CHAPTER 2

Review of Literature
Chapter 2
Review of Literature

2.1. Introduction

According to Descombe, the purpose of literature review is “to avoid repeating research that has been already conducted and to provide a foundation on which to build new research” (158)\(^1\). This chapter enlists through an analysis of various research articles and text books related to the problems of Text Mining and the Text categorization. Text Mining falls under the broader area called Data Mining. In the Text Mining one of the interesting problems found is Text categorization. In general the Text categorization problem has two stages. They are the preprocessing and the text categorization stages. The preprocessing text data includes cleaning the text, organizing the text and selecting the important or relevant features in the text. Hence the review of literature is divided into three parts. They are namely Text Mining, Preprocessing and Categorization of the Text.

2.2. Text Mining

Text Mining can be said as cleaning, structuring, organizing the text in a proper format and extracting relevant information from the text. According to Thomas W. Miller (2008) Text Mining is the automated or partially automated processing of text. It involves imposing structure upon text and extracting relevant information from the text. Miller adds the Text Mining involves non-trivial tasks like classifying text documents, analyzing syntax, identifying relationships among documents, understanding a question

expressed in natural language, extracting meaning from a message and summarizing the
meaning. All these things specified by Thomas W Miller indicates the role of text mining.

During the text mining the important issues are structuring of the text and
information retrieval in the text data bases. Jiawei Han and Micheline Kamber (2003) says
that most text databases are semistructured data in that they are neither completely
unstructured nor completely structured. Moreover, information retrieval techniques, such
as text indexing methods, have been developed to handle the unstructured documents.
Also Han et al indicate that information retrieval is concerned with the organization and
retrieval of information from a large number of text based documents. A typical
information retrieval problem is to locate relevant documents based on user input such as
keywords or example documents. Han et al outline that text mining involves the tasks of
structuring the text and information retrieval.

In his Ph.D. proposal Un Yong Nahm indicates that text mining (2001) is a
relatively new research area at the intersection of data mining, natural language processing,
machine learning, and information retrieval. The goal of Text Mining is to discover
knowledge in the unstructured text. The related task of Information Extraction (IE)
concerns locating specific items of data in natural language documents, thereby
transforming unstructured text into a structured database. Also UN Yong Nahm says that
traditional data mining assumes that the information to be “mined” is already in the form
of a relational database. Unfortunately, for many applications, electronic information is
only available in the form of unstructured natural – language documents rather than
structured databases. Apart from these IE and semi structured databases, the Yong Nahm
mentions in his thesis that Web Mining is a specialized extension for text mining to the
semi-structured texts available on the Internet. Apart from these things this thesis presents a new frame work of Text Mining called DISCOTEX (Discovery from Text Extraction). An alternative rule induction system called TEXTRISE is being developed.

In another paper titled “A Mutually Beneficial Integration of Data Mining and Information Extraction” (cf. Un Yong Nahm and Raymond J Mooney, 2001) Un Yong Nahm have explored the mutual benefit that the integration of IE and KDD can provide. In this paper also the authors emphasizes that Natural – language information extraction methods can transform a corpus of textual documents into a mere structured database.

It is very interesting to note the relationship of other fields with text mining or text data mining. It is easy to identify that text mining is associated with the words, of a particular language. Going one step ahead Marti. A. Hearst says (1999) corpus based computational linguistics is a field engaged in Text data mining. Empirical computational linguistics computes statistics over large text collections in order to discover useful patterns.

More importantly Marti A. Hearst states that Text categorization cannot be considered as Text Data Mining because Text categorization is a boiling down of the specific content of a document into one of a set of pre-defined labels. This does not lead to a discovery of new information. However, later on he quotes one work that uses text category labels (associated with Reuters news wire) to find “unexpected patterns” among text articles. So finally it can be understood that Text Categorization is a concept closer to Text data mining.

Oren Etzioni in his discussion about World Wide Web says that web mining is the use of data mining techniques to automatically discover and extract information from
World Wide Web documents and services. Going by Oren Etzioni’s statement it can be understood that web mining is nothing but text mining done on web documents. (1996). He also adds that Data Mining has been applied to databases traditionally, yet much of the information on the web lies buried in documents designed for human consumption such as home pages as product catalogs. Furthermore, much of the information on the web is presented in natural language text with no machine – readable semantics; HTML annotation structure the display of web pages, but provides little insight into their content. Thus according to Etzioni, the complications involved in handling text available on the web are different from the normal text data. But still web mining very closely related to the text mining. Otherwise it can be said that web mining is a special case of text data mining.

Generally Data Mining can be considered as knowledge Dictionary in Databases (KDD) (cf. Roven Feldman et al, 1998). Also the KDD is mostly concerned with structured databases. There is very little work done on handling huge amount of information that is available only in unstructured textual form. The keyword – frequency approach supports a range of KDD operations and provides a suitable foundation for knowledge discovery and exploration for collections of unstructured data. A system called knowledge Discovery for Text (KDT) discovers patterns or trends from the existing information on the documents.

In this section various ideas on text mining and its related fields were discussed. The major problem of the text mining, i.e., unstructured data, was highlighted here. The problem of Information extraction from the unstructured data was addressed here. A related field of Text mining called as web mining was explained and its close
relationship with Text Mining was also discussed. A system used for identifying patterns in text information called as knowledge Discovery for Text (KDT) was also mentioned here.

2.3. Preprocessing

This section talks about the pre-processing work for the text categorization work. In the previous section itself it was seen that the text was available in the unstructured format. Hence these texts have to be cleaned and converted to a form that can be readily processed.

The pre-processing for a text categorization involves word extraction, stop words removal, word stemming, fixing weights to terms and feature selection. Word extraction is all about identifying the words in text documents which is stored as a stream of characters. One that is over stop words removal should be performed. Stop words are those words which does not contribute anything towards the text categorization process. But these words may play a role in a sentence formation practically. Examples of stop words are ‘is’, ‘was’, ‘are’, ‘he’, ‘she’, ‘have’, ‘the’, ‘in’ etc., These stop words are removed from the document to make the categorization task easier. Next comes the word stemming. Generally in a text words may be present in various forms. But they do have a basic form. This basic form is called as canonical form. For example “seeing”, “seen”, “saw” are the various forms of the basic form or canonical form of “to see” or “see”. Now these words “seeing”, “seen”, “saw” have to be connected to the basic form “see”. (cf. Lam Lai Yin and Dominic Savio, 1996). This process is called stemming. After all these tasks, fixing weights to the terms or words in a document takes place. To do this stems for all the remaining words are selected. If there are any duplicates for a stem they are also removed.
from the document. After the removal of duplicate stems each stem becomes unique in the
document. Next this set of stems, constitute the indexing terms for the set of documents.
This set of indexing terms is called as indexing vocabulary. Now comes the task of
assigning weights to these indexing terms. Before that the reason for fixing weights to the
terms should be understood.

Assigning weights to term in a document indicate the relative importance of term in
the document. Suppose there are two indexing terms are present in the group of
documents. One term is assigned ‘0’ and another tem is assigned ‘1’. The second term
with weight ‘1’ is considered to be more important term than the first term having weight
‘0’. By saying important, a term is considered to be very frequently appearing in a
particular document in the set of documents and the very frequently appearing term will be
helpful in identifying the category of the document very much. The set of indexing terms
in the indexing vocabulary and their corresponding term weights, a document can be
represented as the following feature vector:

\[ D_j = < w_{j1}, w_{j2}, \ldots, w_{jk} > \]

Where \( K \) is the number of indexing terms in the vocabulary, or the vocabulary size,
and \( W_{ji} \) is the term weight of the \( i^{th} \) indexing term in document \( j \).

The term weighting is the key to retrieval effectiveness (cf. Gerard Salton,
Christopher Buckley, 1988). The effective retrieval is said to be dependent on two main
factors one, items likely to be relevant to the user’s need must be retrieved; two, items
likely to be extraneous must be rejected. To state it simply the relevant features or term
should be retrieved and irrelevant features should be rejected. There are two measures to
assess the ability of the system to retrieve the relevant feature and reject the irrelevant
features. They are recall and precision. Recall is the proportion of relevant features retrieved, measured by the ratio of the number of relevant features in the collection. The precision is the proportion of retrieved items that are relevant, measured by the ratio of the number of relevant retrieved features to the total number of retrieved items.

Recall and precision are very useful measures to assess the performance in retrieving or identifying relevant features in a document and rejecting the irrelevant features. A preferred system produces high recall by retrieving everything that is relevant and also high precision by rejecting all the features that are not relevant. The recall function of retrieval appears to be best served by using broad, high frequency terms that occurs in many documents of collection. Such terms are very helpful in selecting many documents, including many relevant documents. The precision factor may be best served by using narrow, highly specific terms that are capable of isolating the few relevant features from the collection of irrelevant ones. To make a compromise between these two measures the terms that are broad enough to achieve reasonable recall level and do not produce a unreasonably low precision are selected.

Due to the differing recall and precision requirements, it favours the use of composite term weighting factors that contain both recall and precision – enhancing components. Three main considerations look very important with reference to this issue. First, the terms that are frequently mentioned in individual documents, or document excerpts, appear to be useful as recall measure. This gives a term frequency (tf) (cf. Karen Sparck Jones, 1972) factor be used as part of the term-weighting system measuring the frequency of occurrence of the terms in the document or query texts.
The term frequency weights have been used for many years in automatic indexing environments.

Second, term frequency factors alone cannot ensure acceptable retrieval performance. Specifically when the high frequency terms are not concentrated in a few particular documents, but instead are prevalent in the whole collection, all documents tend to be retrieved. This affects the precision of the search. Hence a new collection dependent factor must be introduced that favors terms concentrated in a few documents of a collection. The popular inverse document frequency (idf) (or inverse collection frequency) factor performs this function. The idf factor varies inversely with the number of documents \( n \) to which a term is assigned in a collection of \( N \) documents. The idf factor may be computed as \( \log (N/n) \).

The important aspect that should be considered here is term discrimination. This aspect suggests that the best terms for document content identification are those able to distinguish certain individual documents from the remainder of the collection. This says that the best terms should have high term frequencies but low overall collection frequencies. A reasonable measure of term importance may then be obtained by using the product of the term frequency and the inverse document frequency (\( tf \times idf \)).

A third factor in term-weighting, in addition to the term frequency and the inverse document frequency, appears useful in systems with widely varying sector lengths. In many cases, short documents tend to be represented by short-term vectors, whereas much larger – term sets are assigned to the longer documents. When large number of terms are used for document representation, the chance of term matches between queries and documents is high, and hence larger documents have a better chance of being retrieved.
than the short ones. Generally, all the relevant documents should be treated as equally important for retrieval purposes. This indicates that a normalization factor be introduced into the term – weighting formula to equalize the length of the document vectors. Assuming that \( w \) represents the weight of term \( t \), the final term weight might be defined as

\[
\frac{w}{\sum_{vector} i \ w_i}
\]

(or)

\[
\frac{w}{\sqrt{\sum_{vector} i \ w_i^2}}
\]

By the above discussion it can be noted that term weighting is an important issue in the pre-processing step of Text Categorization. Using the Recall and Precision measures to measure the performance of extracting relevant data and term weighting, form the central aspect of this step.

After the discussion on term weighting it will be very essential to note that there is another way of looking at the term weighting. It is said that commonly used feature weighting methods only consider the distribution of a feature in the documents and do not consider the class distribution of a feature/ term in the documents and do not consider the class information for feature or term weighting. Mutual Information (MI) is a method which represent the dependency of a feature in the regarding class, (cf. Morteza Zahedi et.al, 2011) has been used for feature selection. It is argued that use of MI method for feature weighting increases the performance of text classification in terms of average recall and average precision.
The term class refers to the particular domain to which a particular document is classified after the end of text classification process. Suppose there exists a class called ‘sports’ then all the documents relevant to sport will be classified to this class. Now coming to the MI method, it is a method mostly used in statistical approaches. MI can be effectively used as a feature weighting tool. The MI value of a feature in a class represents the dependency of that feature in the regarding class, thus indicating the importance of the feature/ term in that class. Hence to measure the fitness of a feature in a particular class, the MI of that feature will be calculated for that class during the feature selection phase. This value is later on used as the weight of the feature/ term.

MI method considers the distribution of the features in different classes while weighting each feature in each class. In this method the value of each feature in each class is calculated and this value is used for feature weighting based on class dependency. Thus MI method is a recently discovered method for term as feature weighting that tries to include the class information too. But at the same time it should be noted that TF – IDF method too provide very good results in the practical contexts (cf. G. Salton et al, 1973).

When a discussion on term weighting or feature weighting is taking place two concepts require a mention. They are indexing exhaustivity and term specificity. Indexing exhaustivity refers to the accuracy and depth with which the various topic areas relevant to a given document are reflected in the set of index terms assigned to the document, whereas term specificity is a function of the exactness with which a term characterizes a given subject. Generally, increasing exhaustivity implies a better recall performance, while increasing term specificity means better precision. In particular, the more exhaustive the indexing, that is, the more thorough the coverage of the various subject areas, the more
likely it is that relevant items are actually retrieved in response to user queries, thus achieving high recall; similarly, the greater the term specificity, that is, the more precise the definition of each term, the less likely it is that extraneous non-relevant items are also retrieved. G. Salton and C.S. Young says that exhaustivity will ensure the retrieval of relevant terms or features and high term specifying will eliminate the irrelevant terms or features.

In a real working environment, one may consider that indexing exhaustivity has something to do with the number of index terms assigned to given document, particularly the number of higher frequency terms – those largely responsible for the recall performance. The term specificity, on the other hand may be assumed to be related to the number of documents to which a given term is assigned in a given collection. The idea being that the smaller the document frequency, that is, the more concentrated the assignment of a term to only a few documents in a collection, the more likely it is that a given term is reasonably specific.

Discussing about exhaustivity and term specificity, Karen Sparck Jones (1972) says that exhaustivity is a property of index descriptions, and specificity one of index terms. They are most clearly illustrated by a simple keyword or descriptor system. In this case the exhaustivity of a document description is the coverage of its various topics given by the terms assigned to it; and the specificity of an individual term is the level of detail at which a given concept is represented.

Karen Sparck Jones adds more about the exhaustivity and term specificity as follows: If the exhaustivity of a document description is increased by the assignment of
more terms, when the number of terms in the indexing vocabulary is constant, the chance of the document matching a request is created.

Specificity is a semantic property of index terms: a term is more or less specific as its meaning is more or less detailed and precise. This is a natural view for anyone concerned with the construction of an entire indexing vocabulary. Some decision has to be made about the discriminating power of individual terms in addition to their descriptive propriety. For example, the index term “beverage” may be as properly used for documents about tea, coffee, and coca as the terms “tea”, “coffee” and “cocoa”. Whether the more general term “beverage” only is incorporated in the vocabulary, or whether “tea”, “coffee”, and “cocoa” are adopted, depends on judgments about the retrieval utility of distinction between documents made by the later but not of the former. It is also predicted that the more general term would be applied to more documents than the separate term “tea”, “coffee”, and “cocoa”, to the less specific term would have a larger collection distribution than the more specific ones.

It is of course assumed that such choices when a vocabulary is constructed are exclusive: it may be either “beverage” or “tea”, “coffee”, or “cocoa”. If all the four terms are existing that is a different matter. Then “beverage” may be interpreted to mean “other beverages” or explicitly treat it as a related broader term.

The Karen Sparck Jones argument with the “beverage” example clearly states that less specific terms or low specificity terms will retrieve more documents. Hence the rejection of non-relevant documents will also be lower.

Pum – Mo Ryu and Key-Sun choi (2004) define specificity as it is the quantity of domain specific information contained in the term. Specific terms have larger quantity of
domain information than general terms. Some terms have a relatively large quantity of
domain information, and others have a relatively small quantity of domain information.
Ryu and Choi conducted a similar study of the use of frequency in determining specificity
on multiword medical terms, resting around 436 disease names, measuring at in 1,70,000
abstracts (120 MB of text) taken from Medline collection. They found that df determined
specificity with an accuracy of 60.6%.

Generally there exists an assumption that document frequency represents a level of
term, specificity. However, empirical results to support this assumption are limited.
The results show that (cf. Hideo Joho et.al, 2007) the assumption holds only at the very
specific levels that cover the majority of vocabulary. The results also show that a larger
corpus is more accurate at estimating specificity. Hideo Joho and Mark Sanderson started
with the assumption that all 45,000 norms and noun phrases in Word Net would be used in
their experiments. Different corpora were examined for their coverage of the Word Net
terms. The corpora used were large fragment of the TREC collection (the financial times,
Google.

Considering the analysis done by Hideo Joho and Mark Sanderson the results show
that when the document frequency is used as a means of generating a concept hierarchy,
the statistics obtained from a larger collection is likely to provide a better result than a
smaller set. A wider range of vocabulary will also be found in the larger collection.

2.3.1. Feature Selection

Now comes the feature of term selection process. After the stop words removal,
word stemming and Term/Feature weighting, feature selection is the last of the pre
processing step for text classification. The term/feature selection refers to the selection of the appropriate features that should be introduced in the analysis in order to maximize (A. Salappa et al., 2006) the performance of the resulting model. This has significant implications for issues such as: (a) noise reduction through the elimination of noisy features; (b) reduction of the computational effort required to develop and implement an appropriate model; (c) simplification of the resulting models; and (d) facilitation of the easy use and updating of the model.

Feature selection (FS) is usually performed as a pre-processing phase prior to model development, using special algorithms. The development of FS algorithms has been an active research topic in data mining and machine learning. FS algorithms are computational procedures, which are used to select a set of features that optimized an evaluation measure representing the quality of features. The researches in this field have mainly focused on algorithmic developments, experimental evaluations and real-world applications. However, the previous studies done on this field concentrated on a limited number of algorithms and a limited number of methods.

In the feature selection process the important aspect is the identification of relevant features and non-relevant features. In the real world situations, relevant features are often called as unknown apriori. Therefore many candidate features are introduced to better represent (M. Dash et al., 1997) the domain. But many of these candidate features are either partially or completely irrelevant or redundant to the target concept. To state precisely, a relevant feature is neither irrelevant nor redundant to the target concept. Also a non-relevant feature does not affect the target concept in anyway, and a redundant feature does not add anything new to target concept.
In most of the applications, the size of a dataset is very large that learning might not work as well before removing these unwanted features. Reducing the number of irrelevant or redundant features drastically reduces the running time of a learning algorithm and gives a more general concept. This provides in getting a better insight into the underlying concept of a real–world classification problem. In general, the objective of the FS methods is to try and pick a subset of features that are relevant to the target concept.

Feature selection is perceived and defined by various authors in different angles. But mostly those definitions are similar in intuition or content. The following is the list of definitions that are conceptually different and cover a range of definitions.

1. Idealized: find the minimally sized feature sublet that is necessary and sufficient to the target concept.

2. Classical: select a subset of M features from a set of N features, M<N, such that the value of a criterion function is optimized over all lab subset of size M.

3. Improving prediction accuracy: the aim of feature selection is to choose a sublet of features for improving prediction accuracy or decreasing the size of the structure without significantly decreasing prediction accuracy of the classifier built using only the selected features.

4. Approximating original class distribution: the goal of feature selection is to select a small subset such that the resulting class distribution, given only the values for the selected features, is as close as possible to the original class distribution given all feature values.

In the four definitions specified above, the third definition stresses about the prediction accuracy of the classifier, built using only the selected features. The last
definition talks about the class distribution provided the training dataset. These two differ conceptually. Hence M.Dash and H.Liu state that the definition of feature selection considers both these factors and give the following definition:

Feature selection attempts to select the minimally sized subset of features according to the following criteria. The criteria can be:

1. The classification accuracy does not significantly decrease; and
2. The resulting class distribution, given only the values for the selected features, is as close as possible to the original class distribution given all features.

The definition goes like this. To be more precise, feature selection methods search through the subset of features, and try to find the best one among the competing \(2^N\) candidate subsets according to some evaluation function. But this procedure is exhaustive as it tries to find only the best one. It may be too costly and practically prohibitive, even for a medium – sized feature set size (N). Other methods based on heuristic or random search methods attempt to reduce computational complexity by compromising performance. These methods need to stopping criterion to prevent an exhaustive search of subsets. According to M.Dash and H.Liu, (1997) there are four basic steps in a typical feature selection method.

1. A generation procedure to generate the next candidate subset.
2. An evaluation function to evaluate the subset under examination.
3. A stopping criterion to decide when to stop; and
4. A validation procedure to check whether the subset is valid.

The generation procedure mentioned here is a search procedure. Basically, it generates subsets of features for evaluation. The generation procedure can be initiated:
(i) with no features, (ii) with all features, or (iii) with a random subset of features. In these first two cases, features are iteratively added or removed, whereas in the last case, features are either iteratively added or removed or produced randomly thereafter. The role of an evaluation function is to measure the goodness of a subset produced by some generation procedure, and this value is compared with the previous best. If it is found to be better, then it replaces the previous best subject.

![Diagram of Feature Selection process with validation]

**Fig. 2.1: Feature Selection process with validation**

There should be always a stopping criterion to stop the selection criterion, without which FS may run exhaustive or forever through the space of subsets. Generation procedures and evaluation functions can influence the choice for a stopping criterion. Stopping criteria created on the basis of Generation procedure include:

i) Whether a predefined number of features are selected, and (ii) whether a pre-defined number of iterations are reached. Stopping criteria based on an evaluation function can be: (i) whether addition (or deletion) of any feature does not produce a better subset; and (ii) whether an optional subset according to some evaluation function is
obtained. The loop continues until some stopping criterion is fulfilled. Then the FS process halts by outputting a selected subset of features to a validation procedure. The validation tries to test the validity of the selected subset by carrying out different tests, and comparing the results with previously established results, or with the results of competing feature selection methods using artificial datasets, real – world datasets, or both.

In the past there have been quite a few attempts to study feature selection methods based on some framework or structure. Prominent among these are Doak’s (1987) and siedlecki and sklansky(1992) surveys. Siedlecki and sklansky discussed the evolution of FS methods and grouped the methods into past, present, and future categories. Their main focus was the branch and bound methods and its variants. Their survey was published in the year 1987, and since then many new efficient methods have been introduced (Eg. Focus, Relief, LVF). Doak followed a similar approach to siedlecki and sklansky’s survey and grouped the different search algorithms and evaluation function used in FS methods independently, and ran experiments using some combinations of evaluation functions and search procedures.

When the FS is considered, the two important steps, namely, generation procedure and evaluation function should be discussed. It is very useful to categories the generation procedures and evaluation functions. A framework of evaluation functions is presented in which a total of 32 methods are grouped based on the types of generation procedure and evaluation function used in them.

**a) Generation Procedures**

If the original feature set contained N number of features, then the total number of competing candidate subsets to be generated is \(2^N\). This will be very large even if the
N-value is medium – size. There are different approaches for solving this problem, namely complete, heuristic, and random.

b) Complete

This is one type of generation procedure that does a complete search for the optimal subset according to the evaluation function used. Complete represents the exhaustive search. But, according schlimmer (1987), completeness of search does not mean that it must be exhaustive. Different heuristic functions are used to reduce the search without jeopardizing the chances of finding the optimal subset. Hence although the order of search space is $O(2^N)$, a fewer subsets are evaluated. According to the evaluation function the optimality of the feature subset is assured because of the backtrack procedure. Backtracking can be done using various techniques, such as branch and bound, best first search and beam search.

c) Heuristic

In a generation procedure for each iteration, all remaining features yet to be selected or rejected are considered for selection/rejection. Many variations to this simple process exist, but generation of subsets is basically incremental (either increasing/decreasing). The order of the search space $O(N^2)$ or less; some exceptions are Relief DTM. These procedures are very simple to implement and very fast in producing results, because the search space is only quadratic in terms of the number of features.

d) Random

The Random generation procedure is rather new in its use in feature selection methods compared to the other two categories. Even if the search space is $O(2^N)$, these methods typically search less number of subsets than $2^N$ by setting a maximum number of
iterations possible. Optimality of selected subset depends on the source available. Each random generation procedure would require values of some parameters. Assignment of suitable values to this parameter is an important talk for getting good results.

e) Evaluation Functions

An optional subset is always relative to a certain evaluation function. In other words an optional subset chosen using one evaluation function may not be the same as that which uses another evaluation function. The role of an evaluation function is to measure the discriminating ability of a feature or a subset to distinguish the different class labels. Langley grouped different feature selection methods into two broad (i.e. filter and wrapper) based on these dependence on the inductive algorithm that will finally use the selected subset. Filter methods are independent of inductive algorithm that will finally use the selected subset. The filter methods are independent of the inductive algorithm, whereas the wrapper methods use the inductive algorithm, as the evaluation function. Ben-Bassat (1982) grouped the evaluation functions existing until 1982 into three categories: information or uncertainty measures, distance measures, and dependence measures, and suggested that the dependence measures can be divided between the first two categories. In 1982 there was no existing function and hence he has not considered the classification error rate as an existing evaluation function. Doak divided the evaluation functions into three categories; data intrinsic measures, classification error rate, and estimated or incremental error rate, where the third category is basically a variation of the second category. The data intrinsic category includes distance, entropy, and dependence measures. Considering these dimensions the evaluation functions can be divided into five categories: distance, information or uncertainty, dependence, consistency, and classifier
error rate. In the following subsections each of this type of evaluation functions are discussed.

f) **Distance Measures**

   It is also known as separability divergence, or discrimination measure. For a two class problem, a feature $X$ is preferred to another feature $Y$ if $X$ induces a greater difference between the two-class conditional probabilities than $Y$: if the difference is Zero, then $X$ and $Y$ are indistinguishable. Euclidean distance measure is an example.

g) **Information Measure**

   Information gain from a feature is determined by these measures. The information gain from a feature $X$ is defined as the difference between the prior uncertainty and expected posterior uncertainty using $X$. Feature $X$ is preferred to feature $Y$ if the information gain from feature $X$ is greater than that from feature $Y$. Entropy measure is an example for this.

h) **Dependence Measure**

   Dependence measures or correlation measures qualify the ability to predict the value of one variable from the value of another. The co-efficient is a classical dependence measure and can be used to find the correlation between a feature and a class. If the correlation of feature $X$ with class $C$ is higher than the correlation of feature $Y$ with $C$, then feature $X$ is preferred to $Y$. A slight variation of this is to determine the dependence of a feature on other features; this value indicates the degree of redundancy of the feature. All evaluation functions based on dependence measures can be divided between distance and information measures. But these are kept as a separate category as they represent a different point of view.
Table 2.1: Details on Evaluation Functions:

<table>
<thead>
<tr>
<th>Evaluation Function</th>
<th>Generality</th>
<th>Time Complexity</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance Measure</td>
<td>Yes</td>
<td>Low</td>
<td>--</td>
</tr>
<tr>
<td>Information Measure</td>
<td>Yes</td>
<td>Low</td>
<td>--</td>
</tr>
<tr>
<td>Dependence Measure</td>
<td>Yes</td>
<td>Low</td>
<td>--</td>
</tr>
<tr>
<td>Consistency Measure</td>
<td>Yes</td>
<td>Moderate</td>
<td>--</td>
</tr>
<tr>
<td>Classifier Error rate</td>
<td>Yes</td>
<td>High</td>
<td>Very High</td>
</tr>
</tbody>
</table>

i) Consistency Measures

The consistency measures have come into existence very recently. These are characteristically different from other measures, because of their heavy reliance on the training dataset and the use of the Min – Features bias in selecting a subset of features. As in features bias prefers consistent hypothesis definable over as few features as possible. These measures find out the minimally sized subset that satisfies that acceptable inconsistency rate that is usually set by the user.

j) Classifier Error Rate Measures

The methods using this type of evaluation function are called “wrapper methods”. In other words the classifier is the evaluation function here. The features are selected by the classifier and later on these features used by the classifier in predicting the class labels of unseen instances. Though the accuracy level is very high for these methods, the computation is quite costly.

The above table 2.1 shows a comparison of various evaluation functions irrespective of the type of generation procedure used. The different parameters used for the comparison are:
1. Generally: How appropriate is the selected subset for the different classifiers.

2. Time complexity: Time taken for selecting the subset of features; and

3. Accuracy: How accurate is the prediction using the selected subset.

The discussion on the various features selection methods was presented above. This discussion highlighted the way the methods are categorized and the measures used in the method to measure the efficiency of the feature selection process.

The important aspect in the feature selection process is the identification of the relevant and irrelevant features. The relevant features should be accepted and irrelevant should be discarded. But the important question is how to identify the relevant features. For that the relevance of a feature should be defined clearly.

Authors George H. John, Ron Kohavi, Karl P Fleger define the relevance of the features (cf. George H. John et.al, 1994). Initially, according to them two degrees of relevance are needed. They are weak and strong.

The input to a supervised learning algorithm is a set of N training instances. Each instance X is an element of the set F1 × F2 × ……. × Fn, where Fi is the domain of the ith feature. Training instances are tuples (X, Y) where Y is the label, or output. Given an instance the value of the feature Xi is denoted by Xi. The task of the induction algorithm is to induce a structure (example, a decision tree or a neural network) such that, given a new instance, it is possible to accurately predict the label Y. The probability measure p on the space F1 × F2 × ……. × Fn × Y. This discussion does not make any assumptions on the features or on the label; they can be discrete continuous linear, or structured and the label may be single – valued or a multi-valued vector of arbitrary dimension.
The relevance can be defined in the following ways:

Definition 1: A feature $X_i$ is said to be relevant to a concept $C$ if $X_i$ appears in every Boolean formula that represents $C$ and irrelevant otherwise.

Definition 2: $X_i$ is relevant iff there exists some $X_i$ and $Y$ for which $p(X_i = x_i) > 0$ such that $p(Y = y / X_i = x_i) \neq p(Y=y)$

In this definition, $X_i$ is relevant if knowing its value can change the estimates for $Y$, or in other words, if $Y$ is conditionally dependent of $X_i$. Note that this definition fails to capture the relevance of features in the parity concept, and may be changed like this.

Let $S_i$ be the set of all features except $X_i$, i.e. $S_i = (X_1, \ldots X_{i-1}, \ldots, X_{i+1}, \ldots X_m)$.

Denote $S_i$ value – assignment to all features $S_i$.

Definition 3 : $X_i$ is relevant iff there exists some $X_i$, $y$, and $S_i$ for which $p(X_i = x_i) > 0$ such that $p(Y=y, S_i = s_i / X_i = x_i) \neq p(Y = y, S_i = s_i)$.

In this definition, $X_i$ is relevant if the probability of the label (given all features) can change when the knowledge about the $X_i$ is eliminated.

Definition 4 : $X_i$ is relevant iff there exists some $x_i$, $y$, and $s_i$ for which $P(X_i = x_i, S_i = s_i) \neq P(Y=y / S_i = s_i)$

With the help of correlated XOR these definitions can be explained as below. Let the features $X_1 \ldots X_5$ be Boolean. The instance space is such that $X_2$ and $X_3$ are negatively correlated with $X_4$ and $X_5$, respectively, i.e., $X_4 = \overline{X_2}$, $X_5 = \overline{X_3}$. There are only eight possible instances, and it is assumed that they are equiprobable. The (deterministic) target concept is

$Y = X_1 \oplus X_2$ \quad ($\oplus$ denotes XOR)
Note that the target concept has an equivalent Boolean expression, namely $Y = X_1 \oplus X_2$. The features $X_3$ and $X_5$ are irrelevant in the strongest possible sense. $X_1$ is indispensable, and one of $X_2, \ldots, X_4$ can be disposed of, but one of them should be present.

Table 2.2 show that for each definition, the relevant and the irrelevant features.

**Table 2.2 : Feature relevance for the correlated XOR problem under four definitions.**

<table>
<thead>
<tr>
<th>Definition</th>
<th>Relevant</th>
<th>Irrelevant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition 1</td>
<td>$X_2$</td>
<td>$X_2, X_3, X_4, X_5$</td>
</tr>
<tr>
<td>Definition 2</td>
<td>None</td>
<td>All</td>
</tr>
<tr>
<td>Definition 3</td>
<td>All</td>
<td>None</td>
</tr>
<tr>
<td>Definition 4</td>
<td>$X_1$</td>
<td>$X_2, X_3, X_4, X_5$</td>
</tr>
</tbody>
</table>

According to definition 1, $X_3$ and $X_5$ are clearly irrelevant; both $X_2$ and $X_4$ are irrelevant because each can be replaced by the negation of other. By definition 2, all features are irrelevant since for any output value $y$ and feature $x$, there are two instances that agree with the values. By definition 3, every feature is irrelevant, because knowing its value changes the probability of four of the eight possible instances from $1/8$ to Zero. By definition 4, $X_3$ and $X_5$ are clearly irrelevant, and both $X_2$ and $X_4$ are irrelevant, since they do not add any information to $S_4$ and $S_2$, respectively.

GH. John et al mention about strong the weak relevance. The strong relevance implies that the feature is indispensable in the sense that it cannot be removed without loss of prediction accuracy.

Definition on weak relevance: A feature $X_i$ is weakly relevant iff it is not strongly relevant, and there exists a subset of features $S_i$ of $S_i$ for which there exists some $X_i, y_i$, and $S_i$ with $P(\overline{X_i} = x_i, S_2^{\perp} = S_i^{\perp}) > 0$ such that
\[ p\left( Y = y / X_i = x_i, S_i^1 = s_i^1 \right) \neq p(Y = y / S_i^1 = s_i^1) \]

Weak relevance implies that the features can sometimes contribute to prediction accuracy. Features are relevant if they are either strongly or weakly relevant, and are irrelevant otherwise.

Irrelevant features can much contribute to prediction accuracy, by definition.

In the earlier discussion (Correlated XOR), feature \( X_1 \), is strongly relevant; features \( X_2 \) and \( X_4 \) are weakly relevant; and \( X_3 \) and \( X_5 \) are irrelevant.

According to GH John et.al, algorithms such as FOCUS find a minimal set of features that are sufficient to determine the concept. When sufficient data is provided, these algorithms will select all strongly relevant features, none of the irrelevant ones, and a smallest subset of the weakly relevant features that are sufficient to determine the concept. Algorithms such as relief attempt to efficiently approximate the relevant features.

Now after discussing the theoretical background of the feature selection it is important to look into the algorithmic part of it. There are a number of approaches to feature/term selection process. The two major approaches of feature selection are the filter model and wrapper model.

In the filter model three instances are considered there. They are FOCUS, Relief and the method used by Cardie (cf. Claire Cardie, 1993). The focus algorithm originally defined by noise – free Boolean domains, exhaustively examines all subsets of features, selecting the minimal subset of features that is sufficient to determine the label. This is referred as MIN-FEATURES bias. This bias has severe implications when applied blindly without regard for the resulting induced concept. For example,
In a medical diagnosis task, a set of features describing a patient might include the patient’s social security number (SSN). It is assumed, that features other than SSN are sufficient to determine the correct diagnosis. When FOCUS searched for the minimum set of features, it will pick the SSN as the only feature needed to uniquely determine the label. Any Induction algorithm with generalize very poorly if only the SSN is given.

The relief algorithm assigns a “relevance” weight for each feature, which is meant to denote the relevance of the feature, to the target concept. Relief is a Randomized algorithm. It picks up sample instances randomly from the training set and updates the relevance values based on the difference between the selected instance and the two nearest instances of the same and opposite class (the “near-hit” and “near-miss”).

The Relief algorithm does not attempt to determine useful subsets of the weakly relevant features. According to Kenji Kira and Larry A. Rendell (1992), Relief algorithm will not help with redundant features. Also most of the given features are relevant to the concept, it would select most of them even though only a fraction is necessary for concept description.

In the real domains, many features have high correlations, and this many are weakly relevant, and will not be removed by Relief.

The figure 2.2 of feature filter mode, characterizes these algorithms. In this model, the feature subset selection is done as a pre-processing step. The disadvantage of the filter
approach is that it totally ignores the effects of the selected feature subset on the performance of the induction algorithm.

G.H. John et al claim that to determine a useful subset of features, the subset selection algorithm must take into account the biases of the induction algorithm in order to select a subset that will ultimately result in an induced structure with high predictive accuracy on unseen data. Hence the possibility of using a new approach called wrapper model comes into feature selection process.

i) The Wrapper Model:

![Figure 2.3 The Wrapper Model](image)

In the wrapper model, the feature subset selection algorithm exists as a wrapper around the induction algorithm. This is represented in the above figure 2.3. The feature subset algorithm conducts a search for a good subset using the induction algorithm itself as part of the evaluation function.

Given a subset of features, the accuracy of the induced structure using only the given feature is estimated. This feature is estimated for the evaluation process of the subset using n-fold, cross validation. The training data is split into n approximately equal sized partitions. The induction algorithm is the run n times, each time using n-1 partitions as the
training set and the other partition as the test set. The accuracy of the results from each of
the n-runs are then averaged to produce the estimated accuracy.

As the heuristic search most suited for the most machine language algorithms, it is
used here. One simple greedy algorithm, called backward elimination, starts with the full
set of features, and greedily removes the one that most improves performance, or degrades
performance slightly. A similar algorithm called forward selection starts with the empty
set of features, and greedily adds features.

The algorithm can be improved by considering both addition of a feature and
deletion of a feature at each step. For example, during backward elimination, consider
adding one of the deleted features if it improves performance. Thus at each step the
algorithm greedily either adds or deletes. The only difference between the backward and
forward versions is that the backward version starts with no features. The algorithms are
straight forward and are described in many statistics books under the names backward
stepwise elimination and forward stepwise selection. One has to be careful to set the
degradation and improvement marginal so that cycles will not occur.

The above heuristics increase the overall running time of the black-box induction
algorithm by a multiplicative factor of $O(m^2)$ in the worst case, where $n$ is the number of
features, while this may be impractical on some issues, it does not depend on $n$, the number
of instances. As noted in Cohen W.W (1993) divide and conquer need much more time for
pruning than for growing the structure (by a factor of $O(n^2)$ for random data). By pruning
after the feature subset selection, pruning may be faster.

The discussion on feature selection exposed various things like exact definition of
the feature selection, various measures used and the algorithms under the categories of
wrapper model and filter model. Based on these things the feature selection process becomes very important in making an effective text classification. That is the foremost reason for elaborately discussing about the feature selection process in this thesis. After all the pre-processing steps the actual problem of Text categorization is considered. According to Lam Lai Yin and Dominic Savio, (1996) Text Categorization is the classification of text documents into a set of one or more categories. In Text categorization, the categories that the documents are to be classified into pre-defined classes, typically by the designer or maintainer of the categorization system. The end users are usually not involved in the process of defining categories. Nevertheless, once the documents are classified into categories, users can identify a number of categories that may contain documents relevant to his needs, and ignore documents in categories that are unlikely to be relevant. By doing this, the scope of information that the user has to search is largely reduced, which in turn speeds up the process of finding relevant information for the user. In fact, a typical application of text categorization systems is to limit the scope of searching for text retrieval systems. In addition to specifying a query, the user can limit the scope of search by specifying a set of categories to be searched.

Based on the set of categories, there may be overlaps between different categories, in which a document can belong to more than one of the categories. In some cases, some documents may belong to none of the defined categories. Based on the Categorization system, these documents may be removed or put into a special “unclassified” category. Like user profiles in text filtering systems, the set of categories is usually unchanged over a long period of time. In this situation it is essential to understand the concept of text filtering. Text filtering can be described as the selective dissemination of textual
information. Like text retrieval, text filtering involves the selection of a subset from the available text documents, according to the information needs of the user.

Similar to that filtering, text categorization systems are typically required to process a stream of incoming new documents, sometimes in real time. During the categorization process, documents are assigned zero, one, or more categories. The processed documents are then stored in document database together with the list of category labels they are assigned to. As in text retrieval, the membership of a document to a category can be binary, in which a document is either assigned or not assigned a category. In the case of graded categorization each document is given here indicates its relevance to each of the categories. If binary membership is to be obtained from a graded categorization system, a threshold can be set so that a category is assigned to a document only if the relevance score corresponding to the category is above the threshold.

2.4. Text Categorization

According to Thorsten Joachims (1998) the goal of text categorization is the classification of documents into a fixed number of predefined categories. Each document can be in multiple, exactly one, or no category at all. Using machine learning, the objective is to learn classifiers from examples which perform the category assignment automatically. This is a supervised learning problem. Since categories may overlap, each category is treated as a separate binary classification problems.

The first step in text categorization is to transform documents, which typically are strings of characters, into a representation suitable for the learning algorithm and for the classification task. Information Retrieval research suggests that word stems work well as representation units and that their ordering in a document is of minor importance for many
tasks. This leads to an attribute – value representation of text. Each distinct word \( w_i \) corresponds to a feature, with the number of times word \( w_i \) occurs in the document as its value. To avoid unnecessarily large feature vectors, words are considered as features only if they occur in the training data at least 3 times and if they are not “stop records”.

Fuchun Peng and Xiangji Huang (cf. Fuchun Peng et.al, 2006) consider that Text classification addresses the problem of assigning a given passage of text to one or more predefined classes. This is an important area of information retrieval research that has been heavily investigated, although most of the research activity has concentrated on English Text.

Text classification is the problem of assigning a document \( D \) to one of a set of \(|C|\) pre-defined categories. \( C = \{C_1, C_2, \ldots \ldots \ldots C_{|C|}\} \). Normally a supervised learning framework is used to train a text classifier, where a learning algorithm is provided as a set of N labeled training examples \( \{(d_i, C_i) : i = 1, \ldots, N\} \) from which it must produce a classification function \( F : D \rightarrow C \) that maps documents to categories. Here \( d_i \) denotes the \( i^{th} \) training document and \( C_i \) is the corresponding category label of \( d_i \). The random variable \( D \) and \( C \) are used to denote the document and category values respectively. A probabilistic text classifier is formulated as the following decision problem: given a document \( d \), determine the class label \( C^* \) that yields the highest posterior probability \( P(C=C/D=d) \) (written \( P(c/d) \) for simplicity):

\[
C^* = \arg \max_{c \in C} \{P(c|d)\}
\]

and text classifiers that we w
Text categorization can also be considered as a process of classifying documents with regard to a group of one or more existent categories according to themes or concepts present in their contents (cf. Renato Fernandes Correa, et. al, 2002). The most common application of it is in Information Retrieval System (IRS) to document indexing. The organization of text in categories allow the user to limit the target of a search submitted to IRS, to explore the collection and to find relevant information they need with prior knowledge about the keywords of a theme. A method to transform text categorization into a viable task is to use machine learning algorithms to automate text classification, allowing it to be carried out in a fast, in a concise manner and in a broad range.

The effectiveness of the Text classification is determined by the algorithm used. Hence it becomes very important to analyze the various algorithms in the Text classification. But before discussing algorithms it is very essential to understand the role of classifiers in the Text categorization algorithm.

A classifier is a function that maps on (Susan Dumais et al, 1998) input attribute vector, $\vec{x} = (x_1, x_2, x_3,...x_n)$, to a confidence that input belongs to a class that is $f(\vec{x}) = \text{confidence (class)}$. In the case of text classification, the attributes are words in the document and the classes correspond to text categories.

Example of classifiers for the category cricket includes:

- If (bat and wicket) or (Long on), then confidence (Cricket Category) = 0.9
- Confidence (Cricket Category) = 0.3 $\times$ bat + 0.4 $\times$ wicket + 0.7 $\times$ Long on.

Some of the classifiers are probabilistic in the sense that confidence (class) is a probability distribution.
2.4.1. Naive – Bayes Algorithm

In simple terms, a naïve Bayes classifier or simple Bayesian classification assumes that the presence of a particular feature of class is unrelated to the presence of any other feature, given the class variable. For example a bicycle will be considered a two wheeler if it has two wheels, one handle bar, two brakes. Even if these features depend on each other or upon the existence of the other features, a naïve Bayes classifier considers all of these properties to independently contribute to the probability that this two wheeler is our bicycle.

Depending on the precise nature of the probability model, naive bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical application, parameter estimation for naive Bayes models use the method of maximum likelihood; in other words, one can work with the naive Bayes model without believing in Bayesian probability or using any Bayesian methods.

In spite of their simple design and apparently over simplified assumptions, naive Bayes classifiers have worked quite well in many complex real – world situations. In 2004, analysis of the Bayesian classification problem has shown that there are some theoretical reasons for the apparently unreasonable efficacy of naive Bayes classifiers. Still a comprehensive comparison with other classification methods in 2006 showed that Bayes classification is outperformed by more current approaches, such as boosted trees of random forests.

An advantage of the naive Bayes classifier is that it only required a small amount of training after to estimate the parameters necessary for classification. Because independent
variables are assumed, only the variances of the variables for each class need to be determined and not the entire covariance matrix.

It should be noted (cf. Susan Dumais, 1998) that a naive – 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. Bayes theorem is used to estimate probabilities.

\[ P(C=C_k / \theta) = \frac{P(\theta / C=C_k) P(C=C_k)}{P(\theta)} \]

The quantity \( P(\theta / C=C_k) \) is often impractical to compute without simplifying assumption. For the Naive Bayes classifier, it is assumed that given the category variable \( C \), the features \( X_1, \ldots, X_n \) are conditionally independent. This simplifies the computation yielding:

\[ P(\theta / C=C_k) = \prod_{i} P(\theta / C=c_k) \]

Despite the fact the assumption of conditional independence is generally not true for word appearance in documents; the Naive Bayer classifier is effective.

According to Fuchung Peng et.al (2006) Naive Bayes classifiers is a simple yet effective learning algorithm for text classification which is based on a simple application of Bayes’ Rule:

\[ P(c|d) = \frac{P(c) \times P(d|c)}{P(d)} \quad \ldots(A) \]

In text classification, a document \( d \) is normally represented by a vector of \( K \) attributes \( d = (v_1, v_2, \ldots, v_k) \) computing \( P(d|c) \) in this case is not trivial, since the space of possible documents \( d = (v_1, v_2, \ldots, v_k) \) is vast. To simplify this computation, the naive
Bayes model introduces an additional assumption that the entire attribute values, \( v_j \), are independent given the category label, \( C \). That is for \( i \neq j \), \( v_i \) and \( v_j \) are conditionally independent of the given \( C \). This assumption greatly simplifies the computation by reducing (A) to:

\[
P(c|d) = P(c) \times \frac{\prod_{j=1}^{\ell} P(v_j|c)}{P(d)} \quad \text{(B)}
\]

Based on equation (B), a maximum a posteriori (MAP) classifier can be constructed by seeking the optimal category which maximizes the posterior \( P(c|d) \).

\[
c^* = \arg \max_{c \in C} \left\{ P(c) \times \prod_{j=1}^{\ell} P(v_j|c) \right\} \quad \text{(C)}
\]

\[
= \arg \max_{c \in C} \left\{ \prod_{j=1}^{\ell} P(v_j|c) \right\} \quad \text{(D)}
\]

Equation (D) is called the maximum likelihood naïve Bayes classifier, obtained by assuming a uniform prior over categories. There are several variants of the naïve Bayes classifier, including the binary independence model, the multinomial model, the Poisson model, and the negative binary independence model. It has been shown that for text categorization applications, the multinomial model is most often the best choice.

The naïve Bayes classifier is not without the shortcomings. Based on the statement of Kamal Nigam and Andrew McCallum (1998), when naïve Bayes is given just a small set of labeled training data, classification accuracy suffers because of the parameter estimates of the generative model. But still the naïve Bayes algorithm can be considered as an effective fundamental algorithm on text classification.
2.4.2. Back Propagation Network Algorithm

After the naive Bayes algorithm the next significant algorithm of Text classification will be back propagation Neural Network (BPN) algorithm. But before going into the BPN algorithm it is essential to have a little bit of discussion on artificial Neural Network as it forms the background of the algorithm.

An artificial neural network consists of a large number of simple processing units, which are often referred to as neurons. Neurons are connected to form a network by weighted connections, or synapses, through which neurons can communicate by sending signals to each other. (cf. Lam Lai Yin, Dominic Savio, 1996). By local computations performed within each neuron and the signals sent between then through the connections, the neural network as a whole performs computation in a highly parallel and distributed manner.

In many networks, the neurons are arranged in a number of layers. Neural Networks having more than one layer of neurons are sometimes called multi-layer neural networks. Following figure shows an artificial neural network with three layers of neurons.

![Artificial Neural Network Diagram](image)

*Figure 2.4 An artificial neural network.*

*(Courtesy: Lam Lai Yin, Dominic Savio, 1996)*
Based on the way the neuron layers are connected together in a neural network, or the network topology, various neural networks can be broadly classified two classes: feed forward networks and recurrent networks.

In a multi-layer feed forward neural network, the direction of the signal flow through the connections is strictly feed – forward, from one layer to a following layer. Although the signal flow can continue across multiple layers of neurons, there must not exist any feedback connections, going from one layer to a preceding layer. Moreover, there should not be any intra-layer feedback connections, going from one layer to a preceding layer. Also, there should not be any intra-layer connections between the neurons of the same layers. Since loops are not possible in a feed forward neural network, the number of times the signals will flow in a network is finite, limited by the number of layers or depth of the network. After these finite numbers of steps of signal flow, all the activation values in the network will be known.

In a feed –forward network, there is usually a layer called the input layer, which accepts external signals as input to the neural network. Also there is one other layer called output layer in which the activation values of the neurons in this layer will be taken as output of the neural network. The layers in between the input layer and the output layer are sometimes called hidden layers. Consequently, neurons in the input layer are called input units, while those in the output layer and hidden layer are called output units and hidden units, respectively. Perceptrons, adaline and non-linear feed-forward networks trained by Back propagation are some of the illustrations of feed forward neural networks.

Another type of neural networks is called as recurrent neural networks. These recurrent neural networks contain feedback connections. There are many types of
recurrent neural networks proposed in the theory, including the well known model proposed by Hopfield, often called as Hopfield network.

Unlike feed forward networks, there are loops in the recurrent networks. Because of this, signal flow may continue to any number of steps. In this case, the network is said to undergo a relaxation process until the activation values converge to a stable state. In some cases, this stable state may not be reached and the network will continue to change its activation values without becoming stable. When the network does become stable, the set of activation values at the stable state represents the output of the network.

In the BPN a three layer feed-forward network is used as the text classifier. This network consists of an input layer, a hidden layer, and an output layer. All neurons in the neural network are non-linear units with the sigmoid function as the activation function. Based on these things BPN algorithm is found to be very satisfactory when applied to the Test classification problem. Generally, BPN has been one of the most studied (cf. R.Rojas, 1996) and used algorithms for neural networks learning ever since.

The back propagation algorithm looks for the minimum error function in weight space using the method of gradient descent. The combination if weights which minimize the error function is considered to be a solution of learning problem. Since this method required computation of the gradient of the error function at each iteration step, the continuity and differentiability of the error function should be guaranteed. Obviously an activation functions is used one of the more popular activation functions for back propagation networks is the sigmoid, a real function $s_c : \mathbb{R} \rightarrow (0,1)$ defined by the expression.

$$s_c(X) = \frac{1}{1+e^{-cx}}$$
The constant $x$ can be selected arbitrarily and its reciprocal $1/x$ is called the temperature parameter in stochastic neural networks. The activation function plays an important role in the working of BPN algorithm.

According to R. Rojas a feed – forward neural network is a computational graph whose nodes are computing units and whose directed edges transmit numerical information from node to node. Each computing unit is capable of evaluating a single primitive function of its input. In fact the network represents a chain of function compositions which transform an input to an output vector called pattern. The network is a particular implementation of a composite function from input to output space, which is called as network function. The learning problem consists of finding the optimal combination of weights so that the network function $\mu$ approximates a given function $f$ as closely as possible.

Consider a feed – forward network with $n$ input and $m$ output units. It can consist of any number of hidden units and can exhibit any desired feed forward connection pattern. The given training set is $\{(x_1, t_1), \ldots, (x_p, t_p)\}$ consisting of $p$ ordered pairs of $n$- and $m$-dimensional vectors, which are called the input and output patterns. The primitive functions at each node of the network are considered as continuous and differentiable. The weights of the edges are real numbers selected at random. When the input pattern $X_i$ from the training set is presented to this network, it produced an output $O_i$ and $t_i$ identical for $i=1, \ldots, p$, by using a learning algorithm. More precisely error function of the network should be minimized. It is defined as

$$E = \frac{1}{2} \sum_{i=1}^{p} \| o_i - t_i \|^2.$$
After minimizing this function for the training set, new unknown input pattern are presented to the network and it is expected to interpolate. The network must recognize whether a new input vector is similar to learned patterns and produce a similar output.

The BPN algorithm is used to find a local minimum of the error function. The network is initialized with randomly chosen weights. The gradient of error function is computed and used to correct the initial weights. Then the task is to compute this gradient recursively.

Looking into the various internal details of the algorithm, the BPN is considered as a better option for performing text categorization. Moreover it has a relative success and popularity in other domains. (cf. Lim Lai Yin, Dominic Savio, 1996). In general, major hindrance to the application of neural networks in the text categorization task was the high dimensionality of the feature space needed to represent the textual data. The high dimensionality has been shown to cause scalability problems to neural networks. Finally it can be said that back propagation learning in neural networks was able to give good categorization performance.

2.4.3. Ngrams Algorithm

The next important algorithm that has to be considered in the text categorization is Ngrams algorithm. This method is based on the following:

1. A direct relation “term class”, instead of passing by the two relations: “term document” and “document class”. This will reduce the response time and eliminate the influence of the documents size in the performances of text categorization.
2. A reduction of dimensionality by preserving only the terms that characterize best a class compared to the other classes. This will reduce the capacity of the used memory and neglect the most frequent terms that do not bring any information as well as the terms of weak frequencies (Abdellatif Rahmoun et al, 2007).

An N-gram is a sequence of N consecutive characters. In a text, all the n-grams are located and the frequencies are counted. The space character is replaced by “-“, to facilitate detection. This technique is purely statistical and does not require any knowledge of the document language. Another advantage of the N-grams is the automatic capture of the most frequent roots. The tolerance to spelling mistakes and deformations is also a significant property. Lastly this technique does not need to eliminate stop words or to proceed to the lemmatization, or stemming.

In other words NGrams is based on calculating and comparing profiles of N-gram frequencies (William B Cavnar et al, 1994). First the system is used to compute profiles on training set data that represent the various categories, e.g., language samples or newsgroup content samples. Then the system computes a profile for a particular document that is to be classified. Finally, the system computes a distance measure between the document’s profile and each of the category profiles. The system selects the category whose profile has the smallest distance to the document profile. The profiles involved are quite small, typically 10K bytes for a category training set, and less than 4 K bytes for an individual document. Using the N-gram frequency profiles provides a simple and reliable way to categorize documents in a wide range of classification tasks.

Basically NGrams algorithm uses frequency statistics for Text categorization. Zipf’s Law forms a fundamental of this. Zipf’s law states that “The nth most common
word in a human language text occurs with a frequency inversely proportional to \( n \). The implication of this law is very straightforward. There are always a lot of words which dominates most of the other words of the language in terms of frequency of use. Also Zipf’s Law implies that classifying documents with N-Gram frequency statistics will not be very sensitive to cutting off the distributions at a particular rank. From this, it follows that if documents of the same category are compared then they should have similar N-gram frequency distributions.

N-grams based approach to text categorization is tolerant of textual errors. This method is small, fast and robust. This system worked very well for language classification, achieving in one test a 99.8% correct classification rate on articles written in different languages. The system also worked reasonably well for classifying articles from a number of different computer oriented newsgroups according to subject, achieving as on 80% correct classification rate.

It can be understood that N-gram frequency method provides an inexpensive and highly effective way of classifying documents. This is done by using samples of the desired categories rather than resorting to more complicated and costly methods such as natural language parsing or assembling detailed lexicons. Cavnar mentions in his article that this approach defines a “Categorization by example” method. Also this system is considered resistant to various OCR problems, since it depends on the statistical properties of N-gram occurrences and not on any particular occurrence of a word.

2.4.4. **K-Nearest Neighbourhood based on Simulated Annealing**

Finally the algorithm that requires a significant attention is the K-Nearest Neighbourhood Algorithm based simulated annealing (KNN-SA) K-NN and SA are
different methods combined together to achieve better results in text categorization. In fact, KNN is used broadly in text classification, but it has one deficiency – computational efficiency (cf. Chuan Yao Yan et.al, 2007). Hence simulated annealing method is combined with this algorithm to help find out the expected neighbours. Researchers have found that KNN algorithm achieves very good performance in their experiments on different data sets.

KNN was first proposed the by cover and hart in 1968. The idea of KNN algorithm is simple and straightforward. To classify a new document, the system finds the K nearest neighbours among the training documents. Then it uses the categories of the K-nearest to weight the category of candidates. According to Chuan Yao et.al KNN algorithm is less efficient. 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. Moreover, the performance of this algorithm greatly depends on two factors, that is, a suitable similarity function and an appropriate value for the parameter K.

Due to the computational complexity, KNN is seldom used in the real-time scenario. In order to improve the efficiency of KNN, many methods have been proposed which are divided into two categories. The first one is reducing the number of training sets in fast KNN text classification approach based on pruning the training corpus.

The second is to adopt the fast algorithm with the proof of categorization function. A traditional way to do it is by representing the training set as a tree called KD-tree which stores a set of points in K-dimensional space, K being the number of attributes. This is a
binary tree that divides the input space with a hyper-plane and then splits each partition again, recursively. Then the computation is done within this KD-tree.

The Genetic Algorithm (GA) is also used to help K-nearest neighbourhood classification (cf. Anupam Kumar Nath et.al, 2005).

Finally the heuristic search way called as simulated annealing is used to select the K-nearest neighbors of test instances quickly, rather than calculating distance of all instances in training set. The KNN algorithm works like this: Given a set of labeled prototypes or text categories and a test document to be classified, the K-NN method finds its K nearest neighbours among the training documents. The categories of the K neighbours are used to select the nearest category for the test document: each category gets the sum of votes of all the neighbours belonging to it and that one with the highest score is chosen (cf. Edgardo Ferretti et al, 2005). Other strategies calculate these score taking into account the distances between the K neighbours and the test document or, alternatively, using a similarity measure like the scalar product. In this last strategy, each document is represented through a vector of terms and each category gets a score equal to the same of the similarities between K neighbours and text document.

In the general classification task, a document could belong to more than one category and different strategies should be used to consider this case. A possibility is designing a K-NN classifier for each category in order to decide whether test document belongs to a category or not. The basic assumption made here is that each text is assigned to only one category. The goodness of semantic K-NN was measured determining the error percentage which was obtained categorizing texts for each data set.
An important problem for the text categorization and also in all the information retrieval tasks in general is to relate different words but with the “same” information in order to perform a conceptual or semantic search (i.e., based on the meaning of the words). Therefore it is necessary to consider synonyms and words that refer to the same concept (cf. Paolo Rosso et al, 2004).

Earlier it has been mentioned that simulated Annealing is a heuristic method used in KNN algorithm. Simulate annealing has come out as a result of comparing the Central Constructs of Combinatorial optimization and statistical mechanics (cf. S. Kirkpatrick et al, 1983). According S. Kirkpatrick there is a deep and useful connection between statistical mechanics (the behavior of systems with many degrees of freedom in thermal equilibrium at a finite temperature) and multivariate or combinatorial optimization (finding the minimum of a given function depending on many parameters). A detailed analogy with annealing in solids provides a framework for optimization of the properties of very large and complex systems. This connection to statistical mechanics exposes new information and provides an unfamiliar perspective on traditional optimization problems and methods.

The Chuanyao Yang et al conducted an experiment based on KNN algorithm with simulated annealing for the categorization of text. The dataset used was a Chinese corpus. According to them the KNN-SA algorithm has greatly outperformed the conventional one in computational efficiency. The method KNN-SA should be universally applicable to classification problems for data in other languages.

Before concluding this chapter a comparative table on the experimental results of Naive Bayes, BPN, N-Grams and KNN-SA found in the literature is provided below:
Table 2.3 Details on the classification based on the Algorithms (NB, BPN, N-Grams, KNN-SA)

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Categories</th>
<th># of Documents</th>
<th>Performance (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB</td>
<td>10</td>
<td>9603</td>
<td>81.5</td>
</tr>
<tr>
<td>BPN</td>
<td>10</td>
<td>840</td>
<td>80</td>
</tr>
<tr>
<td>N-Grams</td>
<td>14</td>
<td>3478</td>
<td>99.8</td>
</tr>
<tr>
<td>KNN-SA</td>
<td>10</td>
<td>10,000</td>
<td>65</td>
</tr>
</tbody>
</table>

**NB:** Naive Bayes, **BPN:** Back propagation Network, **KNN-SA:** K-Nearest Neighbourhood with simulated Annealing.

In this chapter survey of literature on various technical aspects and information on Data Mining, Text Mining, the importance of feature selection, other pre-processing tasks and finally text categorization were discussed. These details found in the literature provide a platform for the automatic text categorization problem and helps to understand the problem in hand very clear. More than that this literature survey provides or determines the direction of the research work.