

Design and Investigation of Techniques to Improve Information Retrieval on the Web

THESIS

submitted in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

RAJENDRA KUMAR ROUL

2009PHXF0421G

under the supervision of

Dr. Sanjay Kumar Sahay



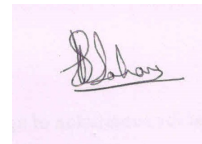
BITS Pilani
Pilani | Dubai | Goa | Hyderabad

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
PILANI (RAJASTHAN), INDIA, 2016

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN)**

CERTIFICATE

This is to certify that the thesis entitled **Design and Investigation of Techniques to Improve Information Retrieval on the Web** submitted by **Rajendra Kumar Roul**, ID No. **2009PHXF0421G** for award of Ph.D. of the Institute, embodies original work done by him under my supervision.

A handwritten signature in black ink on a light pink background. The signature appears to be 'Sanjay Sahay' written in a cursive style.

Signature of the Supervisor

Dr. SANJAY KUMAR SAHAY
Assistant Professor
Department of Computer Science &
Information Systems

Date: 23/11/2016

ACKNOWLEDGEMENT

I would like to thank my Supervisor and Guide **Dr. Sanjay Kumar Sahay** who encouraged and directed me towards doing research. He supported me throughout my thesis with his patience and knowledge whilst allowing me the room to work in my own way. It is his initiation and motivation that this work came into existence and I am greatly indebted to him. One simply could not wish for a better or friendlier supervisor.

I also take this opportunity to thank **Prof. G Raghurama**, Director, Birla Institute of Technology and Science Pilani (BITS Pilani) K K Birla Goa Campus, **Prof. Ashwin Srivivasan**, Deputy Director, BITS Pilani K K Birla Goa Campus for giving me the opportunity to pursue my thesis.

My sincere thanks to **Dr. Bharat M. Deshpande**, Head, Department of Computer Science and Information Systems, **Dr. Neena Goveas**, Department of Computer Science and Information Systems, **Prof. Santonu Sarkar**, Department of Computer Science and Information Systems and **Dr. Prasanta Kumar Das**, Associate Dean of Academic Research Division (ARD), BITS Pilani K K Birla Goa Campus for their valuable suggestions.

My sincere thanks also goes to the Doctoral Advisory Committee (DAC) members **Dr. Biju K. Raveendran**, Assistant Professor of the department of Computer Science and Information Systems and **Dr. Chandradew Sharma**, Assistant Professor of the department of Physics, BITS Pilani K K Birla Goa Campus for preliminary assessment of the thesis and helpful suggestions during my progress seminars. I also thank the faculty members and research scholars of the department of Computer Science, BITS Pilani K K Birla Goa Campus for their kind cooperation.

I heartily thank my friends for their prayers and constant encouragement to finish this work successfully. I am also thankful to my fellow colleagues whose challenges and productive criticism, especially at the progress seminars, have provided improvement in the presentation of the work.

I would also like to thank my parents Mr. Laxmidhar Roul and Mrs. Basanti Roul, my wife Mrs. Malabika Roul and son Mr. Aarpit Kumar Roul for their constant love, confidence, good wishes and for being with me against all odds in this journey of education. This thesis is heartily dedicated to my wife Mrs. Malabika Roul who stood by me at all times.

RAJENDRA KUMAR ROUL

ABSTRACT

The World Wide Web is the main storage for Information Retrieval (IR). According to the latest survey, the web has indexed at least 4.75 billion of documents. To make searching of information much easier for the users, web search engines came into existence. However, with the exponential increase in the number of internet users and the digital documents on the web, it is becoming difficult for the users to find the relevant documents that fulfill their requirements. Aiming in this direction, the thesis presents a detailed design and investigation of different techniques to improve IR on the web.

Different IR models, performance measurement, various challenges in IR and techniques to overcome those challenges are discussed in Chapter 1. Thereafter in Chapter 2, the literature review of different IR techniques are discussed.

The architectures of the Extreme Learning Machine (ELM) and recently designed Multilayer ELM (ML-ELM) which is based on deep learning network are analyzed in Chapter 3. To handle multi-class classification problem, ELM One-Against-One and ELM One-Against-All techniques are discussed in this chapter.

Two new feature selection techniques for text classification named *k-means and Wordnet based feature selection* and *Combined Cohesion, Separation and Silhouette coefficient based feature selection* are discussed in Chapter 4. In both these techniques, deep learning in the form of ML-ELM classifier has been used extensively for experimental work.

Chapter 5 highlights the importance of ML-ELM feature space used for text clustering. Semi-supervised and unsupervised clustering using seeded *k-means* and *k-means*, respectively are done in the feature space of ML-ELM and the results are compared with clustering in the vector space model.

Chapter 6 proposes a new modified apriori approach by cutting down the repeated database scans to improve the association analysis of traditional apriori algorithm for clustering the web documents. Further, the performances of different traditional clustering techniques are measured after combining them with the modified apriori approach. To label these clusters, a cluster labeling mechanism is proposed in Chapter 7. For this purpose, Chi-Square feature

selection along with Wordnet are used to select the top keywords from a cluster. Instead of labeling the clusters with 'bag of words', a concept-driven mechanism has been developed which uses Wikipedia that takes these top keywords of a cluster as input to generate the possible candidate labels. Mutual Information technique is used to rank the candidate labels extracted from Wikipedia and then the topmost candidates are considered as the potential labels of a cluster.

CONTENTS

ACKNOWLEDGEMENT	iii
ABSTRACT	iv
List of Tables	xi
List of Figures	xiv
1 Introduction	1
1.1 Information Retrieval Models	2
1.2 Evaluation of Information Retrieval	3
1.3 Techniques used in Informational Retrieval	5
1.3.1 Feature Selection	5
1.3.2 Text Classification	9
1.3.3 Text Clustering	11
1.3.4 Cluster Labeling	12
1.3.5 Other techniques used in IR	13
1.4 Research Gap in Information Retrieval	13

1.5	Objectives and Organization of the Thesis	16
2	Literature Survey	18
2.1	Introduction	18
2.2	Survey on Feature Selection	18
2.3	Survey on Text Classification	22
2.4	Survey on Text Clustering	26
2.5	Survey on Cluster labeling	29
3	Extreme Learning Machines in Text Classification	32
3.1	Introduction	32
3.2	Methodology	33
3.2.1	Document pre-processing and indexing	33
3.2.2	ELM One-Against-All	33
3.2.3	ELM One-Against-One	35
3.2.4	Multilayer ELM	37
3.3	Experimental Analysis	40
3.4	Summary	44
4	Feature Selection Techniques for Text Classification	45
4.1	Introduction	45
4.2	<i>k-means and Wordnet based feature selection</i>	46
4.2.1	Methodology	47

4.2.2	Experimental Analysis	50
4.3	Combined Cohesion, Separation and Silhouette coefficient based feature Selection	53
4.3.1	Cohesion	55
4.3.2	Separation	55
4.3.3	Silhouette Coefficient	55
4.3.4	Methodology	56
4.3.5	Experimental Analysis	59
4.4	Summary	69
5	Clustering in ML-ELM Feature Space	71
5.1	Introduction	71
5.1.1	Importance of extended feature space of ML-ELM	72
5.1.2	Seeded- k Means/ k Means Algorithm	74
5.2	Methodology	75
5.3	Experimental Analysis	77
5.3.1	Performance evaluation of the clustering	77
5.4	Summary	80
6	Modified Apriori approach for Text clustering	81
6.1	Introduction	81
6.1.1	Modified Apriori Approach	82

6.1.2	Optimization open Traditional Apriori Algorithm for clustering	82
6.2	Methodology	83
6.2.1	Document pre-processing	83
6.2.2	Obtaining initial clusters and their centroids	83
6.2.3	Performing traditional clustering on the centroids of the initial clusters .	84
6.3	Experimental Analysis	84
6.3.1	Performance measurement of traditional and modified apriori approach	84
6.3.2	Performance measurement of traditional clustering algorithms	89
6.4	Summary	93
7	A hybrid approach for cluster labeling	94
7.1	Introduction	94
7.2	Methodology	94
7.2.1	Document Pre-processing	94
7.2.2	Clusters Generation	95
7.2.3	Top Documents selection	95
7.2.4	Representative Keywords Selection	96
7.2.5	Generating Candidate Labels	97
7.2.6	Evaluating Candidate Labels	97
7.3	Experimental Analysis	98
7.3.1	Using 20-Newsgroups dataset	99

7.3.2 Using Reuters dataset	106
7.4 Summary	107
8 Conclusions and Future Directions	109
LIST OF PUBLICATIONS	113
BRIEF BIOGRAPHY OF CANDIDATE	115
BRIEF BIOGRAPHY OF SUPERVISOR	116
.1	117
A Support Vector Machine	118
B Extreme Learning Machine	120
C Multilayer Extreme Learning Machine	125
D Traditional Feature Selection Techniques	130
E Fuzzy C-Means	132
F Cluster Evaluation	133
Bibliography	137

LIST OF TABLES

3.1	Term-document matrix	33
3.2	Accuracy comparisons of different state-of-the-art classifiers	43
3.3	F-measure comparisons of different state-of-the-art classifiers	44
4.1	Synonym list of keywords	46
4.2	ELM (20-Newsgroups)	50
4.3	ML-ELM (20-Newsgroups)	51
4.4	ELM (DMOZ)	52
4.5	ML-ELM (DMOZ)	53
4.6	Reduced term-document matrix	57
4.7	20-NG: performance on top 1% features	61
4.8	20-NG: performance on top 5% features	62
4.9	20 NG: performance on top 10% features	62
4.10	CLASSIC4: performance on top 1% features	62
4.11	CLASSIC4: performance on top 5% features	63
4.12	CLASSIC4: performance on top 10% features	63

4.13	REUTERS: performance on top 1% features	64
4.14	REUTERS: performance on top 5% features	64
4.15	REUTERS: performance on top 10% features	64
4.16	WebKB: performance on top 1% features	65
4.17	WebKB: performance on top 5% features	65
4.18	WebKB: performance on top 10% features	65
4.19	F-measure comparisons using <i>CCSS</i>	66
5.1	Purity of clusters on Classic4 dataset	78
5.2	Entropy of clusters on Classic4 dataset	78
5.3	Purity of clusters on Reuters dataset	78
5.4	Entropy of clusters on Reuters dataset	79
6.1	Performance comparison of different clustering techniques	92
7.1	Term-document matrix of each cluster	95
7.2	Reduced term-document matrix of each cluster	96
7.3	Representative keywords of 20-Newsgroups	101
7.4	Chi-Square values of representative keywords on 20-Newsgroups	102
7.5	Suggested candidate labels of 20-Newsgroups	104
7.6	MI-score for suggested candidate labels on 20-Newsgroups	105
7.7	Representative keywords of Reuters	106
7.8	Chi-square values of representative keywords on Reuters	107

7.9 Suggested candidate labels on Reuters 107

7.10 MI-score for suggested candidate labels on Reuters 107

LIST OF FIGURES

1.1	Architecture of Information Retrieval	2
1.2	Model of feature selection used for classification	6
3.1	ELM One-Against-All	34
3.2	ELM One-Against-One	35
3.3	Multilayer ELM	38
3.4	dmoz-chi-square	41
3.5	dmoz-ig	41
3.6	dmoz-bns	41
3.7	20ng-chi-square	42
3.8	20ng-ig	42
3.9	20ng-bns	42
4.1	Average precision of different classifiers on 20-Newsgroups	51
4.2	Average recall of different classifiers on 20-Newsgroups	51
4.3	Average F-measure of different classifiers on 20-Newsgroups	52

4.4	Average precision of different classifiers on DMOZ	53
4.5	Average recall of different classifiers on DMOZ	54
4.6	Average F-measure of different classifiers on DMOZ	54
4.7	Cohesion and Separation of two terms	56
4.8	Average F-measure on top 1% features	67
4.9	Average F-measure on top 5% features	67
4.10	Average F-measure on top 10% features	68
5.1	Semi supervised clustering (before)	72
5.2	Semi supervised clustering (after)	72
5.3	ML-ELM feature space	73
5.4	TF-IDF and ML-ELM feature vector	76
5.5	Execution time on Classic4 dataset	79
5.6	Execution time on Reuters dataset	79
6.1	Apriori vs. Modified Apriori on CASM	87
6.2	Apriori vs. Modified Apriori on CISI	87
6.3	Apriori vs. Modified Apriori on MED	87
6.4	Apriori vs. Modified Apriori on CRAN	88
6.5	Apriori vs. Modified Apriori on alt	88
6.6	Apriori vs. Modified Apriori on soc	88
6.7	Apriori vs. Modified Apriori on ship	89

6.8	Apriori vs. Modified Apriori on crude	90
6.9	Support count graph for Apriori vs. Modified Apriori	90
6.10	Performance comparison on Classic4	93
7.1	Top keywords of cluster 2 on 20-Newsgroups	101
7.2	Suggested candidate labels and their semantic distances from cluster 2 on 20- Newsgroups	103
7.3	Keyword ranking of top 3 representative keywords of each cluster on 20-Newsgroups	103
7.4	MI-score of each cluster on 20-Newsgroups	104
7.5	Top 3 representative keywords on Reuters	108
7.6	MI-score of each cluster runs on Reuters	108
B.1	Architecture of ELM	124
C.1	Multi-Layer ELM and ELM-Autoencoder	127
F.1	Cohesion	134
F.2	Separation between the centroid of two clusters	134

CHAPTER 1

INTRODUCTION

The process of retrieving unstructured materials (usually documents) that are relevant to an information need from a large collection, is known as *Information Retrieval (IR)*. The term *unstructured data* refers to the information which does not have any clear structure or is not ordered properly. The documents returned by the IR system may or may not match the user query completely. Hence, based on their *relevancy*, the retrieved documents are generally ranked. In information science/IR, relevance denotes how well a retrieved document or set of documents meets the information required by the user and it may include concerns such as timeliness, authority, i.e., from a trusted source or novelty of the document which satisfies the goals of the user and his/her intended use of the information. For ranking, all the documents and queries are indexed first and then the similarity score calculation (using standard similarity mechanism such as cosine-similarity, Euclidean distance, dice coefficient etc.) is performed between them. The general architecture of IR is illustrated in Figure 1.1.

One of the most important applications of the IR system is web-based search engines such as Google, Yahoo, Bing, Ask etc. As every day, the amount of information on the internet is increasing, the demand to retrieve different types of information has also increased which given rise to the interest in other IR related areas that go beyond the document retrieval, like question answering, image classification, audio and video retrievals, bioinformatics etc. Today, IR system has become very important in document retrieval as well as image, audio and video retrievals.

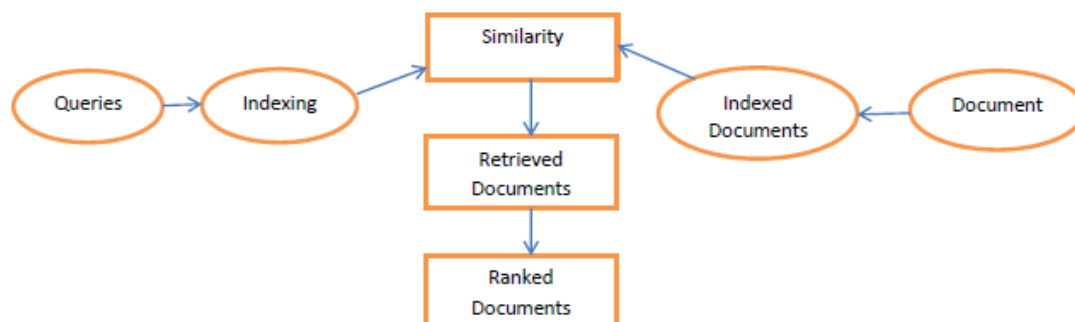


Figure 1.1: Architecture of Information Retrieval

1.1 Information Retrieval Models

To retrieve the relevant documents using IR strategies, documents are generally transformed into a suitable representation. Each retrieval strategy incorporates a specific model for the purpose of document representation and some of the common models are discussed below:

1. *Set-theoretic models* represent the documents as sets of words or phrases. Set-theoretic operations are used on these sets to get the similarity between document and query. Some of the popular set-theoretic models are:
 - i. *Standard boolean model* [1], where a document is represented as a set of keywords and queries are boolean expressions of keywords, connected by boolean operators. The output is whether the document is relevant or not. No partial matching and ranking of documents is possible in this model.
 - ii. *Fuzzy model* [2], in which queries and documents are represented by sets of index terms. The idea is to introduce the notion of a degree of membership associated with the documents in a fuzzy set.
2. *Algebraic models* represent documents and queries as vectors, matrices, or tuples. Some of the common algebraic models are:
 - i. *Vector space model (VSM)* [3], in which the documents and terms are represented by vectors. Documents are ranked based on the similarity score between the query and the documents.

- ii. *Generalized vector space model (GVSM)* [4], which is different from the VSM, where the index term vectors are linearly independent but not pairwise orthogonal. It considers correlations among the index terms.
 - iii. *Topic-based vector space model* [5] extends the VSM of IR by removing the constraint, i.e. 'the term-vectors to be orthogonal'. In contrast to the GVSM, this model does not depend on concurrence-based similarities between the terms.
 - iv. *Extended boolean model* [6], whose idea is to make use of term weight like VSM by combining the boolean query with VSM.
 - v. *Latent semantic indexing (LSI)* [7], takes documents that are semantically similar (talk about the same topics), but are not similar in the vector space (because they use different words) and re-represents them in a reduced concept space in which they have higher similarity.
3. *Probabilistic models* treat the process of document retrieval as a probabilistic inference. Similarities are computed as the probability of document relevant for a given query. Some of the common probabilistic models are:
- i. *Binary independence model* [8] is a probabilistic IR technique that makes some simple assumptions to estimate the probability of similarity between the document and query.
 - ii. *Language model* [9] is a probabilistic model based on the multinomial distribution over sequences of words.
 - iii. *Latent Dirichlet Allocation (LDA)* [10] represents the documents as mixtures of topics that split out words with certain probabilities.

1.2 Evaluation of Information Retrieval

The evaluation of an IR system is based on the assessment of how well a system meets the information required by its users. A collection of documents to be searched along with the search query is considered for performance measurement. All common measures used for performance evaluation indicate whether a document is relevant or non-relevant to a particular query. In practice, queries may be ill-posed and there may be different shades of

relevancy. The following standard measures are generally used for evaluating the performance of IR system.

- i. The performance of IR system is traditionally evaluated by computing the *F-measure* (F) which is weighted harmonic mean of *precision* (P) and *recall* (R)

$$F = 2 * \frac{(P \times R)}{(P + R)}$$

where, P is the ability to retrieve top-ranked documents that are most relevant. In other words, it is the fraction of retrieved documents that are relevant.

$$P = \frac{(\text{relevant}_{documents}) \cap (\text{retrieved}_{documents})}{\text{retrieved}_{documents}}$$

and R is the ability of the search to find all the relevant documents in the corpus. In other way, it is the fraction of relevant documents that are retrieved.

$$R = \frac{(\text{relevant}_{documents}) \cap (\text{retrieved}_{documents})}{\text{relevant}_{documents}}$$

- ii. *Accuracy* (A) refers to the closeness of a measured value to a standard or known value which is different than precision that refers to the closeness of two or more measurements to each other.

$$A = \frac{a + b}{N}$$

where a is the number of documents which are retrieved and are relevant, b is the number of documents which are not retrieved as well as not relevant and N is the total number of documents in the corpus. Some other performance evaluation mechanisms also exist such as:

- iii. *R-Precision* is precision at the R^{th} position in the ranking of results for a query that has R relevant documents.
- iv. *Average Precision* [9] is the average of the precision values of the points at which each relevant document is retrieved.
- v. *Mean Average Precision (MAP)* [9] is the average of the *average precision value* for a set of queries.

- vi. *Precision-at-k* [9] measures the precision of top k results.
- vii. *11-point interpolated average precision* [9] is the standard measure in the early Text REtrieval Conference (TREC) competitions. The precision at 11 levels of recall varying from 0 to 1 by tenths of the documents is taken using interpolation (the value for 0 is always interpolated) and then the precisions are averaged.

1.3 Techniques used in Informational Retrieval

There are various techniques exist to handle the challenges of IR. Many researchers are working in this field to improve the efficiency of IR process. The ultimate reason to enhance this retrieval process is to increase the performance of the search engine which meets the user requirements. Some of the state-of-the-art techniques are discussed below.

1.3.1 Feature Selection

In terms of information retrieval, there is no clear definition of a feature, but it is a distinctive attribute or characteristic of the data. In general, there are some properties which a good feature is likely to have such as meaningful perceptually (for humans), analytically special e.g. maxima, identifiable on images, invariant to a transformation, insensitive to noise. The process of transforming raw data into features which represent the model better, resulting in improved accuracy of the classification technique is known as feature engineering. This process is generally used in pattern classification techniques and can be categorized into three stages:

1. *Feature generation stage*: In this stage, candidate features are generated by pre-determined kind of sensing techniques from the training set.
2. *Feature refinement stage*: Also known as dimensionality reduction stage, where refinement of features is done via. feature extraction or feature selection.
3. *Feature utilization stage*: After feature refinement stage is over, the refined features are used to represent the instances of the dataset. An appropriate classification model is selected to make use of these features. Among the above three stages, feature refinement stage (or dimensionality reduction stage) is most important and has become the main topic of discussion

due to the following reasons:

- i. reduces the time and storage size.
- ii. improves the performance of the model by removing multi-collinearity.
- iii. data visualization is easier when the actual feature space is reduced to low dimensions such as dimension of two or three.

In *feature extraction*, it is assumed that all features are not appropriate though they contain sufficient information. To handle this problem, feature extraction generates new features from the original set by combining or transforming the original one from an extended space to a lower dimensional space, e.g. text clustering [7]. The data transformation may be linear as in the case of principal component analysis (PCA) [11] and linear discriminant analysis (LDA) [12], but many non-linear dimensionality reductions techniques are also in use.

In *feature selection*, the assumption is that the original feature set contains sufficient relevant features which can discriminate clearly between categories and therefore some of the irrelevant feature are eliminated for better efficiency and accuracy. It selects a subset of informative features from the initial feature set and uses it for model construction. Feature selection is important due to the following reasons:

- i. simplifies the model for better understanding.
- ii. makes generalization capacity of the classifier better by reducing overfitting.
- iii. takes lesser time to train the classifier by eliminating irrelevant features.

A general architecture of feature selection used for classification is shown in Figure 1.2.

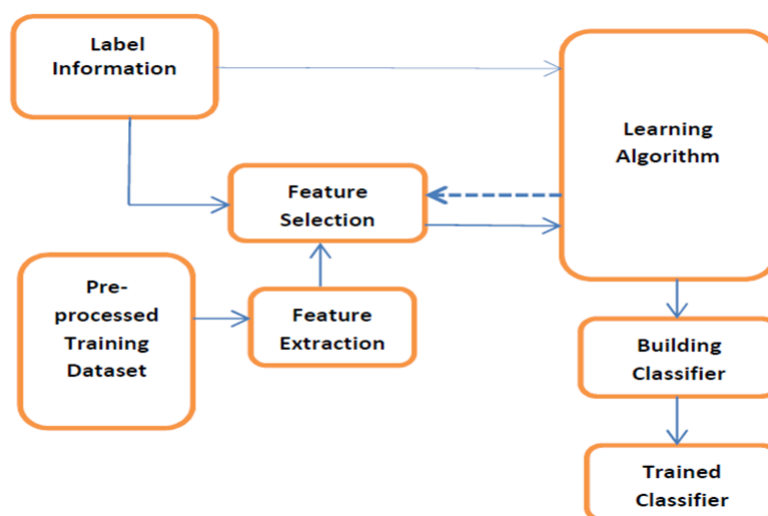


Figure 1.2: Model of feature selection used for classification

Generally, in text classification, feature selection techniques are used when the number of features is very high compared to the number of documents in the corpus. Feature selection increases the accuracy and efficiency of text classification when the original feature sets are of poor quality. Lewis in his work has listed six situations which gives rise to the poor quality of feature sets such as when the feature set does not have sufficiently distinguished instances, concepts from the hypothesis space are excluded from the feature set, large feature set results into ‘very big’ hypothesis space, explicit or implicit assumptions of the learning algorithms are violated by the feature set, feature set contains noisy data and redundancy in the feature set.

Different methods used for feature selection follow four preliminary steps:

1. *selection of the subset*: based on certain strategy, a subset of the candidate features will be selected.
2. *evaluation of the subset*: the subset generated in step 1 will be evaluated according to certain criteria.
3. *criteria to stop the process*: among all the candidate features, the top ‘m%’ features are selected based on the evaluation score, where ‘m’ is the stopping parameter and it is decided empirically.
4. *validation of the selected features*: in the last step, selected ‘m%’ features are validated using some domain knowledge.

Further, the algorithms used for feature selection are classified into the following three categories:

1. *Filter methods* do not use any classifiers for feature selection instead features are selected on the basis of statistical properties [13]. Hence, these methods are fast to compute and capture the usefulness of the feature set, that makes them more practical. Some of the examples include mutual information and correlation coefficient. There can be two categories of filter-based methods: *global*, where a unique score is assigned to each feature which ranks them in the entire corpus and *local*, where for any feature, multiple class-based scores are assigned. Finally, all local scores are converted to a unique global score using some globalization policy. In general, filter methods use cross-validation as the cut-off point for ranking the features. Based on the membership to a class, filter based methods are either *one or two-sided* [14].
 - i. *One-sided*: In this metrics, if the features have membership belonging to classes then

their score is either greater than or equal to zero else score is smaller than zero, depending on whether they have membership or non-membership to the classes. Hence, during feature selection, even if there are no positive features, negative feature cannot be part of top features e.g. odds ratio, correlation coefficient.

ii. *Two sided*: In two-sided metrics, all the features have their score greater than or equal to zero where positive and negative features are implicitly combined. Positive features get higher score compared to negative features which are rarely added to the feature set.

2. *Wrapper methods* generate different subsets of features based on some algorithms and test each subset using a classifier [15]. To find the score of the feature subsets, wrapper methods use a predicative model, whereas filter methods use a proxy measure. Hence, wrapper methods are computationally intensive when the features are very large and increase the overfitting risks, if the number of observations is not sufficient. But they generate the best feature set for a particular type of model as these methods detect the possible interactions between the features. Feature sets of filter methods are less tuned than wrapper methods, hence they are more generalized but usually perform worse than the wrapper methods [16]. As the feature set of filter methods do not use the assumption of predicative model, hence the relationship between the features are not considered which makes them to select the redundant features. Therefore, filter methods are mainly used as a pre-processing step for wrapper methods. Large-scale problems like text categorization mostly do not use wrapper methods due to high computational cost and chance of overfitting[17].

3. *Embedded methods* combine the advantages of both the previous two methods and thus their computational complexity lies between those two methods. In these methods, feature selection is integrated into the training phase of the classifier and thus just like the wrapper methods, these methods are specific to the learning mode. They use their own feature selection algorithm hence, they need to know a good selection in advance which causes degradation in their performance. In these methods, while building the model based on the prediction errors, selected function is either added or deleted, e.g. Recursive feature elimination algorithm, Least Absolute Shrinkage and Selection Operator (LASSO), Elastic Net, Ridge regression etc. are some of the existing embedded methods.

Generally, the feature selection methods are either *unsupervised* or *supervised*. No class labels are required to select the top features in the case of unsupervised feature selection. Super-

vised methods, on the other hand, do require class labels. Some of the unsupervised feature selection methods are ‘Term Strength’[18], ‘Term Contribution’ [18], ‘Document Frequency’ [9], ‘TF-IDF Metric’ [9] etc. Supervised feature selection methods further categorize into two sub-categories - Accuracy and Correlation based.

i. *Accuracy Based Method*: These methods choose the features which maximize the occurrence of features in the positive class and minimize the occurrences of the features in the negative class. Some of the existing methods like Odds Ratio’[19], ‘Probability Ratio’[20], ‘GU Metric’ [16], ‘Bi-Normal Separation (BNS)’ [17], ‘Power Metric’ [16], ‘Fisher Criterion’[21] etc. belong to this sub-category.

ii. *Correlation Based Method*: These methods evaluate the features by finding the correlation of the features with the various classes and choose the features which have the highest correlation score. For example, ‘Chi-square metric’ [9], ‘NGL coefficient’[22], ‘GSS coefficient’ [23], ‘MI-judge’ [9], ‘Information Gain’ etc. are some of the existing correlation based methods.

1.3.2 Text Classification

Documents on the web need to be organized in an effective manner so as to retrieve the data when needed. The process of finding the correct class or category for each document from a given collection of documents is known as text classification or text categorization. The set of categories or classes are already given in the case of supervised classification while they are determined from the collection of documents itself in case of unsupervised classification [9]. The text classification is either *binary* or *multi-class* depending on whether the test instance is classified into one of the two pre-defined classes or more than two classes, respectively. The following are two approaches used to handle the multi-class problem:

1. *One Against All (OAA)* which decomposes the multi-class problem into n binary problems. For each class $c_i \in C$, a binary problem is created where all instances that belong to c_i are considered positive examples, while the remaining instances are considered as negative examples. A binary classifier is then constructed to separate instances of c_i from rest of the classes. If an instance is classified as negative, then all the classes except the positive class receive a vote. This approach may lead to ties among different classes.
2. *One Against One (OAO)* constructs $n(n - 1)/2$ binary classifiers, where each classifier

is used to distinguish between a pair of classes (c_i, c_j). Instances that do not belong to either c_i or c_j are ignored while constructing the binary classifiers for c_i and c_j .

In the above two approaches, a test instance is classified by combining the predictions made by the binary classifiers. A *voting scheme* generally takes place to combine the predictions and the class which receives the highest votes is assigned to the test instance. Text classification is improved by improving the performance of the classifier but doing it manually consumes a lot of time which may lead to errors. Hence, text classification has become a very popular domain of research amongst experts in machine learning and IR in the last decade.

Choice of an efficient classifier is the next important thing in text classification. Based on the nature of the different classifiers, the classification is basically divided into two categories. The first one is *Eager learners*, where given a set of training tuples, it will construct a classification model before receiving new (i.e. test) tuples to classify. We can think of the learned model as being ready and eager to classify previously unseen tuples. Different classification techniques such as decision tree, Bayesian, rule-based, support vector machine artificial neural network etc. fall into this category. The other one of classification is *lazy learner*, in which the learner waits till the last minute before doing any model construction in order to classify a given tuple. In other way, given a training tuple, a lazy learner simply stores it (or does only a little minor processing) and waits until it gets a test tuple. Only when it sees the test tuples, it performs generalization (i.e. classification) in order to classify the tuples based on its similarity to the stored training tuples. Classification techniques such as k -nearest neighbor, case-based reasoning etc. are belong to this category. Lazy learners can be computationally expensive when making classification as they require extra storage space. It does less work when a training tuple is presented and more work when making a classification. However, it naturally support incremental learning. It is also known as instance-based learners as it stores the training tuples.

All the above discussed traditional classifiers have their own limitations. Most of the classifier architectures are based on the approach of neural network, hence certain restrictions are imposed which stops them to solve many complex problems. Extreme Learning Machine [24] is able to approximate any complex non-linear mappings directly from the training samples but it has a shallow architecture similar to traditional SLFNs. Hence, it may need a large network to perfectly fit the highly-variant input data which is difficult to implement. Deep learning, the re-branding of neural network is a sub-branch of machine learning and is based on a set of

network algorithms. The architecture of deep learning is used in many fields such as natural language processing, speech recognition, image processing etc. The common restrictions found in conventional classifiers are not there in deep learning, hence it is able to solve any complex problem [25]. Recently developed Multilayer ELM [26] classifier which relies on deep learning architecture can tackle such common problem generally found in traditional classifiers and is capable to handle a large volume of data.

1.3.3 Text Clustering

Clustering is a technique where a corpus is divided into groups of similar documents. Grouping is such that the documents in a group are more similar to each other as compared to documents in other groups. Clustering can be *unsupervised* (label of the cluster is unknown), *semi-supervised* (partially label is known) and *supervised* (label is known, also known as classification) The unsupervised clustering can be categorized as follows:

1. *Hierarchical Methods* also known as connectivity based clustering where different objects are connected to form clusters. The basic idea behind this method is that at different distances, different clusters are formed. The results of hierarchical clustering are usually presented in a dendrogram (tree structure), especially one showing taxonomic relationships. There are various ways of computing the distance based on singly link, completely link, average link etc. Various techniques are used under hierarchical clustering and mostly come under two categories.
 - i. *Agglomerative* [27] also known as bottom-up method, in which each object is considered as a cluster at first and then pairs of clusters are merged as one moves up the hierarchy.
 - ii. *Divisive* [27] also known as top-down method, in which all the objects are considered to be of one cluster at the beginning and it is recursively divided forming different clusters till one gets each object separately.
2. *Centroid based methods* [27] A central vector represents clusters and it may not be an actual object of the data set. When the number of clusters is known in advance, the algorithm is known as *k*-means clustering algorithm where *k* represents number of clusters. Variations of *k*-means clustering algorithm with different types of optimizations have led to algorithms like *k*-medians, bisecting *k*-means, fuzzy *k*-means etc.

3. *Distribution based methods* [27] can be imagined as objects belonging to same type of distribution. Though theoretically simpler, this model is more complex than the ones mentioned above. Gaussian mixture model is the most prominent example of distribution model.
4. *Density-based methods* [27], where the main idea is that the areas of high density form a cluster and the sparse areas are classified as noise. Density-based scan (DBSCAN) and OPTICS are the two most common examples of density-based clustering models.

1.3.4 Cluster Labeling

After clusters are made, it is important to label them in order to get brief information of each cluster. The purpose of cluster labeling is basically to identify a label for each cluster which can summarize the cluster accurately and distinguish it from other clusters. Strategies for cluster labeling can be broadly classified into two categories:

1. *Differential cluster labeling*, which labels a cluster by comparing term distributions across all clusters. Terms having very low frequency are not the best in representing the whole cluster and can be omitted while labeling the cluster. Two common techniques used for this cluster labeling are: *Point-wise Mutual Information* [28] and *Chi-squared selection*.
2. *Cluster internal labeling*, in which the clusters under consideration (decided based on some techniques) for labeling are just analyzed and its labeling is independent of labels of other clusters. Common techniques used are:
 - i. *Centroid labels*: Centroid of the cluster is calculated by considering all the document vectors. The terms having more weight in the centroid vector are considered for labeling.
 - ii. *Contextualized centroid labels* : Centroid labels may not be effective when cluster contains useless terms of high frequency [29]. To overcome this challenge, cluster is graphically plotted with vectors as roots. This gives a deeper level of interpretation with some semantic knowledge of terms.
 - iii. *Title labels*: The title of the document which is very close to the centroid of the cluster is used for cluster labeling. This technique can also be misleading in cases where a document cannot represent the cluster accurately.

iv. *External knowledge labels*: Labeling process can take advantage of pre-categorized knowledge such as Wikipedia [30] and the candidate label set is taken from Wikipedia after sending the important terms from a cluster. Selection is done from this set using various ranking methods.

1.3.5 Other techniques used in IR

1. Text Summarization

The task of generating summary from one or more texts, which contains important information and is significantly smaller than the original texts (not more than half of the original size) is known as *text summarization*. Extractive and Abstractive are generally two most common methods extensively used for text summarization [31]. In extractive text summarization, various techniques are used to identify the important units of text from the document i.e. the units can be key-phrases or sentences and those units are then extracted and presented in order. Abstractive text summarization is a more ambitious approach where the system generates summaries that are close to the summaries written by humans. This is achieved by the use of various natural language generation techniques.

2. Duplicate and Near-duplicate page detection

Duplicate pages refer to multiple copies of same pages. Near-duplicate pages refer to the pages which are similar but not exact copies of each other. Near-duplicate pages normally have slight modification like insertion, deletion or updation of some data. Algorithms used in near-duplicate page detection are Shingling [32], Spex [33], Simhash [34]).

3. Spam page detection

Web spamming refers to the manipulation of web pages to increase their ranking in the results retrieved by a search engine. There are several algorithms which are used for web spamming and can be broadly classified into three categories: Content-based [35, 36], Link-based [37], Hiding (Clocking and Redirection) [38].

1.4 Research Gap in Information Retrieval

Searching has become the leading paradigm to find the information on WWW. Some of the important challenges are considered as the research gap in IR and discussed below:

i. *Global information access:*

Information is available in a vast number of languages and it is important for a good IR system to take queries in any language and retrieve relevant information regardless of the language of the data. The amount of information on the internet in languages such as mandarin and hindi is increasing rapidly.

ii. *Contextual retrieval:*

Combination of search technique and an understanding of the context of a given query in order to give the most appropriate answer for the query is called contextual retrieval. If all queries are treated equally, the burden is on the user to retrieve the most relevant information from the retrieved results. Understanding context of the user query provides a solution to this problem which helps to retrieve information related to a particular context. In spite of good research, little progress has been made due to various difficulties in understanding the query context.

iii. *Formal representation of natural language:*

A vast number of documents available are in the form of natural language. We need an effective formal representation of such documents in order to retrieve information from them. Often the systems use data extraction instead of translation to capture the imported information from the data. Translation of queries is also important to properly understand what the user is trying to find. Summarization techniques like extraction and abstraction are used and research is being done to improve the techniques further.

iv. *Query-based challenges:*

- **Synonymy:** Different words having the same meaning are known as synonyms. IR systems fail to retrieve results related to synonyms of query words which might be relevant.
- **Polysemy:** If one word has many different meanings then it is known as polysemy. IR systems should know which of the meanings are relevant to the query.
- **Phrases:** Sometimes the search requires us to look for phrases as a whole instead of individual words, because the results of both approaches might not be the same. IR systems should take care of such challenges.

- Object recognition: As dates and currencies can be expressed in different formats, it is important for a good IR system to effectively identify the format and search accordingly.
- Semantics: Sometimes the query can only be understood with semantics e.g. in cases where query is a question and an answer is expected to that question.
- Computation: Some types of queries expect an IR system to compute the value and print the result. IR systems require special functions for computation in such cases e.g. if one types $2 + 3$, he should get 5 as the answer.

v. *Data and performance related challenges:*

- Distributed data: Data to be indexed and retrieved may be distributed on millions of servers worldwide and hence it creates challenges for search engine to retrieve the data efficiently without loss of data.
- Huge data size: The size of the data on the web is increasing everyday . It requires huge storage size to index, even for a small subset of this available data are there to retrieve.
- Volatility of data: Data available is volatile and can be changed. Therefore, for useful data, frequent indexing is required.
- Unstructured and redundant data: No conceptual model/ organization/constraints over data and about 30% of the data available on the web is redundant. It is the job of an IR system to identify such redundant information and retrieve only relevant results.
- Poor quality of data: Anyone from anywhere can upload data on the web. Such data is often outdated, not maintained and may be incorrect e.g. anyone can edit Wikipedia, making it quite an unreliable source of information.
- Heterogeneous data: Multiple media types, formats, languages and alphabets of data are available on the Web.
- Spam data: Spammers exploit the algorithms in IR systems to gain higher rankings in the results. Spam web pages are often devoid of any relevant content and mostly contain advertisements.
- Duplicate and near-duplicate data: These pages increase the computational complexity of the search engine during indexing and crawling.

1.5 Objectives and Organization of the Thesis

The thesis primarily focuses to improve the data and performance related challenges such as volatility of data, unstructured and redundant data, poor quality of data by studying the existing IR techniques such as text classification, feature selection, text clustering and cluster labeling and then designing and investigating some novel techniques by exploring ML-ELM (deep learning) in the domain of IR. The objectives of this thesis are to fulfill the above research gap (i.e. data and performance related challenges) and are listed below:

1. To investigate whether ML-ELM (deep learning) applied to IR can improve the quality of IR in the domain of text classification.
2. To design novel feature selection techniques which can enhance the efficiency of ML-ELM without compromising the efficacy of IR.
3. To investigate how better is the feature space of ML-ELM compared to the TF-IDF vector space for text clustering.
4. To design a novel modified apriori approach which can overcome the problem of traditional apriori approach for text clustering and then test the performance of existing clustering algorithms on the vector space of modified apriori approach. Next, developing an efficient cluster labeling technique to label these clusters generated by the modified apriori approach.

Accordingly, Chapter 2 discusses the prior works done by the researchers on different IR techniques.

ELM (ELM OAO and ELM OAA) and recently developed Multilayer ELM are the two most state-of-the-art classifiers whose architectures design are discussed in Chapter 3. Also, different traditional feature selection techniques are used for experimental work in order to find which technique can improve the performance of ELM and ML-ELM better.

Chapter 4 introduces two novel feature selection techniques for text classification named *k-means and Wordnet based feature selection (KWFS)* and *Combined Cohesion*,

Separation and Silhouette coefficient (CCSS) based feature selection. In *KWFS*, a corpus is divided into different clusters using k -means clustering technique and then Bi-Normal Separation(BNS) along with Wordnet with cosine-similarity generate the reduced feature vector. Whereas in *CCSS*, three important parameters *cohesion*, *separation* and *silhouette coefficient* are combined together to measure the importance of a term in the entire corpus. The total score for a term is computed by adding the score of the silhouette coefficient to the ratio between the separation and cohesion score of that term and finally, based on the total score, important features are selected.

Chapter 5 discusses how semi-supervised and unsupervised clustering using seeded k -means and k -means techniques are done in ML-ELM feature space. The empirical results of seeded k -means outperform the k -means clustering algorithm both in ML-ELM feature space as well as in TF-IDF vector space.

Chapter 6 introduces the techniques to cluster the web documents using modified apriori approach. The proposed approach considered documents as itemset and keywords as transaction so that it ends up with clusters having a minimum frequency support threshold. By making the keywords as transaction and document as itemset, the support threshold idea of association rule mining algorithm is combined with the output like that of a clustering algorithm. This modified apriori technique is run on a corpus of web documents to get some initial clusters. Next, the Fuzzy C-means, k -means and Vector Space Model clustering techniques are run on these initial clusters and their clustering results are compared.

Chapter 7 proposes an approach on cluster labeling which is the follow-up of Chapter 6. To label the clusters, Chi-Square along with Wordnet have been used for selecting top keywords of a cluster. Instead of labeling the cluster with 'bag of words', a concept-driven mechanism has been developed which uses the online encyclopedia like Wikipedia that takes the top keywords of a cluster as input to produce the possible candidate labels. Mutual Information technique is used to rank the candidate labels and the topmost candidates are considered as the potential labels of a cluster.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

The present search engine is facing many challenges due to unstructured nature of the web. IR plays an important role in enhancing the performance of search engine by introducing many techniques which can handle these challenges very efficiently. Much research works is going on in this domain which introduces many new techniques to boost IR. In this chapter, we discuss the prior works which are done on different techniques of IR.

2.2 Survey on Feature Selection

Feature selection is an important technique in IR and lot of research work has already been done in this field. It simplifies the model and decreases the training time. Depending on the selection algorithm and model building, the feature selection methods can be classified into 3 categories namely Filter methods, Wrapper methods and Embedded methods. Filter based techniques evaluate general characteristics of data without the involvement of any learning algorithm. Filtering based techniques are commonly used for the feature selection task. Pinheiro et al. [39] have proposed a filtering technique called ALOFT (at least one feature) for the task of feature selection. This method is based on the knowledge of text categorization domain's specific characteristics. Experimental evaluations on Reuters-21578, 20-Newsgroups and WebKB using k-NN and Naive Bayes classifiers show that ALOFT performs better than

classic variable Ranking method. In the similar lines, an improved version of traditional filter based technique is proposed by Uysal et al.[40] known as Improved Global Feature Selection Scheme (IGFSS). Evaluation done on common public datasets reveal that IGFSS improves the classification in terms of two widely used metrics-Micro and Macro F1. A probabilistic filter based technique for feature selection known as distinguishing feature selector (DFS) is also proposed by Uysal et al.[41]. This technique offers competitive performance with most sophisticated approaches like Chi-Square, information gain, Gini index etc. in terms of computational complexity, accuracy and dimensionality reduction.

Wrapper method was first proposed in 1997 by Kohavi et al.[15]. A relation is established between the relevance and the feature subset. The optimal subset is found out for a particular algorithm and domain. Accuracy is significantly improved as compared to filter methods but it requires more computation time. A new approach for unsupervised feature selection is proposed by Wang et al.[42]. It uses the embedded method in which feature selections is embedded into a clustering algorithm via sparse learning without transformation. Experimental results on six different datasets gave positive results.

The training data can either be labeled, unlabeled or partially labeled which leads to supervised, unsupervised and semi-supervised learning. In supervised learning, relevance of the feature is measured by its correlation with the class (Robnik et al.[43], Weston et al.[44], Song et al.[45] and Li et al.[46]) data variance or data distribution is used to measure the relevance of feature in unsupervised learning (Dash et al.[47], Dy et al.[48] and He et al.[49]) while semi-supervised learning uses small number of labeled data to improve unsupervised learning (Zhao et al.[50]). Feature selection in unsupervised scenario is a much harder task than in supervised learning due to absence of class label which gives relevant information. He et al.[49] have proposed a filter technique for feature selection task which does not require any learning algorithm. The relevance of a feature is determined by its Laplacian score and the top features are selected based on this score. Experimental results show that this method is effective and outperforms other methods like data variance and Fisher score. Feature selection for the task of text clustering, unsupervised learning is commonly used. Liu et al.[51] have proposed techniques namely ‘document frequency’, ‘term contribution’, ‘term variance quality’ and ‘term variance’. Experimentally, it is found that these feature selection techniques not only decrease the dimensionality of features but also increase the accuracy of text clustering.

Multi-label feature selection is a promising technique to increase the efficiency and efficacy of multi-label classification and is harder to implement than single label feature selection. Lee et al.[52] in their paper proposed multi-label feature selection technique based on mutual information using interaction information. In this method, dependencies among multiple variables are measured. Proposed approach effectively selects feature subsets for multi-label classification problems. Similarly, Chen et al.[53] have suggested a framework for multi-label feature selection in which multi-label documents are transformed to a single label before applying feature selection algorithms. They have proposed an approach namely Entropy-based label assignment (ELA) in which label weights are assigned to a multi-label document based on label entropy. Evaluation is done using three standard feature selection algorithms and the results are found to be promising. In another work, Chen et al.[54] have proposed fuzzy ranking analysis paradigm for reduction in dimensionality along with a relevance measure called as Discriminating Power Measure (DPM). DPM has low computational cost and helps in discriminating positive and negative features, also classification is done in parallel and not in serial order. Experimental results show the reduction in dimensionality from thousands to few hundreds with zero rejection and little decline in accuracy (from 84.5% to 80.4%). Research of feature selection on Naive Bayesian classifier is of great importance as it is simple to implement and also is highly sensitive to the features that are selected. In the same direction, Chen et al.[55] also presents Class Discriminating Features (CDM) and Multi-class Odds Ratio (MOR) feature selection metrics for Naive Bayesian classifier. Experiments carried out on multi-class documents using this approach proved that it has better selection capability compared to other common approaches.

Information theory based feature selection is very effective in terms of computation cost, dimensionality, scalability and classifier's independence. Common drawbacks of this method are selection of irrelevant features and lack of knowledge between features. This problem is addressed by Bennasar et al.[56] with the use of two non-linear feature selection techniques known as Joint Mutual Information Maximization (JMIM) and Normalized Joint Mutual Information Maximization (NJMIM). These methods use Mutual Information and maximum/minimum criteria to address the problem of overestimation. Results have shown that these techniques performed better than most of the others on common public datasets. Average class error is reduced by 6% compared to the next best performing technique.

In the context of text categorization, Azam et al.[57] have done a comparison of feature selection metrics based on term frequency and document frequency. The main focus of their work is the relative importance of these frequencies and for this purpose metrics which are based on document frequency like GINI index and discriminative power measure are tested with term frequency. Experimental results on Reuters dataset reveal that the term frequency based metrics are more useful for smaller feature sets. Major problem of text categorization is high dimensionality of feature space. In this context Shang et al.[58] have done study on Gini index theory and proposed a new Gini index based algorithm which helps in reducing the dimensions of features. Construction of measure function of Gini index also helps in text categorization. Meng et al. [59] have done a two step feature selection for the task of text categorization. First, a novel feature selection technique is applied and secondly, a semantic space is constructed between terms based on latent semantic indexing. This two stage process is experimentally found to outperform standard feature selection techniques. Feature selection based on the semantic relation of text documents is suggested by Thangamani et al.[60]. Identification of semantic relation is done using ontology. Relation between term and concept is represented by ontology and evaluations show the effectiveness of the proposed approach. Pent et al.[61] have implemented minimal redundancy maximal relevance (mRMR), an equivalent form of maximal dependency condition for the task of feature selection. In the next stage, mRMR and other common feature selectors are combined to generate the final feature set. Experiments on naive Bayes, SVM and Linear discriminative analysis on four different datasets show improvement in computational complexity and accuracy of the proposed approach. A framework of feature selection based on two measurements namely frequency and ratio measurement has been introduced by Li et al.[62]. A method known as WFO is proposed which combined these two measurements and trained weights. Experimental results show the robustness of this method across various classification tasks.

Ant colony algorithm is based on observation of ants finding their shortest path in search of food. Aghdam et al.[63] have suggested a feature selection algorithm based on ant colony optimization. This method is computationally less complex and outperforms standard algorithms like information gain and Chi-Squared based on the results obtained by simulation on Reuters dataset. A binomial hypothesis test for the estimation of the probability of a feature's relevance based on a threshold value is proposed by Yang et al.[64]. This technique is named as Bi-Test and compared with four most common algorithms such as information

gain, Chi-Squared, Gini index and Poisson distribution. Experimental results have shown that Bi-test outperforms most of these algorithms with the use of common classifiers. A different approach for feature selection task is proposed by Gabrilovich et al.[65]. Their technique is based on domain-specific knowledge. From a given feature subset, features are generated by contextual analysis and word sense disambiguation. Synonyms and polysemes are addressed with the use of ontology to generalize terms to concepts. Experimental analysis have revealed a large reduction in feature subset with this method.

2.3 Survey on Text Classification

Recently ELM and ML-ELM have attracted the attention of many researchers in the field of text classification. Working in this direction, Huang et al. [66] in their approach discussed three important things. First, ELM provides unified learning platform, second, compared to PSVM and LS-SVM, ELM has fewer optimisation constraints and third in theory, ELM can classify any disjoint regions and approximate any target continuous function. Their simulation results show that ELM has good performance and scalability at much faster learning speed compared to SVM and LS-SVM. Zuo Bai et al. [67] worked on sparse ELM and has shown that sparse ELM can reduce the training time and storage space compared to the unified ELM. It has very good performance with faster learning speed compared to the state-of-the-art SVM classifier. It also has the ability to handle large-scale binary classification compared to the unified ELM. Balasundaram et al. [68] have worked on a new 1-norm ELM for regression and multi-class classification and proposed a linear programming work whose solution is obtained by solving its dual exterior penalty problem as an unconstrained minimization using a fast Newton method. The main advantage of their approach is that it leads to a sparse model representation where many components of the optimal solution vector will become zero and hence, the decision function can be determined using much less number of hidden nodes compared to ELM. The experimental results are compared with ELM using additive and radial basis function (RBF) hidden nodes, optimally pruned ELM (OP-ELM) and SVM methods. Similar or better generalization performance of the proposed method on the test data over ELM, OP-ELM and SVM clearly illustrates its applicability and usefulness. Shifei Ding et al. [69] introduced ELM, describing the principles and algorithm of ELM. In their studies, typical variants of ELM like incremental ELM, two-stage ELM, pruning ELM, evolutionary ELM,

error-minimised ELM, online sequential ELM etc. have been described. They have summarized the applications of ELM for classification, function approximation, regression, pattern recognition etc.

Very less research work has been done where ML-ELM is used as a classifier (Ding et al. [25] Mirza et al. [70] Yang et al. [71] Tang et al.[72]). Many other state-of-the-art mechanisms also have been used for text classification. A new web page classification based on SVM weighted voting scheme has been proposed by Rung-Ching et al. [73]. In their work, latent semantic analysis has been used to find the hidden information from the documents and it is also used to extract text features from each web page. This helps the SVM to classify the web pages. Experimental results show that their approach is better than the traditional approaches. Chin et al. [74] proposed a new text document classification which is a combination of k-Nearest Neighbors (k-NN) and SVM techniques. They have tested their approach on many benchmark datasets and the results show that the accuracy of SVM and k-NN combined approach has less impact on the values of the parameters as compared to the traditional k-NN technique. A rough set approach to SVM classification is proposed by Lingras et al. [75] which is mostly useful when handling noisy data. Their work proposed two new approaches, extension (1-v-r) and (1-v-1) to SVM multi-class classification by using the boundary region in rough sets. They have justified that extended (1-v-r) can reduce the training time of the traditional (1-v-r) approach. The experimental results support their theoretical results.

Least squares support vector machines (LS-SVMs) are sensitive to noise or outliers in the training dataset. Working in this direction, Thamrongrat et al. [76] have proposed a novel LS-SVM (RLS-SVM) approach which is based on the truncated least squares loss function for classification and regression with noisy data. In their work, the Newton and an iterative algorithm based on the concave-convex procedure (CCCP) have been used to solve the proposed RLS-SVM approach. They have tested their approach on fourteen benchmark regression and ten benchmark classification datasets. Juan et al. [77] have suggested a method which uses textual content of the documents in order to classify the web documents into a predefined hierarchy. They have developed a Stratified Discriminant Analysis (SDA) technique to reduce the feature vectors of web documents, and to identify the categories with few training examples leading to more robust classification models for those categories. Bai et al. [78] have devel-

oped a model called SUMO (Suggested Upper Merged Ontology) based on text classification which is integrated with Wordnet ontology to classify the web pages. Their method can reduce the dimensionality of the vector space and increase the performance of text classification. Long Li et al. [79] proposed a hierarchical-vertical classification of framework that built a hierarchical classifier after discovering the inherent hierarchical structure of relationships among vertical web pages based on flat datasets. They have used SVM using odds ratio test to select discriminative features which produced best results.

Junchang Xin et al.[80] have proposed a novel distributed ELM based on Map-Reduce framework, named ELM*. The most expensive computation part of the matrix Moore-Penrose generalized inverse operator in the output weight vector calculation is the matrix multiplication operator. As the matrix multiplication operator is decomposable, a distributed ELM (ELM*) based on Map-Reduce framework is developed which handles the expensive computation very efficiently. Myungsook et al. [81] have developed a technique for web pages classification using keywords of documents and random forest learning method. In their work, they identified that the random forest learning method is better than other state-of-the-art machine learning mechanisms for classification. Wen Zhang et al [82] used semi-supervised clustering for text classification. Their approach is based on the assumption that documents of each category have multiple text components which are identifiable by clustering. They have used labeled documents to capture the silhouettes of text components and unlabeled documents are used to adapt the centroids of text components. They categorized an unlabeled document into the class of the text cluster using Euclidean distance. Their approach outperforms SVM, BPNN and DKS methods. An unsupervised URL based web page classifier has been proposed by I. Hernandez et al. [83]. The main aim of their approach is to analyze the URL of web pages in the training set, building a set of patterns which are representative of collection of URLs that refer to the web pages of the same class.

Temporal contexts present in the text document for effective text classification have been used by L. Rocha et al. [84]. They have proposed an algorithm named Chronos to identify such temporal contexts based on the stability of the terms in the training set. The temporal characteristics that their approach explores are class distribution, which is classes appearing, splitting or merging as a consequence of evolution of knowledge. The temporal contexts selected by Chronos are passed onto standard automatic document classifiers such as

SVM, k-NN, Naive Bayes and they have reported that the performance of these classifiers are improved by 10%. R. Johnson et al. [85] effectively used word order for text classification with the help of convolution neural networks (CNN). Their approach relies on CNN that is applied on high dimensional text data for learning embedding of small text regions. The notable feature of their approach is that n-grams can contribute to predict accurately despite not being present in the training data, as long as some of their constituent words appear in training set. They have reported that their error rate to be lower than state-of-the-art classification techniques such as SVM, Naive Bayes + SVM and Naive Bayes + Language model. An approach that relies on the border instances found by routine strategies to construct centroid vectors for centroid-based classifier is proposed by D. Wang et al. [86]. Their approach used a proper subset of documents which are the border instances, rather than all to construct centroid vectors and then iteratively adjust initial centroid vectors after eliminating erroneous instances. The proposed approach outperforms standard classification techniques as k-NN, centroid classification and SVM. JuiHsi et al. [87] have proposed a method called Identifying Possibly Misclassification Documents (IPMD) to classify the Chinese documents accurately. They have verified the documents class labels predicted by SVM. These verified documents are then fed to the IPMD module to determine their distinguishability. The indistinguishable documents are utilized to develop more precise SVM models using learning strategy modules. The used algorithm is semi-supervised in nature and the experimental results show that to classify the Chinese documents the best approach is to feed both indistinguishable and misclassified documents to the training set.

Kwanho et al. [88] have proposed a language-independent semantic kernel that establishes similarity between short-text documents without the used of lexical databases and grammatical tags. They have used semantic and syntactic features of documents in the same kernel. Their approach is robust to the number of categories involved and the number of words per document regardless of the language. A concept model to make the spatial feature more effective was used by Huilin et al. [89]. In addition to using the inherent information from the sentence itself, they also used semantic information connection between sentences. The additional features reveal the similarity between instances. Experiments show that this new feature enhanced the precision and recall. They used ELM instead of SVM for relation extraction as it is faster and better. To capture label dependencies, a supervised topic modeling algorithm, Label set Topic Model (LsTM) was proposed by Ximing et al. [90]. The reason was that a word

can be assigned to a combination of labels rather than a single label. They used two observed label set layers: the super-label set and the sub-label set. The super-label set groups several related labels and the sub-label set assigns combination of these labels to each word. Their results are at par with the existing approaches. Their approach is an extension of L-LDA. An approach to classify short text from scientific documents suggested by Duc-Thuan et al. [91]. They have used topic models from various universal datasets to enhance text features. The idea is to make short text more topic-oriented and less sparse. Machine learning algorithms like SVM, k-NN and Naive Bayes are applied for classification and they asserted that their approach outperforms the existing algorithms to classify short text scientific documents.

All of the above approaches have used traditional classifiers which have their own limitations. Most of them use the shallow neural networks algorithms in which there are certain restrictions for the capabilities to approximate the complex function. Deep learning has aroused interest in the past decade in many research domain such as computer vision, automatic speech recognition, pattern recognition and recently has attracted much attention in the field of machine learning. It is a multilayer perceptron artificial neural network algorithm. There is no such restriction found in deep learning (capabilities to achieve approximating the complex function) which removes the difficulty of optimization associated with the deep models [25] and achieves an approximation of complex function. Extreme Learning Machine [24] is able to approximate any complex non-linear mappings directly from the training samples but it has shallow architecture similar to traditional SLFNs. Hence, it may need a large network to perfectly fit the highly-variant input data which is difficult to implement. Recently designed Multilayer ELM [26] is able to address this issue which combines deep learning (i.e. ELM auto-encoder) with ELM, decomposes the original input data into multiple hidden layers and performs unsupervised learning layer-wise.

2.4 Survey on Text Clustering

In this semantic web, cluster based on semantic meaning of the documents are much needed for the user. The proliferation of web over the past few years has resulted in various modifications and refinements in the methods of searching web documents. Working in this direction, Song et al. [92] have developed a genetic algorithm based on latent semantic

indexing (LSI) for text clustering. LSI helps in dimensionality reduction and take cares of synonyms and polysemes. They proposed a variable string length genetic algorithm which generates the number of required clusters to be formed. Experimental results have shown the efficiency of their approach. Li et al.[93] in their work have used user-related tag expansion method to improve the web document clustering. They designed a new model called Folk-LDA that mutually models expanded and original tags as independent observations. Experimentally they have shown that their technique can be efficiently applied to more than 90% of tagged web documents. Empirical results based on human-edited web directory justified that the tag-based clustering method is better than the word-based methods.

An LSI based multilingual document clustering (MLDC) technique has been proposed by Wei et al. [94]. MLDC organizes the documents in different languages into different categories having similar documents on the basis of their content. Experimentally they have shown that their approach can able to manage a good balance between cross-lingual and monolingual clustering of documents. Cao et al. [95] designed a cluster-based vector space model which used co-occurrence matrix based on text features to represent the documents. Their work first identifies the proper noun from all the documents and then extracts different features in the form of words, phrases, non-contiguous phrases of proper noun and cluster them. Finally, they clusters the text using co-occurrence matrix based on text feature clustering. Huang et al. [96] used a hierarchical represent model with multi-granularity (HRMM) for clustering the web documents. Their approach consists of two-phase clustering process and five-layer representation of data. HRMM captures the structural knowledge hidden in the documents by using granular computing which generates high quality clusters. Experimental results show that HRMM significantly outperformed the vector space model (VSM) and non VSM-based clustering algorithms. A statistical semantic method to enhance the document clustering has been proposed by Farahat et al.[97]. In their work, they measure the term-term correlations which exist in the documents that needs to be clustered in order to represent a corpus specific semantic similarity. To capture this similarity, they compared their approach with Vector Space Model (VSM) and other well known methods. The experimental works on thirteen benchmark datasets show the effectiveness of their approach.

Clustering methods are unsupervised i.e. no labeled data are available [98] [99]. One of the most popular partitioning clustering algorithms is K-Means, which partitions ‘n’

observations into 'k' clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This algorithm can be performed on a large data set with linear time complexity [100]. However, the problem of this algorithm is that an inappropriate choice of clusters 'k' may yield poor results. Improvement of K-Means clustering algorithm based on user tag is done by Jun Tang [101]. It was first used in social annotation data to expand the vector space model of K-Means. Then, it is applied to the links involved in social tagging network to enhance the clustering performance. In case of an ambiguous query, word sense discovery is one of the useful methods for IR in which documents are clustered in corpus. Discovering word senses by clustering the words according to their distributional similarity is done by Patrick et al. [102]. The main drawback of their approach is that they require large training data to make proper cluster and its performance is based on cluster centroid, which changes whenever a new web page is added to it. Hence, identifying relevant cluster will be a tedious work. In 2008, Jiyang Chen et al. [103] have proposed an unsupervised approach to cluster that results by word sense communities.

Clusters are made based on dependency based keywords which are extracted for large corpus and manual label are assigned to each cluster. Chakrabarti [104] also discussed various types of clustering methods and categorized them into partitioning, geometric embedding and probabilistic. Phiradit et al. [105] proposed Suffix Tree Clustering (STC), a phrase-based state-of-the-art algorithm for web clustering that automatically groups semantically related documents based on shared phrases. Their technique combines the hierarchical agglomerative clustering method with phrase based STC to improve the cluster merging process. It outperforms the original STC with 16% increase in F-measure. Peng Li et al. [93] proposed a user-related tag expansion method to overcome this problem of limiting the usage of tags. They have designed a novel generative model called Folk-LDA, which jointly models original and expanded tags as independent observations. Results show that Folk-LDA can alleviate topic drift in expansion, especially for topic-specific documents and the proposed tag-based clustering methods significantly outperform the word-based methods. Xiwu et al. [106] have investigated and evaluated several extended vector space models which can combine social annotation and web page text. In particular, a novel vector space model is proposed by computing the semantic correlations between social annotations and web page words. Using semantic correlations between social tags and web page words that improves the clustering accuracy with an increase of 4%-7% of RI score. Cindy et al. [107] have extracted a similarity

matrix among pages via. in-page and cross-page link structures, based on which a density-based clustering algorithm is developed which hierarchically groups densely linked web pages into semantic clusters. This method is efficient and effective, and sheds light on mining and exploring web structures. Mari et al. [108] experimented three dimensionality reduction methods with a selection of distance measures and showed that after dimensionality reduction into small target dimensionality, such as 10 or below, the superiority of cosine measure does not hold anymore. Malik et al.[109] presented an efficient iterative partition clustering method named CDIM that maximizes the sum of discrimination information provided by documents. A key advantage of CDIM is that its clusters are identified by their highly discriminating terms with high semantic relatedness to their clusters' contexts.

2.5 Survey on Cluster labeling

Good features in a cluster play a vital role during cluster labeling. The features can be found by many ways like extracting the most frequent terms occurring in a cluster, by considering the top weighted terms in the cluster centroid etc. Working in this direction, Glover et al. [110] showed that the extracted label from the extended anchor text of a web page performs better than labels that are extracted from the content of the web page. Extended anchor text refers to the words which appear near links to the target page. For experimental work, DMOZ and Yahoo directory datasets are used. Clusters generated using extended anchor text produce better features and these features are found to be more consistent with the summary of the document. Heerden et al. [111] in their paper suggested an unsupervised labeling method named as 'unsupervised weight-based cluster labeling' using self-organizing map. Their approach assigned significant weights for neurons cluster and then constructing sub-labels by linking those weights. Finally, cluster labels are generated from those sub-labels. Ping et al. [112] proposed a cluster labeling method based on convex decomposition which used the topological property of the dataset. Comparative experiments and time complexity suggests that their approach greatly improves both the quality of the clusters as well as the efficiency. Empirical results on various datasets used in their work like sunflower, wine, iris, ring etc. shows the efficiency of their approach.

Turel et al. [113] introduced the cluster labeling method using cover coefficient-based and sequential k-means algorithm. Cluster labeling is done based on term weighting. A new metric called $Sim_f\text{-measure}$ has been used to measure the effectiveness of the cluster labeling which turns out to be good. For experimental work, AMBIENT and ODP-239 datasets have been used. Comparative study is done on the proposed method to evaluate the relative performance with respect to the Lingo and Suffix tree clustering. Similar to the community mining in the social network, Li et al. [114] in their paper have generated a hierarchy of document clusters which are typically coherent. Cluster labeling is done using the betweenness centrality measure of the term which co-occurs in the network. For experimental work, they constructed a dataset using Google. Padua et al. [115] proposed a labeling technique called Genetic labeling technique (GLM) for association rule clustering. The optimization function of the GLM method is balancing the values of the measures in order to evaluate the cluster labeling. Their method is based on genetic algorithm and gives a very good performance compared to some state-of-the-art methods.

Tholpadi et al. [116] in their research developed a variational approach to show that cluster labeling problem can be handled effectively by multilingual topic models. They designed a novel Scatter/Gather system called ShoBha for multilingual corpora. Empirical results on the entire overlapping Wikipedia of English, the Canadian Hansards corpus, Hindi and Bengali articles, and a trilingual news corpus having 41,000 articles, signifies the effectiveness of their system. Lee et al. [117] in their work used some invariant topological properties of a trained kernel radius function for developing the cluster labeling. Their complexity analysis and experimental results demonstrate the accuracy of their approach. An effective algorithm proposed by D'Orangeville et al. [118] for cluster labeling used Support Vector Clustering (SVC). Their work understands the functionality describing the SVC cluster contours and found the interconnection paths between critical points separating distinct cluster contours. Experimental results on synthetic dataset sampled from 15 uniform density functions signifies the quality of their work. Lopes et al. [119] suggested a cluster labeling mechanism using artificial neural network. They have used both supervised and unsupervised learning along with a discretization model to label the clusters. Iris, Seeds, Glass, Scientia.Net databases are used for their experimental work. Their results labeled clusters with an average of above 88.79% of elements correctly. Li et al. [120] developed a combined approach of both linguistic and statistical perspectives to label the clusters. Performance of their approach is evaluated on 20-Newsgroups

and NewsMiner (Chinese) datasets. Experimental results demonstrate that their algorithm can generate good quality clusters and significantly outperform other existing methods.

Wikipedia is so diverse that it can be considered as a small web in itself. Nayak et al.[121] in their paper suggested an approach to label the Wikipedia clusters which consumes limited resources and time. The obtained results contained thousands of clusters and are evaluated against an external data which turns out to be good. Roitman et al.[122] used two extended fusion methods named CombSUM (CLD) and CombMNZ (CLD) for labeling the clusters. 20-Newsgroups and DMOZ datasets have been used for the experimental work. They concluded that CLD method is a good method compared to all the methods used in their work for labeling the clusters. Geraci et al. [123] have described the working of the meta search engine Arnil. External knowledge is not required for the of clustering and cluster labeling and it is done on the fly by processing only the snippets provided by the search engine. Clustering is done by using furthest point first algorithm and by combining intra and inter cluster labeling. Evaluation is done against Vivisimo, a known industry standard and Arnil outperforms it by 10%. Cluster labeling by using concepts in a machine-readable dictionary has been proposed by Fukomoto et al. [124]. They have made the assumption that the terms in the cluster content have the same hypernym. Experimentally this method is found to improve labeling accuracy. Ji et al. [125] have suggested fuzzy set approach to improve the cluster labeling. A similarity index based on fuzzy binary relation between fuzzy set of the cluster and class is used. This method is proven to be faster than Bayesian maximum likelihood classifier. The idea of creation of the most significant word list to discriminate each document group from the others has been proposed by Moura et al. [126]. This list is generated from the hierarchy of document groups. This list can be used for the task of cluster labeling.

Structural property in documents like hierarchical structure has been utilized by Muhr et al. [127]. Cluster labeling information like relations of sibling, parent, and child have been used in cluster labeling task. Common approaches like maximum minimum term frequency, information gain, Jensen Shannon Divergence and Chi-Square method have used these relationships for labeling. Accuracy is improved with the use of relations as compared to use of these conventional approaches directly. The resultant hierarchy is automatically labeled. Nouns hierarchy of hypernym automatically from text are constructed as given in [128]. For labeling, a set of hypernyms are extracted with the use of linguistic patterns from the text.

CHAPTER 3

EXTREME LEARNING MACHINES IN TEXT CLASSIFICATION

3.1 Introduction

In this chapter, the effectiveness of Extreme Learning Machine (Appendix B) and Multilayer ELM (Appendix C) in the domain of text classification are studied and compared with the existing relevant techniques such as Support Vector Machine (SVM) (Appendix A), which is one of the most popular and effective technique for classifying the text documents. This chapter highlights the importance of ELM in the field of text classification by testing the classifiers based on different interpretations of ELM, analyzing their performances, and studying which existing feature selection techniques are most suited to improve their performances. For multi-class classification problem using ELM, *One-Against-All (OAA)* and *One-Against-One (OAO)* techniques are studied. A multilayer implementation of ELM called ML-ELM which is inspired on deep learning network also has been studied extensively. To the best of our knowledge, no previous research work has used deep learning in the form of ML-ELM for text classification. Hence, the aim of this chapter is to introduce ELM and ML-ELM in the field of text classification and test their performances in order to justify the significance of deep learning.

3.2 Methodology

A detailed design on how ELM is used (*ELM-OAA*) and (*ELM-OAO*) to handle the multi-class classification problem on text data is discussed here. The discussion also includes the design issues of ML-ELM for classification of text data.

3.2.1 Document pre-processing and indexing

Consider a corpus P consisting of different classes $C = \{C_1, C_2, \dots, C_m\}$ of documents. Documents in each class are first parsed and tokenized, stop-words are removed, nouns are selected as the keywords using natural language toolkit (nlk)¹, ignoring other parts of speeches such as verbs, adjectives, adverbs and pronouns and then each document is represented as term vectors in the vector space over the system's vocabulary. For each keyword, TF-IDF values are calculated and the inverted indexes are obtained. Table 3.1 represents the corpus P using term-doc matrix (X) of dimension $n \times N$, where each x_{ij} is the TF-IDF value of i^{th} term (feature) with respect to the j^{th} document.

Table 3.1: Term-document matrix

	d_1	d_2	d_3	...	d_N
x_1	x_{11}	x_{12}	x_{13}	...	x_{1N}
x_2	x_{21}	x_{22}	x_{23}	...	x_{2N}
x_3	x_{31}	x_{32}	x_{33}	...	x_{3N}
.
.
.
x_n	x_{n1}	x_{n2}	x_{n3}	...	x_{nN}

3.2.2 ELM One-Against-All

ELM-OAA is a regular single ELM classifier that performs multi-class classification by training the hidden layer neurons using supervised learning based on N arbitrary distinct training documents (x_j, y_j) of dimension $R_n \times R_m$, where x_j is the input and y_j is the output feature

¹<http://www.nltk.org/>

vector of j^{th} document. The entire process is illustrated in Figure 3.1 and the following steps discussed the training and testing of ELM-OAA mechanism.

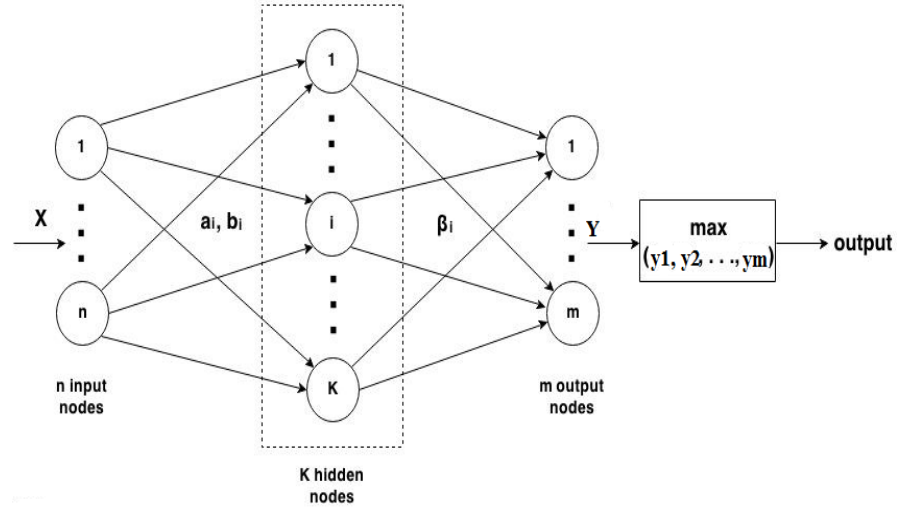


Figure 3.1: ELM One-Against-All

1. *Initialization of input layer, input weight, biases and output layer:*

The number of nodes in the input layer is n , where n is the number of features. Next, the input weight ' a ' and hidden biases ' b ' are selected randomly. In ELM-OAA, the output layer consists of multiple nodes, equal to the number of distinct classes m . Additionally, for each training document x_j , the target output y_j is represented by m bits i.e. (y_{j1}, \dots, y_{jm}) , and for a pattern of class i , only y_{ji} is '1' and rest of the bits are '0'. These are the basic characteristics of ELM-OAA and is the traditional way a simple ELM classifier solves the multi-class classification problem.

2. *Hidden layer matrix construction:*

Next, for N training documents (x_j, y_j) of dimension $R_n \times R_m$ and K hidden nodes, the hidden node parameters $(a_i, b_i, i = 1, \dots, K)$ are randomly assigned. The hidden node output matrix H is computed using the activation function g as mentioned below:

$$g_K(x_j) = \sum_{i=1}^K \beta_i g_i(x_j, a_i, b_i) = y_j \quad (3.1)$$

where $j = 1, \dots, N$.

3. *Computation of output weight (β):*

The output weights β is computed using the following equation:

$$\beta = H^+ Y \quad (3.2)$$

where, H^+ represents the moore-penrose inverse.

4. *Classification of a test document:*

For an input text document x , the ELM-OAA classifier produces an output vector given as:

$$g_K(x) = \sum_{i=1}^K \beta_i g_i(x, a_i, b_i) = y \quad (3.3)$$

which contains m output nodes, ($y = y_1, y_2, \dots, y_m$), and the output node with maximum value indicates the class to which x belongs.

3.2.3 ELM One-Against-One

One of the common approaches for solving the multi-class classification problem is to break up the problem into multiple two-class problems based on the inherent relationships between the classes in training documents. The following steps discuss the techniques to implement the ELM-OAO scheme on text data and the entire process is illustrated in Figure 3.2.

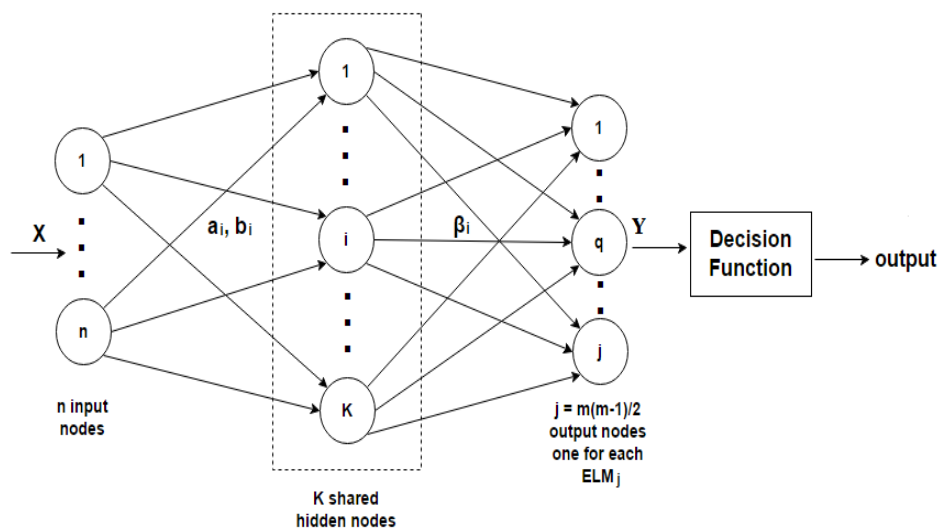


Figure 3.2: ELM One-Against-One

1. *Initialization of input layer, input weight, biases and output layer:*

Initialization of input layer, weight and biases are similar to ELM-OAA. But for output layer, the m classes are decomposed in a pairwise manner to give $m(m-1)/2$ combinations of class pairs and each is trained by one ELM classifier [ELM $_q(j, s)$, $q = 1, 2, \dots, r$ ($r = m(m-1)/2$); $j = 1, 2, \dots, (m-1)$; $s = j+1, \dots, m$].

2. *Hidden layer matrix construction and computation of output weight (β):*

Construction of hidden layer and computation of output weight (β) are same as described in ELM-OAA.

3. *Training the ELM classifier:*

For each ELM $_q(j, s)$, the input and output document used for training are the only documents that are related to both j and s , and all other documents are ignored. In order to differentiate between two distinct classes, the single output node of the ELM $_q(j, s)$ is labeled as '1' for all the documents belonging to class j and '-1' for all those belonging to class s . Since for a m -class problem, we have $r=m(m-1)/2$ binary ELM classifiers, in order to simplify the implementation, the ELM-OAO is considered as a combination of r SLFNs with a common set of shared hidden nodes, which helps to make the network structure more compact. Each of the r ELM binary classifiers are trained independently using their corresponding training document from the collection of r sets of independent training documents (x_q, y_q , $q = 1, 2, \dots, r$). Since each of the r ELM classifiers is trained independently, once the q^{th} group training document (x_q, y_q) is fed to the classifier, the weight β_q connecting the q^{th} binary classifier output neuron with the shared hidden nodes is learned without any affect on the other binary classifier output weights $\beta = [\beta_1, \dots, \beta_{q-1}, \beta_{q+1}, \dots, \beta_L]^T$. Hence,

$$\beta_q = H_q^\dagger y_q \quad (3.4)$$

4. *Classification of a test document:*

The r output nodes are assigned one for each of the r binary ELM classifiers. In order to decide an input document x belongs to which of the m classes, the output values from these r nodes are fed into a decision function. Here, the decision function that employs a loss based approach for decoding the class label that is most consistent with the output y is used, such that the input x is assigned the class label i , if for the document (x, i) ,

the loss is minimum among all the class labels ($i = 1, 2, \dots, m$). The total loss for a document (x, i) is calculated in the following way:

$$d_L(M(i), y(x)) = \sum_{q=1}^r L(M(i, q), y_q(x)) \quad (3.5)$$

Here, M is $m \times r$ matrix, such that a distinct class pair (j, s) is included in each column which has '+1' in row j , '-1' in row s and the others are zero. The function L is the exponential loss function, and $y_q(x)$ represents the output value of the q^{th} binary ELM classifier. Thus, the final class label that is assigned to the document x is the class i for which the total loss based on the equation 3.5 is minimum.

3.2.4 Multilayer ELM

ML-ELM is an artificial neural network having multiple hidden layers. It combines ELM and ELM-autoencoder (ELM-AE) and hence contains all the features of ELM. The design and architecture of Multilayer ELM relies on the mechanism of deep learning and it is shown in Figure 3.3. As already mentioned that in an ELM network for N training documents (x_j, y_j) and K hidden nodes, we have :

$$g_L(x_j) = \sum_{i=1}^K \beta_i g_i(x_j, \mathbf{a}_i, \mathbf{b}_i) = y_j \quad (3.6)$$

The following steps discuss how the text classification is done using ML-ELM.

1. *Initialization of input layer, input weights, biases and output layer:*

Initialization of input layer, weight and biases are similar to ELM except the weight and biases are orthogonal which improves the performance of ELM-AE [129]. The output layer consists of multiple nodes equal to the number of distinct classes m .

2. *ELM-AE Hidden layer matrix construction :*

For each hidden layer, the output feature vector is same as the input feature vector. The random hidden node weights and biases are taken as orthogonal and the input feature vector is mapped to the higher, lower or equal dimensional space of hidden nodes through

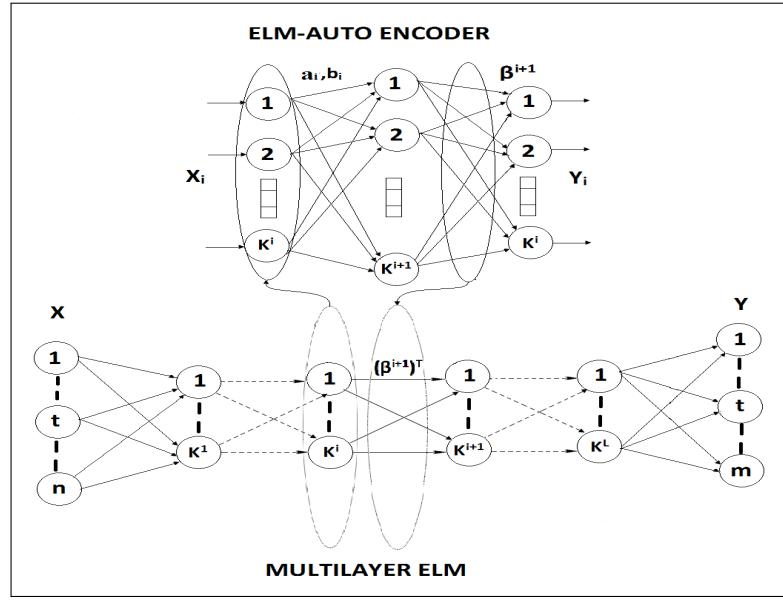


Figure 3.3: Multilayer ELM

the orthogonal hidden weights, $\mathbf{a} = [a_1, a_2, \dots, a_K]$ and the orthogonal hidden biases, $\mathbf{b} = [b_1, b_2, \dots, b_K]$ using the following equation:

$$\mathbf{h} = g(\mathbf{a} \cdot \mathbf{x} + \mathbf{b}), \quad \mathbf{a}^T \mathbf{a} = \mathbf{I}, \quad \mathbf{b}^T \mathbf{b} = 1 \quad (3.7)$$

3. Computation of output weight β :

The value of β is computed as follows:

- i. if $n > K$ (*Compress representation*: mapping the features from a higher dimension of input signal space to a lower dimension of feature signal space)

then

$$\beta = \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T \mathbf{H} \right)^{-1} \mathbf{H}^T \mathbf{X} \quad (3.8)$$

where, C is used as the scaling parameter to adjust the experiential and structural risk. $\frac{\mathbf{I}}{C}$ is the regularization term which improves the generalization performance and makes the solution more robust [66]

- ii. if $n = K$ (*Dimension of equal length representation*: representing the features of training dataset in an equal dimensional space i.e. the dimensions of input signal space and feature signal space are equal), then

$$\beta = \mathbf{H}^{-1} \mathbf{X} \quad (3.9)$$

- iii. if $n < K$ (*Sparse representation*: mapping the features from a lower dimension of input signal space to a higher dimension of feature signal space), then

$$\beta = \mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{X} \quad (3.10)$$

4. *Training the classifier:*

ML-ELM makes use of ELM-AE to train the parameters in each layer. In other words, the hidden layer weights of ML-ELM are initialized by ELM-AE from layer to layer using unsupervised learning, and ML-ELM hidden layer activation functions can be either linear or non-linear piecewise. All output weights are determined analytically. The output of the i^{th} hidden layer of ML-ELM can be obtained from the output of $(i-1)^{th}$ hidden layer and the output weight of β^i of the i^{th} hidden layer. The output weight of β^i of the i^{th} hidden layer is obtained layer wise from the ELM-AE, and its transpose. ML-ELM with ' K ' hidden nodes can be represented as

$$\mathbf{H}^n = g((\beta^n)^T \mathbf{H}^{n-1}) \quad (3.11)$$

For $n = 0$, the input layer \mathbf{X} can be considered as the 0^{th} hidden layer. The transformations of the data from the feature space of one layer to the next, and so on are carried out as shown in equation 3.11, until it reaches the last hidden layer before the output layer \mathbf{Y} . The final output matrix \mathbf{Y} can be obtained by computing the results between the last hidden layer and the output layer using the regularized least squares technique [130].

5. *Classification of a test document:*

In a similar way like ELM, the feature vector of the test document \mathbf{x} is fed to the ML-ELM, and with the known value of β in each layer, it computes the next hidden layer feature vector and the process is repeated till the last layer and finally, one of the output layer node receives the maximum value to which the test document \mathbf{x} is assigned.

3.3 Experimental Analysis

The performance of three different implementations of ELM against other traditional classifiers are tested experimentally. For this purpose, from all categories of 20-Newsgroups (Table ??), 11,293 documents are considered and similarly from all categories of DMOZ (Table ??), 24,410 documents are considered to generate the corpus. Three different traditional feature selection techniques such as Chi-Square (Appendix D), Bi-normal separation (BNS) (Appendix D) and Information gain (IG) (Appendix D) are used to select the important features from the corpus of each dataset and those features are used as the input feature vector (length lies between 500 to 2500). Accordingly, the hidden layer nodes are set which is higher than the length of the input feature vector ($n < K$). The number of hidden layers on both datasets are set to 3 (decided empirically)² using ML-ELM.

One can observe how the accuracy of simple ELM, i.e. ELM (OAA), ML-ELM and SVM varies based on the feature selection methods used, and for different number of features within each method starting from 500 to 2500 incrementing 500 each time. It is also observed from the experiment that the accuracy of ELM-OAO is lower than ELM-OAA.

Based on the experimental results on both DMOZ and 20-Newsgroups, it can be seen that Chi-Square feature selection technique gives better results. In 20-Newsgroup dataset, the Chi-Square technique outperformed the IG and BNS for all features ranges from 500-2500, however in the DMOZ dataset, its superior performance is witnessed at the higher feature range of 2500. ML-ELM gives the overall best accuracy using 2500 features under Chi-Square for both datasets, dominating over the other two ELM techniques and SVM. Thus, one can say that the Chi-Square feature selection technique is one of the most suitable to boost the ELM's performance, especially when higher number of features are considered.

Performance Evaluation:

It is intended to show how suitable is ELM to accomplish the task of text classification with high accuracy comparable to or better than the current state-of-the-art techniques. As can be seen from the results, even ELM-OAA almost always outperforms the SVM or is neck to neck with it in the worst case. ELM not only performs better than SVM when the feature

²the result for which the proposed approach obtained the better performance

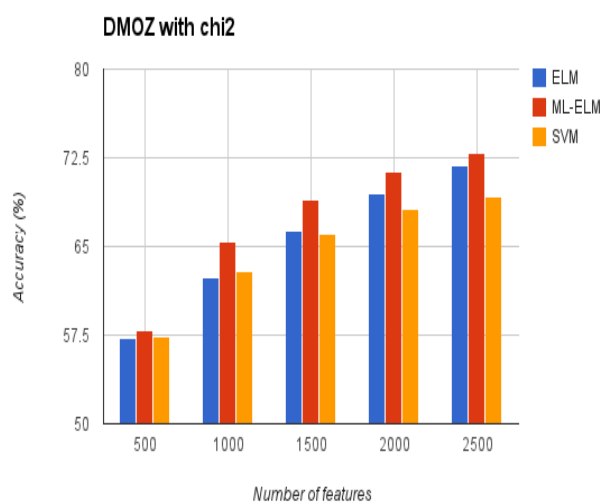


Figure 3.4: dmoz-chi-square

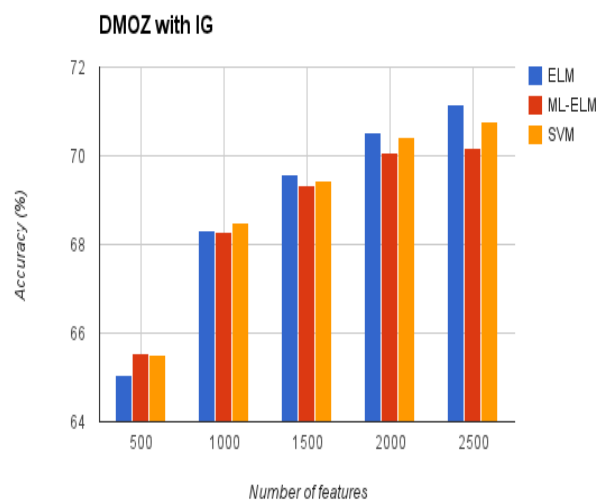


Figure 3.5: dmoz-ig

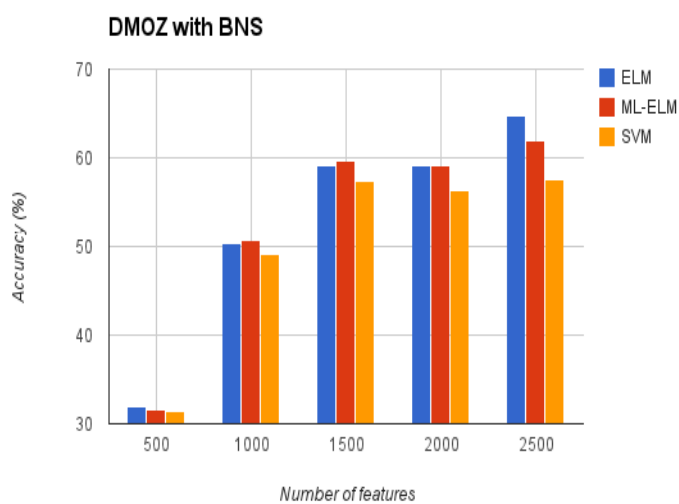


Figure 3.6: dmoz-bns

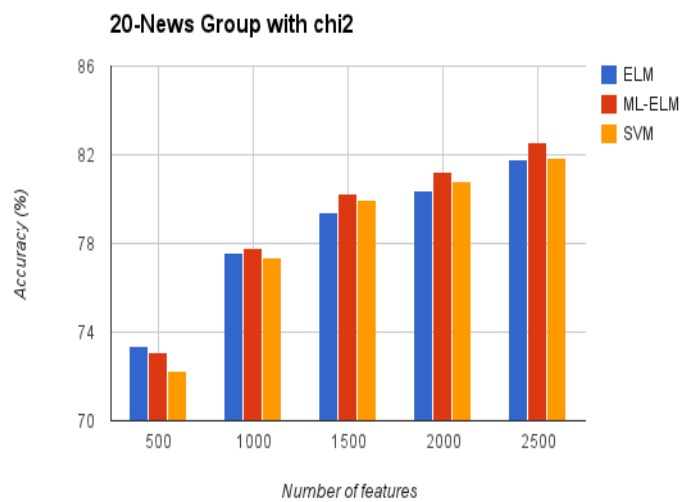


Figure 3.7: 20ng-chi-square

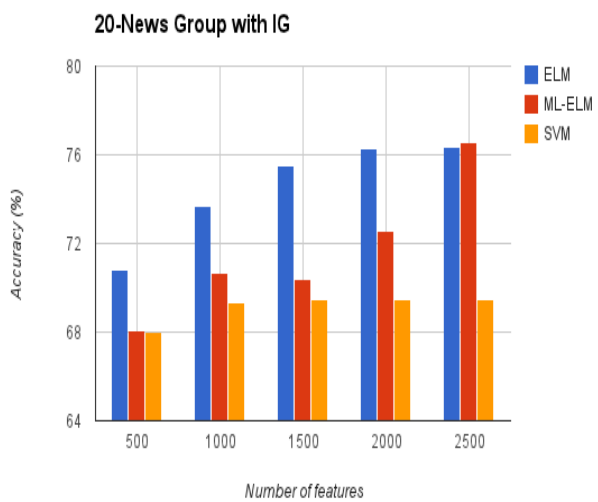


Figure 3.8: 20ng-ig

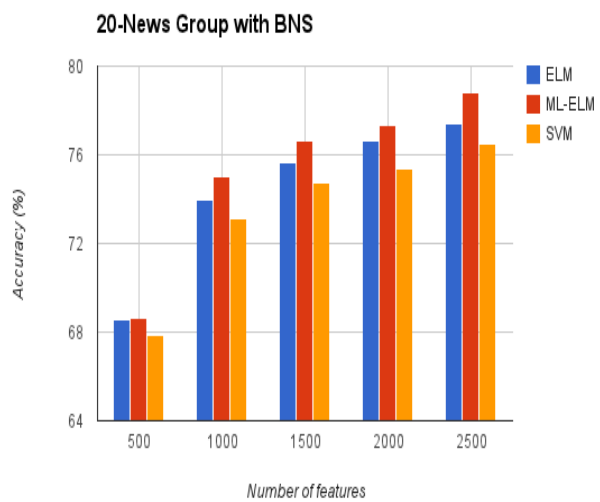


Figure 3.9: 20ng-bns

selection technique is a suitable one, but in fact performs significantly better in comparison even when the feature selection technique is not suitable. Thus, it is less vulnerable to poor feature selection than SVM. From the figures, it can be seen that ML-ELM in most cases is better than the simple ELM, and its performance is more affected by the suitability of the feature selection technique with the drop or rise in accuracy being somewhat compounded over that found in simple ELM. Thus, ML-ELM performs better than ELM when feature selection is favorable and vice-versa. Figures 3.4 - 3.6 demonstrate the accuracy of ELM and ML-ELM over SVM for DMOZ with different selection techniques. Similarly, Figures 3.7 - 3.9 demonstrate the accuracy of ELM and ML-ELM over SVM for 20-Newsgroup with different selection techniques. As can be seen from the figures that ELM outperforms SVM in almost all cases, and the best performance for each dataset is achieved using ML-ELM having 82.55 on 20-Newsgroups, and 72.83 on the DMOZ datasets over ELM and SVM. Table 3.2 and 3.3 show the accuracy and F-measure (input feature vector length 2500) comparisons of ELM and ML-ELM with other traditional classifiers. Here, one can observe that the accuracy and F-measure of ML-ELM dominated almost all the classifiers for both datasets with different feature selection techniques. What is important to note is that even the simple ELM almost always outperforms the SVM (in accuracy) or is at worst comparable in performance (in F-measure). The superior performance of the ML-ELM over the traditional state-of-the-art SVM classifier and other existing relevant classifiers proves its importance as a highly successful and suitable technique for text classification. The probable reasons of why ML-ELM performance is better compared to other traditional classifiers are discussed in Chapter 4.

Table 3.2: Accuracy comparisons of different state-of-the-art classifiers

Classifier	20-Newsgroups (Accuracy-%)			DMOZ (Accuracy-%)		
	Chi-Square	BNS	IG	Chi-Square	BNS	IG
ELM-OAA	81.78	77.43	76.37	71.87	64.79	71.17
ML-ELM	82.55	78.84	76.58	72.83	61.91	70.17
ELM-OAO	73.88	70.03	62.04	69.11	54.08	63.29
SVM (Linear SVC)	81.87	76.52	69.48	69.20	57.47	70.78
SVM (Linear kernel)	75.57	75	46.61	66.19	55.91	66.05
KNN (K=5)	52.44	56.73	46.41	45.80	33.59	26.35
Gussian NB	68.97	58.51	58.30	40.05	30.48	48.87
Multinomial NB	79.50	77.89	60.54	53.53	58.95	63.40
Bernoulli NB	78.31	74.37	68.95	49.19	50.33	48.65
Decision Trees	59.57	61.74	56.94	41.80	32.33	49.69

Table 3.3: F-measure comparisons of different state-of-the-art classifiers

Classifier	20- Newsgroups (F-Measure-%)			DMOZ (F-Measure-%)		
	Chi-Square	BNS	IG	Chi-Square	BNS	IG
ELM-OAA	77.54	78.61	77.44	68.24	59.96	68.98
ML-ELM	81.39	81.53	81.34	68.32	60.68	71.47
SVM (LinearSVC)	80.20	79.45	80.33	68.20	59.00	69.19
SVM (Linear kernel)	72.57	72.00	63.48	62.39	54.11	64.35
KNN (K=5)	53.45	55.28	48.64	41.72	38.62	35.45
Multinomial Naive Bayes	69.50	72.89	67.76	54.34	54.67	53.40
Gaussian Naive Bayes	58.53	58.62	59.33	41.35	36.38	42.63
Bernoulli Naive Bayes	68.31	69.37	64.25	48.27	51.66	47.47
Decision Trees	58.23	60.40	57.89	42.67	42.45	48.85

3.4 Summary

This chapter analyzed and compared the performance of SVM, ELM and ML-ELM classifiers to demonstrate the potential of the ELM (for accuracy) and ML-ELM (both accuracy and F-measure) as a highly successful and suitable technique for text classification. It is also observed from the experimental results that ML-ELM can outperform other existing classifiers. Additionally, our results can serve as complementary knowledge to further strengthen the understanding of the essential relationship between SVM, ELM and ML-ELM. Since the ELM and ML-ELM classifiers outperformed the traditional classifiers including state-of-the-art SVM classifier in the majority of cases, and achieved the best performance overall for both datasets, it shows the importance of the ML-ELM classifier in being able to achieve high performance, which is comparable and most often greater than the most popular and cutting edge methods in the domain of text classification. The state-of-the-art SVM classifiers like LinearSVC, with which we have compared the ELM's performance, have been tuned and perfected using penalties like L1, L2 to further improve the accuracy by reducing the error margin.

After experimenting ML-ELM on different benchmark text classification datasets by using different traditional feature selection techniques, next we took interest to develop two new feature selection techniques on which the performance of ML-ELM is tested. Detailed discussions on these two feature selection techniques and their performances are done in the next chapter.

CHAPTER 4

FEATURE SELECTION TECHNIQUES FOR TEXT CLASSIFICATION

4.1 Introduction

The incredible increase of online documents in digital form on the web has renewed the interest in text classification. However, the poor quality of feature selection, extremely high dimensional feature space and complexity of natural languages became the roadblock for classification process. Hence, in order to work with the text data and to increase the efficiency of the classifier, choice of quality features is of paramount importance. Feature selection is used for three main reasons: simplification of models for making them easier to interpret by researchers/users, shorter training times and enhanced generalization by reducing overfitting. The central premise when using a feature selection technique is that the document contains many features which may either be redundant or irrelevant, and can thus be minimized without incurring much loss of information.

To address these issues, this chapter introduces two novel feature selection techniques for efficient text classification named *k-means and Wordnet based feature selection* and *Combined Cohesion, Separation and Silhouette coefficient based feature Selection*. In both the proposed techniques, the performance of ML-ELM has been tested on different benchmark datasets and the results found are promising.

4.2 *k*-means and Wordnet based feature selection

k-means and Wordnet based feature selection (*KWFS*) finds the relationships between the terms so as to ensure that any redundant or irrelevant features are discarded and only the important (top) features are selected from a given corpus consisting many documents from different classes. The top features are selected using the traditional *k*-means clustering technique [27], where a class (named as supervised cluster) is divided into a number of sub-classes (or sub-clusters) so that it can further bring more similar documents into the same group which in turn strengthen the relationship between the features (or keywords) in the sub-cluster. From each sub-cluster, the top features are selected using cosine-similarity after forming the synonym list (Wordnet¹ is used for this purpose) of each feature (Table 4.1) selected based on their Bi-normal separation (BNS) score (appendix D) (the best performing measures in the probit classifier [131]). Finally, all the top features of each sub-cluster are combined to form a reduced feature vector. To make the above discussion more formal, detailed steps are given in

Table 4.1: Synonym list of keywords

Keyword	Synonym list
amazing	astonishing, astounding, extraordinary, fabulous, fantastic, improbable, incredible, unbelievable, wonderful,
begin	ample, broad, gigantic, great, enormous, tall, huge, substantial, immense, vast, gargantuan, tremendous, colossal, large, sizable, grand, mammoth, astronomical, expansive, spacious, stout, titanic, mountainous
calm	aloof, composed, quiet, peaceful, still, mild, serene, smooth, tranquil, collected, unruffled, level-headed, unexcited, detached
describe	portray, recount, report, characterize, represent, narrate, picture, relate, record
great	powerful, worthy, distinguished, noteworthy, remarkable, considerable, grand, much, mighty
important	notable, significant, primary, necessary, vital, critical, indispensable, distinguished, valuable, essential, principal, considerable, famous, well-known
plan	plot, contrivance, design, scheme, method, map, diagram, draw, procedure, arrangement, intention, way, device, blueprint
think	assume, believe, consider, deem, judge, contemplate, mediate, reflect

the next section and for implementation purpose, two algorithms (Algorithm 4.1 and 4.2) are developed.

¹<https://wordnet.princeton.edu/wordnet>

4.2.1 Methodology

Step 1 *Document pre-processing:*

Consider a corpus P consists of n number of classes i.e. supervised clusters $C = \{C_1, C_2, \dots, C_n\}$ of documents. The documents are pre-processed (as discussed in Chapter 3) and keywords are selected as nouns to create a dictionary of the corpus.

Step 2 *Term-document matrix generation:*

The documents of each C_i are then converted to vectors using vector space model and these vectors are aggregated to form the term-document matrix. BNS value of every keyword is computed for each C_i of the corpus before generating the sub-clusters of C_i .

Step 3 *Sub-cluster generation:*

k -means clustering algorithm is run on term-vectors of C_i and it returns k sub-clusters ($C_i = \{c_1, c_2, \dots, c_k\}, \forall C_i \in C$) where each sub-cluster represents a concept of related terms or keywords.

Step 4 *Top keywords selection:*

For each $c_j \in C_i$, the centroid (C') of c_j is computed and then the cosine similarities² of all the keywords of c_j with the centroid C' are calculated. Selection of top keywords which can represent c_j is done using the following steps:

- (i) Select a keyword X having the highest BNS score from the keyword-list of c_j and using Wordnet prepare an initial synonym-list for X .
- (ii) Next, check those keywords which are present both in the keyword-list of c_j and synonym-list of X . If any such keywords are found then create a new synonym-list (new-synonym-list) of X and add all those keywords one by one to the new-synonym-list of X , after discarding them from the keyword-list of c_j . In this way, the new-synonym-list of X is created. Now, discard the initial synonym-list of X .
- (iii) Repeat step 4(i) and (ii) till the keyword-list of c_j gets exhausted. At the end, the new-synonym-list for those keywords which are selected based on their BNS values

²<https://radimrehurek.com/gensim/tutorial.html>

Algorithm 4.1 Generating sub-clusters

```

1: Input:Pre-processed text dataset of a cluster,  $C_i, i \in [1, n]$ 
2: Output: Sub-clusters of  $C_i$  with cosine-similarity
3: content[] ← contains all the keywords
4:  $c_j \leftarrow$  sub-clusters return by  $k$ -means algorithm,  $j \in [1, k]$ 
5: tf_idf[][] ← stores the tf-idf values of each keyword
6:  $cs \leftarrow$  stores the cosine-similarity value of a keyword
7:  $S[][][] \leftarrow$  stores the cosine-similarity values of keywords of each  $c_j \in C_i$ 
8: for all  $C_i$  do
9:   for all  $d \in C_i$  do
10:     content[] ← read all keywords of  $d \in C_i$ 
11:   end for
12:   tf_idf[][] ← TfIdfVectorizer(content[]) // compute the tf-idf values of keywords
13:    $c_j \leftarrow k$ -means(no_of_sub-clusters, tf_idf[])
14:   for all  $c_j \in C_i$  do
15:     for all Term-Vector  $\in c_j$  do
16:        $cs \leftarrow$  cosine_similarity(Term-Vector, Centroid of  $c_j$ )
17:        $S[\text{cluster\_no}][\text{sub-cluster\_no}][\text{term-vector\_no}] \leftarrow cs$ 
18:     end for
19:   end for
20: end for
21: return  $S$ 

```

from the keyword-list of c_j are generated. This gives a list of new-synonym-list of c_j .

- (iv) Select top $m\%$ keywords (determined by experiment) from each new-synonym-list of c_j which have highest cosine-similarity (tightly bound to the centroid of c_j) values. Collect all these top keywords of c_j from each new-synonym-list and discard the rest of the keywords to obtain the reduced feature vector for c_j .
- (v) Repeat step 4 (i-iv) for every $c_j \in C_i$ and at the end, merge all the top keywords of each c_j into a list to obtain the reduced feature vector of C_i .

Step 5 *Final reduced feature vector generation:*

Repeat steps 2 to 4 for each cluster C_i and combine the resulting features after removal of duplicates to obtain the final reduced feature vector of the entire corpus.

Step 6 *Training and performance evaluation of classifiers:*

The final reduced feature vector can then be used to train ML-ELM and other traditional classifiers for text classification. Using the output predictions and the known class labels

Algorithm 4.2 Selecting best features of a sub-cluster

```

1: Input:Sub-Cluster  $c_j$  generated by Algorithm 4.1 with cosine-similarity values of each
   keyword
2: Output:Reduce feature vector ( $FV$ ) of  $c_j$ 
3: Keyword_List( $KL$ )  $\leftarrow \phi$ 
4: Synonym_Listw( $SL_x$ )  $\leftarrow \phi$ 
5: New_Synonym_Listx( $NSL_x$ )  $\leftarrow \phi$ 
6: List_of_List( $LL$ )  $\leftarrow \phi$  //A two dimensional list
7:  $KL \leftarrow$  keywords from all documents  $D \in c_j$ 
8: for all keyword  $X \in KL$  (selected based on their BNS score) do
9:    $SL_x \leftarrow$  all the synonyms of  $X$  found in Wordnet
10:  for all keyword  $U \in KL$  do
11:    if  $U \in SL_x$  then
12:       $NSL_x = NSL_x \cup \{U\}$  //add  $U$  to the synonym_required.list of  $X$ 
13:       $KL = KL - \{U\}$  //drop  $U$  from  $KL$ 
14:    end if
15:  end for
16:   $LL \leftarrow LL \cup NSL_x$  //appended the synonym required list of  $X$  to  $LL$ 
17:   $NSL_x \leftarrow \phi$ 
18:   $KL \leftarrow KL - \{X\}$  //drop keyword  $X$  from  $KL$ 
19:  ( $SL_x$ )  $\leftarrow \phi$ 
20: end for
21: for all  $NSL_x \in LL$  do
22:   select the top m% keywords  $K$ (determined by experiment) having highest cosine-
   similarity values from  $NSL_x$ 
23:    $FV \leftarrow FV \cup K$  //append all the top features into a list
24: end for
25: return  $FV$ 

```

of test document, precision, recall and F-measure are calculated to quantify the performance of different classifiers as follows:

$$Average_{precision} = \frac{\sum_{i=1}^n p_i}{n} \quad (4.1)$$

$$Average_{recall} = \frac{\sum_{i=1}^n r_i}{n} \quad (4.2)$$

$$Average_{F-measure} = \frac{\sum_{i=1}^n f_i}{n} \quad (4.3)$$

where, p_i , r_i and f_i are the precision, recall and F-measure of the i^{th} category and n is the total number of categories in a dataset.

4.2.2 Experimental Analysis

The proposed method is tested on 20-Newsgroups and DMOZ datasets. The algorithm was run on these two datasets separately. For k -means clustering, k was set as 10 (decided empirically). Different classifiers including ensemble classifiers are used for comparison purposes such as Support Vector Machine (LinearSVC), Bernoulli naive-bayes (B-NB), Multinomial Naive-Bayes (M-NB), Gaussian Naive-Bayes (G-NB), Decision Tree (DT), Extra Trees (ET), Gradient Boosting (GB), Random Forest (RF), ELM and ML-ELM. The algorithm also was tested for various input feature vector lengths, with different number of hidden layer nodes used in ELM and a number of hidden layers used in ML-ELM. But for performance purpose, those values of the these parameters are considered for which the proposed approach obtained the best results.

20-Newsgroups dataset:

All 7 categories of 20-Newsgroups dataset are considered for experimental work. Out of 18,846 documents of this dataset (includes 25 empty documents), 11,318 are used for training and the remaining 7,528 are used for testing. The length of the input feature vector is set to 1852. The number of internal (hidden) nodes is set to ‘2000’ for both ELM and ML-ELM with number of hidden layers set as ‘3’ for ML-ELM. For demonstration purpose, the category-wise performance of ELM and ML-ELM are shown in the Tables 4.2 and 4.3, respectively. Figures 4.1 - 4.3 show the average precision, recall and F-measure of different classifiers on 20-Newsgroups dataset respectively. From the Figure 4.3, it is observed that ML-ELM outperformed other established classifiers.

Table 4.2: ELM (20-Newsgroups)

Category	No. of Test Documents	Precision	Recall	F-Measure
Alt	320	0.6201	0.6261	0.6231
Computers	1952	0.8137	0.8444	0.8288
Miscellaneous	390	0.7431	0.7083	0.7253
Recreation	1590	0.7980	0.8189	0.8083
Science	1580	0.7430	0.7059	0.7240
Social	399	0.7610	0.5459	0.6357
Talk	1297	0.7001	0.7499	0.7241
Average	1075	0.7398	0.7142	0.7242

Table 4.3: ML-ELM (20-Newsgroups)

Category	Total Testing Documents	Precision	Recall	F-Measure
Alt	320	0.6242	0.6224	0.6234
Computers	1952	0.8186	0.8429	0.8306
Miscellaneous	390	0.7769	0.7030	0.7381
Recreation	1590	0.9298	0.8140	0.8680
Science	1580	0.8010	0.7015	0.7480
Social	399	0.6299	0.7635	0.6903
Talk	1297	0.7058	0.7608	0.7323
Average	1075	0.7551	0.7440	0.7472

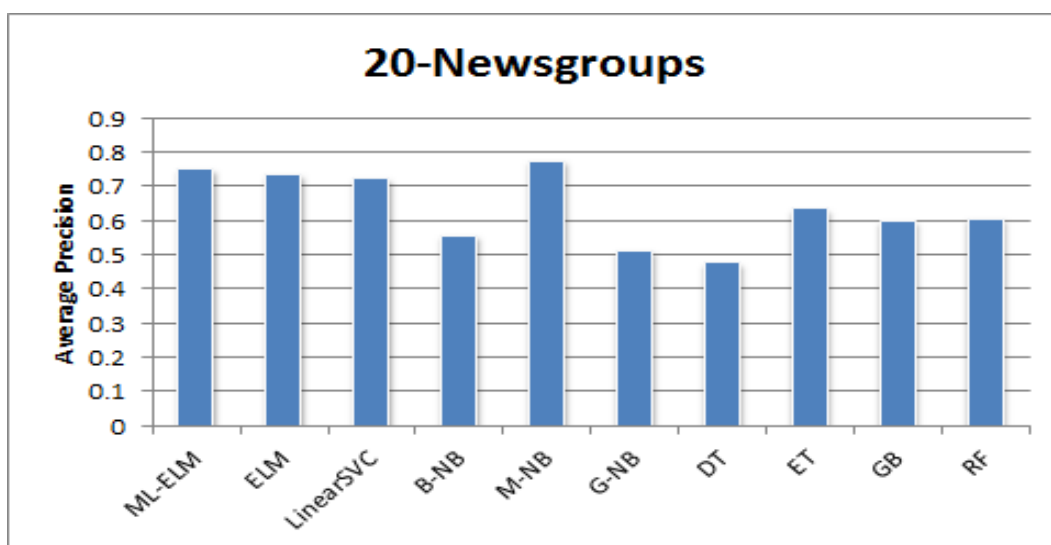


Figure 4.1: Average precision of different classifiers on 20-Newsgroups

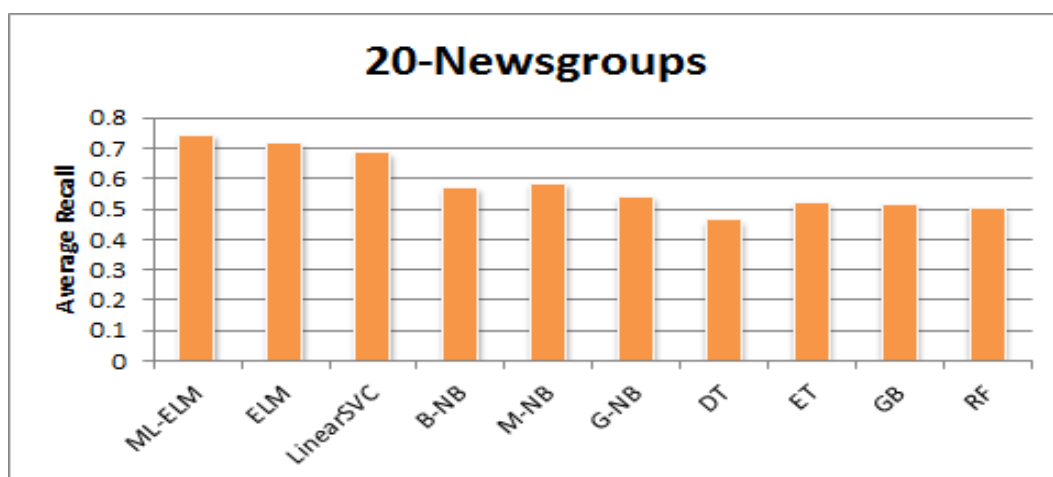


Figure 4.2: Average recall of different classifiers on 20-Newsgroups

DMOZ dataset:

All 14 categories of DMOZ dataset are considered for experimental work. 38,834 documents

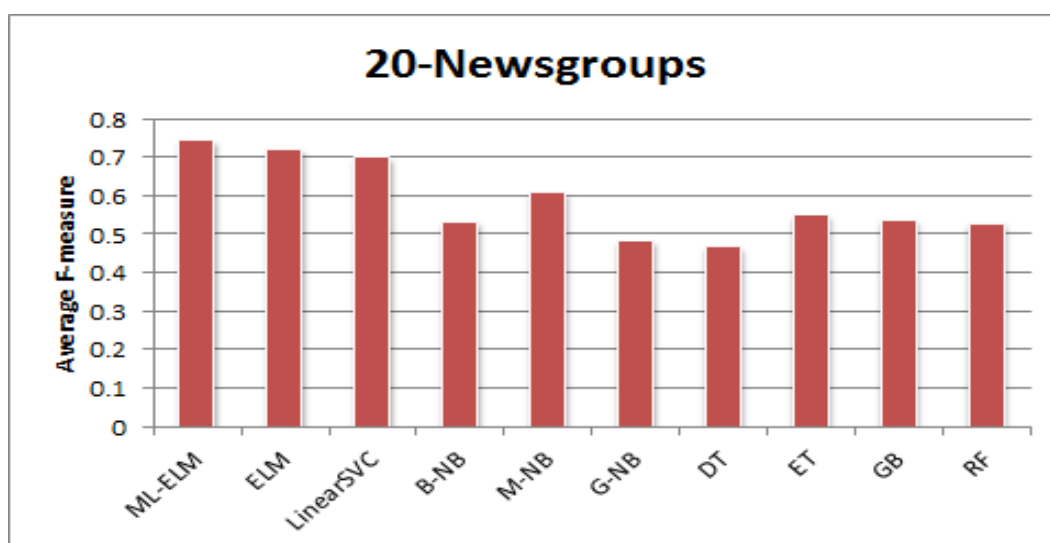


Figure 4.3: Average F-measure of different classifiers on 20-Newsgroups

are used for training and 31,068 documents are used for testing. The length of the input feature vector is set to 2260. The number of internal nodes is set to '2500' for both ELM and ML-ELM with number of hidden layers set as '5' for ML-ELM. For demonstration purpose, category-wise performance of ELM and ML-ELM are shown in the Tables 4.4 and 4.5 respectively. The average precision, recall and F-measure of different classifiers are shown in Figures 4.4 - 4.6. From the Figure 4.6, it is clear that ML-ELM has obtained the better results compared to other classifiers which justifies its prominence in the field of text classification.

Table 4.4: ELM (DMOZ)

Category	No. of Test Documents	Precision	Recall	F-Measure
Arts	1396	0.7388	0.6815	0.7090
Business	3384	0.7556	0.7014	0.7275
Computers	1494	0.7434	0.6912	0.7164
Games	5757	0.6914	0.6845	0.6879
Health	1491	0.6958	0.7019	0.6988
Homes	1405	0.7314	0.6855	0.7077
News	1504	0.7411	0.6518	0.6936
Recreation	1410	0.6969	0.7015	0.6992
Reference	1301	0.7015	0.6716	0.6862
Regional	1307	0.7114	0.6515	0.6801
Science	1390	0.7275	0.6610	0.6927
Shopping	6209	0.7281	0.6525	0.6882
Society	1505	0.7445	0.6871	0.7146
Sports	1515	0.7215	0.6659	0.6926
Average	2219	0.7235	0.6778	0.6996

Table 4.5: ML-ELM (DMOZ)

Category	Total Testing Documents	Precision	Recall	F-measure
Arts	1396	0.7321	0.6819	0.7061
Business	3384	0.7645	0.6923	0.7266
Computers	1494	0.7115	0.7238	0.7176
Games	5757	0.7662	0.7149	0.7397
Health	1491	0.7550	0.6822	0.7168
Homes	1405	0.7139	0.7237	0.7188
News	1504	0.7425	0.6912	0.7159
Recreation	1410	0.7917	0.6823	0.7329
Reference	1301	0.7239	0.6779	0.7001
Regional	1307	0.7732	0.6645	0.7147
Science	1390	0.7882	0.7012	0.7422
Shopping	6209	0.7121	0.7249	0.7184
Society	1505	0.7312	0.6843	0.7070
Sports	1515	0.7442	0.6898	0.7160
Average	2219	0.7464	0.6954	0.7195

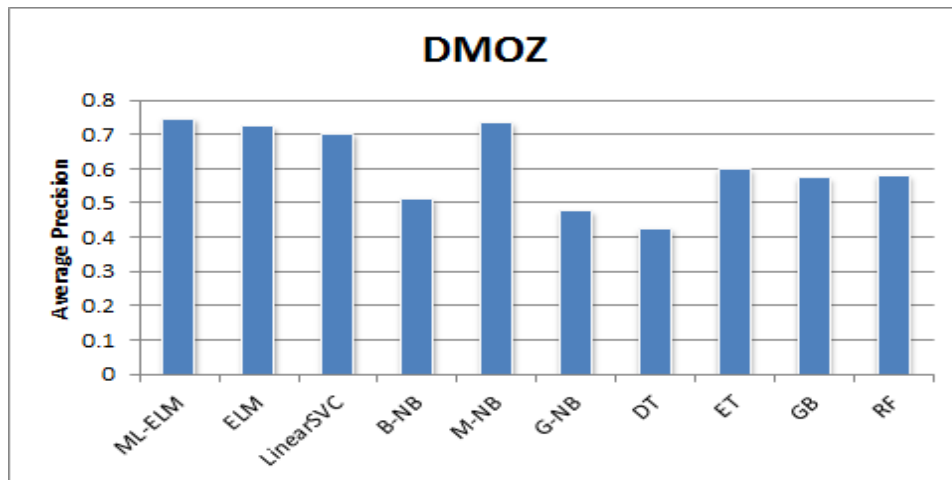


Figure 4.4: Average precision of different classifiers on DMOZ

4.3 Combined Cohesion, Separation and Silhouette coefficient based feature Selection

Combined Cohesion, Separation and Silhouette coefficient based feature selection (CCSS) is an innovative technique for selection of prominent features (or terms) from a given corpus in order to prepare the reduced feature vector. Three important parameters, namely *cohesion*, *separation* and *silhouette coefficient* are used to measure the weight of a term in

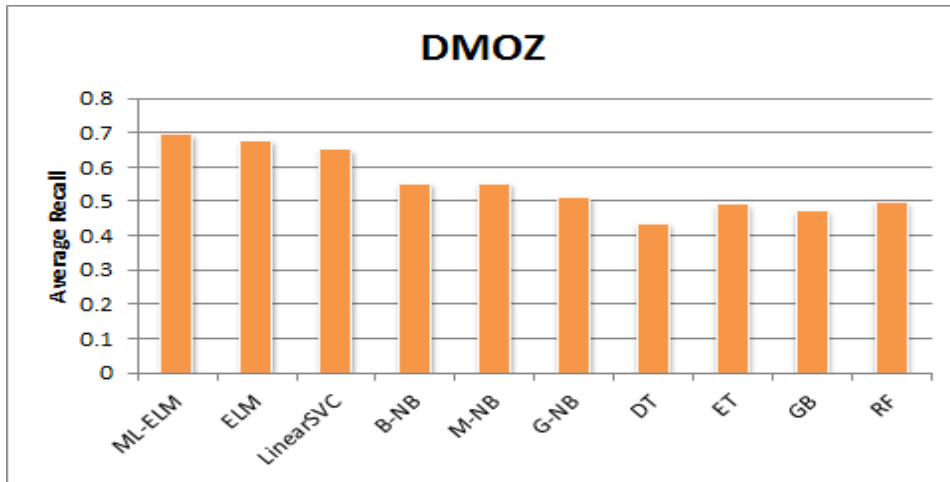


Figure 4.5: Average recall of different classifiers on DMOZ

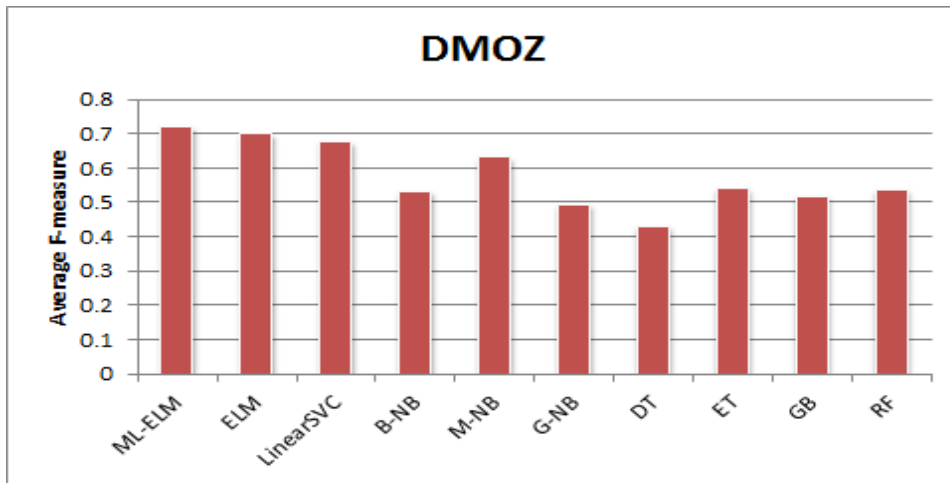


Figure 4.6: Average F-measure of different classifiers on DMOZ

a corpus which is discussed in the next section. In this technique, first, the traditional k -means clustering algorithm is run on a corpus P of terms collected from the documents of all classes and it generates k term-document clusters. Next, in each term-document cluster, the cohesion, separation and silhouette coefficient scores of every terms are calculated which helps to compute the *total-score* of each term. Based on the total score, the terms are ranked in each term-document cluster and the top $m\%$ terms are selected from each cluster which are finally merged together to generate the reduced feature vector. To make the above discussion more formal, detailed steps are given in the next section and for implementation purpose, an algorithm (Algorithm 5.1) is developed.

4.3.1 Cohesion

Cohesion (Appendix F), determines the closeness of terms in a cluster c by computing the distance of each term t from the centroid (mean of all the terms of c) of the cluster c . If the term t is highly cohesive i.e. the distance between the term t and the centroid of the cluster c is very small compared to other terms in c then t defines the cluster c very well. If the term t is present at the border of the cluster c (far away from the centroid of c) then t is poorly cohesive to the cluster i.e. assignment of t to the cluster c is not a good idea. Euclidean distance is used to measure the distance of a term t from the centroid (c') of the cluster c as follows: If $\vec{t} = (t_1, t_2)$ and $\vec{c}' = (c'_1, c'_2)$ then $\|\vec{c}' - \vec{t}\| = \sqrt{(c'_1 - t_1)^2 + (c'_2 - t_2)^2}$, where \vec{t} and \vec{c}' are term and centroid vectors, respectively.

4.3.2 Separation

Separation (Appendix F) determine how distinct or well separated the term t is from other clusters. The separation score of t is the minimum among all the distances of the term t from the centroid of other clusters in which t is not a member. If the distance of t from the centroid of its neighboring cluster is high then t is well separated from that cluster. High separation shows two clusters are well separated and hence, the clustering technique is good. Figure 4.7 shows the cohesion and separation of two terms t and x from its own and neighboring clusters, respectively. As x is near to its centroid and far away from the centroid of its neighboring cluster compared to the term t , hence x will obtain a high cohesion and separation score compared to the term t .

4.3.3 Silhouette Coefficient

The *Silhouette Coefficient* (Appendix F) measures how much a term t is similar to its own cluster (i.e. cohesion) compared to other clusters (i.e. separation).

$$silhouette(t) = \frac{s(t) - c(t)}{\max(c(t), s(t))}$$

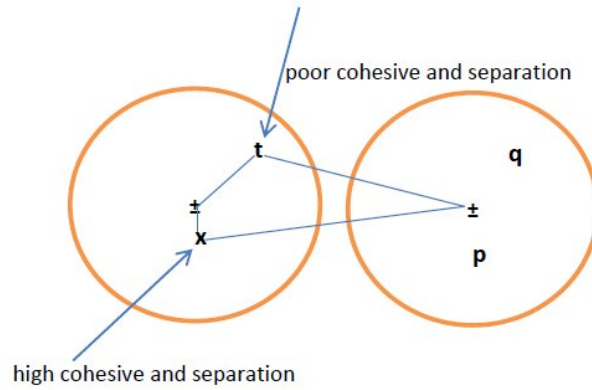


Figure 4.7: Cohesion and Separation of two terms

where $c(t)$ and $s(t)$ are the cohesion and separation scores of the term t , respectively. Silhouette coefficient needs to be positive i.e. $c(t) < s(t)$, and $c(t)$ to be close to 0 i.e. the term t should be very close to its centroid as far as possible, since the coefficient obtains its maximum value of 1, when $c(t)$ is 0. Hence, a high separation score and a low cohesion score of a term t shows the importance of t in a cluster.

4.3.4 Methodology

Step 1. *Preprocessing the corpus:*

Consider a corpus P having classes $C = \{C_1, C_2, \dots, C_n\}$. The documents are identified with an index i.e. doc-id and the corresponding target labels or class labels are stored for future evaluation of the classification metrics. The documents $d = \{d_1, d_2, \dots, d_m\}$ of each class are pre-processed (discussed in chapter 3) and converted into vector form. The documents from all n classes are collected which make the dimension of P as $r \times p$, where r and p are the total number of terms and documents of the corpus P .

Step 2. *Term-Document Cluster formation:*

Next, the traditional k -means clustering algorithm is run on P which generates k term-document (term-doc) clusters, $td = \{td_1, td_2, \dots, td_k\}$. The reason to form clusters is not only to bring the related terms into the same group but also, it helps to compute the total score of each term (discussed in the next step) for which we need clusters. The dimension of each td_i , $i = 1, \dots, k$ is now reduced and shown in the table 4.6.

Table 4.6: Reduced term-document matrix

	d_1	d_2	d_3	...	d_p
t_1	t_{11}	t_{12}	t_{13}	...	t_{1p}
t_2	t_{21}	t_{22}	t_{23}	...	t_{2p}
t_3	t_{31}	t_{32}	t_{33}	...	t_{3p}
·	·	·	·	...	·
·	·	·	·	...	·
·	·	·	·	...	·
t_q	t_{q1}	t_{q2}	t_{q3}	...	t_{qp}

Step 3. *Selection of important features from each cluster:*

The aim is to select important features from each of the k clusters for maintaining uniformity without excluding any collection and is done as follows:

- i. *Computation of centroid for each term-doc cluster:* From each term-doc cluster td_i , the centroid of all the term vectors $\vec{t}_j, j = 1, \dots, q$ is computed by using the following equation:

$$\vec{c}_i = \frac{\sum_{j=1}^q \vec{t}_j}{q} \quad (4.4)$$

where, \vec{c}_i is the centroid vector (dimension of $1 \times p$) of i^{th} term-doc cluster, td_i . Transpose of \vec{c}_i (i.e. \vec{c}_i^T) is shown below.

$$\vec{c}_i^T = \begin{bmatrix} \frac{(t_{11} + t_{21} + t_{31} + \dots + t_{q1})}{q} \\ \frac{(t_{12} + t_{22} + t_{32} + \dots + t_{q2})}{q} \\ \frac{(t_{13} + t_{23} + t_{33} + \dots + t_{q3})}{q} \\ \cdot \\ \cdot \\ \cdot \\ \frac{(t_{1p} + t_{2p} + t_{3p} + \dots + t_{qp})}{q} \end{bmatrix}$$

- ii. *Computation of cohesion score for each term:*

To measure how cohesive is the term, $\vec{t}_j \in td_i$ to the centroid, $\vec{c}_i \in td_i$, the Euclidean distance is computed between \vec{t}_j and \vec{c}_i as follows:

$$cohesion(\vec{t}_j) = (||\vec{c}_i - \vec{t}_j||) \quad (4.5)$$

iii. *Computation of separation score for each term:*

The following equation is used to measure how well separated is a term, $\vec{t}_j \in \mathbf{td}_i$ from the centroid of other clusters, \vec{c}_s , $s = 1, \dots, k$ and $s \neq i$:

$$separation(\vec{t}_j) = \min(||\vec{c}_s - \vec{t}_j||) \quad (4.6)$$

where \vec{c}_s is the centroid of the s^{th} cluster. In other words, it can be said that *separation* (\vec{t}_j) finds the minimum separate distance among all the distances computed between the term \vec{t}_j and centroid of other clusters in which \vec{t}_j is not a member.

iv. *Computation of silhouette coefficient for each term :*

The silhouette coefficient of the term $\vec{t}_j \in \mathbf{td}_i$ is calculated as follows:

$$silhouette(\vec{t}_j) = \frac{(separation(\vec{t}_j) - cohesion(\vec{t}_j))}{\max(cohesion(\vec{t}_j), separation(\vec{t}_j))} \quad (4.7)$$

v. *Computation of total-score for each term of a term-doc cluster:*

Total score of the term $\vec{t}_j \in \mathbf{td}_i$ is calculated as follows:

$$total-score(\vec{t}_j) = \frac{separation(\vec{t}_j)}{cohesion(\vec{t}_j)} + silhouette(\vec{t}_j) \quad (4.8)$$

The reason why we divided the separation score of a term with its cohesion score is that for any important term, the minimum separation value should be high i.e. the term should be well separated from its nearest neighboring cluster and the cohesion value should be low i.e. the term should be tightly bound to the centroid which in turn represents the cluster well, hence the *total-score* will be high. But if it is considered in a reverse order (i.e. $\frac{cohesion}{separation}$) then the top (or important) terms will lose their score (or importance) in the cluster as the ratio will generate very low scores for them.

vi. *Selection of top m% terms from each term-doc cluster and merging them to generate the final feature vector:*

- By repeating steps 4 (ii-v), terms of all k clusters will obtain their respective *total-scores*.

- Rank the terms in each term-doc cluster based on their *total-scores*.
- Select top $m\%$ terms from each term-doc cluster, td_i and merge them into a list, which generates an efficient reduced feature vector.

Step 4. *Performance measurement of different classifiers:*

The reduced feature vector along with the class label of each document generate the training feature vector.

The following equations are used to compute the average performance of the classifiers.

$$Average_{precision} = \frac{\sum_{i=1}^n X_i}{\#documents} \quad (4.9)$$

$$Average_{recall} = \frac{\sum_{i=1}^n Y_i}{\#documents} \quad (4.10)$$

$$Average_{F-measure} = \frac{\sum_{i=1}^n Z_i}{\#documents} \quad (4.11)$$

where, X_i , Y_i and Z_i represent $(p_i * d_i)$, $(r_i * d_i)$ and $(f_i * d_i)$, respectively. d_i , f_i , p_i and r_i are ‘the number of test documents’, ‘F-measure’, ‘precision’ and ‘recall’ of the i^{th} category, respectively. $\#documents$ represents the ‘Total number of test documents’.

4.3.5 Experimental Analysis

The experimental work of the proposed method is tested on four benchmark datasets (20-Newsgroups, Classic4 (Table ??), Reuters (Table ??) and WebKB (Table ??)). For the k -means clustering, k was set as 10 (decided empirically) for all the datasets. The percentage of terms to be selected from each cluster was set as 1%, 5% and 10%. After the feature selection is done, a new term-document matrix was constructed using the reduced features which was then used for text classification. Different classifiers including many ensemble classifiers are used for comparison purpose like LinearSVC, LinearSVM, Nearest Centroid (NC), Gaussian Naive Bayes (GNB), Multinomial Naive-Bayes (M-NB), Adaboost, Decision Tree (DT), Random Forest (RF), ELM and ML-ELM. The number of hidden layers is set as 5 for 20-Newsgroups and 3 for the remaining three datasets (decided empirically). The algorithm was tested using

Algorithm 4.3 *Important terms selection using CCSS*

```

1: Input:  $k$  clusters and their corresponding term-doc matrix
2: Output:  $Top[]$   $\leftarrow$  important terms of a term-doc cluster,  $td_i$ 
3:  $q \leftarrow$  number of terms in  $td_i$ 
4:  $coh[] \leftarrow \phi$  // stores the cohesion score
5:  $sep[] \leftarrow \phi$  // stores the separation score
6:  $sil[] \leftarrow \phi$  // stores the silhouette coefficient score
7:  $total-score[] \leftarrow \phi$  // stores the total score
8:  $S \leftarrow \phi$  // stores the ranked terms of  $td_i$  in descending order of their  $total-score$ 
9:  $\vec{c}_i \leftarrow \phi$  // centroid vector of  $td_i$ 
10:  $Top[] \leftarrow \phi$ 
11: // computing the centroid of cluster  $td_i$ 
12: for all terms  $\vec{t}_j, j \in [1, q]$  do
13:    $\vec{c}_i \leftarrow \vec{c}_i + \vec{t}_j$ 
14: end for
15:  $\vec{c}_i \leftarrow \frac{\vec{c}_i}{q}$ 
16:  $\vec{c}_s \leftarrow$  stores the centroids of all other  $(k-1)$  clusters after computing them in the same
   manner like  $\vec{c}_i$ 
17: //computing the cohesion score of all the terms of  $td_i$ 
18: for all terms  $\vec{t}_j, j \in [1, q]$  do
19:    $coh[\vec{t}_j] \leftarrow ||\vec{c}_i - \vec{t}_j||$  //Euclidean distance
20: end for
21: // computing the separation score of all the terms of  $td_i$ 
22: for all terms  $\vec{t}_j, j \in [1, q]$  do
23:    $A_j[] \leftarrow \phi$ 
24:   for all centroids  $c_s, s \in [1, k]$  and  $s \neq i$  do
25:      $A_j[] \leftarrow (||\vec{c}_s - \vec{t}_j||)$  //Euclidean distance
26:   end for
27:    $sep[\vec{t}_j] \leftarrow \min(A_j[])$ 
28: end for
29: // computing the silhouette coefficient of all the terms of  $td_i$ 
30: for all terms  $\vec{t}_j, j \in [1, q]$  do
31:    $sil[\vec{t}_j] \leftarrow \frac{sep[\vec{t}_j] - coh(\vec{t}_j)}{\max(coh[\vec{t}_j], sep[\vec{t}_j])}$ 
32: end for
33: // computing the total score of all the terms of  $td_i$ 
34: for all terms  $\vec{t}_j, j \in [1, q]$  do
35:    $total-score[\vec{t}_j] \leftarrow \frac{sep[\vec{t}_j]}{coh[\vec{t}_j]} + sil[\vec{t}_j]$ 
36: end for
37: for all terms  $\vec{t}_j, j \in [1, q]$  do
38:    $S \leftarrow$  rank each  $\vec{t}_j$  based on their total-score
39: end for
40:  $Top[] \leftarrow$  select top  $m\%$  terms from  $S$ 
41: return  $Top$ 

```

different number of hidden layer nodes for ELM and ML-ELM and the best results are obtained when the number of hidden layer nodes is more than the nodes in the input layer. In the Tables 4.7 - 4.18, the bold mark indicates the highest average F-measure obtained by *CCSS* using a classifier in comparison with other traditional selection techniques.

20-Newsgroups dataset:

For experimental purpose, three classes of 20-Newsgroups dataset are considered (*alt.atheism*, *misc.forsale* and *soc.religion.christian*) for experimental purpose, which comprises of total 1663 training and 1107 testing documents. The total vocabulary i.e. terms, contained in all of these documents is 20422 and that in training documents is 16270. Tables 4.7 - 4.9 show the average F-measure using different feature selection techniques on different classifiers for the top 1%, 5% and 10% features, respectively. Classifier-wise *CCSS* obtained the highest average F-measure using Multilayer ELM for all three selected feature formats.

Table 4.7: 20-NG: performance on top 1% features

Classifier	Chi-Square	BNS	IG	GINI	CCSS
LinearSVC	0.89121	0.87163	0.87361	0.88949	0.87515
LinearSVM	0.90235	0.89968	0.90152	0.90445	0.89429
NC	0.85584	0.83785	0.84085	0.86064	0.83225
G-NB	0.88226	0.86308	0.86993	0.87515	0.81783
M-NB	0.86302	0.83861	0.83797	0.85795	0.88662
Adaboost	0.87320	0.88318	0.88383	0.88473	0.87334
DT	0.85992	0.85846	0.85906	0.84245	0.85589
RF	0.89673	0.87608	0.88646	0.88230	0.87761
ELM	0.89243	0.87042	0.87543	0.89576	0.88421
ML-ELM	0.90704	0.90236	0.89512	0.90667	0.90803

Classic4 dataset:

For experimental purpose, all the 4 categories of Classic4 dataset are considered in evaluation. The total vocabulary contained in all documents is 21299 and that for training documents is 15971. Tables 4.10 - 4.12 show the average F-measure using different feature selection techniques on different classifiers for the top 1%, 5% and 10% features, respectively. Classifier-wise *CCSS* obtained the highest average F-measure using Multilayer ELM for top 1% and 10% and using LinearSVC for top 5% features.

Table 4.8: 20-NG: performance on top 5% features

Classifier	Chi-Square	BNS	IG	GINI	CCSS
LinearSVC	0.93460	0.92817	0.92815	0.94015	0.93454
LinearSVM	0.94377	0.93461	0.93729	0.94375	0.94509
NC	0.89954	0.88561	0.89489	0.89591	0.87908
G-NB	0.93517	0.88817	0.90253	0.92879	0.86015
M-NB	0.93121	0.90104	0.91608	0.92510	0.89190
Adaboost	0.89076	0.88367	0.86260	0.87113	0.87182
DT	0.85999	0.86257	0.85810	0.85762	0.85619
RF	0.90423	0.88002	0.90226	0.89424	0.88718
ELM	0.93550	0.92675	0.93012	0.93685	0.92334
ML-ELM	0.93874	0.93885	0.93667	0.94587	0.94743

Table 4.9: 20 NG: performance on top 10% features

Classifier	Chi-Square	BNS	IG	GINI	CCSS
LinearSVC	0.94742	0.93732	0.94648	0.95377	0.94921
LinearSVM	0.95286	0.94559	0.94647	0.95654	0.94536
NC	0.90147	0.89582	0.89861	0.90510	0.89582
G-NB	0.93997	0.91011	0.93352	0.93993	0.87138
M-NB	0.93823	0.92347	0.93271	0.93733	0.91938
Adaboost	0.88267	0.86260	0.86342	0.87254	0.86688
DT	0.86373	0.84928	0.86608	0.85916	0.85643
RF	0.89299	0.89245	0.89470	0.89279	0.89571
ELM	0.93226	0.94053	0.93441	0.94674	0.92886
ML-ELM	0.95637	0.94880	0.93567	0.95814	0.95926

Table 4.10: CLASSIC4: performance on top 1% features

Classifier	Chi-Square	BNS	IG	GINI	CCSS
LinearSVC	0.91492	0.88287	0.89941	0.91785	0.90146
LinearSVM	0.93632	0.90663	0.92185	0.93799	0.91670
NC	0.90966	0.83336	0.88875	0.88188	0.83345
G-NB	0.83280	0.79882	0.87912	0.86602	0.80022
M-NB	0.84198	0.76851	0.80706	0.85318	0.86169
Adaboost	0.89163	0.88096	0.88438	0.88987	0.85394
DT	0.86047	0.84002	0.85311	0.86367	0.82878
RF	0.91775	0.89213	0.91535	0.91803	0.88719
ELM	0.90289	0.88146	0.89946	0.92384	0.89185
ML-ELM	0.93650	0.91223	0.91886	0.94567	0.91707

Reuters dataset:

For experimental purpose, all class documents of Reuters dataset are considered for evaluation.

The total vocabulary comprising all of these documents is 17582 and training documents is

Table 4.11: CLASSIC4: performance on top 5% features

Classifier	Chi-Square	BNS	IG	GINI	CCSS
LinearSVC	0.95597	0.94338	0.95106	0.96303	0.96659
LinearSVM	0.96512	0.95914	0.96053	0.96795	0.96020
NC	0.93413	0.92599	0.92953	0.93950	0.93279
G-NB	0.92631	0.90764	0.91491	0.90219	0.90986
M-NB	0.93815	0.92328	0.93096	0.94526	0.94668
Adaboost	0.87346	0.88183	0.84974	0.85481	0.84586
DT	0.84928	0.84844	0.84886	0.85555	0.84843
RF	0.91894	0.91892	0.91657	0.91920	0.89555
ELM	0.94534	0.92521	0.95678	0.95178	0.94576
ML-ELM	0.95277	0.94889	0.96254	0.96549	0.96540

Table 4.12: CLASSIC4: performance on top 10% features

Classifier	Chi-Square	BNS	IG	GINI	CCSS
LinearSVC	0.96546	0.96124	0.96337	0.96868	0.96550
LinearSVM	0.95541	0.96653	0.96759	0.97286	0.96769
NC	0.94022	0.93595	0.93950	0.94239	0.93992
G-NB	0.92032	0.90687	0.91549	0.91055	0.89721
M-NB	0.95096	0.94560	0.94777	0.95353	0.95917
Adaboost	0.85481	0.85481	0.85481	0.84586	0.84586
DT	0.85070	0.84739	0.85209	0.84575	0.85084
RF	0.91527	0.92683	0.91858	0.91438	0.91673
ELM	0.92335	0.94675	0.95637	0.96943	0.95632
ML-ELM	0.97108	0.95875	0.96889	0.97940	0.97078

13531. Tables 4.13 - 4.15 show the average F-measure using different feature selection techniques on different classifiers for the top 1%, 5% and 10% features, respectively. Classifier-wise CCSS obtained the highest average F-measure using Multilayer ELM for top 5% and 10% and using LinearSVM for top 1% features.

WebKB dataset:

For experimental purpose, all class documents of WebKB are considered in evaluation. The total vocabulary comprising all these documents is 7606 and training documents is 7522. Tables 4.16 - 4.18 show the average F-measure using different feature selection techniques on different classifiers for the top 1%, 5% and 10% features, respectively. Classifier-wise CCSS obtained the highest average F-measure using Multilayer ELM for top 5% and 10% features and using LinearSVM for top 1% features.

Table 4.13: REUTERS: performance on top 1% features

Classifier	Chi-Square	BNS	IG	GINI	CCSS
LinearSVC	0.93365	0.92337	0.92968	0.93972	0.90524
LinearSVM	0.95241	0.94919	0.95147	0.95951	0.92926
NC	0.83338	0.83042	0.83156	0.83834	0.81451
G-NB	0.86442	0.85535	0.85345	0.85344	0.85148
M-NB	0.87203	0.84187	0.85836	0.86013	0.84526
Adaboost	0.64005	0.64005	0.64005	0.76295	0.77928
DT	0.89169	0.88853	0.89004	0.89518	0.86448
RF	0.92233	0.92945	0.92360	0.93204	0.90925
ELM	0.94567	0.95023	0.93143	0.93376	0.90327
ML-ELM	0.95414	0.95673	0.95678	0.96759	0.92601

Table 4.14: REUTERS: performance on top 5% features

Classifier	Chi-Square	BNS	IG	GINI	CCSS
LinearSVC	0.95123	0.94782	0.95008	0.95425	0.94076
LinearSVM	0.96229	0.96386	0.96648	0.96555	0.95492
NC	0.84276	0.83726	0.84064	0.84359	0.83986
G-NB	0.82964	0.87128	0.85761	0.85329	0.84877
M-NB	0.90245	0.89553	0.90329	0.91179	0.88668
Adaboost	0.63823	0.65844	0.62867	0.67428	0.77314
DT	0.90289	0.90602	0.90376	0.90350	0.89668
RF	0.92824	0.92324	0.93328	0.91606	0.91753
ELM	0.94567	0.94897	0.95607	0.93452	0.93142
ML-ELM	0.96393	0.96314	0.95837	0.96876	0.96938

Table 4.15: REUTERS: performance on top 10% features

Classifier	Chi-Square	BNS	IG	GINI	CCSS
LinearSVC	0.95733	0.95171	0.95453	0.95695	0.95477
LinearSVM	0.96486	0.96618	0.96685	0.96819	0.96781
NC	0.84553	0.84320	0.84261	0.84546	0.84519
G-NB	0.79522	0.84728	0.83480	0.81193	0.79145
M-NB	0.90079	0.90557	0.90810	0.90971	0.88693
Adaboost	0.63800	0.63800	0.63428	0.63428	0.63800
DT	0.90557	0.89854	0.90680	0.89942	0.90653
RF	0.91901	0.91690	0.91905	0.91982	0.91076
ELM	0.92336	0.95667	0.95542	0.95502	0.94556
ML-ELM	0.96721	0.95764	0.96778	0.96324	0.96958

Discussion:

Figures 4.8 - 4.10 show the performance comparison among different classifiers for top 1%, 5% and 10% features, respectively on different datasets using CCSS feature selection tech-

Table 4.16: WebKB: performance on top 1% features

Classifier	Chi-Square	BNS	IG	GINI	CCSS
LinearSVC	0.84017	0.81562	0.83208	0.77056	0.78836
LinearSVM	0.86636	0.85418	0.87036	0.81684	0.87785
NC	0.76794	0.75209	0.76273	0.74439	0.73456
G-NB	0.65442	0.63748	0.67973	0.53566	0.55930
M-NB	0.68787	0.65050	0.66735	0.61533	0.60170
Adaboost	0.83838	0.81693	0.81550	0.79223	0.79302
DT	0.79120	0.77399	0.77503	0.74938	0.73599
RF	0.84676	0.82846	0.83196	0.81568	0.81502
ELM	0.81267	0.80382	0.84784	0.78940	0.80536
ML-ELM	0.87504	0.82365	0.85637	0.83657	0.82516

Table 4.17: WebKB: performance on top 5% features

Classifier	Chi-Square	BNS	IG	GINI	CCSS
LinearSVC	0.87806	0.86861	0.87754	0.88150	0.86492
LinearSVM	0.87914	0.89136	0.89868	0.88725	0.85949
NC	0.79428	0.78230	0.78507	0.78554	0.77728
G-NB	0.70748	0.74339	0.73449	0.70752	0.71888
M-NB	0.77868	0.75186	0.76395	0.75465	0.74448
Adaboost	0.79831	0.81163	0.81849	0.80973	0.80282
DT	0.78483	0.78760	0.79159	0.78826	0.77702
RF	0.83868	0.82967	0.84827	0.84487	0.83076
ELM	0.86721	0.83251	0.84512	0.87523	0.83675
ML-ELM	0.89675	0.88568	0.89034	0.89675	0.89910

Table 4.18: WebKB: performance on top 10% features

Classifier	Chi-Square	BNS	IG	GINI	CCSS
LinearSVC	0.88882	0.88007	0.88358	0.88188	0.87571
LinearSVM	0.87054	0.89838	0.89880	0.90048	0.85602
NC	0.80388	0.79180	0.79743	0.79185	0.78611
G-NB	0.70038	0.73236	0.71898	0.70539	0.70298
M-NB	0.78504	0.77862	0.78699	0.77905	0.77623
Adaboost	0.81144	0.79831	0.79831	0.80097	0.80883
DT	0.78083	0.79578	0.79057	0.78474	0.77786
RF	0.83460	0.83170	0.83144	0.82889	0.81987
ELM	0.86621	0.88765	0.89012	0.87564	0.84243
ML-ELM	0.90236	0.90432	0.86432	0.88987	0.91278

nique. Table 4.19 shows the average F-measure comparisons of different established classifiers using *CCSS* technique. The bold result indicates the maximum F-measure obtained by a classifier using *CCSS* and it signifies that Multilayer ELM leaves behind other standard classifiers

Table 4.19: F-measure comparisons using CCSS

Classifier	20- Newsgroups (F-Measure-%)			Classic4 (F-Measure-%)			Reuters (F-Measure-%)			WebKB (F-Measure-%)		
	1%	5%	10%	1%	5%	10%	1%	5%	10%	1%	5%	10%
LinearSVC	87.515	93.454	94.921	90.146	96.659	96.550	90.524	94.076	95.477	78.836	86.492	87.571
Linear	89.429	94.509	94.536	91.670	96.020	96.769	92.926	95.492	96.781	87.785	85.949	85.602
SVM												
NC	83.225	87.908	89.582	83.345	93.279	93.992	81.451	83.986	84.519	73.456	77.728	78.611
G-NB	81.783	86.015	87.138	80.022	90.986	89.721	85.148	84.877	79.145	55.930	71.888	70.298
M-NB	88.662	89.190	91.938	86.169	94.668	95.917	84.526	88.668	88.693	60.170	74.448	77.623
Adaboost	87.334	87.182	86.688	85.394	84.586	84.586	77.928	77.314	63.800	79.302	80.282	80.883
DT	85.589	85.619	85.643	82.878	84.843	85.084	86.448	89.668	90.653	73.599	77.702	77.786
RF	87.761	88.718	89.571	88.719	89.555	91.673	90.925	91.753	91.076	81.502	83.076	81.987
ELM	88.421	92.334	92.886	89.185	94.576	95.632	90.327	93.142	94.556	80.536	83.675	84.243
ML-ELM	90.803	94.743	95.926	91.707	96.540	97.078	92.601	96.938	96.958	82.516	89.910	91.278

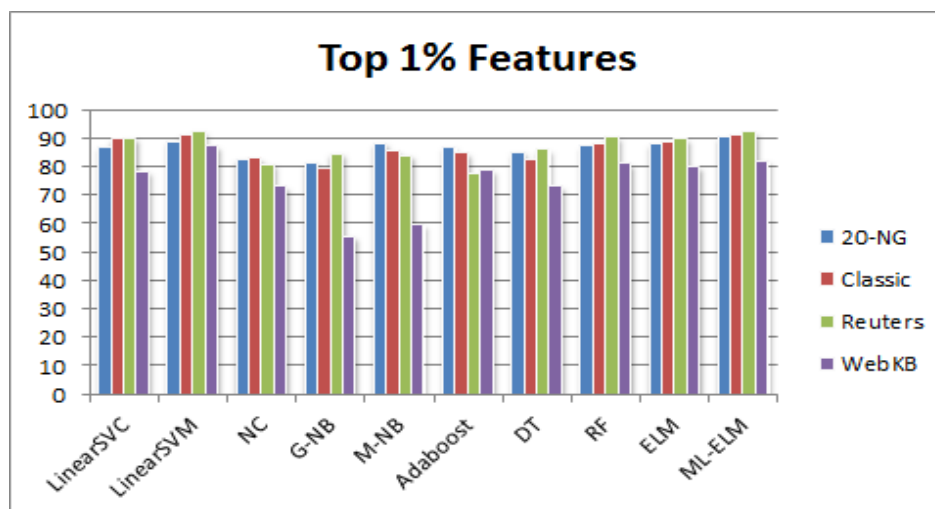


Figure 4.8: Average F-measure on top 1% features

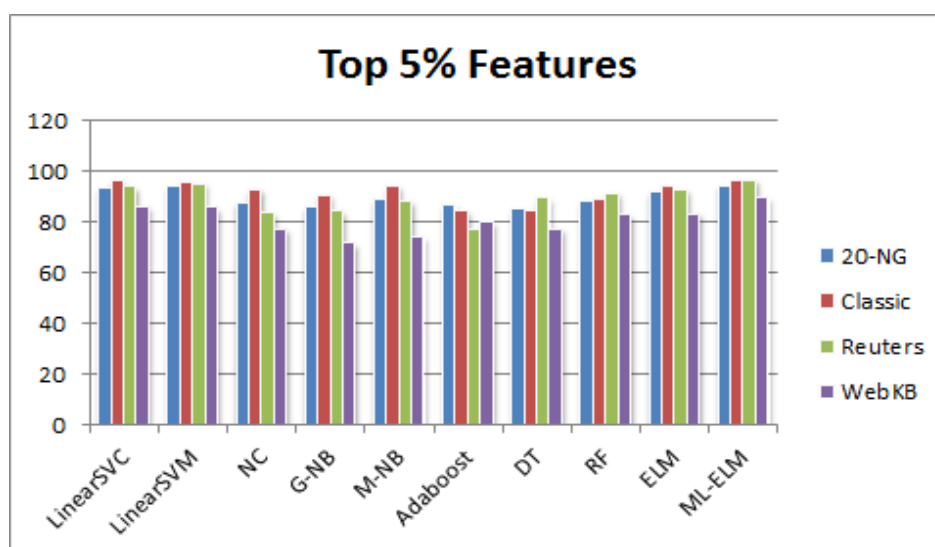


Figure 4.9: Average F-measure on top 5% features

except for some cases like top 5% features of Classic4 (maximum F-measure obtained by LinearSVC), top 1% features of Reuters (maximum F-measure obtained by Linear SVM) and top 1% features of WebKB (maximum F-measure obtained by Linear SVM) datasets where the results of Multilayer ELM are very near to the maximum result obtained by any other classifier.

In both the novel feature selection techniques (*KWFS* and *CCSS*), from the obtained results, it has been noticed that ML-ELM outperformed the state-of-the-art classifiers. This justifies the

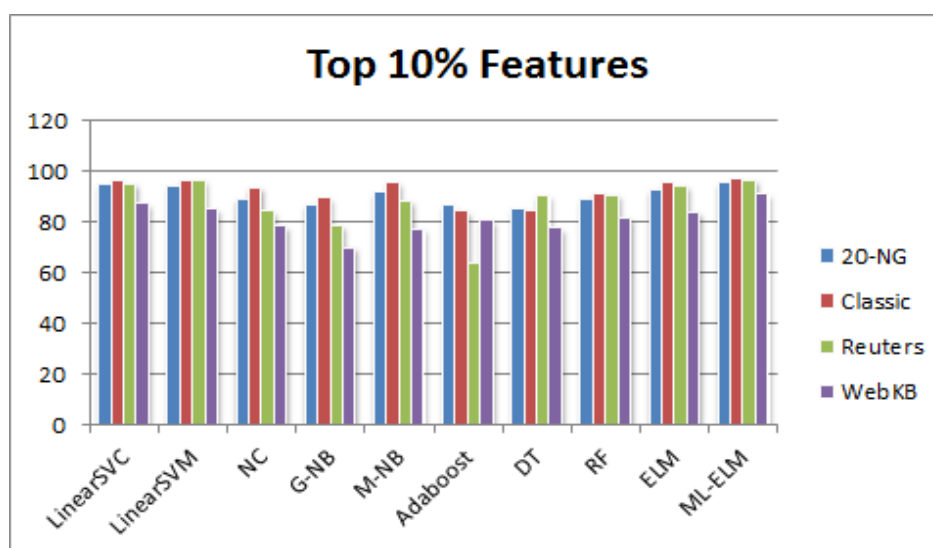


Figure 4.10: Average F-measure on top 10% features

prominence of deep learning for classifying the text documents. The following points discuss some of the probable reasons of “*why Multilayer ELM is outperforming other classifiers ?*”:

- i. Multilayer ELM uses the ELM feature mapping mechanism (discussed in Chapter 5) to map the training feature vector into an extended feature space i.e. $n < L$ and hence makes the features much simpler and linearly separable in the high dimensional space which enhances its performance [132]. Classifier like Extreme Learning Machine (ELM) can approximate any complex non-linearly mappings directly from the training samples, but to fit the highly-variant input data perfectly, it needs a large network that is difficult to implement. This is because of its shallow architecture similar to other ancient SLFNs, which can easily be overcome by Multilayer ELM.
- ii. impressed with an underlying design of the deep network.
- iii. multiple non-linear transformations of input data are possible using multiple layers.
- iv. higher level abstraction of data is captured by Multilayer ELM using multiple layers, whereas the networks having a single layer fail to achieve it.
- v. by using representation learning, a different form of the input is learned at each layer within the network.

4.4 Summary

This chapter proposed two novel feature selection techniques (*KWFS* and *CCSS*) for text classification. In *KWFS*, initially, each cluster is divided into k sub clusters by using k -means algorithm. Then with the help of Wordnet and cosine-similarity a reduced feature vector is generated for the entire corpus. For text classification, ELM and ML-ELM classifiers have been used. This technique is tested on 20-Newsgroups and DMOZ dataset and the results witness the suitability and importance of ELM and ML-ELM in the field of text classification.

It can be observed from the tables and figures of both datasets that ML-ELM performs the best out of all the traditional classifiers. ML-ELM has an impressive average F-measure of 0.7472 on the 20-Newsgroups dataset and 0.7195 on the DMOZ dataset, which signifies the strength of deep learning in the domain of text classification. In the second feature selection technique (*CCSS*), three important parameters (cohesion, separation and silhouette coefficient) are combined to generate a reduced feature vector. Multilayer ELM as the classifier has been used for classifying the text document and its importance also has been extensively measured and discussed in this chapter. This technique can be summarized as follows:

1. *term clustering and total-score calculation*: k -means clustering algorithm is run on a corpus to group the similar terms into k clusters. Cohesion, separation and silhouette coefficient scores of every term in each cluster are computed and finally, with the help of these three scores, the *total-score* is computed.
2. *final reduced feature vector preparation*: At the end, all the terms in each cluster are ranked together based on their total-score and top $m\%$ terms are selected as the important terms from a cluster. These important terms are merged into a list to finalize the reduced feature set.
3. *training Multilayer ELM and other standard classifiers*: The reduced feature vector along with the labels of classes is used to train all conventional classifiers.
4. *performance measurement of the classifiers*: F-measure of Multilayer ELM and other classifiers are computed by comparing the prediction of the classifier about a test document d with the actual class label of d .

It is evident from all the results that the performance of Multilayer ELM outperforms the other well known classifiers. The encouraging results of two techniques show the stability and effectiveness of Multilayer ELM compared to different progressive classifiers in the domain of text classification. This justifies the stability, efficiency and effectiveness of deep learning. Although Multilayer ELM performs well, but still there are some shortcomings which need more attention in order to improve them and can be added to the future work such as:

- Determining activation function and the number of nodes for each hidden layer.
- Similarly in ELM, the behavior and a correct number of hidden units is still debatable and can be added to the future study.
- Also, if we want to understand deeply the functionality of ELM and Multilayer ELM by considering them as an approximation of infinite network, then the variance of hidden layer weights is still open for future study.

Combining the ELM and Multilayer ELM feature space with other traditional classifiers will further reinforce the classification results. Similarly, the feature space of ELM and Multilayer ELM can also be used for clustering process, as the features become more simple and linearly separable by representing them in an extended space. This may outperform the clustering process done in TF-IDF vector space and is the primary focus of the next chapter.

CHAPTER 5

CLUSTERING IN ML-ELM FEATURE SPACE

5.1 Introduction

Clustering is a technique which puts together the similar data items into individual groups. Traditional clustering techniques are generally *unsupervised* which means that there are no class labels available to guide the clustering process. Many researchers have found that if one combines the unlabeled data with limited amount of label data then the performance of the clustering process will be improved [133][134][135]. The clustering technique where some knowledge of supervised data in the form of class labels are used along with the unlabeled data is called *semi-supervised clustering*. Assuming that we have some labeled data which can be used to obtain an initial model, we only have to know what data belongs to which clusters and therefore the actual clusters are not needed. We can begin the clustering process from this initial model which means that we decided the starting point of the search using the supervised information. Having this little known information, the search process continues till it finds the labels of all unlabeled data. Figure 5.1 demonstrates such example by marking the labeled data as ‘×’ and unlabeled data as ‘•’. With this small information, the semi-supervised clustering begins by using the labeled data to form the centroids and place the unlabeled data into their respective clusters (Figure 5.2). It has been acknowledged that the data structure becomes much simpler, or the data will become more linearly separable if it can be transformed into an extended dimensional feature space using a nonlinear data transformation. Generally, clustering approaches use kernel techniques [136] and obtain better results. However, ELM

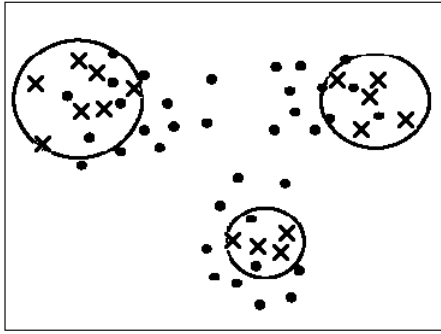


Figure 5.1: Semi supervised clustering (before)

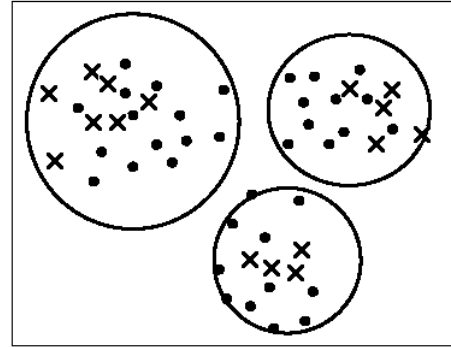


Figure 5.2: Semi supervised clustering (after)

feature mapping is an explicit and simpler feature mapping technique that outperforms kernel-based methods and gives better performance [137][138]. Hence, solving the clustering problem using ML-ELM feature mapping carries significant weight.

This chapter discusses an approach where semi-supervised (seeded- k Means) and unsupervised clustering (traditional k -means) are done in ML-ELM feature space.

5.1.1 Importance of extended feature space of ML-ELM

According to the ELM classification capability (equation 5.1) and universal approximation conditions, a very large number of nodes in the hidden layer ensures that the data become more *linearly separable*. As it is well known that Multilayer ELM uses all the properties of ELM, and hence takes the advantages of this *ELM feature mapping*, which makes the input feature vector much simpler and linearly separable in the extended space. The feature space of ELM is prepared by specifying two parameters: number of hidden layer nodes, and the activation function. L can be set in accordance with the size of the input feature vector ' n ' depending on the following conditions:

- i. $n < L$: Sparse feature vector
- ii. $n = L$: Equal dimension feature vector
- iii. $n > L$: Compressed feature vector

The activation function may be any non-linear continuous function such as *Gaussian*, *Sigmoid* or *Cosine*. The process of mapping the input training data into the feature space of ML-ELM is discussed below.

An input layer feature vector of n -dimension $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$, maps into an ML-ELM feature space of $L (> n)$ dimension.

$$\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), \dots, h_i(\mathbf{x}), \dots, h_L(\mathbf{x})]^T = [g(w_1, b_1, \mathbf{x}), \dots, g(w_i, b_i, \mathbf{x}), \dots, g(w_L, b_L, \mathbf{x})]^T \quad (5.1)$$

Using equation 5.1, it is very easy to transform the features of training dataset from a low dimensional input data space into an extended feature space. ELM can approximate any con-

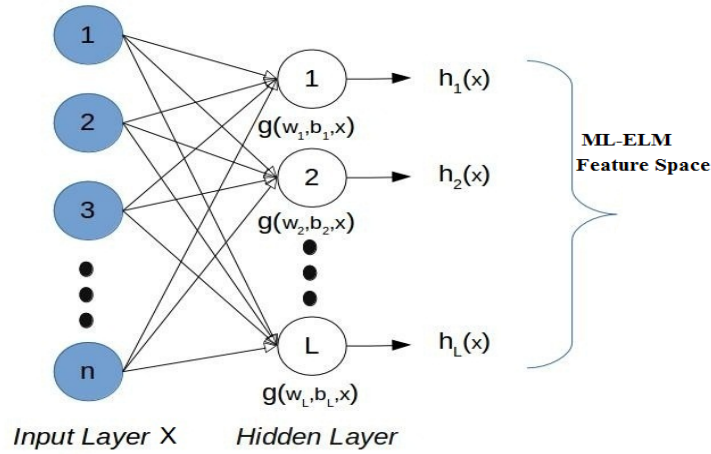


Figure 5.3: ML-ELM feature space

tinuous target function using the following equation [139].

$$\lim_{L \rightarrow +\infty} \|y_L(\mathbf{x}) - y(\mathbf{x})\| = \lim_{L \rightarrow +\infty} \left\| \sum_{i=1}^L \beta_i h_i(\mathbf{x}) - y(\mathbf{x}) \right\| = 0 \quad (5.2)$$

If $\mathbf{h}(\mathbf{x})\beta$ can approximate any continuous function, then any decision regions regardless of their shape can be separated by ELM [66]. This important feature mapping of ELM helps us to solve the clustering problem in feature space of ML-ELM.

5.1.2 Seeded- k Means/ k Means Algorithm

Seeded- k Means proposed by Basu et al. [140] requires a labeled subset S of the dataset X (which contains documents belonging to k different categories), while the identity of the unlabeled parts of X remains unknown. It is assumed that at least one document from each category will be there in the labeled subset, ($S = S_1, S_2, \dots, S_k$). The documents of i^{th} category are represented by S_i while the mean of S_i provides the centroid μ_i . In this manner, all centroids μ_1 to μ_k are initialized which serve as the initial centroids for their respective clusters. It is different from the traditional k Means algorithm, where the initial centroids (w_i) are calculated based on randomly selected k documents (initially consider as k clusters) from the dataset X . Once the initial centroids are decided for each clustering technique (i.e. Seeded- k Means/ k Means), then the other steps to compute the final clusters are similar for both techniques. At every step of the iteration, each document is (re)assigned to a cluster, based on the Euclidean distance of the document from the centroid of the cluster. Next, the new centroids of each cluster are (re)computed and the process is repeated till convergence or the maximum number of iterations is reached. Algorithm 5.1 and 5.2 illustrate the formal analysis of k Means and seeded- k Means.

Algorithm 5.1 k Means

- 1: **Input:** number of clusters k , dataset $X = \{x_1, x_2, \dots, x_N\}$
 - 2: **Output:** k clusters
 - 3: Initially, select k cluster centers randomly.
 - 4: Compute the initial centroid w_1, \dots, w_k for each k cluster as $w_i \leftarrow (\frac{1}{s}) \sum_{j=1}^s D_j$ where s is the number of documents in i^{th} cluster and D_j is the j^{th} document of i^{th} cluster
 - 5: $y \leftarrow 0$
 - 6: repeat:
 - 7: Assign each document x_1, \dots, x_N to one of the cluster x_1, \dots, x_k based on $\min_{i=1, \dots, k} \|x - w_i\|$
 - 8: Recalculate centroids w_1, \dots, w_k as $w_i \leftarrow (\frac{1}{s}) \sum_{j=1}^s D_j$
 - 9: $y \leftarrow y + 1$
 - 10: until convergence achieved or number of iterations exceed a threshold
 - 11: return set of k clusters
-

Algorithm 5.2 Seeded- k Means

- 1: **Input:** number of clusters k , dataset $X = \{x_1, x_2, \dots, x_N\}$, set $S \subset X$ // S contains the labeled data
 - 2: **Output:** k clusters
 - 3: Initialize centroids μ_1, \dots, μ_k from labeled set S as $\mu_i \leftarrow \frac{(\sum_{y \in S_i} y)}{\|S_i\|}$
 - 4: $t \leftarrow 0$
 - 5: repeat:
 - 6: Assign each document x_1, \dots, x_N to one of the cluster x_1, \dots, x_k based on $\min_{i=1, \dots, k} \|x - \mu_i\|$
 - 7: Recalculate centroids μ_1, \dots, μ_k as $\mu_i \leftarrow \frac{(\sum_{y \in X_i} y)}{|S_i|}$
 - 8: $t \leftarrow t + 1$
 - 9: until convergence achieved or number of iterations exceed a threshold
 - 10: return set of k clusters
-

5.2 Methodology

The following steps discuss the k Means and seeded- k Means using the feature space of ML-ELM and TF-IDF vector space.

Step 1. *Pre-processing the documents of the corpus:*

Consider a corpus P having classes $C = \{C_1, C_2, \dots, C_n\}$. The documents are identified with an index (i.e. doc-id) and the corresponding target labels or class labels are stored for future evaluation of the classification metrics. All the documents are pre-processed (as discussed in Chapter 3). The documents from all n classes are collected and converted to term vectors which make P as dimension of $p \times r$, where p and r are total number of documents and terms respectively.

Step 2. *Mapping P into ML-ELM feature space:*

Using equation 5.1, the input feature vector will map into the ML-ELM feature space. According to equation 5.2, by setting the hidden layer nodes more than the input layer nodes ($L > n$), the input feature vector can be represented in an extended dimensional space which will ensure that the features are linearly separable.

Step 3. *Cluster formation:*

After mapping the feature vector into the ML-ELM feature space, seeded- k Means and traditional k Means are run separately in the ML-ELM feature space and on the corpus

P directly (using TF-IDF vector space) to generate required k clusters (Algorithm 5.3).

The steps to implement the complete approach are illustrated in Figure 5.4.

Algorithm 5.3 Seeded- k Means/ k Means clustering in ML-ELM feature space

- 1: **Input:** number of clusters k , number of hidden layer nodes L , Corpus P
 - 2: **Output:** k clusters
 - 3: Map the documents of corpus P into the feature space of ML-ELM, H using $h(\mathbf{x}) = [h_1(\mathbf{x}), \dots, h_i(\mathbf{x}), \dots, h_L(\mathbf{x})]^T$.
 - 4: Call Algorithm 5.1 by passing H , number of clusters // for traditional k Means
 - 5: Retain $n\%$ of labels of the corpus and mark the labels of other documents as ‘0’. Build a subset S using H and retain the labels. Call Algorithm 5.2 by passing H , number of clusters k and labels S . // for seeded- k Means
-

Step 4. *Performance comparison:*

Performance of each cluster is computed using different supervised clustering evaluation techniques viz. purity and entropy (Appendix F).

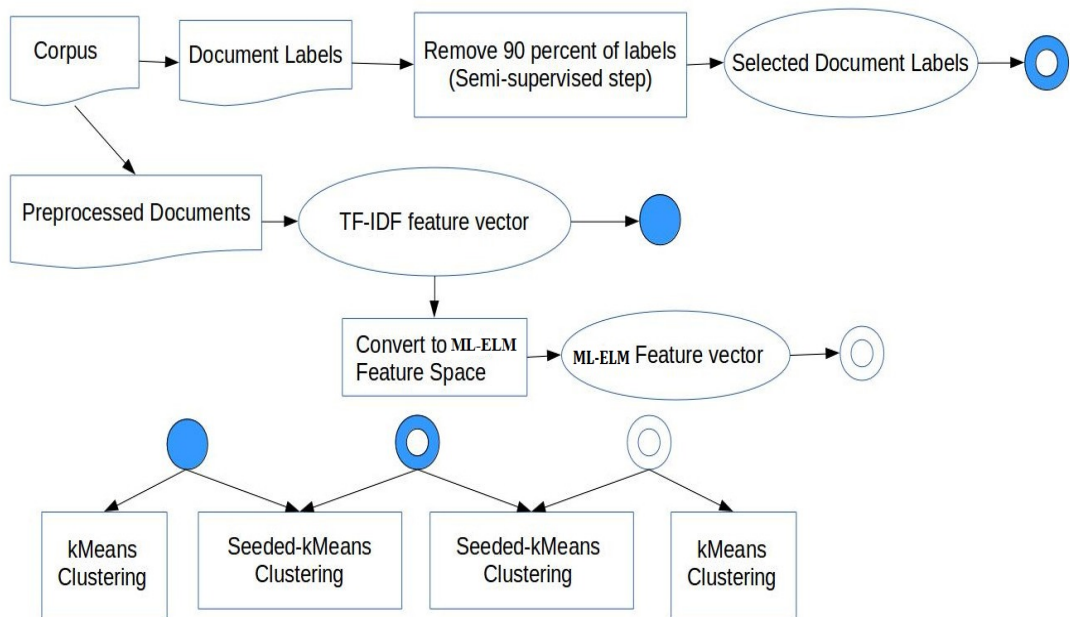


Figure 5.4: TF-IDF and ML-ELM feature vector

5.3 Experimental Analysis

Experimental work of the proposed approach is carried out on two benchmark datasets (Classic4 and Reuters). Dataset wise, results of the proposed approach are discussed and compared with the traditional k Means applied on TF-IDF vector space and feature space of ML-ELM.

5.3.1 Performance evaluation of the clustering

For evaluating the performance of each cluster, different supervised techniques are used where the class details are known in prior. Before evaluating the performance of each cluster, first we identified each cluster belongs to which class of the corpus P . To do this, we considered the number of clusters and classes to be same. A cluster i belongs to a class j , if i contains most of the documents of j compared to other classes in the corpus. Two supervised clustering evaluation techniques such as purity and entropy are used to measure the performance of the clustering process in TF-IDF and ML-ELM feature space.

Classic4 dataset:

All four categories of Classic4 dataset are considered for experimental work. 250 documents are taken from each category to form the corpus. The total vocabulary (terms) of all of these documents is 7916. Tables 5.1 and 5.2 show the purity and entropy of k Means and seeded- k Means using TF-IDF vector space and feature space of ML-ELM respectively. Figure 5.5 shows the execution time of k Means and seeded- k Means using TF-IDF vector space and feature space of ML-ELM.

Reuters dataset:

1459 documents are taken to form the corpus and the total vocabulary of all these documents is 7539. Table 5.3 and 5.4 show the purity and entropy of k Means and seeded- k Means using TF-IDF vector space and feature space of ML-ELM. Figure 5.6 shows the execution time of k Means and seeded- k Means using TF-IDF vector space and feature space of ML-ELM.

Table 5.1: Purity of clusters on Classic4 dataset

Purity	TF-IDF vector space	(L = 0.6n)	(L = 0.8n)	(L = n)	(L = 1.2n)	(L = 1.4n)
<i>k</i> Means (3 iterations)	0.693	0.711	0.726	0.757	0.791	0.824
Seeded- <i>k</i> Means (3 iterations)	0.857	0.948	0.948	0.952	0.962	0.973
<i>k</i> Means (5 iterations)	0.695	0.732	0.747	0.768	0.821	0.837
Seeded- <i>k</i> Means (5 iterations)	0.857	0.953	0.957	0.958	0.967	0.974

Table 5.2: Entropy of clusters on Classic4 dataset

Entropy	TF-IDF vector space	(L = 0.6n)	(L = 0.8n)	(L = n)	(L = 1.2n)	(L = 1.4n)
<i>k</i> Means (3 iterations)	0.734	0.723	0.719	0.688	0.62	0.587
Seeded- <i>k</i> Means (3 iterations)	0.365	0.360	0.358	0.353	0.331	0.313
<i>k</i> Means (5 iterations)	0.739	0.702	0.68	0.637	0.578	0.522
Seeded- <i>k</i> Means (5 iterations)	0.361	0.353	0.323	0.321	0.311	0.302

Table 5.3: Purity of clusters on Reuters dataset

Purity	TF-IDF vector space	(L = 0.6n)	(L = 0.8n)	(L = n)	(L = 1.2n)	(L = 1.4n)
<i>k</i> Means (3 iterations)	0.451	0.537	0.541	0.569	0.618	0.660
Seeded- <i>k</i> Means (3 iterations)	0.765	0.806	0.811	0.82	0.837	0.856
<i>k</i> Means (5 iterations)	0.475	0.586	0.602	0.632	0.642	0.669
Seeded- <i>k</i> Means (5 iterations)	0.799	0.867	0.868	0.872	0.885	0.896

Table 5.4: Entropy of clusters on Reuters dataset

Entropy	TF-IDF vector space	(L = 0.6n)	(L = 0.8n)	(L = n)	(L = 1.2n)	(L = 1.4n)
<i>k</i> Means (3 iterations)	1.355	1.354	1.32	1.271	1.071	0.981
Seeded- <i>k</i> Means (3 iterations)	0.963	0.911	0.908	0.906	0.880	0.868
<i>k</i> Means (5 iterations)	1.354	1.275	1.252	1.271	1.012	0.976
Seeded- <i>k</i> Means (5 iterations)	0.944	0.861	0.840	0.832	0.812	0.804

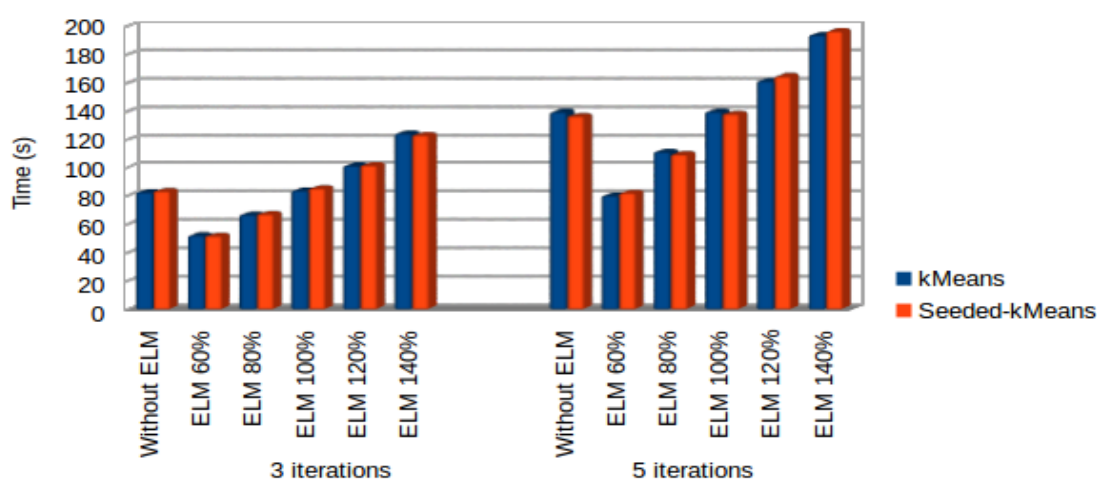


Figure 5.5: Execution time on Classic4 dataset

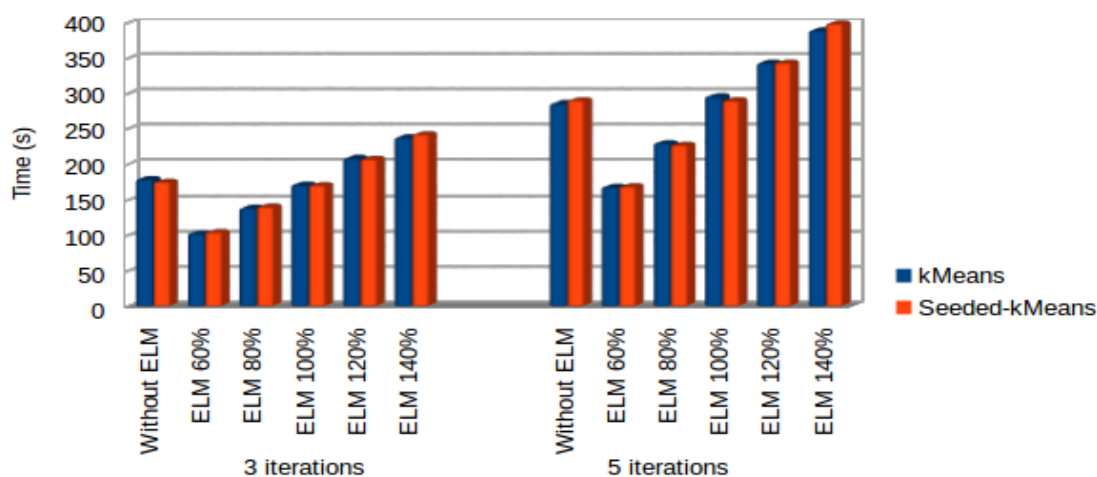


Figure 5.6: Execution time on Reuters dataset

5.4 Summary

This chapter proposes an approach where the feature space of ML-ELM is used for semi-supervised clustering using seeded- k Means algorithm. The number of hidden layers is set to 3 (decided empirical) for ML-ELM on both datasets. The complete work is carried out in two phases as discussed below:

1. While testing the clustering process in the feature space of ML-ELM, the hidden layer nodes L are varied with fixed number of input nodes n such as $L = 1.4n$ (140 %), $L = 1.2n$ (120 %), $L = n$ (100 %), $L = 0.8n$ (80 %) and $L = 0.6n$ (60 %). The first two conditions represent a sparse vector representation where the data is expanded into a linearly separable space which facilitates a more robust cluster formation. $L = n$ shows the importance of non-linearity for clustering where it performs better than the TF-IDF vector space of the same dimensions.
2. Both the clustering techniques are run for many iterations and for the experimental purpose, here we have shown the results after third and fifth iterations. The following points can be observed from the purity (Tables 5.1 and 5.3) and entropy tables (Tables 5.2 and 5.4) on both the datasets:
 - results using *ELM feature space* outperforms the results of *without using ELM feature space* (i.e. TF-IDF vector space).
 - Seeded- k Means performs well compared to k Means in all aspects regardless of the parameters set (i.e. number of clusters to be formed, % of labels to be considered in seeded- k Means). This demonstrates the superiority of an seeded- k Means clustering technique over the traditional one even very few labels (only 10 % of the labels are considered) are provided.
 - results of purity and entropy are close in ML-ELM compressed or equal dimension space ($L \leq n$) whereas it is better in ML-ELM extended feature space ($L > n$) on both datasets.
 - It is also observed that after a certain stage (i.e. threshold point) in extended space, the performance of the clustering process remains unchanged (i.e. further increasing L compared to n have no effect). This may be due to the excessive sparse representation of the features which is not included in the results.

CHAPTER 6

MODIFIED APRIORI APPROACH FOR TEXT CLUSTERING

6.1 Introduction

Association rules of data mining state that the knowledge of frequent itemset can be used to find out how an itemset is influenced by the presence of another itemset in the corpus [9]. An itemset is frequent, if it is present in at least $x\%$ of the total transactions z in the database D , where x is the support threshold. When the number of items included in the database transaction is high and we are finding itemset with small minimum support, the number of frequent itemsets found are quite large, and it makes the problem very expensive to solve, both in time and space. Hence, the minimum support count affects the computational cost of the higher (say k^{th}) iteration of apriori algorithm. Thus, one can say that the cardinality of C_k and the size of D affect the overall computational cost. The traditional apriori algorithm generates and tests the candidate itemset in a level-wise manner using iterative database scan which makes the computational cost high [141]. This chapter discusses the following:

- 1) develop a novel clustering technique called *Modified Apriori Approach*, which improves the traditional apriori approach by removing some of its limitations that save time and space.
- 2) measure the performance of different traditional clustering algorithms after combining

them with association based clustering. For this purpose, Fuzzy C-means (FCM) [142], Vector space model (VSM) [3] and k -means [27] techniques are run on the initial clusters generated by the proposed modified apriori approach.

6.1.1 Modified Apriori Approach

This approach considers *documents as itemset* and *keywords as transaction* so that we end up with clusters having a minimum frequency support threshold. By making the keywords as transaction and document as itemset, we combined the support threshold idea of association rule mining algorithm with the output like that of a clustering algorithm. The salient features of the proposed approach in comparison with traditional apriori algorithm can be listed as follows:

- i. reduces the time for accessing the transactions.
- ii. avoid repeated database scan.
- iii. nullifies the transactions which are no longer in use.
- iv. decreases the number of candidate itemset during the candidate generation step and hence, save the space.

6.1.2 Optimization open Traditional Apriori Algorithm for clustering

1. Apriori algorithm generates frequent candidate set by generating all possible candidate sets and then checking which sets cross the minimum support count. Whereas the proposed approach uses the following rule:
If an itemset occurs $(k-1)$ times in the set of $(k-1)$ frequent itemset, only then it is considered for k^{th} frequent candidate set (since only then it has a chance to come in a ' k ' sized frequent itemset). So, the proposed algorithm rejects number of unwanted candidate sets than the traditional apriori (which are generally not going to be frequent in the next iteration).
2. Apriori counts the occurrence of an itemset even if it does not appear in any of the frequent candidates, but the proposed approach removes that itemset from the array (ini-

tially a 2D-array is considered to store all itemsets and their corresponding transactions) by setting 0 across it so that less number of checks are made for the itemset.

3. Apriori does not take into account the unnecessary computations made if the size of the transaction is less than the size of the candidate set being generated. So, to improve upon this, the proposed approach ignores the corresponding transactions by putting a null (an indicator used) in the 2D-array [143].
4. All of the above statements basically remove the information which is no longer required from the 2D-array created initially and this reduces the unnecessary comparisons in the subsequent steps.

6.2 Methodology

6.2.1 Document pre-processing

Given a corpus P having classes $C = \{C_1, C_2, \dots, C_m\}$ of documents. All the documents are pre-processed (discussed in Chapter 3) and then each document is represented as term vector in the vector space over the system's vocabulary.

6.2.2 Obtaining initial clusters and their centroids

After converting each document to their vector form, traditional and modified apriori approach are run on the corpus P by taking *minimum support* as the input. The frequent sets generated are of frequency greater than the minimum support. Finally, the modified and traditional apriori approaches stop where further frequent itemsets generation is not possible depending on the minimum support. But the modified apriori will take less time and space compared to traditional apriori approach. Each maximum frequent itemset is treated as one cluster which gives the initial clusters. In this way, the generated frequent sets are the ones which have particular set of keywords in common and hence are closely related. This helps in deciding the number of clusters and also the centroids of these clusters which is simply the centroid of the respective frequent itemsets. The details are discussed in Algorithm 6.1.

6.2.3 Performing traditional clustering on the centroids of the initial clusters

After generating the initial clusters by the proposed approach, next is to see how the traditional clustering algorithms perform on these initial clusters. It is known that most of the traditional clustering algorithms need the number of clusters to be formed as the input before the clustering process starts. This problem is handled by generating the initial clusters either at the end of the execution of the proposed approach or based on the requirement, stopping the process at some stage which satisfies the minimum support. This is similar to hierarchical clustering where we stop the clustering process at some stage and the number of clusters generated are sent as the input to a traditional clustering technique such as k -means. Next, FCM, VSM and k -means techniques are run on these initial clusters to generate the final clusters (FC_i) (Algorithms 6.2-6.4). For VSM, the cosine-similarity between each document and the centroid of each cluster are calculated. A document d is assigned to a cluster c , if d 's cosine-similarity score is maximum for c compared to other clusters. This process is repeated till the centroid of each cluster is not changed.

6.3 Experimental Analysis

6.3.1 Performance measurement of traditional and modified apriori approach

For comparison purpose, Classic4 , 20-Newsgroups and Reuters datasets are used. All the four categories (shown in Figures 6.1-6.4) of Classic4, two categories (shown in Figures 6.5 and 6.6) of 20-Newsgroups and two categories (shown in Figures 6.7 and 6.8) of Reuters datasets are considered for experimental work. Each category of Classic4 dataset contains over 1000 documents, but for comparing the modified apriori approach with the conventional apriori approach, both algorithms are run over CASM, CISI, MED and CRAN document sets by using 200, 400, 600, 800 and 1000 documents from each set separately. Similarly, documents of different size have been considered for 20-Newsgroups and Reuters datasets. The support count used is 12 % of the number of documents of any category of a dataset. Figures 6.1 - 6.8 justify that the proposed algorithm has a better running time than the traditional apriori algorithm. It is understood from Figure 6.9 that even on varying the support count for the dataset (600 docu-

Algorithm 6.1 Modified Apriori Approach

- 1: **Input:** Term-document matrix (database T with keywords as transaction and documents as itemset) and minimum support (min_sup)
 - 2: **Output:** Maximum frequent itemset
 - 3: read the database T into a 2D-array and store the information of T in binary form in the array with transactions as rows and itemset as columns
 - 4: $k \leftarrow 1$.
 - 5: find frequent itemset, L_k from C_k , the set of all candidate itemset
 - 6: form C_{k+1} from L_k
 - 7: prune the frequent candidates by removing itemset from C_k whose elements do not come atleast $k-1$ times in L_k
 - 8: modify the entry in the 2D-array in memory to be zero for the itemset which is not occurring in any of the candidates in L_k
 - 9: check the size of transaction (SOT) attribute and remove transaction from 2D-array where $SOT \leq k$
 - 10: $k \leftarrow k+1$.
 - 11: repeat *step 7-10* until C_k is empty or transaction database T is empty
 - 12: *step 5* is called the frequent itemset generation step
 - 13: *step 6* is called as the candidate itemset generation step and *step 7-10* are prune steps
 - 14: details of first two steps are described below
 - 15: *Frequent itemset generation:*
 - 16: Scan database T and count each itemset in C_k , if the count is greater than min_sup, then add that itemset to L_k
 - 17: *Candidate itemset generation:*
 - 18: **for** $k = 1$ **do**
 - 19: $C_1 =$ (all itemset of length = 1)
 - 20: **end for**
 - 21: **for** $k > 1$ **do**
 - 22: generate C_k from L_{k-1} as follows:
 - 23: *The join step:*
 - 24: $C_k = (k-2)$ way join of L_{k-1} with itself.
 - 25: if both $a_1, \dots, a_{k-2}, a_{k-1}$ and a_1, \dots, a_{k-2}, a_k are in L_{k-1} , then add $a_1, \dots, a_{k-2}, a_{k-1}, a_k$ to C_k
 - 26: **end for**
-

ments of MED), the modified algorithm still outperforms the traditional apriori algorithm. The modified apriori algorithm focuses on the limitations of the traditional apriori algorithm and it can be seen from all the figures of different datasets that modified apriori approach outruns the traditional apriori approach. It can also be observed from the figures that when the number of documents increases, the difference in running time between the traditional and modified apriori algorithm becomes significant.

Algorithm 6.2 Performing FCM on initial clusters

```

1: Input: Initial clusters  $C_i$ , value of fuzziness parameter  $m$  and the document vector  $d_{i_{new}}$ 
2: Output: Final clusters,  $FC_i$ 
3: centroid calculation //find cluster centroids
4: for all frequent set  $f_i \in C_i$  do
5:    $c_i \leftarrow \phi$  // centroid
6:    $k \leftarrow \text{length}(f_i)$ 
7:   for all document  $d_j \in f_i$  do
8:     // all documents belongs to a frequent set
9:      $c_i \leftarrow c_i + d_j$ 
10:  end for
11:   $c_i \leftarrow c_i / k$ 
12: end for
13: // assign the documents to their respective clusters:
    Final clusters  $FC_i$  are generated by applying FCM algorithm (Appendix E) on the set
    of document vectors  $d_{i_{new}}$ , using initial clusters  $C_i$ , the centroid  $c_i$  and the fuzziness
    parameter  $m$ 
14: return  $FC_i$ 

```

Algorithm 6.3 Performing VSM on initial clusters

```

1: Input: Initial clusters  $C_i$  and the document vector  $d_{i_{new}}$ 
2: Output: Final clusters  $FC_i$ 
3: centroid calculation //find cluster centroids
4: for all frequent set  $f_i \in C_i$  do
5:    $c_i \leftarrow \phi$  // centroid
6:    $k \leftarrow \text{length}(f_i)$ 
7:   for all document  $d_j \in f_i$  do
8:      $c_i \leftarrow c_i + d_j$ 
9:   end for
10:   $c_i \leftarrow c_i / k$ 
11: end for
12: //assign the documents to their respective clusters
13: for all  $d_i \in d_{i_{new}}$  do
14:   for all  $c_j \in C_j$  (i.e. centroid of each cluster) do
15:      $\text{Similarity}[i][j] \leftarrow \text{cosineSimilarity}(d_i, c_j)$ 
16:   end for
17:   // find the maximum similarity cluster  $C_j$  for the document  $d_i$ 
   // stores the similarity between  $i^{th}$  document and  $j^{th}$  cluster
18:   // among the clusters from 0 to  $j - 1$ 
19:    $k \leftarrow \max(\text{Similarity}[i][0] \dots \text{Similarity}[i][j - 1]) // \forall k \in j$ 
20:    $C_k \leftarrow d_i$  //initial clusters now updated
21: end for
22: repeat step 5 till 21 until the cluster centroids are not changed and it gives the final clusters
     $FC_i$ .
23: return  $FC_i$ 

```

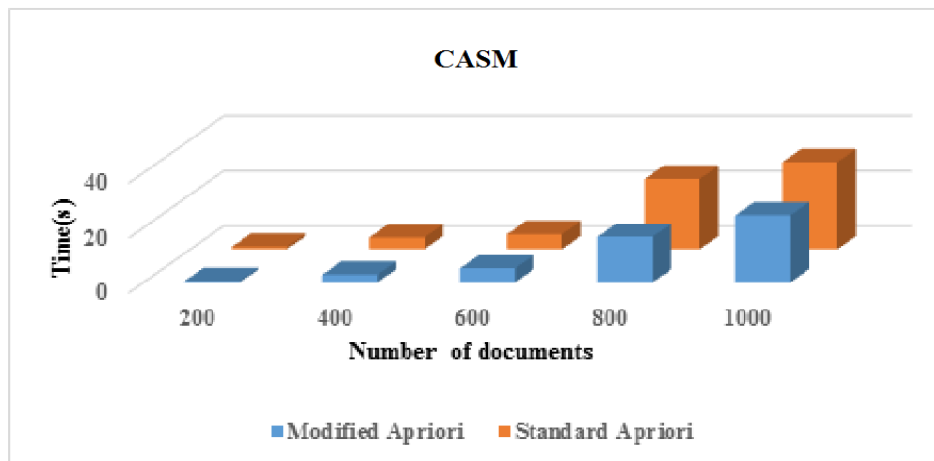


Figure 6.1: Apriori vs. Modified Apriori on CASM

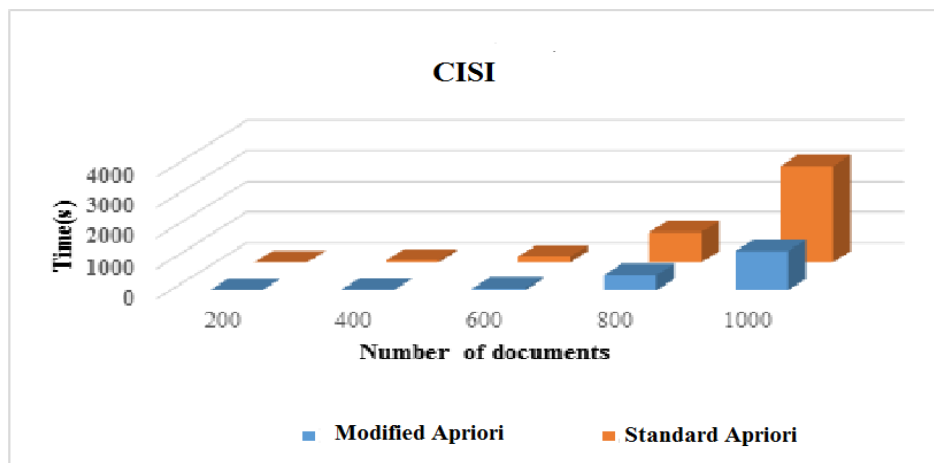


Figure 6.2: Apriori vs. Modified Apriori on CISI

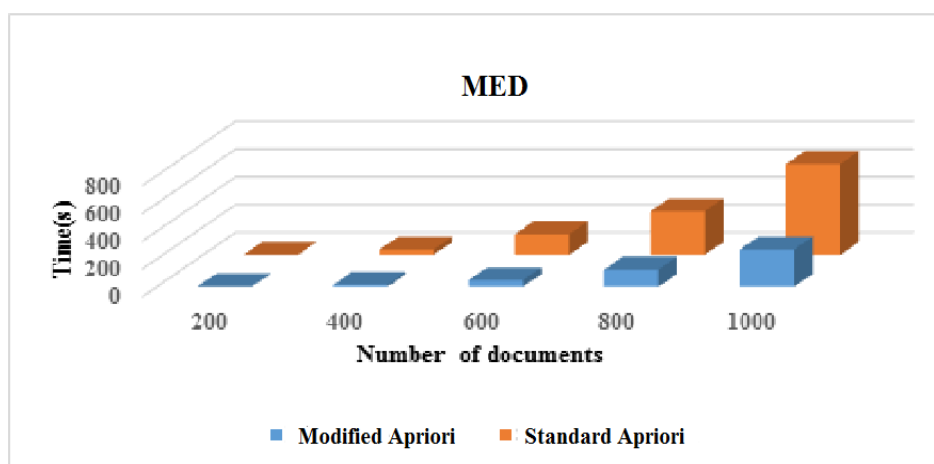


Figure 6.3: Apriori vs. Modified Apriori on MED

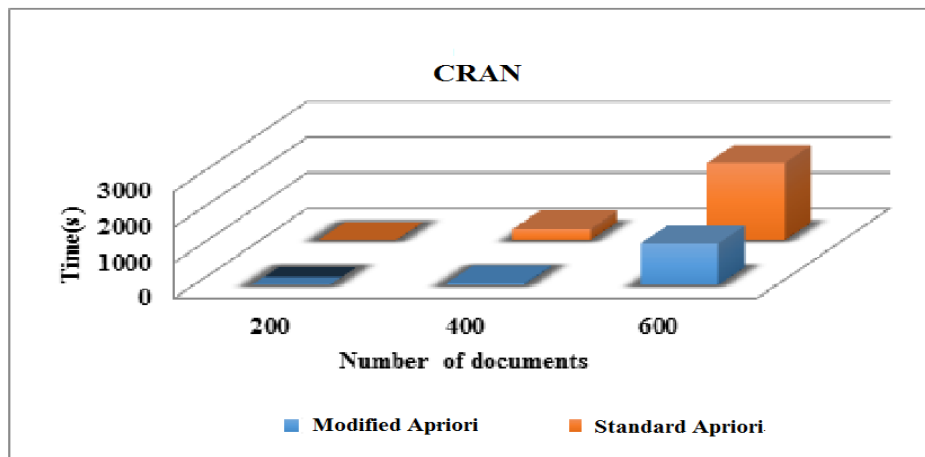


Figure 6.4: Apriori vs. Modified Apriori on CRAN

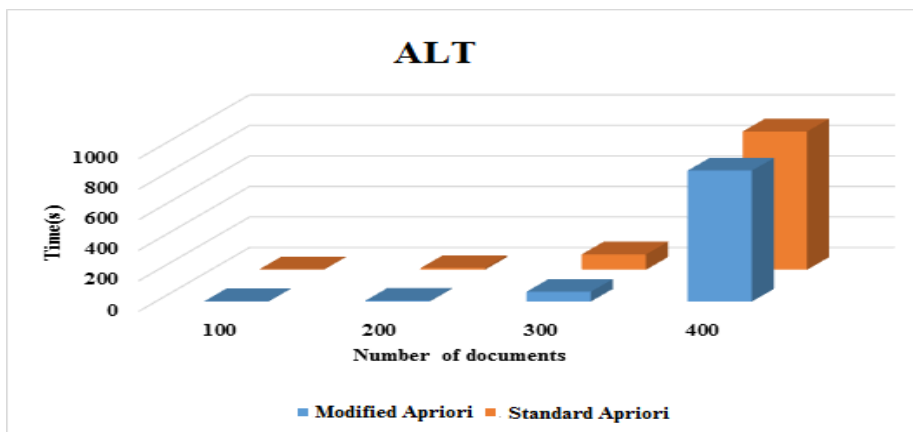


Figure 6.5: Apriori vs. Modified Apriori on alt

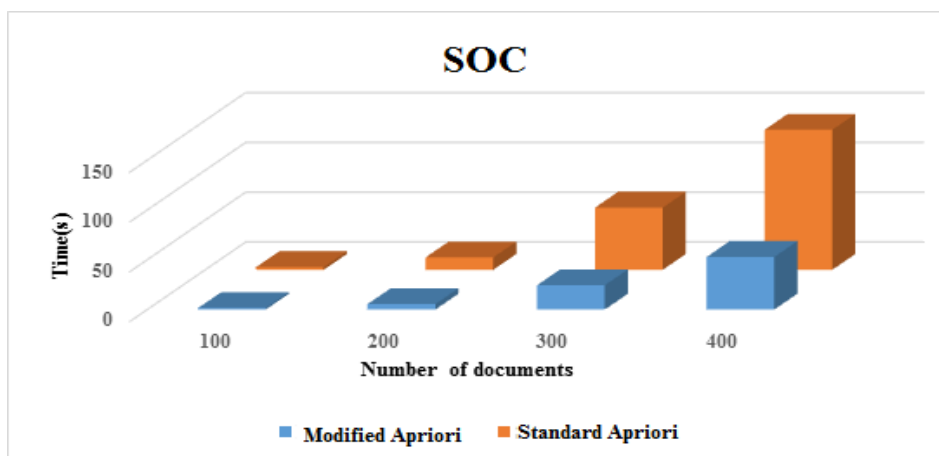


Figure 6.6: Apriori vs. Modified Apriori on soc

Algorithm 6.4 Performing k -means on initial clusters

```

1: Input: Initial clusters  $C_i$  and the document vector  $d_{i_{new}}$ 
2: Output: Final clusters  $FC_i$ 
3: centroid calculation //find cluster centroids
4: for all frequent set  $f_i \in C_i$  do
5:    $c_i \leftarrow \phi$  // centroid
6:    $k \leftarrow \text{length}(f_i)$ 
7:   for all document  $d_j \in f_i$  do
8:      $c_i \leftarrow c_i + d_j$ 
9:   end for
10:   $c_i \leftarrow c_i / k$ 
11: end for
12: //assign the documents to their respective clusters
13: for all  $d_i \in d_{i_{new}}$  do
14:   for all  $c_j \in C_j$  (i.e. centroid of each cluster) do
15:      $\text{distance}[i][j] \leftarrow \text{euclideanDistance}(d_i, c_j)$  // stores the distance between
        $i^{\text{th}}$  document and  $j^{\text{th}}$  cluster
16:   end for
17:   // find the minimum distance cluster  $C_j$  for the document  $d_i$ 
18:   // among the clusters from 0 to  $j - 1$ 
19:    $k \leftarrow \min(\text{distance}[i][0] \dots \text{distance}[i][j - 1])$  //  $\forall k \in j$ 
20:    $C_k \leftarrow d_i$  //initial clusters now updated
21: end for
22: repeat step 5 till 21 until the cluster centroids are not changed and it gives the final clusters
        $FC_i$ .
23: return  $FC_i$ 

```

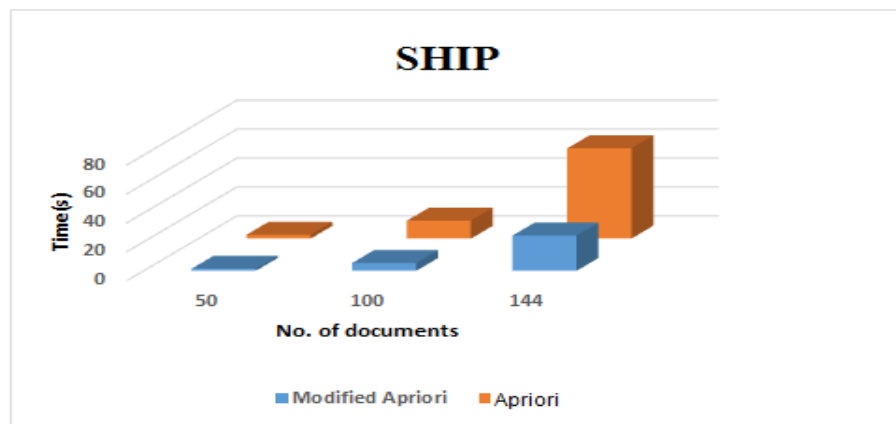


Figure 6.7: Apriori vs. Modified Apriori on ship

6.3.2 Performance measurement of traditional clustering algorithms

To measure the performance of the traditional clustering algorithms (FCM, VSM and k -means) on the initial clusters generated by the modified apriori approach, we form two separate big cor-

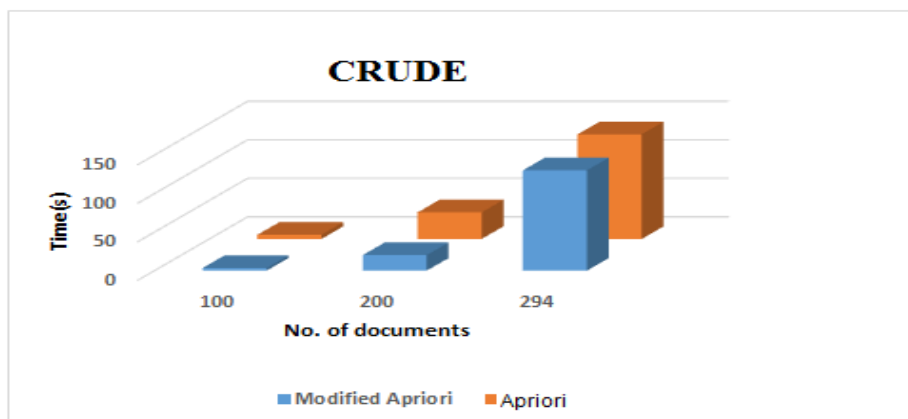


Figure 6.8: Apriori vs. Modified Apriori on crude

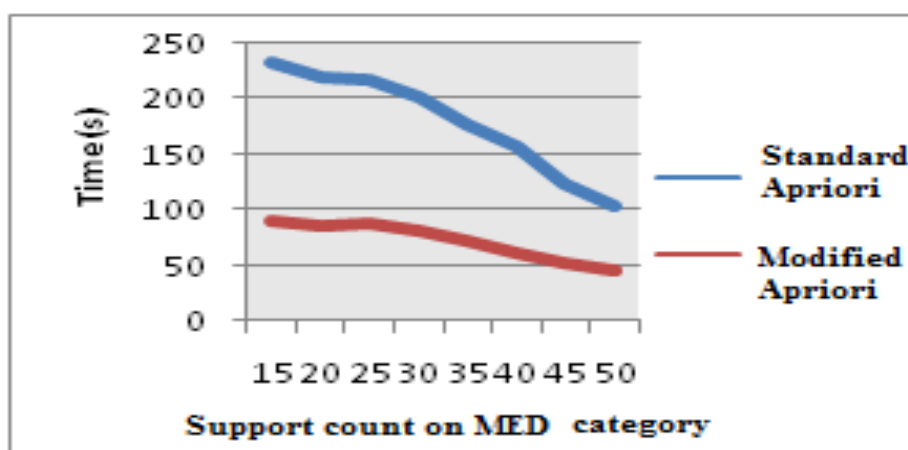


Figure 6.9: Support count graph for Apriori vs. Modified Apriori

pus, one for each dataset by collecting approximately 18800 and 7600 documents from all 7 categories (20 classes) of 20-Newsgroups and 8 categories (8 classes) of Reuters datasets, respectively. As classic4 dataset has only 4 categories (4 classes), hence, we prepare small corpus of different sizes of 200, 400, 600, 800 and 1000 documents collected from all 4 categories. Now, it means that the corpus is having documents from all categories mixed together in one place. On each corpus, we first ran our modified apriori approach and obtained the initial clusters. Then other state-of-the-art clustering algorithms i.e. FCM, VSM and k -means are run separately on these initial clusters to form the final clusters. Our modified apriori approach generates 22 clusters for 20-Newsgroups, 10 clusters for Reuters and 4 clusters for each set of documents of Classic4 datasets depending on the value of minimum support. From 22 clusters

of 20-Newsgroups, we then checked to which class out of 20 classes each cluster belonged by computing the cosine-similarity between each document of that cluster with all the documents of each class. A cluster C_i belongs to a class CG_j iff most of the documents of CG_j fall into C_i . This process is repeated for all the obtained clusters of 20-Newsgroups. Finally, all 22 clusters are distributed among 20 classes of 20-Newsgroups based on their cosine-similarities score so that some of the classes received more than one clusters. The same process is also repeated for Reuters dataset, where all 10 clusters are distributed among 8 classes. But for Classic4, one-to-one mapping is done as 4 clusters are obtained from each set and there are 4 classes. Algorithm 6.5 illustrates the complete mechanism to assign different clusters to their respective classes. After assigning the respective clusters to their corresponding class, if any class received more than one cluster, then we merge them into a single cluster so that each class should have only one cluster ($Cluster_{new}$). This gives number of classes equal to the number of new clusters i.e. $Cluster_{new}$, which makes the performance measurement process of different clustering technique more simple. Next step is to measure the performance of each $Cluster_{new}$. For this, the precision and recall are calculated as follows:

$$precision = \frac{a}{b}, \quad recall = \frac{a}{d} \quad (6.1)$$

where, ‘ a ’ is the number of documents of a particular class found in it’s correspond $Cluster_{new}$, ‘ b ’ is the number of documents in $Cluster_{new}$ and ‘ d ’ is the number of documents in that particular class. The average performances of both datasets are calculated using the following equations:

$$Average_{precision} = \frac{\sum_{i=1}^n (p_i \cdot d_i)}{N} \quad (6.2)$$

$$Average_{recall} = \frac{\sum_{i=1}^n (r_i \cdot d_i)}{N} \quad (6.3)$$

$$Average_{F-measure} = \frac{\sum_{i=1}^n (f_i \cdot d_i)}{N} \quad (6.4)$$

where, n represents the number of classes, f_i, p_i, r_i, d_i are the F-measure, precision, recall and the total number of documents present in i^{th} class, respectively. N is the total number of documents considered for clustering. Figure 6.10 shows the performance of different clustering techniques on CISI datasets. Similarly, in Table 6.1, the performance measurement of different

clustering techniques on 20-Newsgroups and Reuters are shown. Results shows that FCM outperforms the other two clustering algorithms.

Algorithm 6.5 Assigning the respective cluster(s) to the corresponding class

```

1: Input: Cluster  $C_i$  generated by the proposed approach and  $m$  classes ( $CG_j$ ) of a dataset
2: Output: Categories with their respective clusters
3: for  $i$  in 1 to  $k$  do
4:   //  $k$  clusters
5:   for  $j$  in 1 to  $m$  do
6:     //  $m$  classes
7:      $count \leftarrow 0$ 
8:     for all  $d \in i$  do
9:       for all  $d' \in j$  do
10:         $cs \leftarrow cosineSimilarity(d, d')$ 
11:        if  $cs = 1$  then
12:           $count \leftarrow count + 1$ 
13:        end if
14:      end for
15:    end for
16:     $a[i][j] \leftarrow count$ 
17:  end for
18:   $maximum \leftarrow a[i][1]$  // let class 1 received the maximum number of documents of
   $i^{th}$  cluster
19:  for  $l$  in 2 to  $m$  do
20:    if  $a[i][l] > maximum$  then
21:       $maximum \leftarrow a[i][l]$ 
22:       $n \leftarrow l$ 
23:    end if
24:  end for
25:   $CG_n \leftarrow C_i \forall n \in m$  // assign the  $i^{th}$  cluster to  $n^{th}$  class because  $n^{th}$  class received
  the maximum number of documents of  $i^{th}$  cluster
26: end for

```

Table 6.1: Performance comparison of different clustering techniques

Cluster	20- Newsgroups			Reuters		
	Precision	Recall	F-measure	Precision	Recall	F-measure
k -means	72.12	74.42	73.25	61.44	65.60	63.45
VSM	72.74	76.24	74.45	66.72	64.25	65.46
FCM	79.42	77.52	78.46	66.39	69.88	68.09

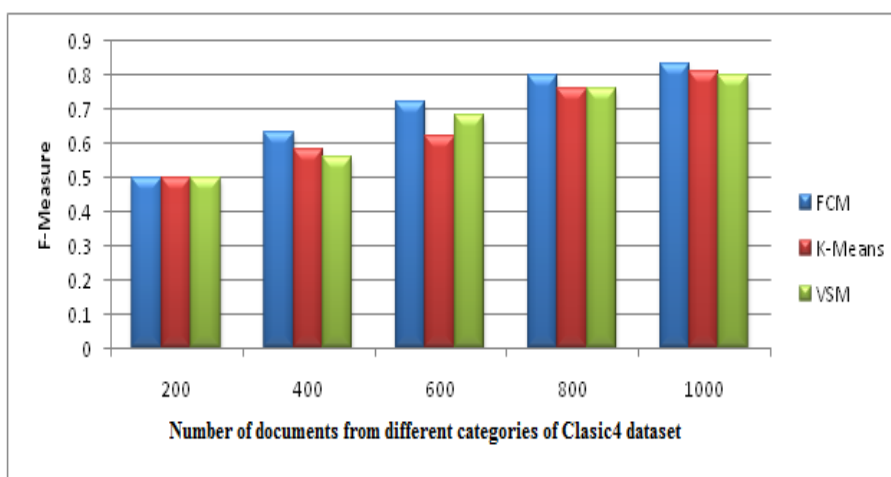


Figure 6.10: Performance comparison on Classic4

6.4 Summary

This chapter proposes a novel clustering technique based on apriori approach to cluster the text documents. In this technique, new modified apriori approach has been proposed and it is compared with traditional apriori algorithm. The proposed modified apriori approach when run to a corpus of web documents produced the same clusters that a traditional apriori approach can. However, experimentally it has been proved that modified apriori approach is more efficient and faster than traditional apriori approach. This is so because at each step, the information, i.e. documents from the corpus have been removed, which is no longer required and in turn it reduces the unnecessary comparisons. Hence, it saves a lot of time. First the initial clusters are formed, then FCM, VSM and k -means techniques are run on it separately. Classic4, 20-Newsgroups and Reuters datasets are used for experimental purpose. We found that FCM gives better clusters compared to VSM and k -means. This work is further extended where each cluster is labeled based on their content and is discussed in the next chapter.

CHAPTER 7

A HYBRID APPROACH FOR CLUSTER LABELING

7.1 Introduction

The objective of this chapter is to label the clusters (clusters generated by the modified apriori approach as discussed in Chapter 6) of a corpus. To obtain the labels, first the important keywords are selected from each cluster. Next, these keywords of each cluster are sent to Wikipedia for generating the potential (candidate) labels. Using *Mutual Information (MI)-score* technique, the candidate labels are ranked and the top ranked candidates are recommended as the actual labels of a cluster. Neither do any standard benchmarks exist for comparing different cluster labeling techniques nor do any standard evaluation mechanism exist for labeling the clusters. The uniqueness of this approach is that it is able to find out the actual labels of a cluster within the least possible number (just top three) of suggested labels generated by Wikipedia.

7.2 Methodology

7.2.1 Document Pre-processing

Let P be a corpus consisting of different classes $C = \{C_1, C_2, \dots, C_m\}$ of documents. Documents of different classes are pre-processed (discussed in Chapter 3) and then each document

is represented in vector form using vector space model. Then, documents of all m classes are collected together into one place which makes the dimension of P as $r \times l$, where r and l are number of terms and documents, respectively.

7.2.2 Clusters Generation

After generating the term-document matrix of P , the modified apriori algorithm (discussed in Chapter 6) is used to cluster the documents based on the minimum support count. This generates n initial clusters. Then, FCM clustering technique (one of the techniques which gives good performance compared to VSM and k -means as discussed in Chapter 6) is run on these n initial clusters to obtain the n final clusters, $FC = \{c_1, c_2, \dots, c_n\}$. The dimension of each $c_j (j = 1, \dots, n)$ is $r \times p$ as shown in Table 7.1.

Table 7.1: Term-document matrix of each cluster

	d_1	d_2	d_3	...	d_p
t_1	t_{11}	t_{12}	t_{13}	...	t_{1p}
t_2	t_{21}	t_{22}	t_{23}	...	t_{2p}
t_3	t_{31}	t_{32}	t_{33}	...	t_{3p}
·	·	·	·	...	·
·	·	·	·	...	·
·	·	·	·	...	·
t_r	t_{r1}	t_{r2}	t_{r3}	...	t_{rp}

7.2.3 Top Documents selection

For selecting the top documents from each c_j , first the centroid (\vec{c}_j) of each c_j is computed as follows:

$$\vec{c}_j = \frac{\sum_{i=1}^p \vec{d}_i}{p} \quad (7.1)$$

Then the cosine-similarity between each document $\vec{d}_i \in c_j$ with \vec{c}_j is calculated. Next, the top $m\%$ (decided empirically) of documents which have highest cosine-similarity values from each c_j are selected and other documents are discarded. The purpose of selecting top documents from each cluster is to check how well these documents represent their corresponding cluster because it has been acknowledged that the closer (having maximum similarity) the document to the centroid of a cluster, the better it represents that cluster [9]. By discarding

unimportant documents, the dimension of each c_j is reduced to $r \times s$, where $s < p$ (Table 7.2).

Table 7.2: Reduced term-document matrix of each cluster

	d_1	d_2	d_3	...	d_s
t_1	t_{11}	t_{12}	t_{13}	...	t_{1s}
t_2	t_{21}	t_{22}	t_{23}	...	t_{2s}
t_3	t_{31}	t_{32}	t_{33}	...	t_{3s}
.
.
.
t_r	t_{r1}	t_{r2}	t_{r3}	...	t_{rs}

7.2.4 Representative Keywords Selection

After generating the reduced term-document matrix for each cluster c_j , top keywords from each c_j are selected. For this, first the normalized chi-square score of all keywords of each cluster are computed by applying Chi-Square feature selection technique on each cluster c_j . To find out the top ranked keywords of each c_j , the following steps are followed :

- 1) Randomly select a keyword W from c_j 's keyword list and then using Wordnet, prepare a synonym-list of W .
- 2) Search for those keywords which are common both to the synonym-list of W and keyword-list of c_j .
- 3) Remove those common keywords from the keyword-list of c_j and at the same time add them to a new list called *synonym-required-list* of W .
- 4) Repeat steps 1 - 3, till the keyword-list of c_j gets exhausted. At the end, synonym-required-lists are generated, one for each of those keywords selected randomly from c_j 's keyword list.
- 5) Now, consider one keyword from each synonym-required-list of c_j which has the highest chi-square value in that list, and merge these keywords to a new list called *important keyword-list* of c_j . Finally, select top ' k ' keywords (based on their chi-square values) from the *important keyword-list* called the representative keywords $R(c_j)$ of c_j .

7.2.5 Generating Candidate Labels

After obtaining the top k keywords of c_j , the next step is to generate the candidate labels and for this, Wikipedia is used. First, the queries for Wikipedia are generated by using the top k keywords of c_j . The reason behind selecting Wikipedia as the external source is that it is a better media through which one can get quality information. The following steps brief how the candidate labels are generated using Wikipedia:

- 1) Given a set of top k keywords as queries q , the Wikipedia database is searched for the results. A Python module called Python Wikipediabot Framework¹ (a set of tools provided to automate the work on MediaWiki sites like Wikipedia, etc.) is used for this purpose.
- 2) Further, to strengthen the relative importance of keywords of the query q , q is executed against the index of Wikipedia which is a collection of disjunctions of the top k keywords. As a result, a collection of sorted documents $D_j(q)$ based on their similarity score with q is returned.
- 3) For each document $d_j \in D_j(q)$, a set of categories associated with d_j and the title of d_j have been considered as the potential candidate labels for c_j (represented as $L(c_j)$).

7.2.6 Evaluating Candidate Labels

Further, to evaluate which candidate labels are better among all the candidate labels generated by Wikipedia for a cluster c_j , the following steps are used:

- 1) To evaluate the candidate labels generated by Wikipedia, Mutual Information (MI-Score)² is used, which judges the candidate labels $L(c_j)$ based on their semantic relationship with the associated documents in the cluster c_j . MI-score is used because it is considered as a good indicator of relevance between two random variables³.
- 2) The average point-wise mutual information (*PWMI*)[144] of a label $l \in L(c_j)$ with the set of top keywords of c_j generates the MI-scores for $l \in L(c_j)$. The average *PWMI* of l

¹<https://www.mediawiki.org/wiki/Manual:Pywikibot/Scripts>

²<http://nlp.stanford.edu/IR-book/html/htmledition/mutual-information-1.html>

³T.M Cover and J.A Tomas *Elements of Information Theory* John Wiley & Sons, 1991

with the set of top keywords of c_j gives the *semantic distance* between l and the content of c_j . Minimum distance indicates that label l is very close to c_j .

- 3) The collections of Wikipedia are used as a data source (i.e. the proposed approach used the text of the first ‘ n ’ result received after passing the query q to the Wikipedia index) for MI-score as they are large enough to give the correct results as well as relevant enough to the content of the cluster c_j . The following equation determines the MI-score for a label $l \in L(c_j)$.

$$MI(l, R(c_j)) = \sum_{i \in R(c_j)} PWMI(l, i|X) * w(i) \quad (7.2)$$

where $R(c_j)$ are the representative keywords (a set of top k keywords) of c_j , $w(i)$ is the strength or weight of the top keyword, $i \in R(c_j)$. $PWMI$ will be measured in an external textual source X i.e. corpus. Measurement of $PWMI$ between a pair of keywords is done as follows:

$$PWMI(l, i|X) = \log \left(\frac{prob(l, i|X)}{prob(l|X) * prob(i|X)} \right) \quad (7.3)$$

Maximum likelihood estimation⁴ approximates the probability of ‘ y ’ (denotes a pair of terms or a single term) in the given corpus as follows:

$$prob(y|X) = \frac{No. \ of \ occurrence \ of \ y \ in \ X}{Total \ number \ of \ keywords \ in \ X} \quad (7.4)$$

7.3 Experimental Analysis

The proposed algorithm is tested on 20-Newsgroups and Reuters datasets. The accuracy of the proposed approach is measured on both datasets by using the following equation:

$$Accuracy = \frac{a}{b} \quad (7.5)$$

where, ‘ a ’ represents the number of clusters whose labels are found within the top suggested labels returned by Wikipedia and ‘ b ’ represents the total number of clusters considered for labeling. The top 3 suggested labels returned by Wikipedia are considered for measuring the accuracy on each dataset. The reason to choose only the top 3 suggested labels of Wikipedia

⁴<https://onlinecourses.science.psu.edu/stat414/node/191>

as the candidate labels is that we try to find out the original labels within the least possible number of suggested labels in order to show the efficiency of the proposed approach. The MI-score of each of the suggested labels generated by Wikipedia on both datasets are shown in an increasing order of the label, i.e. candidate ‘Label 1’ has less MI-score than candidate ‘Label 2’ and finally candidate ‘Label 3’ has the maximum MI-score. This measures the semantic distance between the candidate labels and the associated documents of the cluster C_i . Hence, candidate ‘Label 1’ has minimum semantic distance and candidate ‘Label 3’ has maximum semantic distance from the cluster C_i which indicates candidate ‘Label 1’ is very close to C_i compared to candidate ‘Label 2’ and candidate ‘Label 3’.

7.3.1 Using 20-Newsgroups dataset

20-Newsgroups contains a certain number of documents belonging to every cluster along with a name given to that cluster. The name of the clusters is hierarchical with ‘.’ as the level discriminator, e.g. the name comp.graphics represents the newsgroup graphics under section computers. The pre-processed newsgroups documents of different classes (clusters consider here) are grouped together to form a corpus and then divided into different clusters by first applying our earlier modified apriori approach on each cluster to generate n (here $n = 22$, decided empirically) initial clusters, and then applying FCM technique on these n initial clusters to obtain n final clusters. Next, the assignment of each cluster to their corresponding class is done (same way as discussed in section 6.2.3). Then, top k (here k is set as 3) keywords based on their chi-square values from the *important keyword list* are extracted from each cluster of 20-Newsgroups. The generated representative keywords are shown in Table 7.3. while Table 7.4 shows their chi-square values. For in-depth discussion, we have just explained about one cluster (cluster 2) explicitly as follows:

As one can see that the proposed algorithm returned keywords corresponding to the leaf of the title (i.e. leaf of ‘comp.graphics’ is ‘graphics’). The algorithm also tried to return the intermediate titles as can be seen in cluster 2 where both ‘comp’ and ‘graphics’ are returned for the title ‘comp.graphics’. The result of cluster 2 shows that the combination of representative keywords gives us a more appropriate label. Figure 7.1 demonstrates some top keywords of cluster 2 and their corresponding chi-square values, out of which, top 3 keywords known as representative keywords of cluster 2 are selected. All the combinations of these top 3 representative keywords

are passed to Wikipedia and the resulting suggested labels are evaluated with MI-score. Top 3 suggested labels are recommended as the candidate labels based on their MI-score which is good enough to label the cluster. Figure 7.2 demonstrates the semantic distance between candidate labels generated by Wikipedia from cluster 2. The less the semantic distance, the more the chances of considering that candidate label as the appropriate label of the corresponding cluster.

Table 7.5 shows the suggested labels for all the clusters of 20-Newsgroups by the proposed approach. Table 7.6 shows the MI-Score of candidate labels respectively. Figures 7.3 and 7.4 demonstrate the chi-square values of top 3 keywords and the MI-score of top 3 suggested labels of each cluster respectively. In Figure 7.3, one can see that the keyword ('politics') of cluster 17 received the highest chi-square value of 0.42 followed by 'sci' of cluster 15 having chi-square value of 0.406 compared to all the 20 clusters. Similarly, in Figure 7.4, 'Label 1' ('Christian') of cluster 1 received the lowest MI-value of 3.34 compared to all the 20 clusters.

Cluster by cluster analysis shows the advantages of the proposed approach. In cluster 1, due to the closeness of words 'christian' and 'religion' in the external corpus, the corresponding labels have both almost equal MI-values while label 'atheism' is valued above both of them as it contains both of these words with high frequency. Results of cluster 2 suggest label 'Computer Graphics' which is a semantically better label than both 'computer' and 'graphics'. Results of all the clusters except clusters 3, 4 and 5 are good. As one can see from the original labels, clusters 3, 4 and 5 are semantically very close to each other. On top of that, they all use word 'comp' in high frequency rather than the full form, 'computer'. This disambiguation produces confusion which rates unrelated titles above the actual titles. For cluster 3, the suggested title 'Window', disambiguation for Window (structure) and Window (computing) validates our claim. This pattern is also observed when word 'sci' and 'misc' are used in place of 'science' and 'miscellaneous'. These short forms are not part of the language and hence are needed to be taken care of by some external corpus.

The representative keywords generation of our approach is both efficient and accurate as can be seen from Table 7.3. The accuracy of our proposed approach is more than 85%, as out of 20 clusters, labels of 17 clusters are good and the remaining 3 clusters' (cluster 3, 4 and 5) labels are satisfactory.

Table 7.3: Representative keywords of 20-Newsgroups

Cluster No.	Cluster Name	Representative Keywords
1	alt.atheism	atheism, religion, christian
2	comp.graphics	comp, graphic, computer
3	comp.os.ms-windows.misc	comp.os.ms-windows.misc
4	comp.sys.ibm.pc.hardware	ibm, sys, comp
5	comp.sys.mac.hardware	comp, sys, hardware
6	comp.windows.x	comp, window, computer
7	misc.forsale	forsale, misc, computer
8	rec.auto	auto, car, sport
9	rec.motorcycles	motorcycle, bike, car
10	rec.sport.baseball	sport, baseball, comp
11	rec.sport.hockey	sport, hockey, med
12	sci.crypt	crypt, sci, comp
13	sci.electronics	sci, electronics, computer
14	sci.med	med, sci, misc
15	sci.space	sci, space, computer
16	soc.religion.christian	hedrick, religion, christian
17	talk.politics.guns	politics, gun, auto
18	talk.politics.mideast	mideast, politics, religion
19	talk.politics.misc	politics, misc, religion
20	talk.religion.misc	misc, religion, atheism

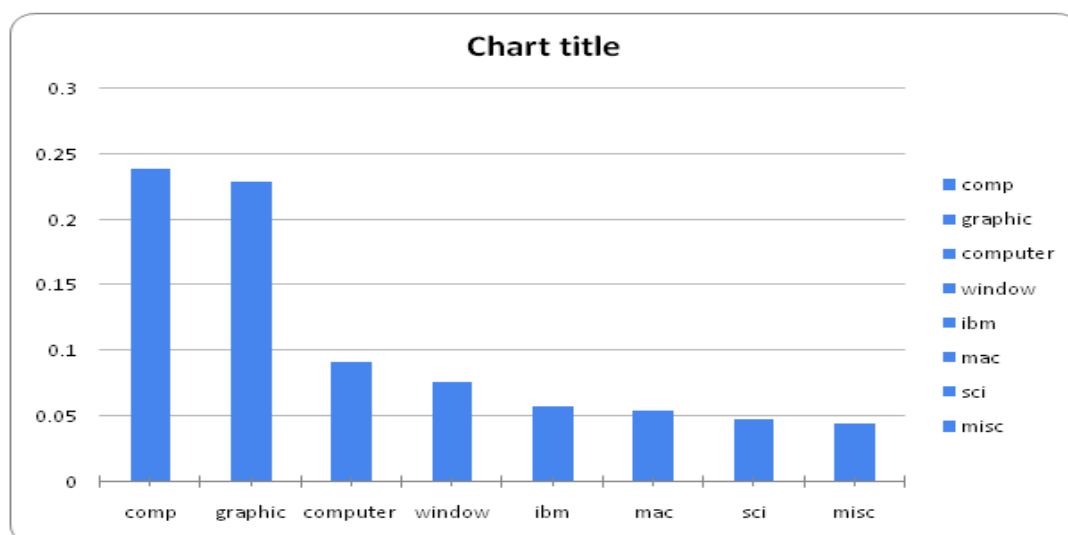


Figure 7.1: Top keywords of cluster 2 on 20-Newsgroups

Table 7.4: Chi-Square values of representative keywords on 20-News groups

Cluster No.	Cluster Name	Keyword 1	Chi-2 value	Keyword 2	Chi-2 value	Keyword 3	Chi-2 value
1	alt. atheism	atheism	0.0542	religion	0.0094	christian	0.0065
2	comp. graphics	comp	0.2392	graphic	0.2286	computer	0.0917
3	comp. os. ms-windows. misc	window	0.1150	comp	0.1133	misc	0.1133
4	comp. sys. ibm. pc. hardware	ibm	0.2300	sys	0.2300	comp	0.2300
5	comp. sys. mac. hardware	comp	0.2129	sys	0.2129	hardware	0.2129
6	comp. windows. x	comp	0.3309	window	0.2681	computer	0.0994
7	misc. forsale	forsale	0.2680	misc	0.1121	computer	0.0741
8	rec.auto	auto	0.0603	car	0.0158	sport	0.0029
9	rec. motorcycles	motorcycle	0.0363	bike	0.0076	car	0.0032
10	rec. sport. baseball	sport	0.0448	baseball	0.0054	comp	0.0007
11	rec. sport. hockey	sport	0.0701	hockey	0.0012	med	0.0003
12	sci. crypt	crypt	0.2741	sci	0.2741	comp	0.1182
13	sci. electronics	sci	0.2928	electronics	0.2928	computer	0.0492
14	sci. med	med	0.3596	sci	0.3596	misc	0.0883
15	sci space	sci	0.4016	space	0.2590	computer	0.0663
16	soc. religion. christian	hedrick	0.2740	religion	0.1885	christian	0.1414
17	talk. politics. guns	politics	0.4201	gun	0.3154	auto	0.0043
18	talk. politics. mideast	mideast	0.2071	politics	0.2071	religion	0.0015
19	talk. politics. misc	politics	0.2207	misc	0.2207	religion	0.0413
20	talk. religion. misc	misc	0.2551	religion	0.2515	atheism	0.1317

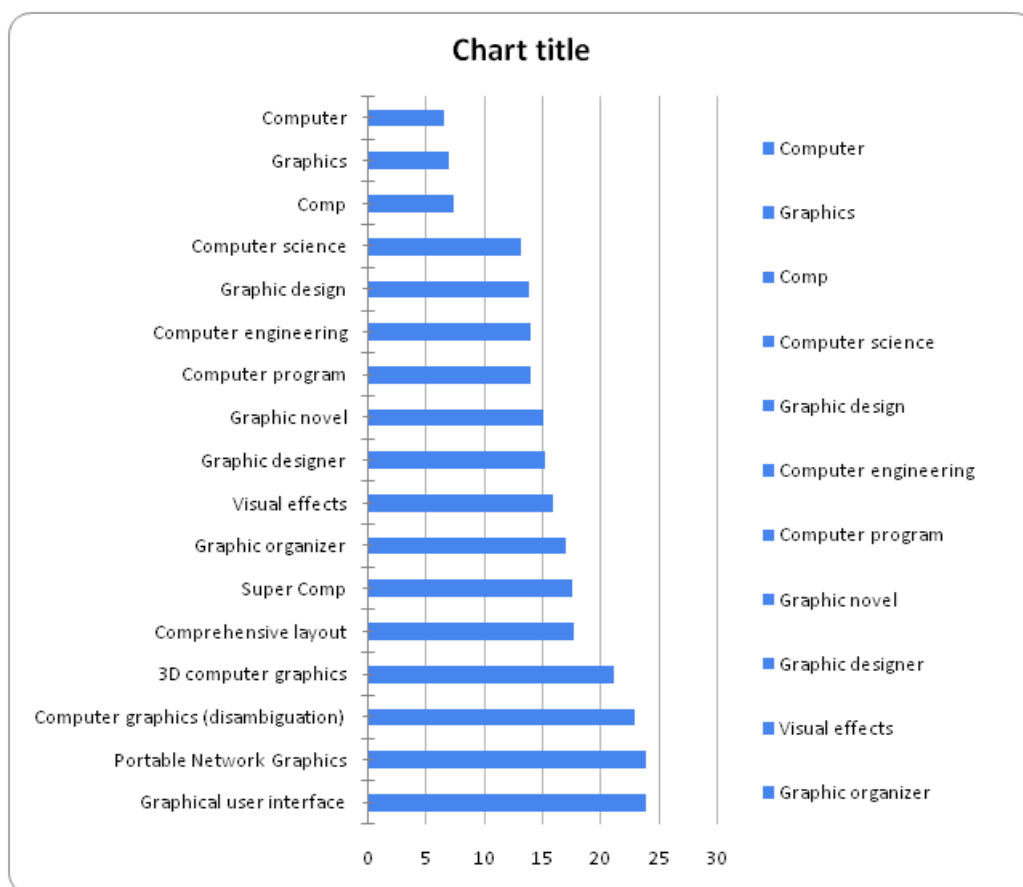


Figure 7.2: Suggested candidate labels and their semantic distances from cluster 2 on 20-News groups

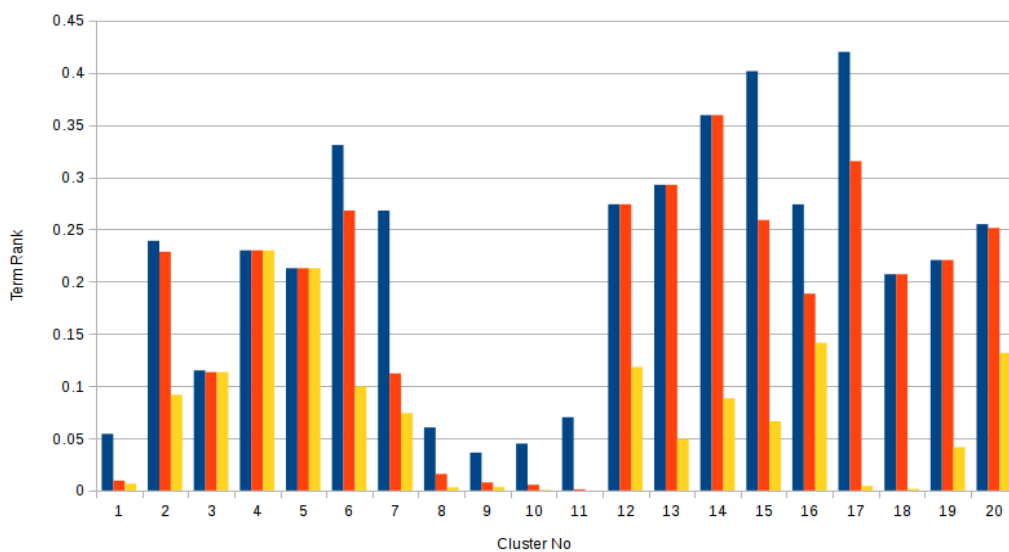


Figure 7.3: Keyword ranking of top 3 representative keywords of each cluster on 20-News groups

Table 7.5: Suggested candidate labels of 20-Newsgroups

Cluster Name	Suggested Label 1	Suggested Label 2	Suggested Label 3
alt. atheism	Christian	Religion	Atheism
comp. graphics	Computer	Graphics	Computer Graphics
comp. os. ms-windows. misc	Window	OpenBSD	DOS
comp. sys. ibm. pc. hardware	IBM	DOS	Comp
comp. sys. mac. hardware	DOS	Comp	Unix
comp. windows. x	Computer	DOS	Window
misc. forsale	Computer	Usenet	Misc
rec. autos	Auto	C.a.R.	Car
rec. motorcycles	C.a.R.	Car	MotorCycle
rec. sport. baseball	Baseball	Sport	Comp
rec. sport. hockey	Sport	Hockey	Injury
sci. crypt	Comp	SCI	Crypt
sci. electronics	Computer	Computing	Electronics
sci. med	SCI	Misc	FAQ
sci space	Computer	SPACE	Space
soc. religion. christian	Christian	Religion	Christianity
talk. politics. guns	Auto	Gun	Politics
talk. politics. mideast	Religion	Politics	Islam
talk. politics. misc	Religion	Politics	Misc
talk. religion. misc	Religion	Atheism	Usenet

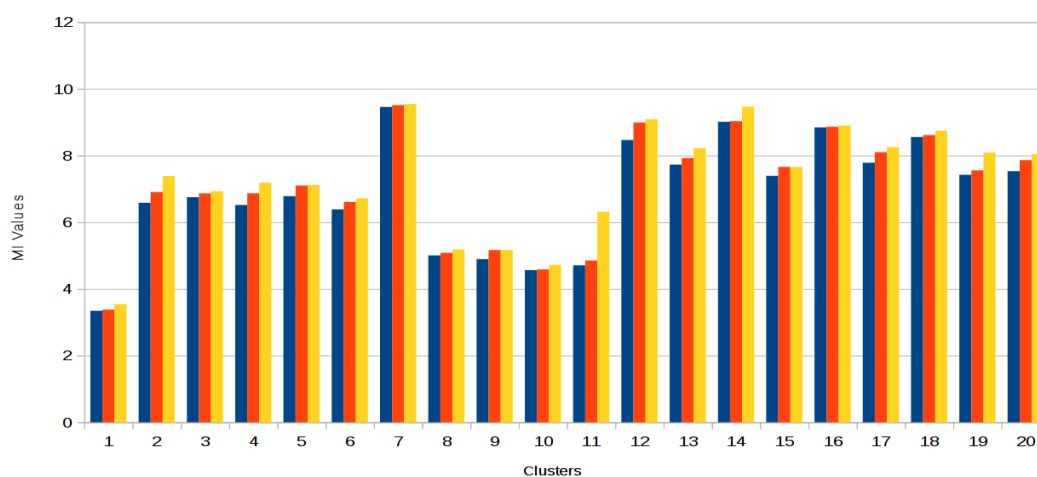


Figure 7.4: MI-score of each cluster on 20-Newsgroups

Table 7.6: MI-score for suggested candidate labels on 20-News groups

Cluster No.	Cluster Name	Label 1	MI-Value	Label 2	MI-Value	Label 3	MI-Value
1	alt. atheism	Christian	3.343	Religion	3.381	Atheism	3.545
2	comp. graphics	Computer	6.586	Graphics	6.908	Comp	7.386
3	comp. os. ms-windows. misc	Window	6.752	OpenBSD	6.864	DOS	6.933
4	comp. sys. ibm. pc. hardware	IBM	6.519	DOS	6.869	Comp	7.188
5	comp. sys. mac. hardware	DOS	6.783	Comp	7.102	Unix	7.121
6	comp. windows. x	Computer	6.385	DOS	6.611	Window	6.719
7	misc. forsale	Computer	9.459	Usenet	9.513	Misc	9.547
8	rec.auto	Auto	5.006	Car	5.084	Sport	5.180
9	rec. motorcycles	Car	4.895	MotorCycle	5.167	Motorcycle	5.167
10	rec. sport. baseball	Sport	4.565	Baseball	4.584	Comp	4.720
11	rec. sport. hockey	Sport	4.708	Hockey	4.852	Injury	6.315
12	sci. crypt	Comp	8.466	SCI	8.992	Crypt	9.089
13	sci. electronics	Computer	7.731	Computing	7.924	Electronics	8.221
14	sci. med	SCI	9.012	Misc	9.032	FAQ	9.468
15	sci space	Computer	7.394	SPACE	7.661	Space	7.661
16	soc. religion. christian	Christian	8.845	Religion	8.866	Christianity	8.902
17	talk. politics. guns	Auto	7.785	Gun	8.101	Politics	8.254
18	talk. politics. mideast	Religion	8.557	Politics	8.616	Islam	8.748
19	talk. politics. misc	Religion	7.424	Politics	7.558	Misc	8.089
20	talk. religion. misc	Religion	7.533	Atheism	7.863	Usenet	8.046

7.3.2 Using Reuters dataset

The pre-processed documents of different classes of Reuters dataset are divided into different clusters by first applying the earlier modified apriori approach on each cluster to generate n (here $n = 10$, decided empirically) initial clusters and then applying FCM technique on these n initial clusters to obtain the n final clusters. Next, assignments of all 8 clusters to their corresponding classes are done (discussed in Chapter 6). The top 3 representative keywords of each cluster of Reuters dataset which are generated by using the proposed feature selection technique are shown in the Table 7.7. Table 7.8 shows the chi-square values of these 3 keywords based on which they have been selected from a cluster (also known as the representative keywords). After sending these top keywords of each cluster to Wikipedia, the suggested labels generated by Wikipedia and their MI-score are shown in Tables 7.9 and 7.10 respectively. Figures 7.5 and 7.6 demonstrate the chi-square values of top 3 top keywords and MI-Score of 3 candidate labels respectively. One can see that in Figure 7.5, the keyword ('pct') of cluster 5 has the highest chi-square value of 0.294 compared to all 8 clusters. Similarly, 'Label 1' ('bank') of cluster 6 in Figure 7.6 received the lowest MI-value of 6.36 among all the 8 clusters. From Table 7.10, it has also been observed that except clusters 1 and 3, the remaining clusters' label either directly match with the suggested labels or are semantically similar to them. Out of eight clusters, six clusters' labels are matching with the original labels and another two clusters' (clusters 1 and 3) labels are satisfactory. If we go deeper in accepting the suggested labels from Wikipedia, that is probably 'Label 4' or beyond some labels of it, then it may give the exact original label for clusters 1 and 3. This shows that the proposed approach's accuracy is more than 75%.

Table 7.7: Representative keywords of Reuters

Cluster No.	Cluster Name	Representative Keywords
1	acq	dtrs, company, mln
2	crude	barrel, oil, dtrs
3	earn	sh, rct, mln
4	grain	grain, pct, trade
5	interest	pct, bank, rate
6	money-fx	bank, pct, rate
7	ship	port, ship, pct
8	trade	reuter, trade, lrs

Table 7.8: Chi-square values of representative keywords on Reuters

Cluster No.	Cluster	Keyword 1	Chi-2 value	Keyword 2	Chi-2 value	Keyword 3	Chi-2 value
1	acq	dlrs	0.2027	company	0.1837	mln	0.1764
2	crude	barrel	0.1727	oil	0.1521	dlrs	0.1413
3	earn	sh	0.2065	rct	0.1826	mln	0.1737
4	Grain	grain	0.2284	pct	0.0670	trade	0.0352
5	interest	pct	0.2940	bank	0.2734	rate	0.2664
6	money-fx	bank	0.2484	pct	0.1154	rate	0.0868
7	ship	port	0.1371	ship	0.1037	pct	0.0947
8	trade	reuter	0.1810	trade	0.1162	lrs	0.0967

Table 7.9: Suggested candidate labels on Reuters

Cluster Name	Suggested Label 1	Suggested Label 2	Suggested Label 3
acq	company dlrs	railway	company
crude	iron	oil	barrel
earn	cts	hr	botany
grain	trade	grain	glencore
interest	bank	rate	hsbc
money-fx	bank	rate	hsbc
ships	ship	port	pct
trade	iron	trade	germany

Table 7.10: MI-score for suggested candidate labels on Reuters

Cluster No.	Cluster Name	Label 1	MI-Value	Label 2	MI-Value	Label 3	MI-Value
1	acq	company dlrs	8.448	railway	8.520	company	15.249
2	crude	iron	8.233	oil	8.241	barrel	8.297
3	earn	cts	7.591	hr	8.307	botany	8.415
4	grain	trade	6.925	grain	6.964	glencore	7.517
5	interest	bank	7.478	rate	7.686	hsbc	7.860
6	money-fx	bank	6.369	rate	6.607	hsbc	6.723
7	ship	ship	6.785	port	6.893	pct	6.912
8	trade	iron	8.991	trade	9.029	germany	9.043

7.4 Summary

This chapter proposed a cluster labeling technique which is the extension of the work discussed in Chapter 6. Using a feature selection technique, first top k keywords known as representative keywords are selected from each cluster which are later sent to Wikipedia for getting candidate labels. The candidate labels generated from Wikipedia are evaluated using MI-score. The approach generates the actual labels by just considering the top-3 candidate labels of Wikipedia. For experimental work, 20-Newsgroups and Reuters datasets are considered. The experimental results illustrate the accuracy of the proposed approach which is more

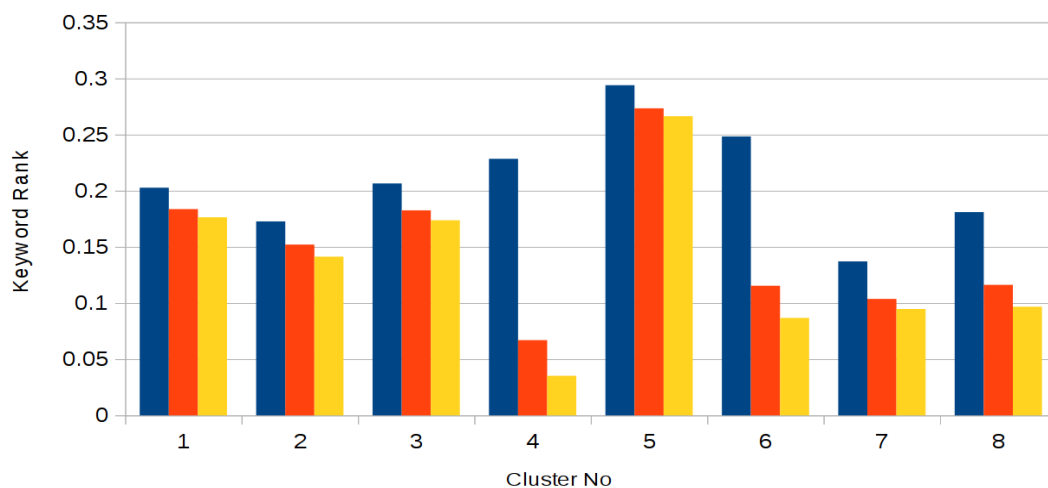


Figure 7.5: Top 3 representative keywords on Reuters

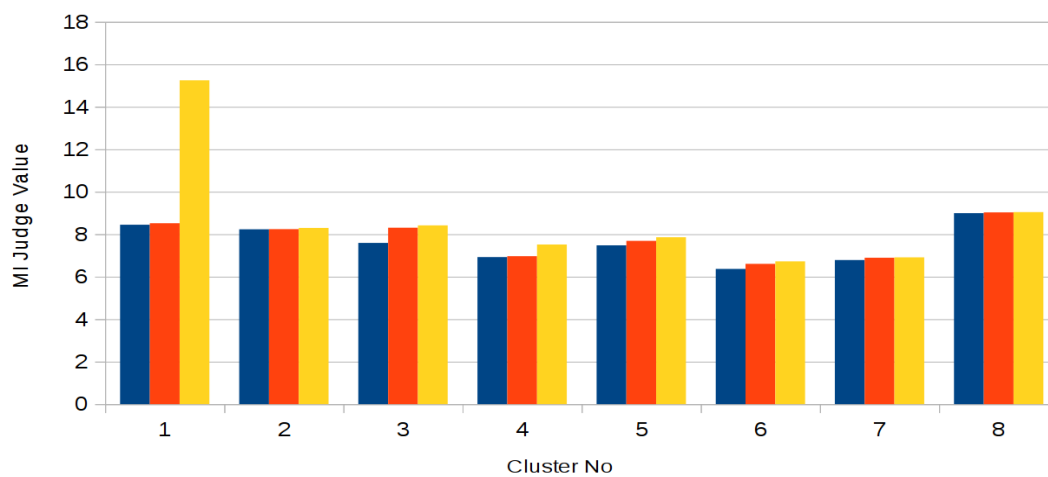


Figure 7.6: MI-score of each cluster runs on Reuters

than 85% and 75% by generating good labels (that match with the actual labels) for most of the clusters of 20-Newsgroups and Reuters datasets, respectively. This justifies the efficiency of the proposed approach.

CHAPTER 8

CONCLUSIONS AND FUTURE DIRECTIONS

Information retrieval is the activity of obtaining information resources relevant to an information need from a collection of information resources. Deep learning approaches which have taken the machine learning community by storm have a high impact on IR. It has been successfully applied for image processing, speech recognition and NLP. The main advantage of deep learning over conventional approaches is that it is completely data driven with stacked layers of neural networks progressively “learning” the data with increasing levels of abstraction, without the necessity of manually hand-coded features. In the context of NLP, word embedding is the starting point of transforming a categorical feature, e.g. a word from a vocabulary, into a continuous representation of a real-valued vector in the Cartesian space of ‘p’ dimensions.

This thesis primarily focused on how deep learning can be useful for text data and some other aspects of IR. For this purpose, ML-ELM is used for text classification and the feature space of ML-ELM has been tested extensively for clustering the text data. The work is carried out by initially analyzing and comparing the performance of SVM, ELM and ML-ELM classifiers to demonstrate the potential of the ELM (for accuracy) and ML-ELM (both accuracy and F-measure) as a highly successful and suitable technique for text classification. It is also observed from the experimental results that ML-ELM can outperform other existing classifiers. Additionally, the results can serve as complementary knowledge to further strengthen the understanding of the essential relationship between SVM, ELM and ML-ELM. Empirical results show that ML-ELM outperformed the traditional classifiers in the majority of cases, and

achieved the best performance overall for both datasets. The experimental results also indicate the high suitability and effectiveness of ELMs in this field as well.

After testing ML-ELM on different benchmark datasets using different traditional feature selection techniques, the interest has been taken to develop two novel feature selection techniques (*KWFS and CCSS*) on which the performance of ML-ELM is tested. In *KWFS*, initially, each cluster is divided into k sub-clusters using k -means algorithm. Then, with the help of Wordnet and cosine-similarity, a reduced feature vector is generated for the entire corpus. For text classification, ELM and ML-ELM classifiers are used. This technique is tested on 20-Newsgroups and DMOZ datasets and the results show the importance of ML-ELM in the field of text classification. In the second feature selection technique (*CCSS*), three important parameters (cohesion, separation and silhouette coefficient) are combined to generate a reduced feature vector. Multilayer ELM as the classifier has been used for classifying the text document and its importance also has been extensively measured. It is evident from all the experimental results that the performance of Multilayer ELM outperforms the other well known classifiers. The encouraging results of two proposed techniques show the stability and effectiveness of Multilayer ELM compared to different progressive classifiers in the domain of text classification. This justifies the stability, efficiency and effectiveness of deep learning.

Combining the ELM and Multilayer ELM feature space with other traditional classifiers will further reinforce the classification results. Similarly, the feature space of ELM and Multilayer ELM can also be used for the clustering process, as the features became more simple and linearly separable by representing them in an extended space. This may outperform the clustering process done in TF-IDF vector space. For this propose, the feature space of ML-ELM is used for semi-supervised clustering using seeded- k Means algorithm. From the experimental results on Classic3 and Reuters datasets, following points are observed:

- results using *ELM feature space* always outperform the results of *without using ELM feature space* (i.e. *TF-IDF vector space*).
- Seeded- k Means performs very well compared to k Means in all aspects regardless of the parameters set (i.e. number of clusters to be formed, % of labels to be considered in seeded- k Means). This demonstrates the superiority of the seeded- k Means clustering technique over the traditional one even if very few labels (only 10 % of the labels are considered) are provided.

- results of purity and entropy are close in ML-ELM compressed or equal dimension space ($L \leq n$) whereas it is better in extended feature space ($L > n$) on both datasets.
- It is also observed that after a certain stage (i.e. threshold point) in extended space, the performance of the clustering process remains unchanged (i.e. further increasing L compared to n has no effect). This may be due to the excessive sparse representation of the features which is not included in the results.

Next, some other important aspects of IR has been discussed by proposing a novel clustering technique based on apriori approach to cluster the text documents. In this technique, a new modified apriori approach has been proposed and it is compared with the traditional apriori algorithm. The proposed modified apriori approach when run on a corpus of web documents produced the same clusters that a traditional apriori approach could. However, experimentally it has been proved that modified apriori approach is more efficient and faster than the traditional apriori approach. This is so because at each step the documents from the corpus are removed, which is no longer required and in turn it reduces the unnecessary comparisons. Hence, it saves a lot of time. Classic4, 20-Newsgroups and Reuters datasets are used for experimental purposes. It is found that FCM gives better clusters compared to VSM and k -means.

In order to label the clusters, a novel cluster labeling technique is developed. Using a feature selection technique, first top- k keywords known as representative keywords are selected from each cluster which are later sent to Wikipedia for getting candidate labels. The candidate labels generated from Wikipedia are evaluated using MI-score. The approach generates the actual labels by just considering the top-3 candidate labels of Wikipedia. For experimental work, 20-Newsgroups and Reuters datasets are considered. The experimental results illustrate the accuracy of the proposed approach which is more than 85% and 75% by generating good labels (that match with the actual labels) for most of the clusters of 20-Newsgroups and Reuters datasets, respectively. This justifies the efficiency of the proposed approach.

Although Multilayer ELM performs well, but still there are some shortcomings which need more attention and can be added to the future work are:

- Determining activation function and the number of nodes for each hidden layer.
- Similarly in ELM, the behavior and a correct number of hidden units is still debatable.
- Also, if one wants to deeply understand the functionality of ELM and Multilayer ELM

by considering them as an approximation of infinite network, then the variance of hidden layer weights is still an open question.

Also, the feature space of ELM and ML-ELM can be used for text classification. Similarly, in cluster labeling, separation of terms into categories of discriminative, short and common terms are explicitly required. Other enhancements like metadata associated with each document apart from the title of the documents can also be used as potential labels. As our approach successfully encompasses all the important milestones needed for automatically providing a label to each cluster of documents, all the steps can be coupled in a single script, which can serve as an important software tool to generate the label for any cluster.

APPENDIX A

DATASETS

1. *20-Newsgroups Dataset* is a standard machine learning dataset and it has 11293 training documents and 7528 test documents classified into 20 classes. All these 20 classes are divided into group of 7 categories such as 'Alt', 'Computer', 'misc', 'rec', 'sci', 'soc' and 'talk'. Table ?? shows the specification details of 20-Newsgroups dataset.
2. *Classic4 Dataset* is a well known benchmark dataset in text mining. It has 4285 training documents and 2839 test documents classified into 4 categories such as *cacm*, *cisi*, *cran*, *med*, having 3204, 1460, 1400 and 1033 documents, respectively. Table ?? shows the specification details of Classic4 dataset.
3. *Reuters-21578 R8 Dataset* is a widely used text mining dataset. It has 5485 training documents and 2189 testing documents classified into 8 classes. Table ?? shows the specification details of Reuters dataset.
4. *WebKB Dataset* is a widespread text mining dataset in which the web pages are collected from four different college websites. It has 2803 training documents and 1396 test documents classified into four classes. Table ?? shows the specification details of WebKB dataset.
5. *DMOZ Dataset* is an open directory project consisting of 14 categories of web pages. The categories are namely "Arts", "Business", "Computers", "Games", "Health", "Home", "News", "Recreation", "Reference", "Regional", , "Science",

Table A.1: Specification of 20-Newsgroups Dataset

Class	Train documents	Test documents	Total number of documents
alt.atheism	480	319	799
comp.graphics	584	389	973
comp.os.ms-windows.misc	572	394	966
comp.sys.ibm.pc.hardware	590	392	982
comp.sys.mac.hardware	578	385	963
comp.windows.x	593	392	985
misc.forsale	585	390	975
rec.autos	594	395	989
rec.motorcycles	598	398	996
rec.sport.baseball	597	397	994
rec.sport.hockey	600	399	999
sci.crypt	595	396	991
sci.electronics	591	393	984
sci.med	594	396	990
sci.space	593	394	987
soc.religion.christian	598	398	996
talk.politics.guns	545	364	909
talk.politics.mideast	564	376	940
talk.politics.misc	465	310	775
talk.religion.misc	377	251	628
Total	11293	7528	18821

Table A.2: Specification of Classic4 Dataset

Class	Train documents	Test documents	Total number of documents
casm	1922	1282	3204
cisi	876	584	1460
cran	840	560	1400
med	620	413	1033
Total	4285	2839	7097

“Shopping”, “Society”, “Sports”. Table ?? shows the specification details of DMOZ dataset.

Table A.3: Specification of Reuters R8 Dataset

Class	Train documents	Test documents	Total number of documents
acq	1596	696	2292
crude	253	121	374
earn	2840	1083	3923
grain	41	10	51
interest	190	81	271
money-fx	206	87	293
ship	108	36	144
trade	251	75	326
Total	5485	2189	7674

Table A.4: Specification of WebKB Dataset

Class	Train documents	Test documents	Total number of documents
project	336	168	504
course	620	310	930
faculty	750	374	1124
student	1097	544	1641
Total	2803	1396	4199

Table A.5: Specification of DMOZ Dataset

Class	Train documents	Test documents	Total number of documents
Arts	1745	1396	3141
Business	4230	3384	7614
Computers	1868	1494	3362
Games	7196	5757	12953
Health	1864	1491	3355
Homes	1756	1405	3161
News	1880	1504	3384
Recreation	1762	1410	3172
Reference	1626	1301	2927
Regional	1634	1307	2941
Science	1737	1390	3127
Shopping	7761	6209	13970
Society	1881	1505	3386
Sports	1894	1515	3409
Total	38834	31068	69902

APPENDIX B

SUPPORT VECTOR MACHINE

In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification analysis. Given a set of training examples, each marked as belonging to one of the two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. Formally, given a set of training data $(\mathbf{x}_i, \mathbf{t}_i)$, $i = 1, \dots, N$, where $\mathbf{x}_i \in \mathbf{R}^d$ and $\mathbf{t}_i \in \{1, -1\}$, due to the nonlinear separability of these training data in the input space, in most cases, one can map the training data \mathbf{x}_i from the input space to a feature space \mathcal{Z} through a nonlinear mapping $\phi : \mathbf{x}_i \rightarrow \phi(\mathbf{x}_i)$. The distance between two different classes in the feature space \mathcal{Z} is $(2/\|\mathbf{w}\|)$. To maximize the separating margin and to minimize the training errors, ξ_i , is equivalent to

$$\text{Minimize : } L_{PSVM} = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i \quad (\text{B.1})$$

$$\text{subject to : } t_i(\mathbf{w} \cdot \phi(\mathbf{x}_i) + b) \geq (1 - \xi_i), \xi_i \geq 0, i = 1, \dots, N \quad (\text{B.2})$$

where, C is a user-specified parameter and provides a trade off between the distance of the separating margin and the training error. Based on the Karush-Kuhn-Tucker

(KKT) theorem [145], to train such SVM is equivalent to solving the following dual optimization problem:

$$L_{DSVM} = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N t_i t_j \alpha_i \alpha_j \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) - \sum_{i=1}^N \alpha_i \quad (\text{B.3})$$

$$\text{subject to : } \sum_{i=1}^N t_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C \quad (\text{B.4})$$

where each Lagrange multiplier α_i corresponds to a training sample (\mathbf{x}_i, t_i) . Vectors \mathbf{x}'_i s for which $t_i(\mathbf{w} \cdot \phi(\mathbf{x}_i) + b) = 1$ are termed support vectors [146]. Kernel functions $K(\mathbf{u}, \mathbf{v}) = \phi(\mathbf{u}) \cdot \phi(\mathbf{v})$ are usually used in the implementation of SVM learning algorithm. In this case, we have

$$\text{Minimize : } L_{DSVM} = 1/2 \sum_{i=1}^N \sum_{j=1}^N t_i t_j \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^N \alpha_i \quad (\text{B.5})$$

$$\text{subject to : } \sum_{i=1}^N t_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C \quad (\text{B.6})$$

The SVM kernel function $K(\mathbf{u}, \mathbf{v})$ needs to satisfy Mercer's condition [146]. The decision function of SVM is

$$f(\mathbf{x}) = \text{sign} \left(\sum_{s=1}^{N_s} \alpha_s t_s K(\mathbf{x}, \mathbf{x}_s) + b \right) \quad (\text{B.7})$$

where N_s is the number of support vectors \mathbf{x}'_s . Although above we have focused on SVMs for two-class classification, however this can easily be extended to a k-class classification by constructing 'k' two-class classifiers. In simple words, the geometrical interpretation of support vector classification (SVC) is that the algorithm searches for the optimal separating surface, i.e. the hyperplane that is, in a sense, equidistant from the two classes.

APPENDIX C

EXTREME LEARNING MACHINE

Extreme Learning Machine(ELM), a classification technique proposed by Huang et al. [24] is a combination of Single Layer feed-forward neural networks(SLFNs) and Support Vector Machine [147]. Neural networks and SVM are two state-of-the-art machine learning techniques. Despite of their superiority, they have the following limitations.

Limitations of Neural Networks:

- rate of learning is very slow compared to their expected rate
- high level training is required
- computationally expensive as it needs more resources
- several training cycles are required to obtain an optimal structure of the network
- the results of training complexity depend on the initialization as the error function is not convex
- they are merely the approximation of a required solution as errors in them are much expected
- black-box in nature
- proneness to over fitting

Limitations of SVM:

- algorithmic complexity is very high
- training time for standard SVM is $O(n^3)$ and space complexity is $O(n^2)$, where n is the training set size
- running time both in training and testing phase is slow
- unstandardized probabilities of class membership
- handling multi-class classification is computationally expensive
- difficult to interpret the parameters for a solved model
- unable to handle non linear separable input data
- sensitive to noisy data
- lack of transparency of results

Initially ELM was SLFNs and later it extended to the generalized SLFNs, [66] where the hidden layer does not require tuning, nor has the need to be neuron alike, [137]. ELM has potential to become a good classifier over the other traditional classifiers is due to the following reasons:

- ◇ adjustment of input weights and hidden layer biases which consumes more time are not required here as they are assigned randomly
- ◇ neither hidden layer require to be tuning nor to be neuron alike
- ◇ easy to implement and very fast learning speed
- ◇ can be able to handle a large volume of data
- ◇ no back propagation and no over fitting
- ◇ gives very good performance with less human intervention
- ◇ avoids local minimization

- ◇ parallelization of computation
- ◇ produces one optimal solution with almost no errors

ELM as a Model:

For N arbitrary distinct samples $(\mathbf{x}_i, \mathbf{y}_i)$, where $\mathbf{x}_i = [x_{i1}, \dots, x_{in}]^T \in \mathbf{R}^n$ and $\mathbf{y}_i = [y_{i1}, \dots, y_{im}]^T \in \mathbf{R}^m$, such that $(\mathbf{x}_i, \mathbf{y}_i) \in \mathbf{R}^n \times \mathbf{R}^m$ where $(i = 1, 2, \dots, N)$, along with L hidden nodes, and an activation function $g(x)$. The output function of ELM for a given input \mathbf{x} is:

$$\mathbf{g}_L(\mathbf{x}_j) = \sum_{i=1}^L \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + \mathbf{b}_i) = \mathbf{y}_j, j = 1, \dots, N \quad (\text{C.1})$$

where,

- $(\mathbf{w}_i, \mathbf{b}_i), i = 1, \dots, L$ are the randomly generated hidden node parameters such that $\mathbf{w}_i = [w_{i1}, w_{i2} \dots w_{in}]^T$ is the weight vector connecting the i^{th} hidden node to n input nodes and \mathbf{b}_i is the i^{th} hidden node bias.
- $\beta = [\beta_1, \dots, \beta_L]^T$ is the weight vector between the i^{th} hidden node and the output nodes.
- $\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x}), \dots, g_L(\mathbf{x})]$ is the output (row) vector of the hidden layer with respect to the input \mathbf{x} which maps the n -dimensional input space to L -dimensional feature space, H (called ELM feature space, also known as hidden layer output matrix).

Compact format of equation B.1 can be written as follows:

$$\mathbf{H}\beta = \mathbf{Y} \quad (\text{C.2})$$

where

$$\mathbf{H} = \begin{bmatrix} g(w_1 \cdot x_1 + b_1) & \dots & g(w_L \cdot x_1 + b_L) \\ g(w_1 \cdot x_2 + b_1) & \dots & g(w_L \cdot x_2 + b_L) \\ \cdot & \dots & \cdot \\ \cdot & \dots & \cdot \\ \cdot & \dots & \cdot \\ g(w_1 \cdot x_N + b_1) & \dots & g(w_L \cdot x_N + b_L) \end{bmatrix}_{N \times L} \quad (\text{C.3})$$

$$\beta = \begin{bmatrix} \beta_{11} & \dots & \beta_{1m} \\ \beta_{21} & \dots & \beta_{2m} \\ \cdot & & \\ \cdot & & \\ \cdot & & \\ \beta_{L1} & \dots & \beta_{Lm} \end{bmatrix}_{L \times m} \quad \mathbf{Y} = \begin{bmatrix} y_{11} & \dots & y_{1m} \\ y_{21} & \dots & y_{2m} \\ \cdot & & \\ \cdot & & \\ \cdot & & \\ y_{N1} & \dots & y_{Nm} \end{bmatrix}_{N \times m} \quad (\text{C.4})$$

The i^{th} column of \mathbf{H} is the i^{th} hidden node output w.r.t inputs x_1, x_2, \dots, x_N . According to Huang [148], as long as the number of hidden nodes is large enough, the parameters of the network do not all need to adjust. ELM tends to reach not only the smallest norm of output weights but also the smallest training error(similar concept in SVM) and can be represented as follows:

$$\text{minimize} : \|\mathbf{H}\beta - \mathbf{Y}\|^2 \text{ and } \|\beta\| \quad (\text{C.5})$$

The minimal norm least square solution of the above linear system is given by:

$$\beta = \mathbf{H}^+ \mathbf{Y} \quad (\text{C.6})$$

where \mathbf{H}^+ is the Moore-Penrose [149] generalized inverse of matrix \mathbf{H} .

Figure B.1 shows the system diagram of ELM.

Conclusion of ELM Algorithm:

In conclusion, using ELM to obtain the output weights β can be divided into the three following steps:

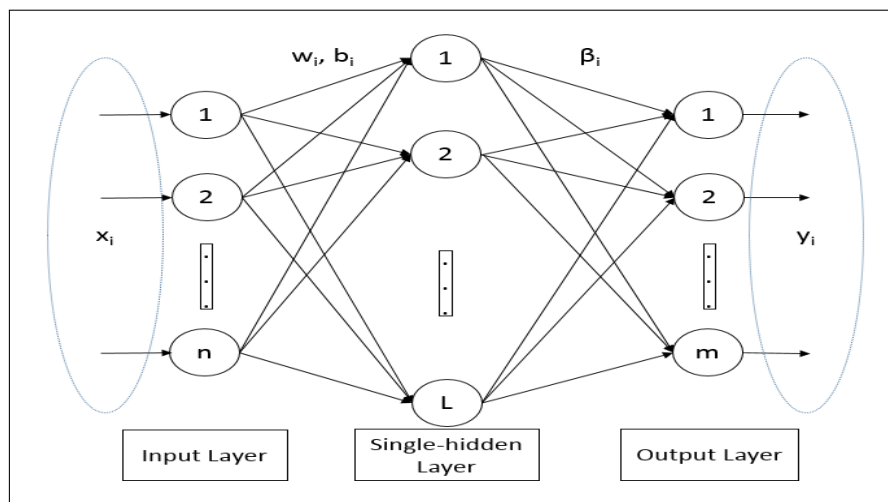


Figure C.1: Architecture of ELM

1. Randomly select numerical values between 0 and 1 to set input weights w_i and biases of the hidden layer b_i , ($i = 1, \dots, L$)
2. Calculate the hidden layer output matrix H
3. Calculate the output weight vector β as follows:

$$\beta = H^+Y$$

APPENDIX D

MULTILAYER EXTREME LEARNING MACHINE

Multi-layer ELM (ML-ELM) is a machine learning technique of artificial neural networks which uses deep learning(a multi-layer perceptron) and ELM extensively. Deep learning was first proposed by Hinton et al. [150] who in their work used deep structure of multi-layer auto encoder and establish a multilayer neural network on the unsupervised data. In their proposed method, they first used an unsupervised training to obtain the parameters in each layer. Next, the network is fine tuned by supervised learning. Hinton [151] who proposed the deep belief network can outperforms the traditional multi-layer neural network, SVMs, SLFNs but has slow learning speed. Kasun et al. [152] first proposed the multi-layer form of ELM known as ML-ELM in which unsupervised learning is performed from layer to layer and it does not require any fine tuning and hence does not need to spend a long time on the network training. It has a better or comparable performance in comparison with any state-of-the-art deep networks. Cambria et al. [129] proposed a multi-layer implementation of ELM that performs layer by layer unsupervised learning in a manner that resembles deep networks and has a better or comparable performance to deep networks for the task of image recognition. We have applied a similar technique to ELM for the purpose of text classification and have found improved performance over conventional classifiers. Figure C.1 shows how ML-ELM combines both ELM-AE and ELM together.

ELM-AE

Autoencoder is an unsupervised neural network. The outputs and inputs of the autoencoder are same. Like ELM, ELM-AE has ' n ' input layer nodes, single hidden layer of ' L ' nodes and ' n ' output layer nodes. In spite of many resemblances between these two, there are two major differences exist between them which are as follows:

- i. ELM is a supervised neural network and the output of ELM is a class label while ELM-AE is a unsupervised one and it's output is same as input.
- ii. Input weights and biases of the hidden layer are random in case of ELM, but they are orthogonal in ELM-AE.

Depending on the number of hidden layer nodes, the ELM-AE can be divided into the following three categories.

i) Compressed representation($n > L$):

In compressed representation, features of training dataset needs to be represented from a higher(or sparse) dimensional input signal space to a lower(or compressed) dimensional feature space.

ii) Equal dimension representation($n = L$):

In this representation of features, the dimension of input signal space and feature space need to be equal.

iii) Sparse representation($n < L$):

It is just the reverse of compressed representation where features of training dataset needs to be represented from a lower dimensional input signal space to a higher(or sparse) dimensional feature space.

The performance of machine learning algorithms very much depend on how well a dataset features are engineered, since it determines how well the most important aspects of the data are captured and represented. Similar to the way deep

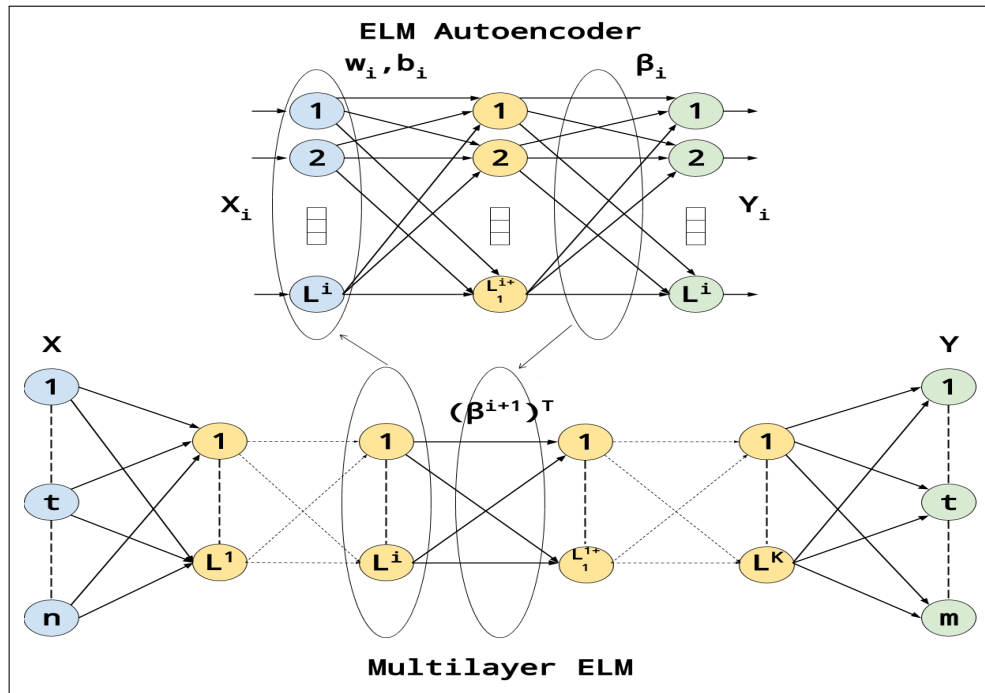


Figure D.1: Multi-Layer ELM and ELM-Autoencoder

networks which are based on restricted Boltzmann machines and auto encoders, use the features engineered by them to train the multi-layer network, the ELM multi-layer implementation also stacks on top of ELM auto encoders which represent the features as singular values, based on which the layer by layer unsupervised learning is carried out. The ML-ELM algorithm achieves similar performance as compared to deep networks but is significantly faster as it does not have to undergo iterative tuning. The ELM-AE, just like regular ELMs is a good universal approximator, and its main aim is to represent the input features in a meaningful way by transforming the input data to a N dimensional feature space of the hidden nodes. Thus the resulting representation is either a compressed, equal or sparse representation, depending on whether the input features are mapped to a lower, equal or higher dimensional feature space than their own. As we are aware that in an ELM network for N training examples (x_j, y_j) and L hidden nodes we have :

$$g_L(x_j) = \sum_{i=1}^L \beta_i g_i(x_j, w_i, b_i) = y_j, \quad j = 1, \dots, N \quad (\text{D.1})$$

where (w_i, b_i) , $i = 1, \dots, L$ are the randomly generated hidden node parameters and

H is the hidden layer output matrix. The output weights β which map the hidden node feature space to the that of the output nodes, can be computed using Equation C.4,C.5 or E.1 depending on the number of training samples greater than, equal to, or less than the hidden layer nodes and this is different than the computation of β in case of ELM. The ELM-AE works in a similar manner like a regular ELM as shown above, except for a few modifications in order to perform unsupervised learning:

- 1) For each of the hidden layers, the output data is taken to be the input data itself. Thus, for each input data \mathbf{x} :

$$\mathbf{y} = \mathbf{x} \quad (\text{D.2})$$

- 2) The random hidden node weights and biases are taken to be orthogonal, which tends to improve the ELMs generalization performance. Thus the input data is mapped to the higher, lower or equal dimensional space of hidden nodes through the orthogonal hidden weights $\mathbf{w} = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_L]$ and the orthogonal hidden biases $\mathbf{b} = [\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_L]$ using the following equations:

$$\begin{aligned} h &= g(\mathbf{w} \cdot \mathbf{x} + \mathbf{b}) \\ \mathbf{w}^T \mathbf{w} &= \mathbf{I}, \mathbf{b}^T \mathbf{b} = 1 \end{aligned} \quad (\text{D.3})$$

- 3) While calculating the output weights β , since in the case of ELM-AE, the output data is equal to the input data ($\mathbf{y} = \mathbf{x}$), the output weight β is responsible for learning the transformation from the hidden layer feature space to input data, and is given by:

- i. if $n > L$ (i.e number of input layer nodes is more than the hidden layer nodes) then

$$\beta = \left(\frac{\mathbf{I}}{\lambda} + \mathbf{H}^T \mathbf{H} \right)^{-1} \mathbf{H}^T \mathbf{X} \quad (\text{D.4})$$

- ii. if $n = L$ (i.e number of input layer nodes is equal to the hidden layer nodes) then

$$\beta = \mathbf{H}^{-1} \mathbf{X} \quad (\text{D.5})$$

- iii. if $n < L$ (i.e number of input layer nodes is less than the hidden layer

nodes) then

$$\beta = \mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{X} \quad (\text{D.6})$$

ML-ELM makes use of ELM-AE to train the parameters in each layer. In other words, the hidden layer weights of ML-ELM are initialized by ELM-AE from layer to layer using unsupervised learning, and ML-ELM hidden layer activation functions can be either linear or non-linear piecewise. All output weights are determined analytically. The output of the i^{th} hidden layer of ML-ELM can be obtained from the output of $(i-1)^{th}$ hidden layer and the output weight of β^i of the i^{th} hidden layer. The output weight of β^i of the i^{th} hidden layer is obtained layer wise from the ELM-AE, and its transpose. ML-ELM with ' L ' hidden nodes can be represented as

$$\mathbf{H}^n = g((\beta^n)^T \mathbf{H}^{n-1}) \quad (\text{D.7})$$

For $n = 0$, the input layer \mathbf{X} can be considered as the 0^{th} hidden layer. The transformations of the data from the feature space of one layer to the next, and so on are carried out as shown in Equation E.2, until reach the last layer before the output layer \mathbf{y} . The final output matrix \mathbf{y} can be obtained by computing the results between the last hidden layer and the output layer using the regularized least squares technique [130].

APPENDIX E

TRADITIONAL FEATURE SELECTION TECHNIQUES

i. Chi-Square

Chi-Square (χ) [9] with T number of training samples which compares the expected data with the observed data based on a specific hypothesis and can be defined as follows:

$$\chi^2(t, c) = \frac{T \times (ps - qr)}{(p + r) \times (q + s) \times (p + q) \times (r + s)} \quad (\text{E.1})$$

where, t is a term, c is a class, p is the no. of true positive cases, q is the no. of false negative cases, r is the no. of false positive and s is the no. of true negative cases, respectively.

ii. Bi-normal separation

Bi-normal separation (BNS) can be considered a statistical modification of the accuracy selection method. According to Forman [17], BNS of a term t in a class c_i can be defined as:

$$BNS(t, c_i) = \left| \phi^{-1} \left(\frac{n_{it}}{n_i} \right) - \phi^{-1} \left(\frac{n_{\bar{i}t}}{n_{\bar{i}}} \right) \right| \quad (\text{E.2})$$

where, ϕ and ϕ^{-1} are the standard normal distribution and its corresponding inverse, respectively,

n_i : no. of documents $\in c_i$,

$n_{\bar{i}}$: no. of documents $\notin c_i$,

n_{it} : no. of documents $\in c_i$ having the term t and

$n_{\bar{it}}$: no. of documents $\notin c_i$ having the term t .

iii. Information Gain

Introduced by Yang and Pedersen [153], Information Gain (IG) is one of the most popular feature selection technique very frequently used in the domain of machine learning. IG of a term t measures how much entropy decreases when the term is given (i.e. present in the class c) rather than not given (i.e. not present in the class c). It can be defined by the following equation:

$$IG(t) = \sum_{c \in c_i, !c_i} \sum_{t \in t_i, !t_i} P(t, c) \log \frac{P(t, c)}{P(t)P(c)} \quad (E.3)$$

iv. Gini Index

Gini index (GI) which is as an improved version of an attribute selection algorithm, is a global feature selection technique used to classify the text data [58]. The following equation describes to find the GI of a term t in the class C_i .

$$GI(t) = \sum_{i=1}^m p(t|C_i)^2 p(C_i|t)^2 \quad (E.4)$$

where m is the number of classes.

APPENDIX F

FUZZY C-MEANS

Fuzzy C-Means (FCM) algorithm [142] tries to distribute a finite collection of n objects into c clusters. It returns a list of c cluster centroids along with a matrix which shows the degree of membership of each object to different clusters. It aims to minimize the following function:

$$T_m = \sum_{i=1}^n \sum_{j=1}^c v_{ij}^m \|d_{ij}\|^2$$

where, distance $d_{ij} = \mathbf{x}_i - \mathbf{c}_j$, m (generally set to 2) is the fuzzy coefficient, \mathbf{c}_j is the centroid(vector) of cluster j , \mathbf{x}_i is the i^{th} object and $v_{ij} \in \text{range } [0, 1]$ is the degree of membership of \mathbf{x}_i with respect to \mathbf{c}_j subject to the following conditions:

$$\sum_{j=1}^c v_{ji} = 1, \quad i = 1, 2, \dots, n \text{ and } 0 < \sum_{i=1}^n v_{ij} < n, \quad j = 1, 2, \dots, c$$

One can iteratively find the values of \mathbf{c}_j and v_{ij} updated with each iteration by using the following equations.

$$\mathbf{c}_j = \frac{\sum_{i=1}^n v_{ij}^m \mathbf{x}_i}{\sum_{i=1}^n v_{ij}^m} \quad (\text{F.1})$$

$$v_{ij} = \frac{1}{\sum_{k=1}^c \left(\frac{\|d_{ij}\|}{\|d_{ik}\|} \right)^{\frac{2}{m-1}}} \quad (\text{F.2})$$

APPENDIX G

CLUSTER EVALUATION

From the above discussion, it is understood that since there is a number of different types of clusters and each clustering algorithm defines its own type of cluster, hence each clustering might require a different evaluation measure. For instance, centroid-based clustering like k -means can be evaluated using *sum square error (SSE)* whereas for density-based clustering, which need not be globular, *SSE* won't work to measure the tendency. A list of some important issues for cluster evaluation is given below:

- Determining the clustering tendency of a set of data and deciding the correct number of clusters.
- Without referring to external information, compute how well the results of a cluster analysis fit the data.
- Comparing the cluster evaluation results with some external information such as class labels and comparing two sets of clusters to determine which is better.

Based on the above issues, cluster evaluation technique are classified into the following two categories:

- 1) *Unsupervised techniques* measure the cluster tendency without the help of any external information. It means they use only the information available in the dataset. They are divided into two classes:

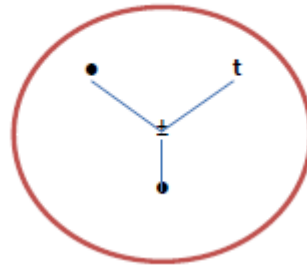


Figure G.1: Cohesion

- *Cluster Cohesion* determines how closely or tightly the objects are inside a cluster. It checks the compactness or tightness of a cluster. Figure F.1 shows the cluster cohesion. *SSE* method is used to measure the cohesion of a cluster C_i . It is computed as the sum of the Euclidean distance of all the objects from the centroid of C_i .

$$\text{Total SSE} = \sum_{i=1}^k \sum_{y \in C_i} \text{distance}(c_i, y)^2 \quad (\text{G.1})$$

where C_i is the i^{th} cluster and c_i is the centroid of C_i and k is the number of clusters.

- *Cluster Separation* determine how distinct or well separated a cluster is from another cluster. It checks the isolation of a cluster. Figure F.2 shows the separation between two clusters. Between groups sum of squares (*SSB*)

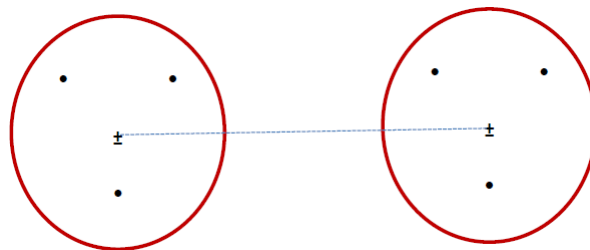


Figure G.2: Separation between the centroid of two clusters

method is used to measure the separation of a cluster. It is computed as sum of the squared distance of a cluster centroid, c_i , to overall mean, c , of all

the data points, n_i .

$$Total\ SSB = \sum_{i=1}^k n_i * distance(c_i, c)^2 \quad (G.2)$$

- *Silhouette Coefficient* [154] is the most popular method generally used to evaluate individual clusters and objects by combining both cohesion and separation. It measures how much an object t is similar to its own cluster (i.e. cohesion) compared to other clusters (separation) and it can vary between -1 and 1. If the silhouette coefficient of the object t is high then it highly matches to its own cluster and poorly matches to other clusters. If most of the objects in a cluster have high coefficient values then the cluster is well configured. It can be represented as follows:

$$silhouette(t) = \frac{s(t) - c(t)}{\max(c(t), s(t))} \quad (G.3)$$

where $c(t)$ and $s(t)$ are the cohesion and separation score of the object t , respectively. Silhouette coefficient needs to be positive (i.e. $c(t) < s(t)$), and $c(t)$ to be close to 0 (i.e. the object t should be very close to its centroid) as far as possible, since the coefficient receives its maximum value of 1 when $c(t)$ is 0.

2) *Supervised techniques* measure the clustering tendency by comparing the results of the clustering with the supplied external information. For instance, measuring the degree of correspondence between the generated cluster labels and the supplied class labels. There are two types of measures used for supervised techniques:

- *classification-oriented* measures evaluate the extent to which a cluster contains objects of a single class. Below are the examples which are based on classification-oriented measures.

Purity is a simple technique which measures the extent to which a cluster

contains documents of a single class.

$$purity = \frac{1}{N} \sum_{i=1}^k \max_j p_{ij} \quad (G.4)$$

where, $p_{ij} = \frac{n_{ij}}{n_i}$. Here, N is the total documents to be clustered and k is the number of clusters. n_{ij} is the number of documents of class j in cluster i and n_i is the number of documents of cluster i .

Entropy is another supervised measures shows the degree to which each cluster of documents belongs to a single class.

$$entropy = \sum_{i=1}^k \frac{n_i}{n} e_i \quad (G.5)$$

where $e_i = - \sum_{j=1}^C p_{ij} \log_2 p_{ij}$. C is the number of classes in the corpus. *Precision* is the fraction of a cluster i that consists of objects of a specified class j .

$$precision(i, j) = p_{ij} \quad (G.6)$$

Recall is the extent to which a cluster i contains all objects of a specified class j .

$$recall(i, j) = \frac{n_{ij}}{n_j} \quad (G.7)$$

- *Similarity-oriented* measures the extent to which two objects that are in the same class are in the same cluster and vice versa. Below are two such examples.

$$Rand\ statistic = \frac{f_{00} + f_{11}}{f_{00} + f_{11} + f_{10} + f_{01}} \quad (G.8)$$

$$Jaccard\ coefficient = \frac{f_{11}}{f_{11} + f_{10} + f_{01}} \quad (G.9)$$

where, f_{00} : number of pair of points having a different class and different cluster, f_{01} : number of pair of points having a different class and same cluster, f_{10} : number of pair of points having a same class and different cluster, f_{11} : number of pair of points having a same class and same cluster

BIBLIOGRAPHY

- [1] “Information retrieval on-line. f. w. lancaster and e. g. fayen,” *Journal of the American Society for Information Science*, vol. 25, no. 5, pp. 336–337, 1974.
- [2] Y. Ogawa, T. Morita, and K. Kobayashi, “A fuzzy document retrieval system using the keyword connection matrix and a learning method,” *Fuzzy sets and systems*, vol. 39, no. 2, pp. 163–179, 1991.
- [3] G. Salton, A. Wong, and C.-S. Yang, “A vector space model for automatic indexing,” *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.
- [4] S. M. Wong, W. Ziarko, and P. C. Wong, “Generalized vector spaces model in information retrieval,” in *Proceedings of the 8th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 18–25, ACM, 1985.
- [5] J. Becker and D. Kuropka, “Topic-based vector space model,” in *Proceedings of the 6th international conference on business information systems*, pp. 7–12, 2003.
- [6] G. Salton, E. A. Fox, and H. Wu, “Extended boolean information retrieval,” *Communications of the ACM*, vol. 26, no. 11, pp. 1022–1036, 1983.
- [7] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, “Indexing by latent semantic analysis,” *Journal of the American society for information science*, vol. 41, no. 6, pp. 391–407, 1990.
- [8] C. T. Yu and G. Salton, “Precision weighting an effective automatic indexing method,” *Journal of the ACM (JACM)*, vol. 23, no. 1, pp. 76–88, 1976.

- [9] C. Manning and P. Raghavan, "Introduction to information retrieval," vol. 1, no. 1, 2008.
- [10] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.
- [11] I. S. Bajwa, M. Naweed, M. N. Asif, and S. I. Hyder, "Feature based image classification by using principal component analysis," *ICGST Int. J. Graph. Vis. Image Process. GVIP*, vol. 9, pp. 11–17, 2009.
- [12] E. K. Tang, P. N. Suganthan, X. Yao, and A. K. Qin, "Linear dimensionality reduction using relevance weighted lda," *Pattern recognition*, vol. 38, no. 4, pp. 485–493, 2005.
- [13] K. Kira and L. A. Rendell, "The feature selection problem: Traditional methods and a new algorithm," in *AAAI*, vol. 2, pp. 129–134, 1992.
- [14] H. Ogura, H. Amano, and M. Kondo, "Distinctive characteristics of a metric using deviations from poisson for feature selection," *Expert Systems with Applications*, vol. 37, no. 3, pp. 2273–2281, 2010.
- [15] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial intelligence*, vol. 97, no. 1, pp. 273–324, 1997.
- [16] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *The Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.
- [17] G. Forman, "An extensive empirical study of feature selection metrics for text classification," *The Journal of machine learning research*, vol. 3, pp. 1289–1305, 2003.
- [18] W. J. Wilbur and K. Sirotkin, "The automatic identification of stop words," *Journal of information science*, vol. 18, no. 1, pp. 45–55, 1992.
- [19] C. Van Rijsbergen, D. J. Harper, and M. F. Porter, "The selection of good search terms," *Information Processing & Management*, vol. 17, no. 2, pp. 77–91, 1981.
- [20] D. Mladenic and M. Grobelnik, "Feature selection for unbalanced class distribution and naive bayes," in *ICML*, vol. 99, pp. 258–267, 1999.

- [21] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 7, pp. 711–720, 1997.
- [22] H. T. Ng, W. B. Goh, and K. L. Low, "Feature selection, perceptron learning, and a usability case study for text categorization," in *ACM SIGIR Forum*, vol. 31, pp. 67–73, ACM, 1997.
- [23] L. Galavotti, F. Sebastiani, and M. Simi, "Experiments on the use of feature selection and negative evidence in automated text categorization," in *International Conference on Theory and Practice of Digital Libraries*, pp. 59–68, Springer, 2000.
- [24] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006.
- [25] S. Ding, N. Zhang, X. Xu, L. Guo, and J. Zhang, "Deep extreme learning machine and its application in eeg classification," *Mathematical Problems in Engineering*, vol. 2015, pp. 1–11, 2015.
- [26] L. L. C. Kasun, H. Zhou, G.-B. Huang, and C. M. Vong, "Representational learning with extreme learning machine for big data," *IEEE Intelligent Systems*, vol. 28, no. 6, pp. 31–34, 2013.
- [27] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.
- [28] K. W. Church and P. Hanks, "Word association norms, mutual information, and lexicography," *Computational linguistics*, vol. 16, no. 1, pp. 22–29, 1990.
- [29] F. Role and M. Nadif, "Beyond cluster labeling: Semantic interpretation of clusters contents using a graph representation," *Knowledge-Based Systems*, vol. 56, pp. 141–155, 2014.
- [30] D. Carmel, H. Roitman, and N. Zwerdling, "Enhancing cluster labeling using wikipedia," in *Proceedings of the 32Nd International ACM SIGIR Conference*

on Research and Development in Information Retrieval, SIGIR '09, (New York, NY, USA), pp. 139–146, ACM, 2009.

- [31] K. Ganesan, C. Zhai, and J. Han, “Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions,” in *Proceedings of the 23rd international conference on computational linguistics*, pp. 340–348, Association for Computational Linguistics, 2010.
- [32] A. Z. Broder, “Identifying and filtering near-duplicate documents,” in *Combinatorial pattern matching*, pp. 1–10, Springer, 2000.
- [33] Y. Bernstein and J. Zobel, “Accurate discovery of co-derivative documents via duplicate text detection,” *Information Systems*, vol. 31, no. 7, pp. 595–609, 2006.
- [34] M. S. Charikar, “Similarity estimation techniques from rounding algorithms,” in *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, pp. 380–388, ACM, 2002.
- [35] G.-G. Geng, C.-H. Wang, Q.-D. Li, L. Xu, and X.-B. Jin, “Boosting the performance of web spam detection with ensemble under-sampling classification,” in *Fuzzy Systems and Knowledge Discovery, 2007. FSKD 2007. Fourth International Conference on*, vol. 4, pp. 583–587, IEEE, 2007.
- [36] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly, “Detecting spam web pages through content analysis,” in *Proceedings of the 15th international conference on World Wide Web*, pp. 83–92, ACM, 2006.
- [37] B. Wu and B. D. Davison, “Identifying link farm spam pages,” in *Special interest tracks and posters of the 14th international conference on World Wide Web*, pp. 820–829, ACM, 2005.
- [38] J.-L. Lin, “Detection of cloaked web spam by using tag-based methods,” *Expert Systems with Applications*, vol. 36, no. 4, pp. 7493–7499, 2009.

- [39] R. H. Pinheiro, G. D. Cavalcanti, R. F. Correa, and T. I. Ren, “A global-ranking local feature selection method for text categorization,” *Expert Systems with Applications*, vol. 39, no. 17, pp. 12851–12857, 2012.
- [40] A. K. Uysal, “An improved global feature selection scheme for text classification,” *Expert Systems with Applications*, vol. 43, pp. 82–92, 2016.
- [41] A. K. Uysal and S. Gunal, “A novel probabilistic feature selection method for text classification,” *Knowledge-Based Systems*, vol. 36, pp. 226–235, 2012.
- [42] S. Wang, J. Tang, and H. Liu, “Embedded unsupervised feature selection,” in *AAAI*, pp. 470–476, 2015.
- [43] M. Robnik-Šikonja and I. Kononenko, “Theoretical and empirical analysis of relieff and rrelieff,” *Machine learning*, vol. 53, no. 1-2, pp. 23–69, 2003.
- [44] J. Weston, A. Elisseeff, B. Schölkopf, and M. Tipping, “Use of the zero norm with linear models and kernel methods,” *The Journal of Machine Learning Research*, vol. 3, pp. 1439–1461, 2003.
- [45] L. Song, A. Smola, A. Gretton, K. M. Borgwardt, and J. Bedo, “Supervised feature selection via dependence estimation,” in *Proceedings of the 24th international conference on Machine learning*, pp. 823–830, ACM, 2007.
- [46] T. Li, C. Zhang, and M. Ogihara, “A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression,” *Bioinformatics*, vol. 20, no. 15, pp. 2429–2437, 2004.
- [47] M. Dash and H. Liu, “Feature selection for clustering,” in *Knowledge Discovery and Data Mining. Current Issues and New Applications*, pp. 110–121, Springer, 2000.
- [48] J. G. Dy and C. E. Brodley, “Feature selection for unsupervised learning,” *The Journal of Machine Learning Research*, vol. 5, pp. 845–889, 2004.
- [49] X. He, D. Cai, and P. Niyogi, “Laplacian score for feature selection,” in *Advances in neural information processing systems*, pp. 507–514, 2005.

- [50] Z. Zhao and H. Liu, "Spectral feature selection for supervised and unsupervised learning," in *Proceedings of the 24th international conference on Machine learning*, pp. 1151–1157, ACM, 2007.
- [51] L. Liu, J. Kang, J. Yu, and Z. Wang, "A comparative study on unsupervised feature selection methods for text clustering," in *Natural Language Processing and Knowledge Engineering, 2005. IEEE NLP-KE'05. Proceedings of 2005 IEEE International Conference on*, pp. 597–601, IEEE, 2005.
- [52] J. Lee and D.-W. Kim, "Mutual information-based multi-label feature selection using interaction information," *Expert Systems with Applications*, vol. 42, no. 4, pp. 2013–2025, 2015.
- [53] W. Chen, J. Yan, B. Zhang, Z. Chen, and Q. Yang, "Document transformation for multi-label feature selection in text categorization," in *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pp. 451–456, IEEE, 2007.
- [54] C.-M. Chen, H.-M. Lee, and Y.-J. Chang, "Two novel feature selection approaches for web page classification," *Expert systems with Applications*, vol. 36, no. 1, pp. 260–272, 2009.
- [55] J. Chen, H. Huang, S. Tian, and Y. Qu, "Feature selection for text classification with naïve bayes," *Expert Systems with Applications*, vol. 36, no. 3, pp. 5432–5435, 2009.
- [56] M. Bennisar, Y. Hicks, and R. Setchi, "Feature selection using joint mutual information maximisation," *Expert Systems with Applications*, vol. 42, no. 22, pp. 8520–8532, 2015.
- [57] N. Azam and J. Yao, "Comparison of term frequency and document frequency based feature selection metrics in text categorization," *Expert Systems with Applications*, vol. 39, no. 5, pp. 4760–4768, 2012.
- [58] W. Shang, H. Huang, H. Zhu, Y. Lin, Y. Qu, and Z. Wang, "A novel feature selection algorithm for text categorization," *Expert Systems with Applications*, vol. 33, no. 1, pp. 1–5, 2007.

- [59] J. Meng, H. Lin, and Y. Yu, "A two-stage feature selection method for text categorization," *Computers & Mathematics with Applications*, vol. 62, no. 7, pp. 2793–2800, 2011.
- [60] M. Thangamani and P. Thangaraj, "Integrated clustering and feature selection scheme for text documents," vol. 6, no. 5, pp. 536–541, 2010.
- [61] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 8, pp. 1226–1238, 2005.
- [62] S. Li, R. Xia, C. Zong, and C.-R. Huang, "A framework of feature selection methods for text categorization," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pp. 692–700, Association for Computational Linguistics, 2009.
- [63] M. H. Aghdam, N. Ghasem-Aghaee, and M. E. Basiri, "Text feature selection using ant colony optimization," *Expert systems with applications*, vol. 36, no. 3, pp. 6843–6853, 2009.
- [64] J. Yang, Y. Liu, Z. Liu, X. Zhu, and X. Zhang, "A new feature selection algorithm based on binomial hypothesis testing for spam filtering," *Knowledge-Based Systems*, vol. 24, no. 6, pp. 904–914, 2011.
- [65] E. Gabrilovich and S. Markovitch, "Feature generation for text categorization using world knowledge," in *IJCAI*, vol. 5, pp. 1048–1053, 2005.
- [66] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, no. 2, pp. 513–529, 2012.
- [67] Z. Bai, G.-B. Huang, D. Wang, H. Wang, and M. B. Westover, "Sparse extreme learning machine for classification," *Cybernetics, IEEE Transactions on*, vol. 44, no. 10, pp. 1858–1870, 2014.

- [68] S. Balasundaram, D. Gupta, *et al.*, “1-norm extreme learning machine for regression and multiclass classification using newton method,” *Neurocomputing*, vol. 128, pp. 4–14, 2014.
- [69] S. Ding, X. Xu, and R. Nie, “Extreme learning machine and its applications,” *Neural Computing and Applications*, vol. 25, no. 3-4, pp. 549–556, 2014.
- [70] B. Mirza, S. Kok, and F. Dong, “Multi-layer online sequential extreme learning machine for image classification,” in *Proceedings of ELM-2015 Volume 1*, pp. 39–49, Springer, 2016.
- [71] Y. Yang and Q. J. Wu, “Multilayer extreme learning machine with subnetwork nodes for representation learning,” *IEEE transactions on cybernetics*, vol. 46, no. 11, pp. 2570–2583, 2016.
- [72] J. Tang, C. Deng, G.-B. Huang, and J. Hou, “A fast learning algorithm for multi-layer extreme learning machine,” in *Image Processing (ICIP), 2014 IEEE International Conference on*, pp. 175–178, IEEE, 2014.
- [73] R.-C. Chen and C.-H. Hsieh, “Web page classification based on a support vector machine using a weighted vote schema,” *Expert Systems with Applications*, vol. 31, no. 2, pp. 427–435, 2006.
- [74] C. H. Wan, L. H. Lee, R. Rajkumar, and D. Isa, “A hybrid text classification approach with low dependency on parameter by integrating k-nearest neighbor and support vector machine,” *Expert Systems with Applications*, vol. 39, no. 15, pp. 11880–11888, 2012.
- [75] P. Lingras and C. Butz, “Rough set based 1-v-1 and 1-vr approaches to support vector machine multi-classification,” *Information Sciences*, vol. 177, no. 18, pp. 3782–3798, 2007.
- [76] P. Thamrongrat, L. Preechaveerakul, and W. Wettayaprasit, “A novel voting algorithm of multi-class svm for web page classification,” in *Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on*, pp. 327–331, IEEE, 2009.

- [77] J. C. Gomez and M.-F. Moens, “Hierarchical classification of web documents by stratified discriminant analysis,” in *Multidisciplinary Information Retrieval*, pp. 94–108, Springer, 2012.
- [78] B. Rujiang, W. Xiaoyue, and H. Zewen, “A novel web pages classification model based on integrated ontology,” in *Software Engineering, Business Continuity, and Education*, pp. 1–10, Springer, 2011.
- [79] L. Li, D. Song, and L. Liao, “Vertical classification of web pages for structured data extraction,” in *Information Retrieval Technology*, pp. 486–495, Springer, 2012.
- [80] J. Xin, Z. Wang, C. Chen, L. Ding, G. Wang, and Y. Zhao, “Elm: distributed extreme learning machine with mapreduce,” *World Wide Web*, pp. 1–16, 2013.
- [81] M. Klassen and N. Paturi, “Web document classification by keywords using random forests,” in *Networked Digital Technologies*, pp. 256–261, Springer, 2010.
- [82] W. Zhang, X. Tang, and T. Yoshida, “Tesc: An approach to text classification using semi-supervised clustering,” *Knowledge-Based Systems*, vol. 75, pp. 152–160, 2015.
- [83] I. Hernández, C. R. Rivero, D. Ruiz, and R. Corchuelo, “Cala: an unsupervised url-based web page classification system,” *Knowledge-Based Systems*, vol. 57, pp. 168–180, 2014.
- [84] L. Rocha, F. Mourão, H. Mota, T. Salles, M. A. Gonçalves, and W. Meira Jr, “Temporal contexts: Effective text classification in evolving document collections,” *Information Systems*, vol. 38, no. 3, pp. 388–409, 2013.
- [85] R. Johnson and T. Zhang, “Effective use of word order for text categorization with convolutional neural networks,” *The 2015 Annual Conference of the North American Chapter of the ACL*, pp. 103–112, 2015.
- [86] D. Wang, J. Wu, H. Zhang, K. Xu, and M. Lin, “Towards enhancing centroid classifier for text classification: a border-instance approach,” *Neurocomputing*, vol. 101, pp. 299–308, 2013.

- [87] J. Fu and S. Lee, "A multi-class svm classification system based on learning methods from indistinguishable chinese official documents," *Expert Systems with Applications*, vol. 39, no. 3, pp. 3127–3134, 2012.
- [88] K. Kim, B.-s. Chung, Y. Choi, S. Lee, J.-Y. Jung, and J. Park, "Language independent semantic kernels for short-text classification," *Expert Systems with Applications*, vol. 41, no. 2, pp. 735–743, 2014.
- [89] H. Liu, C. Jiang, C. Hu, and L. Zhang, "Efficient relation extraction method based on spatial feature using elm," *Neural Computing and Applications*, vol. 27, no. 2, pp. 271–281, 2016.
- [90] X. Li, J. Ouyang, and X. Zhou, "Labelset topic model for multi-label document classification," *Journal of Intelligent Information Systems*, vol. 46, no. 1, pp. 83–97, 2016.
- [91] D.-T. Vo and C.-Y. Ock, "Learning to classify short text from scientific documents using topic models with various types of knowledge," *Expert Systems with Applications*, vol. 42, no. 3, pp. 1684–1698, 2015.
- [92] W. Song and S. C. Park, "Genetic algorithm for text clustering based on latent semantic indexing," *Computers & Mathematics with Applications*, vol. 57, no. 11, pp. 1901–1907, 2009.
- [93] P. Li, B. Wang, and W. Jin, "Improving web document clustering through employing user-related tag expansion techniques," *Journal of Computer Science and Technology*, vol. 27, no. 3, pp. 554–566, 2012.
- [94] C.-P. Wei, C. C. Yang, and C.-M. Lin, "A latent semantic indexing-based approach to multilingual document clustering," *Decision Support Systems*, vol. 45, no. 3, pp. 606–620, 2008.
- [95] C. Qimin, G. Qiao, W. Yongliang, and W. Xianghua, "Text clustering using vsm with feature clusters," *Neural Computing and Applications*, vol. 26, no. 4, pp. 995–1003, 2015.

- [96] F. Huang, S. Zhang, M. He, and X. Wu, "Clustering web documents using hierarchical representation with multi-granularity," *World Wide Web*, vol. 17, no. 1, pp. 105–126, 2014.
- [97] A. K. Farahat and M. S. Kamel, "Statistical semantics for enhancing document clustering," *Knowledge and information systems*, vol. 28, no. 2, pp. 365–393, 2011.
- [98] B. S. Everitt, S. Landau, M. Leese, and D. Stahl, "Hierarchical clustering," *Cluster Analysis, 5th Edition*, pp. 71–110, 2001.
- [99] A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical pattern recognition: A review," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 1, pp. 4–37, 2000.
- [100] R. Xu, D. Wunsch, *et al.*, "Survey of clustering algorithms," *Neural Networks, IEEE Transactions on*, vol. 16, no. 3, pp. 645–678, 2005.
- [101] J. Tang, "Improved k-means clustering algorithm based on user tag," *Journal of Convergence Information Technology*, vol. 5, no. 10, pp. 124–130, 2010.
- [102] P. Pantel and D. Lin, "Discovering word senses from text," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 613–619, ACM, 2002.
- [103] J. Chen, O. R. Zaïane, and R. Goebel, "An unsupervised approach to cluster web search results based on word sense communities," in *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology-Volume 01*, pp. 725–729, IEEE Computer Society, 2008.
- [104] S. Chakrabarti, *Mining the Web: Discovering knowledge from hypertext data*, vol. 1. Elsevier, 2002.
- [105] P. Worawitphinyo, X. Gao, and S. Jabeen, "Improving suffix tree clustering with new ranking and similarity measures," in *Advanced Data Mining and Applications*, pp. 55–68, Springer, 2011.

- [106] X. Gu, X. Wang, R. Li, K. Wen, Y. Yang, and W. Xiao, “A new vector space model exploiting semantic correlations of social annotations for web page clustering,” in *Web-Age Information Management*, pp. 106–117, Springer, 2011.
- [107] C. X. Lin, Y. Yu, J. Han, and B. Liu, “Hierarchical web-page clustering via in-page and cross-page link structures,” in *Advances in Knowledge Discovery and Data Mining*, pp. 222–229, Springer, 2010.
- [108] M.-S. Paukkeri, I. Kivimäki, S. Tirunagari, E. Oja, and T. Honkela, “Effect of dimensionality reduction on different distance measures in document clustering,” in *Neural Information Processing*, pp. 167–176, Springer, 2011.
- [109] M. T. Hassan and A. Karim, “Clustering and understanding documents via discrimination information maximization,” in *Advances in Knowledge Discovery and Data Mining*, pp. 566–577, Springer, 2012.
- [110] E. Glover, D. M. Pennock, S. Lawrence, and R. Krovetz, “Inferring hierarchical descriptions,” in *Proceedings of the eleventh international conference on Information and knowledge management*, pp. 507–514, ACM, 2002.
- [111] W. S. van Heerden and A. P. Engelbrecht, “Unsupervised weight-based cluster labeling for self-organizing maps,” in *Advances in Self-Organizing Maps*, vol. 198, pp. 45–54, Springer, 2013.
- [112] Y. Ping, Y.-J. Tian, Y.-J. Zhou, and Y.-X. Yang, “Convex decomposition based cluster labeling method for support vector clustering,” *Journal of Computer Science and Technology*, vol. 27, no. 2, pp. 428–442, 2012.
- [113] A. Turel and F. Can, “A new approach to search result clustering and labeling,” in *Information Retrieval Technology*, vol. 7097, pp. 283–292, Springer, 2011.
- [114] X. Li, J. Chen, and O. Zaiane, *Text document topical recursive clustering and automatic labeling of a hierarchy of document clusters*. 2013.
- [115] R. de Padua, V. O. de Carvalho, and A. B. de Souza Serapião, “Labeling association rule clustering through a genetic algorithm approach,” in *New Trends in Databases and Information Systems*, pp. 45–52, Springer, 2014.

- [116] G. Tholpadi, M. K. Das, C. Bhattacharyya, and S. Shevade, “Cluster labeling for multilingual scatter/gather using comparable corpora,” in *Advances in Information Retrieval*, vol. 7224, pp. 388–400, Springer, 2012.
- [117] J. Lee and D. Lee, “An improved cluster labeling method for support vector clustering,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 27, no. 3, pp. 461–464, 2005.
- [118] V. D’Orangeville, M. A. Mayers, M. E. Monga, and M. S. Wang, “Efficient cluster labeling for support vector clustering,” *Knowledge and Data Engineering, IEEE Transactions on*, vol. 25, no. 11, pp. 2494–2506, 2013.
- [119] L. A. Lopes, V. P. Machado, and R. d. A. Rabelo, “Automatic cluster labeling through artificial neural networks,” in *Neural Networks (IJCNN), 2014 International Joint Conference on*, pp. 762–769, IEEE, 2014.
- [120] Z. Li, J. Li, Y. Liao, S. Wen, and J. Tang, “Labeling clusters from both linguistic and statistical perspectives: A hybrid approach,” *Knowledge-Based Systems*, vol. 76, pp. 219–227, 2015.
- [121] R. Nayak, R. Mills, C. De-Vries, and S. Geva, “Clustering and labeling a web scale document collection using wikipedia clusters,” in *Proceedings of the 5th International Workshop on Web-scale Knowledge Representation Retrieval & Reasoning*, pp. 23–30, ACM, 2014.
- [122] H. Roitman, S. Hummel, and M. Shmueli-Scheuer, “A fusion approach to cluster labeling,” in *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pp. 883–886, ACM, 2014.
- [123] F. Geraci, M. Pellegrini, M. Maggini, and F. Sebastiani, “Cluster generation and cluster labelling for web snippets: A fast and accurate hierarchical solution,” in *String Processing and Information Retrieval*, pp. 25–36, Springer, 2006.
- [124] F. Fukumoto and Y. Suzuki, “Cluster labelling based on concepts in a machine-readable dictionary,” in *IJCNLP*, pp. 1371–1375, 2011.

- [125] M. Ji, “Using fuzzy sets to improve cluster labelling in unsupervised classification,” *International Journal of Remote Sensing*, vol. 24, no. 4, pp. 657–671, 2003.
- [126] M. F. Moura and S. O. Rezende, “Choosing a hierarchical cluster labelling method for a specific domain document collection,” *New Trends in Artificial Intelligence*, pp. 812–823, 2007.
- [127] M. Muhr, R. Kern, and M. Granitzer, “Analysis of structural relationships for hierarchical cluster labeling,” in *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’10, (New York, NY, USA), pp. 178–185, ACM, 2010.
- [128] S. A. Caraballo, “Automatic construction of a hypernym-labeled noun hierarchy from text,” in *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL ’99, (Stroudsburg, PA, USA), pp. 120–126, Association for Computational Linguistics, 1999.
- [129] E. Cambria and G.-B. Huang, “Extreme learning machines,” *IEEE intelligent systems*, vol. 28, no. 6, pp. 30 – 31, 2013.
- [130] R. Rifkin, G. Yeo, and T. Poggio, “Regularized least-squares classification,” *Nato Science Series Sub Series III Computer and Systems Sciences*, vol. 190, pp. 131–154, 2003.
- [131] S. Eyheramendy and D. Madigan, “A novel feature selection score for text categorization,” in *Proceedings of the Workshop on Feature Selection for Data Mining, in conjunction with the 2005 SIAM International Conference on Data Mining*, pp. 1–8, 2005.
- [132] G.-B. Huang, X. Ding, and H. Zhou, “Optimization method based extreme learning machine for classification,” *Neurocomputing*, vol. 74, no. 1, pp. 155–163, 2010.
- [133] S. Basu, M. Bilenko, A. Banerjee, and R. J. Mooney, “Probabilistic semi-supervised clustering with constraints,” *Semi-supervised learning*, pp. 71–98, 2006.

- [134] S. Basu, M. Bilenko, and R. J. Mooney, "Comparing and unifying search-based and similarity-based approaches to semi-supervised clustering," in *Proceedings of the ICML-2003 workshop on the continuum from labeled to unlabeled data in machine learning and data mining*, pp. 42–49, 2003.
- [135] A. Demiriz, K. P. Bennett, and M. J. Embrechts, "Semi-supervised clustering using genetic algorithms," *Artificial neural networks in engineering (ANNIE-99)*, pp. 809–814, 1999.
- [136] L. Zhang, W.-D. Zhou, and L. Jiao, "Kernel clustering algorithm," *CHINESE JOURNAL OF COMPUTERS-CHINESE EDITION-*, vol. 25, no. 6, pp. 587–590, 2002.
- [137] G.-B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, no. 16, pp. 3056–3062, 2007.
- [138] G.-B. Huang and L. Chen, "Enhanced random search based incremental extreme learning machine," *Neurocomputing*, vol. 71, no. 16, pp. 3460–3468, 2008.
- [139] G.-B. Huang, L. Chen, C. K. Siew, *et al.*, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879–892, 2006.
- [140] S. Basu, A. Banerjee, and R. Mooney, "Semi-supervised clustering by seeding," in *In Proceedings of 19th International Conference on Machine Learning (ICML-2002)*, pp. 19–26, Citeseer, 2002.
- [141] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. Verkamo, "Fast discovery of association rules," *Advances in Knowledge Discovery and Data Mining*, pp. 307–328, 1996.
- [142] J. C. Bezdek, R. Ehrlich, and W. Full, "Fcm: The fuzzy c-means clustering algorithm," *Computers & Geosciences*, vol. 10, no. 2, pp. 191–203, 1984.
- [143] J. Singh, H. Ram, and D. J. Sodhi, "Improving efficiency of apriori algorithm using transaction reduction," *International Journal of Scientific and Research Publications*, vol. 3, no. 1, pp. 1–4, 2013.

- [144] A. J. Butte and I. S. Kohane, “Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements,” in *Pac Symp Biocomput*, vol. 5, pp. 418–429, 2000.
- [145] R. Fletcher, *Practical methods of optimization*. John Wiley & Sons, 2013.
- [146] C. Cortes and V. Vapnik, “Support-vector networks,” *Mach. Learn.*, vol. 20, pp. 273–297, Sept. 1995.
- [147] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [148] G.-B. Huang, “Learning capability and storage capacity of two-hidden-layer feedforward networks,” *Neural Networks, IEEE Transactions on*, vol. 14, no. 2, pp. 274–281, 2003.
- [149] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, “A fast and accurate online sequential learning algorithm for feedforward networks,” *Neural Networks, IEEE Transactions on*, vol. 17, no. 6, pp. 1411–1423, 2006.
- [150] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
- [151] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.
- [152] H. G. V. Kasun, Zhou H, “representational learning with elms for big data scholarly article,” *IEEE Intelligent System(In Press)*, 2013.
- [153] Y. Yang and J. O. Pedersen, “A comparative study on feature selection in text categorization,” in *ICML*, vol. 97, pp. 412–420, 1997.
- [154] P. J. Rousseeuw, “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis,” *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

Design and Investigation of Techniques to Improve Information Retrieval on the Web

THESIS

submitted in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

RAJENDRA KUMAR ROUL

2009PHXF0421G

under the supervision of

Dr. Sanjay Kumar Sahay



BITS Pilani
Pilani | Dubai | Goa | Hyderabad

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
PILANI (RAJASTHAN), INDIA, 2016**

CHAPTER 8

CONCLUSIONS AND FUTURE DIRECTIONS

Information retrieval is the activity of obtaining information resources relevant to an information need from a collection of information resources. Deep learning approaches which have taken the machine learning community by storm have a high impact on IR. It has been successfully applied for image processing, speech recognition and NLP. The main advantage of deep learning over conventional approaches is that it is completely data driven with stacked layers of neural networks progressively “learning” the data with increasing levels of abstraction, without the necessity of manually hand-coded features. In the context of NLP, word embedding is the starting point of transforming a categorical feature, e.g. a word from a vocabulary, into a continuous representation of a real-valued vector in the Cartesian space of ‘p’ dimensions.

This thesis primarily focused on how deep learning can be useful for text data and some other aspects of IR. For this purpose, ML-ELM is used for text classification and the feature space of ML-ELM has been tested extensively for clustering the text data. The work is carried out by initially analyzing and comparing the performance of SVM, ELM and ML-ELM classifiers to demonstrate the potential of the ELM (for accuracy) and ML-ELM (both accuracy and F-measure) as a highly successful and suitable technique for text classification. It is also observed from the experimental results that ML-ELM can outperform other existing classifiers. Additionally, the results can serve as complementary knowledge to further strengthen the understanding of the essential relationship between SVM, ELM and ML-ELM. Empirical results show that ML-ELM outperformed the traditional classifiers in the majority of cases, and

achieved the best performance overall for both datasets. The experimental results also indicate the high suitability and effectiveness of ELMs in this field as well.

After testing ML-ELM on different benchmark datasets using different traditional feature selection techniques, the interest has been taken to develop two novel feature selection techniques (*KWFS and CCSS*) on which the performance of ML-ELM is tested. In *KWFS*, initially, each cluster is divided into k sub-clusters using k -means algorithm. Then, with the help of Wordnet and cosine-similarity, a reduced feature vector is generated for the entire corpus. For text classification, ELM and ML-ELM classifiers are used. This technique is tested on 20-Newsgroups and DMOZ datasets and the results show the importance of ML-ELM in the field of text classification. In the second feature selection technique (*CCSS*), three important parameters (cohesion, separation and silhouette coefficient) are combined to generate a reduced feature vector. Multilayer ELM as the classifier has been used for classifying the text document and its importance also has been extensively measured. It is evident from all the experimental results that the performance of Multilayer ELM outperforms the other well known classifiers. The encouraging results of two proposed techniques show the stability and effectiveness of Multilayer ELM compared to different progressive classifiers in the domain of text classification. This justifies the stability, efficiency and effectiveness of deep learning.

Combining the ELM and Multilayer ELM feature space with other traditional classifiers will further reinforce the classification results. Similarly, the feature space of ELM and Multilayer ELM can also be used for the clustering process, as the features became more simple and linearly separable by representing them in an extended space. This may outperform the clustering process done in TF-IDF vector space. For this propose, the feature space of ML-ELM is used for semi-supervised clustering using seeded- k Means algorithm. From the experimental results on Classic3 and Reuters datasets, following points are observed:

- results using *ELM feature space* always outperform the results of *without using ELM feature space* (i.e. *TF-IDF vector space*).
- Seeded- k Means performs very well compared to k Means in all aspects regardless of the parameters set (i.e. number of clusters to be formed, % of labels to be considered in seeded- k Means). This demonstrates the superiority of the seeded- k Means clustering technique over the traditional one even if very few labels (only 10 % of the labels are considered) are provided.

- results of purity and entropy are close in ML-ELM compressed or equal dimension space ($L \leq n$) whereas it is better in extended feature space ($L > n$) on both datasets.
- It is also observed that after a certain stage (i.e. threshold point) in extended space, the performance of the clustering process remains unchanged (i.e. further increasing L compared to n has no effect). This may be due to the excessive sparse representation of the features which is not included in the results.

Next, some other important aspects of IR has been discussed by proposing a novel clustering technique based on apriori approach to cluster the text documents. In this technique, a new modified apriori approach has been proposed and it is compared with the traditional apriori algorithm. The proposed modified apriori approach when run on a corpus of web documents produced the same clusters that a traditional apriori approach could. However, experimentally it has been proved that modified apriori approach is more efficient and faster than the traditional apriori approach. This is so because at each step the documents from the corpus are removed, which is no longer required and in turn it reduces the unnecessary comparisons. Hence, it saves a lot of time. Classic4, 20-Newsgroups and Reuters datasets are used for experimental purposes. It is found that FCM gives better clusters compared to VSM and k -means.

In order to label the clusters, a novel cluster labeling technique is developed. Using a feature selection technique, first top- k keywords known as representative keywords are selected from each cluster which are later sent to Wikipedia for getting candidate labels. The candidate labels generated from Wikipedia are evaluated using MI-score. The approach generates the actual labels by just considering the top-3 candidate labels of Wikipedia. For experimental work, 20-Newsgroups and Reuters datasets are considered. The experimental results illustrate the accuracy of the proposed approach which is more than 85% and 75% by generating good labels (that match with the actual labels) for most of the clusters of 20-Newsgroups and Reuters datasets, respectively. This justifies the efficiency of the proposed approach.

Although Multilayer ELM performs well, but still there are some shortcomings which need more attention and can be added to the future work are:

- Determining activation function and the number of nodes for each hidden layer.
- Similarly in ELM, the behavior and a correct number of hidden units is still debatable.
- Also, if one wants to deeply understand the functionality of ELM and Multilayer ELM

by considering them as an approximation of infinite network, then the variance of hidden layer weights is still an open question.

Also, the feature space of ELM and ML-ELM can be used for text classification. Similarly, in cluster labeling, separation of terms into categories of discriminative, short and common terms are explicitly required. Other enhancements like metadata associated with each document apart from the title of the documents can also be used as potential labels. As our approach successfully encompasses all the important milestones needed for automatically providing a label to each cluster of documents, all the steps can be coupled in a single script, which can serve as an important software tool to generate the label for any cluster.