule says: Set learning rate to 1/t, where t is the number of iterations through the training set	M-way	$I_j = \sum_i w_{ij} O_i + \theta_j$ Which, computes the net input of unit j with respect to the previous layer, i. $O_j = \frac{1}{1+e^{-I_j}}$ Output of each unit j.
Regression Model (Linear Least Squares Fit (LLSF) mapping method) [15]	Training time on M categories → $O(N^2 K_s)$ Testing time per document → $O(ML_V)$ N: The number of training documents L_V: The average number of unique words in a document M: The number of training set in categories (M<N)	M-way	The optimization problem in LLSF is : $W^* = \arg\min_W \{ \sum_{i=1}^{N} \sum_{j=1}^{M} ((\vec{x}_i, \vec{w}_j) - y_{ij})^2 \}$ Where X is a document-term matrix whose elements are term weights in training documents, and Y is a document category matrix whose elements $y_{ij} \in \{0,1\}$ indicates whether the ith training document belongs to the jth category. The solution W is a term-category matrix whose element w_{tj} is the regression coefficient ("weight") of term t in the prediction of category c_j.
Support Vector Machines (SVM)	Training time on M categories → $O(MN^C)$	2- class or M-class	The optimization of linear SVM is to minimize

[4][6][7] [17]	Testing time per document → $O(ML_V)$ N: The number of training documents L_V: The average number of unique words in a document M: The number of training set in categories (M<N)		$\vec{\alpha}^* = \arg\min_{\vec{\alpha}}\{-\sum_{i=1}^{n}\alpha_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (\vec{x_i}, \vec{x_j})\}$ Subject to: $\sum_{i=1}^{n}\alpha_i y_i = 0; 0 \le \alpha_i \le C$
The Rocchio Method (Linear Classifier) [4][7][14] [16]	Training → $\Theta(\|D\|L_{ave} + \|C\|\|V\|)$ Testing→ $\Theta(L_a + \|C\|M_a) = \Theta(\|C\|M_a)$ Complexity of computing parameter is $\Theta(\|C\|\|V\|)$ since the set of parameters consists of $\|C\|\|V\|$ conditional probabilities and $\|C\|$ priors and $\|D\|$ = Number of documents L_{ave} = average length of a document L_a = Number of tokens M_a = types, in the test document	Binary	Rocchio's method computes a classifier for category c_i by means of the formula: $w_{ki} = \beta . \sum_{dj \in POS_i} \frac{w_{kj}}{\|POS_i\|} - \gamma . \sum_{dj \in POS_i} \frac{w_{kj}}{\|NPOS_i\|}$ Where w_{kj} is the weight of t_k in document d_j, POS and NPOS are positive and near positive factors, β and γ are control parameters.