Volume 4, Issue 6, June 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Machine learning approach to text mining: A review

Manzoor Ahmad *
Scientist 'D', Department of Computer Sciences,
University of Kashmir, Srinagar, J&K,
India,190001

Abstract— World Wide Web has become very popular in last decades and brought us a powerful platform to disseminate information and retrieve information as well as analyze information, and nowadays the Web has been known as a big data repository consisting of a variety of data types, as well as a knowledge base, in which informative Web knowledge is hidden. This enormous amount of information stored in unstructured texts cannot simply be used for further processing by computers, which typically handle text as simple sequences of character strings. Therefore, specific pre-processing methods and algorithms are required in order to extract useful patterns. Text mining refers generally to the process of extracting interesting information and knowledge from unstructured text. It can be viewed as one of a class of non-traditional Information Retrieval (IR) strategies which attempt to treat entire text collections holistically, avoid the bias of human queries, and objectify the IR process with principled algorithms. These strategies share many techniques such as semantic parsing and statistical clustering, and the boundaries between them are fuzzy. In this paper different existing Text Mining Algorithms i.e. Classification Algorithm, Clustering Algorithm is briefly reviewed, stating the merits / demerits of the algorithms.

Keywords— Data Mining, Text Mining, Classification, Clustering, Association, agglomerative, Divisive, Information Retrieval, Information Extraction.

I. INTRODUCTION

Labour-intensive manual text-mining approaches first surfaced in the mid-1980s, but technological advances have enabled the field to advance swiftly during the past decade. Text mining is an interdisciplinary field which draws on information retrieval, data mining, machine learning, statistics, and computational linguistics. As most information (common estimates say over 80%) is currently stored as text, text mining is believed to have a high commercial potential value. Increasing interest is being paid to multilingual data mining: the ability to gain information across languages and cluster similar items from different linguistic sources according to their meaning. Text mining, sometimes alternately referred to as text data mining, roughly equivalent to text analytics, refers to the process of deriving high-quality information from text. High-quality information is typically derived through the divining of patterns and trends through means such as statistical pattern learning. Text mining usually involves the process of structuring the input text (usually parsing, along with the addition of some derived linguistic features and the removal of others, and subsequent insertion into a database), deriving patterns within the structured data, and finally evaluation and interpretation of the output. 'High quality' in text mining usually refers to some combination of relevance, novelty, and interestingness. Most Text mining research assumes that the information to be "mined" is already in the form of a relational database. Unfortunately, for many applications, available electronic information is in the form of unstructured natural-language documents rather than structured databases. Consequently, the problem of text mining, i.e. discovering useful knowledge from unstructured text, is becoming an increasingly important aspect of KDD

II. TEXT MINING PROCESS

Current research in the area of text mining tackles problems of text representation, classification, clustering, information extraction or the search for and modelling of hidden patterns. In this context the selection of characteristics and also the influence of domain knowledge and domain-specific procedures plays an important role. Therefore, an adaptation of the known data mining algorithms to text data is usually necessary. In order to achieve this, one frequently relies on the experience and results of research in information retrieval, natural language processing and information extraction. In all of these areas we also apply data mining methods and statistics to handle their specific tasks:

A. INFORMATION RETRIEVAL

Information retrieval is the finding of documents which contain answers to questions and not the finding of answers itself. In order to achieve this goal statistical measures and methods are used for the automatic processing of text data and comparison to the given question. Information retrieval in the broader sense deals with the entire range of information processing, from data retrieval to knowledge retrieval. IR systems allow us to narrow down the set of documents that are relevant to a particular problem. The most well known IR systems are search engines such as Google, which identify those documents on the World Wide Web that are relevant to a set of given words.

B. NATURAL LANGUAGE PROCESSING (NLP) is one of the oldest and most difficult problems in the field of artificial intelligence. It is the analysis of human language so that computers can understand natural languages as humans do. Although this goal is still some way off, NLP can perform some types of analysis with a high degree of success. Shallow parsers identify only the main grammatical elements in a sentence, such as noun phrases and verb phrases, whereas deep parsers generate a complete representation of the grammatical structure of a sentence. The role of NLP in text mining is to provide the systems in the information extraction phase (see below) with linguistic data that they need to perform their task. Often this is done by annotating documents with information like sentence boundaries, part-of-speech tags, parsing results, which can then be read by the information extraction tools.

C. DATA MINING (DM) is the process of identifying patterns in large sets of data. The aim is to uncover previously unknown, useful knowledge. When used in text mining, DM is applied to the facts generated by the information extraction phase. We put the results of our DM process into another database that can be queried by the end-user via a suitable graphical interface. The data generated by such queries can also be represented visually.

III. METHODS OF TEXT MINING

Most data-mining research assumes that the information to be "mined" is already in the form of a relational database. Unfortunately, for many applications, available electronic information is in the form of unstructured natural-language documents rather than structured databases. Consequently, the problem of text mining, i.e. discovering useful knowledge from unstructured text, is becoming an increasingly important aspect of KDD. Much of the work in text mining does not exploit any form of natural-language processing (NLP), treating documents as an unordered "bag of words" as is typical in information retrieval. The existing methods for structuring collections either try to assign keywords to documents based on a given keyword set (classification or categorization methods) or automatically structure document collections to find groups of similar documents (clustering methods).

i. CLASSIFICATION

Text classification aims at assigning pre-defined classes to text documents. An example would be to automatically label each incoming news story with a topic like "sports", "politics", or "art". Whatever the specific method employed, a data mining classification task starts with a *training set D* = (d_1,..., d_n) of documents that are already labelled with a class $L \in$L (e.g. sport, politics). The task is then to determine a *classification model*

$$f{:}D \rightarrow \mathrm{L} \qquad f(d) = L$$

which is able to assign the correct class to a new document *d* of the domain. To measure the performance of a classification model a random fraction of the labelled documents is set aside and not used for training. We may classify the documents of this test set with the classification model and compare the estimated labels with the true labels. The fraction of correctly classified documents in relation to the total number of documents is called *accuracy* and is a first performance measure. Often, however, the target class covers only a small percentage of the documents. Then we get a high accuracy if we assign each document to the alternative class. To avoid this effect different measures of classification success are often used. *Precision* quantifies the fraction of retrieved documents that are in fact relevant, i.e. belong to the target class. *Recall* indicates which fraction of the relevant documents is retrieved.

$$\text{Precision} = \frac{\#\{\text{relevant} \cap \text{retrieved}\}}{\#\text{retrieved}}$$

$$\text{Recall} = \frac{\#\{\text{relevant} \cap \text{retrieved}\}}{\#\text{relevant}}$$

Obviously there is a trade off between precision and recall. Most classifiers internally determine some "degree of membership" in the target class. If only documents of high degree are assigned to the target class, the precision is high. However, many relevant documents might have been overlooked, which corresponds to a low recall. When on the other hand the search is more exhaustive, recall increases and precision goes down. The *F*-score is a compromise of both for measuring the overall performance of classifiers.

$$F = \frac{2}{1/\text{Recall} + 1/\text{Precision}}$$

P is inversely related to R (unless additional knowledge is given)

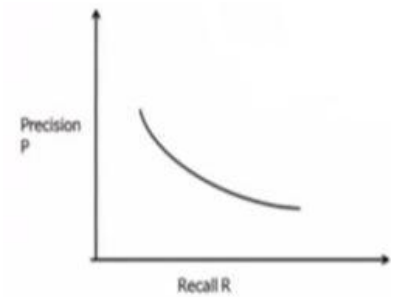

Fig 1. Relation between P & R

ii. THE BOOLEAN MODEL

The Boolean model is a simple model based on set theory in that both the documents to be searched and the user's query are conceived as sets of terms. Retrieval is based on whether or not the documents contain the query terms. Given a finite set

$T = \{t_1, t_2, ..., t_j, ..., t_m\}$

of elements called index terms (e.g. words or expressions - which may be stemmed describing or characterising documents such as keywords given for a journal article), a finite set

D = {D1,..., Di, ..., Dn}, where Di is an element of set called documents. Given a Boolean expression in a normal form Q called a query as follows:

Q = (Wi OR Wk OR ...) AND ... AND (Wj OR Ws OR ...),

with

Wi=ti, Wk=tk, Wj=tj, Ws=ts, or Wi=NON ti, Wk=NON tk, Wj=NON tj, Ws=NON ts

where ti means that the term ti is present in document Di, whereas NON ti means that it is not.

Equivalently, Q can be given in a disjunctive normal form, too. An operation called retrieval, consisting of two steps, is defined as follows:

1. The sets Sj of documents are obtained that contain or not term tj (depending on whether Wj=tj or Wj=NON tj):

Sj = {Di|Wj element of Di}

2. Those documents are retrieved in response to Q which are the result of the corresponding sets operations, i.e. the answer to Q is as follows:

UNION (INTERSECTION Sj)

iii. DRAWBACKS OF BOOLEAN MODEL

- Retrieval based on binary decision criteria with no notion of partial matching.
- No ranking of documents is provided i.e. absence of grading scale.
- Information need has to be translated in to Boolean expression which most users find awkward.
- The Boolean queries formulated by the users are most often too simplistic.
- As a consequence, Boolean model frequently returns either too few or too many documents in response to a user query.

IV. NAIVE BAYES CLASSIFIER

Probabilistic classifiers start with the assumption that the words of a document $d_i$have been generated by a probabilistic mechanism. It is supposed that the class $L(d_i)$ of document d_i has some relation to the words which appear in the document. This may be described by the conditional distribution $p\ (t_1,..., t_{ni}/L(d_i))$ of the $n_i$words given the class. Then the *Bayesian formula* yields the probability of a class given the words of a document.

$$p(L_c/t_1,... , t_{ni}) = \frac{p(t_1,... , t_{ni}/L_c)p(L_c)}{\sum_{L\in L} p\ (t_1,..., t_{ni}/L)p(L)}$$

Note that each document is assumed to belong to exactly one of the k classes in L. The prior probability $p(L)$ denotes the probability that an arbitrary document belongs to class L before its words are known. Often the prior probabilities of all classes may be taken to be equal. The conditional probability on the left is the desired *posterior probability* that the document with words $t_1,..., t_{ni}$belongs to class L_c. We may assign the class with highest posterior probability to our document. For document classification it turned out that the specific order of the words in a document is not very important. Even more we may assume that for documents of a given class a word appears in the document irrespective of the presence of other words. This leads to a simple formula for the conditional probability of words given a class L_c

$$p(t_1,... , t_{ni}/L_c) = \prod_{j=1}^{n_i} p(t_j/L_c)$$

Combining this "naive" independence assumption with the Bayes formula defines the *Naive Bayes classifier*. Simplifications of this sort are required as many thousand different words occur in a corpus. The naive Bayes classifier involves a learning step which simply requires the estimation of the probabilities of words $p(tj/Lc)$ in each class by its relative frequencies in the documents of a training set which are labelled with *Lc*. In the classification step

the estimated probabilities are used to classify a new instance according to the Bayes rule. In order to reduce the number of probabilities $p\ (tj\ /Lm)$ to be estimated, we can use index term selection methods Although this model is unrealistic due to its restrictive independence assumption it yields surprisingly good classifications. It may be extended into several directions. As the effort for manually labelling the documents of the training set is high, some authors use unlabeled documents for training. Assume that from a small training set it has been established that word $t_i$is highly correlated with class *Lc*. If from unlabeled documents it may be determined that word $t_j$is highly correlated with t_i, then also $t_j$is a good predictor for class *Lc*. In this way unlabeled documents may improve classification performance. In the recent past

authors used a combination ofExpectation-Maximization (EM) and a naive Bayes classifier and were able to reduce the classification error by up to 30%

V. NEAREST NEIGHBOUR CLASSIFIER

What is called supervised learning is the most fundamental task in machine learning. In supervised learning, we have training examples and test examples. A training example is an ordered pair (x, y) where 'x' is an instance and 'y' is a label.

A test example is an instance 'x' with unknown label. The goal is to predict labels for test examples. The name "supervised" comes from the fact that some supervisor or teacher has provided explicit labels for training examples.

Assume that instances 'x' are members of the set X, while labels y are members of the set Y. Then a classifier is any function f: $X \rightarrow Y$. A supervised learning algorithm is not a classifier. Instead, it is an algorithm whose output is a classifier. Mathematically, a supervised learning algorithm is a higher-order function: it is a function of type $(X \times Y)^n \rightarrow (X \rightarrow Y)$ where n is the cardinality of the training set.

The nearest neighbour method is perhaps the simplest of all algorithms for predicting the class of a test example. The training phase is trivial, simply store every training example with its label. To make a prediction for a test example, first compute its distance to every training example. Then keep the k closest training examples, where $k \geq 1$ is a fixed integer. Look for the label that is most common among these examples. This label is the prediction for this test example. Using the same set notation as above, the nearest-neighbour method is a function of type $(X \times Y)^n \times X \rightarrow Y$. A distance function has type $X \times X \rightarrow R$.

This basic method is called the kNN algorithm. There are two major designs

choices to make: the value of k, and the distance function to use. When there are two alternative classes, in order to avoid ties the most common choice for k is a small odd integer, for example k = 3. Nearest neighbour classification is a nonparametric method and it can be shown that for large data sets the error rate of the 1-nearest neighbour classifier is never larger than twice the optimal error rate .Several studies have shown that *k*-nearest neighbour methods have very good performance in practice . Their drawback is the computational effort during classification, where basically the similarity of a document with respect to all other documents of a training set has to be determined.

VI. DECISION TREE

Decision trees are classifiers which consist of a set of rules which are applied in a sequential way and finally yield a decision. Decision Trees are a non-parametric supervised learning method used for *classification* .The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. It uses a divide and conquer strategy. The decision tree has three types of nodes.

- A root node that has no incoming edges and zero or more outgoing edges.
- Internal nodes each of which has exactly one incoming edge and two or more outgoing edges.
- Leaf or terminal nodes each of which has exactly one incoming edge and no outgoing edge.

In a decision tree each leaf node is assigned a class label. The non terminal nodes, which include the root and other internal nodes, contain attribute test conditions to separate records that have different characteristics i.e a tree can be "learned" by splitting the source set into subsets based on an attribute value test and information gain which is a good measure for deciding the relevance of an attribute. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node has all the same value of the target variable, or when splitting no longer adds value to the predictions.

Decision trees are standard tools in data mining .They are fast and scalable both in the number of variables and the size of the training set. It is also possible to validate the model using statistical tests and enhances the reliability of the model. For text mining, however, they have the drawback that the final decision depends only on relatively few terms. A decisive improvement may be achieved by boosting decision trees i.e. determining a set of complementary decision trees constructed in such a way that the overall error is reduced.

VII. SUPPORT VECTOR MACHINES AND KERNEL METHODS

SVMs are a class of algorithms that combine the principles of statistical learning theory with optimisation techniques and the idea of a kernel mapping. They were introduced by Boser et al. (1992), and in their simplest version they learn a separating hyperplane between two sets of points so as to maximise the margin (distance between plane and closest point). A single SVM can only separate two classes—a positive class $L1$ (indicated by $y = +1$) and a negative class $L2$ (indicated by $y = ¡1$). Given a set of training data $\{x_1, x_2, \ldots, x_l\}$ in some space R^p and their labels $\{y_1, y_2, \ldots, y_l\}$, where $y_i \in \{-1, +1\}$, estimate a prediction function 'f' such that it can classify an unseen data point x.The SVM learning algorithm aims to find a linear function of the form

$$f(x) = w^T x + b,$$

with $w \in R^p$ and $b \in R$ such that data point 'x'is assigned to a label +1 iff(x) > 0, and a label −1 otherwise.

The SVM algorithm determines a hyperplane which is located between the positive and negative examples of the training set. The parameters are adapted in such a way that the distance ξ– called *margin* – between the hyperplane and the closest positive and negative example documents is maximized. This amounts to a constrained quadratic optimization problem which can be solved efficiently for a large number of input vectors. The documents having distance ξ from the hyperplane are called *support vectors* and determine the actual location of the hyperplane.

SVMs can be used to solve various real world problems:

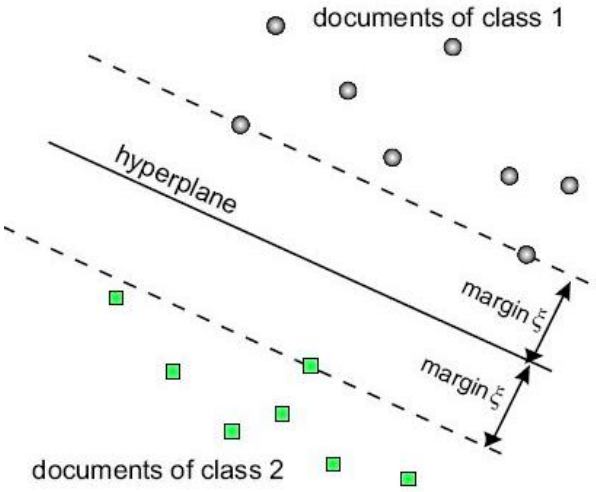

Fig ii). Support Vector machines

- SVMs are helpful in text and hypertext categorization as their application can significantly reduce the need for labelled training instances in both the standard inductive and transductive settings.
- Classification of images can also be performed using SVMs. Experimental results show that SVMs achieve significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback.
- SVMs are also useful in medical science to classify proteins with up to 90% of the compounds classified correctly.

But SVM are sensitive to outliers and noises and has limited interoperability. Another problem with SVM is that it has high computational complexity also the feature selection an important method in the high vector space text is not realized. Motivated by above problems a new method named Support Vector Machines with mixture of Kernel methods (SVM-KM) is used to classify the text. In this method the Kernel is a convex combination of many finite basic Kernels. Each basic Kernel has a Kernel coefficient and is provided with a single feature. As a result its objective function turns into a linear programming parameter iterative learning procedure and greatly reduces computational complexity.

VIII. CLASSIFIER EVALUATIONS

During the last years text classifiers have been evaluated on a number of benchmark document collections. It turns out that the level of performance of course depends on the document collection. Concerning the relative quality of classifier SVMs, and k-nearest neighbours usually deliver top-notch performance, while naive Bayes and decision trees are less reliable.

METHOD	F1-VALUE
Naive Bayes	0.795
Decision tree	0.794
K-Nearest Neighbour	0.856
SVM	0.870

Performance of Different Classifiers for the Reuters collection

IX. CLUSTERING

Clustering method can be used in order to find groups of documents with similar content. The result of clustering is typically a set of clusters *P*. Each cluster consists of a number of documents *d*. Objects in our case documents. Each cluster consists of objects that are similar between themselves and dissimilar to objects of other clusters.

I. Clustering Approaches:- Hierarchical clustering also known as Connectivity based clustering is based on the core idea of objects being more related to nearby objects than to objects farther away. These algorithms connect "objects" to form "clusters" based on their distance. A cluster can be described largely by the maximum distance needed to connect parts of the cluster. At different distances, different clusters will form, which can be represented using a dendrogram, which explains where the common name "hierarchical clustering" comes from: these algorithms do not provide a single partitioning of the data set, but instead provide an extensive hierarchy of clusters that merge with each other at certain

distances. In a dendrogram, the y-axis marks the distance at which the clusters merge, while the objects are placed along the x-axis such that the clusters don't mix. Strategies for hierarchical clustering generally fall into two types:

II. Agglomerative: This is a "bottom up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.

III. Divisive: This is a "top down" approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

The agglomerative procedure considers initially each document *d* of the whole document set *D* as an individual cluster. It is the first cluster solution. It is assumed that each document is member of exactly one cluster. One determines the similarity between the clusters on the basis of this first clustering and selects the two clusters *p, q* of the clustering P with the minimum distance *dist* (*p, q*). Both clusters are merged and one receives a new clustering. One continues this procedure and re-calculates the distances between the new clusters in order to join again the two clusters with the minimum distance dist (p, q). The algorithm stops if only one cluster is remaining. A divisive clustering starts with one cluster of all data points and recursively splits the most appropriate cluster. The process continues until a stopping criterion (frequently, the requested number k of clusters) is achieved. Connectivity based clustering is a whole family of methods that differ by the way distances are computed. Apart from the usual choice of distance functions, the user also needs to decide on the linkage criterion (since a cluster consists of multiple objects, there are multiple candidates to compute the distance to) to use. Popular choices are known as single-linkage clustering (the minimum of object distances) and complete linkage clustering (the maximum of object distances).

X. K-MEANS

Often referred to as Lloyd's algorithm. In basic terms, the algorithm has three steps. The first step chooses the initial centroids, with the most basic method being to choose k samples from the dataset X. After initialization, K-means consists of looping between the two other steps. The first step assigns each sample to its nearest centroid. The second step creates new centroids by taking the mean value of all of the samples assigned to each previous centroid. The difference between the old and the new centroids are computed and the algorithm repeats these last two steps until this value is less than a threshold. In other words, it repeats until the centroids do not move significantly.

1. Choose a value for *K*, the total number of clusters.
2. Randomly choose *K* points as cluster centers.
3. Assign the remaining instances to their closest cluster center.
4. Calculate a new cluster center for each cluster.
5. Repeat steps 3-5 until the cluster centers do not change.

I. Bi-Section-*k*-means one fast text clustering algorithm, which is also able to deal with the large size of the textual data is the Bi-Section-*k*-means algorithm. Bi-Section-*k*-means is a fast and high-quality clustering algorithm for text documents which is frequently outperforming standard *k*-means as well as agglomerative clustering techniques.
Bi-Section-*k*-means is based on the *k*-means algorithm. It repeatedly splits the
largest cluster (using *k*-means) until the desired number of clusters is obtained. Another way of choosing the next cluster to be split is picking the one with the largest variance.

II. Clustering Using the EM-Algorithm Clustering can also be viewed from a statistical point of view. If we have *k* different clusters we may either assign a document d_i with certainty to a cluster (hard clustering) or assign d_i with probability q_{ic} to P_c(soft clustering), where $q_i = (q_{i1},...,q_{ik})$ is a probability vector $\sum_{c=1}^{k} q_{ic} = 1$.Each cluster $P_c$is a mixture component. The mixture probabilities q_c describe an unobservable "cluster variable" *z* which may take the values from {1,..., k}.
It follows an iterative approach which tries to find the parameters of the probability distribution that has the maximum likelihood of its attributes. In general lines, the algorithm's input are the data set, the total number of clusters (k), the accepted error to converge (e) and the maximum number of iterations. For each iteration, first is executed the E-Step (Expectation), that estimates the probability of each point belongs to each cluster, followed by the M-step (Maximization), that re-estimate the parameter vector of the probability distribution of each class. The algorithm finishes when the distribution parameters converge or reach the maximum number of iterations.

III. Information Extraction (IE)
Natural language text contains much information that is not directly suitable for automatic analysis by a computer. However, computers can be used to sift through large amounts of text and extract useful information from single words, phrases or passages. Therefore information extraction can be regarded as a restricted form of full natural language understanding, where we know in advance what kind of semantic information we are looking for. The main task is to extract parts of text and assign specific attributes to it.
A typical IE system has basic phases for input tokenization, lexical analysis, name entity recognition, syntactical analysis, and identifying the interesting information required in a particular application . Applying information extraction on text is linked to the problem of text simplification in order to create a structured view of the information present in free text.

The overall goal being to create a more easily machine readable text to process the sentences. Depending on the particular requirements of the application, IE systems may also include other modules.

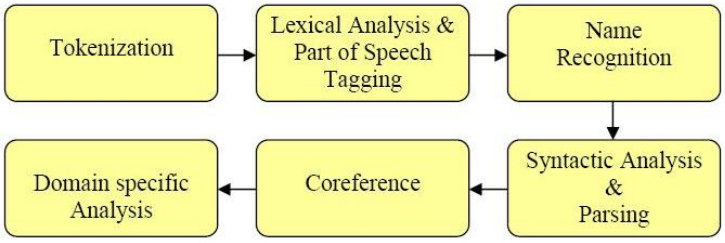

Fig iii) Modules for a Typical IE System

IV. Conclusion

The continuous expansion of textual data has led to the need for text mining techniques and methodologies in order to better study and exploit the content oriented relations between text documents. In this article, we tried to give a brief introduction to the broad field of text mining and presented a brief overview of currently available text mining methods. Taking into account the language the text is written in is important since the language highlights the morphological or syntactic analysis needed. Text mining generally consists of the analysis of (multiple) text documents by extracting key phrases, concepts, etc. and the preparation of the text processed in that manner for further analyses with numeric data mining techniques. NLP interacts with text mining, measurable results though are needed so as to find out which NLP techniques can be applied to what text mining applications. Classifying a text collection into categories may enable the text processing. In contrast to classification, which relies on a pre specified grouping, clustering procedures label documents in a new way by studying the words and phrases that characterize a cluster. New, previously unknown knowledge can also be identified by studying semantic relations between information stored in databases and the existing literature.

References

[1] Michael W. Berry and Malu Castellanos, Editors "Survey of Text Mining : Clustering, Classification, and Retrieval, Second Edition" Springer,September 30, 2007.
[2] J. Han and M. Kamber, "Data mining: concepts and Techniques", Second edition.
[3] K. Nigam, A. K. McCallum, S. Thrun and T. Mitchell, "Text classification from labelledand unlabeled documents using EM," Mach. Learn., vol. 39, pp. 103–134, May 2000
[4] T. Joachims. A Statistical Learning Model of Text Classification for Support Vector Machines. ACM SIGIR Conference, 2001.
[5] Charu C. Aggarwal and Cheng.x.Zhai , "A survey of classification algorithms", pp.164-168,167-181
[6] Y. Li, A. Jain. Classification of text documents. The Computer Journal, 41(8), pp. 537–546, 1998.
[7] E. Youn, M. K. Jeong , "Class dependent feature scaling method using naive Bayes classifier for text datamining" Pattern Recognition Letters , 2009
[8] Sarika Y. Pabalkar ," Web Text Mining for news by Classification ", IJARCCE ,2012
[9] Tam, v., Santaso and sationo, R, "A comparative study of centroid based, neighbourhood based and statistical approaches for effective document categorization" proceedings of 16th international conference on pattern recognition 2002. http://www.seasr.org/documentation/text-mining-overview/
[10] Alex Markov and mark Last, "A simple structure sensitive approach for web document classification", Springer AWIC 2005, pp. 293-298, 2005.
[11] Sourav Sahay, Support vector Machines and document classification",
[12] URL:http://www.static.cc.gatech.edu/souravsahay7001-2.pdf
[13] Yang Y. and Liu X., 1999. A Re-examination of Text Categorization Methods [A]. In: Proceedings of 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval [C]. 42-49. http://www.site.uottawa.ca/~diana/csi4107/L8.pdf
[14] Yang Y., Zhang J. and Kisiel B, "A scalability analysis of classifiers in text categorization," ACM SIGIR'03, 2003.
[15] Jason Kroll, "Decision Tree Learning for Arbitrary Text Classification," Sept 2003
[16] Y. H. Li and A. K. Jain, "Classification of Text Documents", THE COMPUTER JOURNAL, Vol. 41, No. 8, 1998
[17] Li Baoli1, Yu Shiwen1, and Lu Qin2 ," An Improved k-Nearest Neighbour Algorithm
[18] for Text Categorization", 20th International Conference on Computer Processing of Oriental Languages, Shenyang, China, 2003.
[19] Quinlan, J. R. (1986). Induction of decision trees. Machine Leaning, vol (1), pp.81-106
[20] Wenliang Du and Zhijun Zhan ," Building Decision Tree Classifier on Private Data" , 2002
[21] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In C. Nedellec and C. Rouveirol, editors, European Conf. on Machine Learning (ECML), 1998

