CHAPTER 3

SEMANTICS BASED FEATURE EXTRACTION FOR TOPIC CLASSIFICATION

3.1 INTRODUCTION

Web contents represent a universal repository of knowledge. A convenient way of storing also requires a convenient way of retrieving. The web documents such as digital libraries, blogs, reviews and online news require automatic classification to categorize them in a pre-defined class, thereby enhancing the IR system and content management. Most of the TC models that are built using BOW representation of the document involve only the frequency of words but not their semantics. It is also observed that training a classifier with huge amount of relevant features leads to the increase in computational cost. Moreover, the contribution of infrequent terms projects a better prediction of topics. Hence a new approach for Web content classification is proposed by identifying the concept of single words in the document.

The concepts are extracted as Hypernyms that serves as semantic features in addition with the original feature for training the classifier. Each features of document corpus are weighted through TF-IDF measure to focus infrequent terms and highly weighted features are extracted for training. The Hypernym word tree is created that represent the relationship between the Hypernyms. For the fast retrieval of the Hypernyms , radix trie (ret-rie-val) is constructed where word weight is computed based on Hypernym number present in a radix trie. This chapter discusses the development of efficient

feature extraction methods using weighted TF-IDF and Hypernyms by generating the topic directories to provide better indexing for IR system. The performance of the proposed system is evaluated through the k-NN and Naïve Bayes classifier using standard WebKB dataset to show the classification accuracy, precision and recall.

3.2 PROPOSED METHODOLOGY OF SEMANTIC FEATURE EXTRACTION

Initially documents taken for topic classification are bifurcated into training documents for building the classifier and testing documents to validate the classifier. The training documents with class labels are initially preprocessed and relevant words are extracted. The Hypernyms of each word are semantically extracted and ranked using TF-IDF weighting scheme. The Hypernyms are identified by using WordNet ontology.

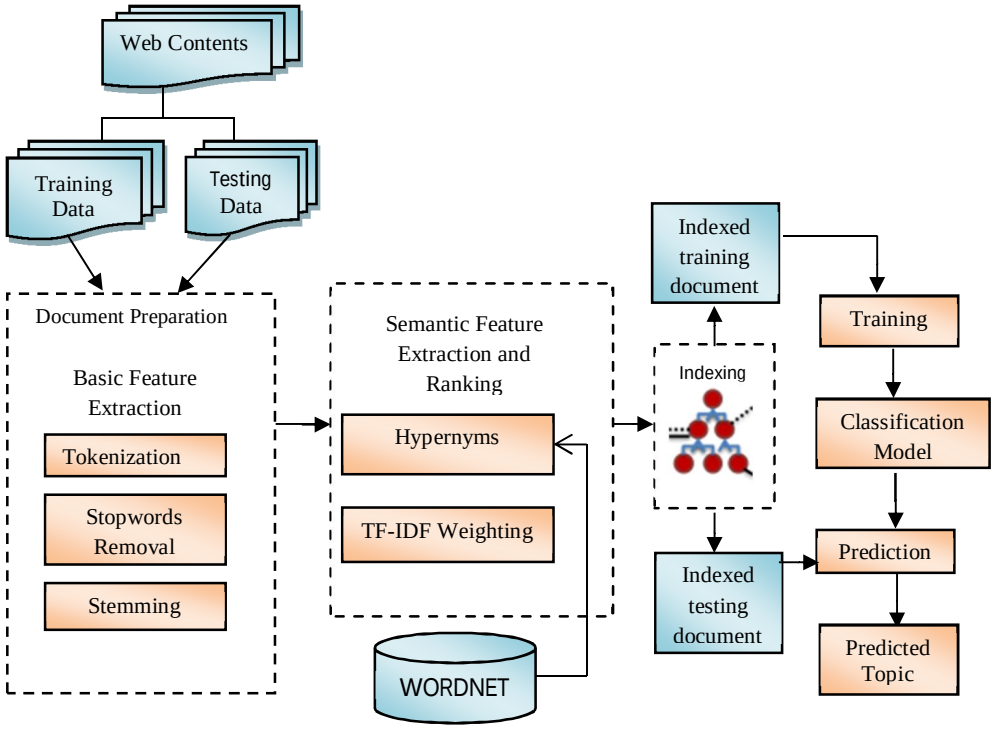

Figure 3.1 Workflow Model for Semantic Feature Extraction and Indexing for TC

For the fast retrieval, Hypernym word tree is created using radix trie(re-trie-val). The word tree serves as an index for training the classifier. This trained model is subsequently used for predicting the class of the unknown documents. The proposed workflow model for semantic feature extraction and indexing approach is depicted in Figure 3.1.

3.2.1 Document Preparation

In this phase, the documents are pre-processed to restrict the irrelevent features from the documents and hence tries to prove the hypothisis, "By neglecting terms that do not contribute to the assessment of a document‘s relevance will improve the efficiency of the IR system".

Preprocessing is the process of basic feature extraction using NLP tasks such as tokenization, removal of stopwords and stemming. Before applying the preprocessing techniques, the input documents are divided into training documents and testing documents. The training documents contain the class label or topic, whereas there is no need for class label in the testing documents. In tokenization task, a document is treated as a string, and then partitioned into a list of tokens. Tokenization is applied to all the documents taken for training and testing. Stopwords are the words in documents that occur frequently but are meaningless regarding IR, some of the very common stopwords are to, and, for, a, an, etc. These stopwords are removed from the tokens.

The resultant tokens are subjected to stemming process which conflates a group of inflected words with same stem into a single feature. For example, words such as “adhere”, “adheres”, “adhering”, “adhesion” and

“adhesive” can be reduced to the single word stem “adhere”. A highly used stemming algorithm is the Porter stemmer(Porter 1980). Two flavors of morphological analysis are carried out, the first one is a normal stemming where words reduced to their stem and another one is light-stemming where the common affixes are removed from words. The above mentioned feature extraction processes are demonstrated through the example given below.

For instance, to find all documents containing information about car accidents which happend in Chennai and to know about the people injured. This information need is expressed as a query. IR do not search through the documents but through the representation such as topic matrix or index. Suppose there are three documents D1, D2 and D3 holding the information about the accident are given as :

D1	: Heavy accident	-	Because of a heavy car accident four people died and two people got injury yesterday morning at Chennai.
D2	: More vehicles	-	In this year more cars became registered in Chennai.
D3	: Lorry causes accident	-	In Chennai a lorry hit the car which is from Egmore, four people were injured.

The information needed to be retrieved is the document containing information about "accidents with heavy vehicles in Chennai". The expected result is in D3, but not all terms of the query occur in the document D3, the terms such as "accident" and "heavy" are also occur in D1. In Table 3.1, the initial representation of terms in the document is shown.

Table 3.1 Initial Representation of Documents

Term	Document-IDs	Term	Document-IDs
Because	D1	at	D1
of	D1	chennai	D1,D2,D3
a	D1,D3	became	D2
heavy	D1	cars	D2
car	D1	in	D1,D2,D3
accident	D1,D3	more	D2
four	D1,D3	hit	D3
people	D1,D3	year	D2
died	D1	registered	D2
and	D1	lorry	D3
two	D1	vehicles	D2
got	D1	egmore	D3
injury	D1	...	
yesterday	D1		
morning	D1		

From the above example, the documents are initailly decomposed into set of terms and then preprocessing is applied to capture the relevent terms for indexing. The Table 3.2 shows the process of tokenisation, stop words removal and stemming.

Table 3.2 Preprocessing with Tokenisation, Stopwords Removal and Stemming

Process	Example
Initial Representation of terms	because of a heavy car accident four people died and two people got injury yesterday morning at chennai in this year more cars became registered lorry hit the which is from Egmore were injured
Tokenisation	because | of | a | heavy | car |accident |four |people | died |and | two | people |got |injury |yesterday | morning |at | chennai | in |this |year |more | cars | became | registered |lorry |hit |the |which |is| from | egmore | were | injured
Stopwords Removal	because | ~~of~~ | ~~a~~ | heavy | car |accident |four |people | died |~~and~~| two | people |~~got~~ |injury |yesterday | morning |~~at~~ |Chennai |~~In~~ |this |year |more | cars | became | registered |lorry |hit |~~the~~ |which | ~~is~~| ~~from~~ | Egmore | ~~were~~ | injured
Stemming	because | heavy | car |accident |four |people | died | two | people | injury |yesterday | morning | chennai |this |year |more | ~~cars~~ | became | registered |lorry |hit |which | egmore | ~~injured~~

The resulting features extracted after stemming process is subjected to further processing.

3.2.2 Relevant Feature Extraction using TF-IDF Weighting

The terms extracted with above methods serves as an index terms for training. In this phase, the importance of the index terms is justified by associating weights to the terms. Let T_i denotes an i^{th} term of j^{th} document D_j in the corpus and W_{ij} denotes the weight associated with (T_i,D_j). Hence the weight W_{ij} is used to quantify the importance of the particular index term for representing the document contents.

The weight values of each index terms are estimated with a numerical statistical measure known as Term Frequency–Inverse Document Frequency(TF-IDF). The importance of index term to a document in a corpus is easily identified by this measure. Term frequency is calculated by finding the number of times each term occurs in a given document. Inverse Document Frequency always diminishes the weight of terms that occur very frequently in a document set and at the same time increases the weight of the terms that occur rarely which controls the fact that some words are more common than others.

The term frequency of i^{th} term in j^{th} document is specified as TF_{ij} and it is given in the Equation (3.1). Similarly Inverse Document Frequency of i^{th} document is given in the Equation (3.2)

$$TF_{ij} = \frac{F_{ij}}{\max\{F_{1j},F_{2j},\ldots,F_{|p|j}\}} \tag{3.1}$$

where F_{ij} denotes the occurrences of i^{th} term in j^{th} document, p denotes the total number of terms in the document.

$$IDF_i = log\frac{N}{Df_i} \tag{3.2}$$

where N denotes the total number of documents taken for training, Df_i denotes the document frequency of i^{th} term.

$$w_{i,j} = TF_{i,j} * IDF_i \tag{3.3}$$

where $w_{i,j}$ in the Equation (3.3) denotes the weight of the i^{th} term in j^{th} document.

The index terms along with its weight values makes to form a Vector Space Model for a given corpus. The corresponding topic index with associated weights is shown in the Table 3.3. The vectors with high TF-IDF

weights are identified as representative features. For example, the weight of the term "hit" is calculated as

$w_{hit,D1}$ = 0/2 * log(3/1) gives 0

$w_{hit,D2}$ = 0/2 * log(3/1) gives 0

$w_{hit,D3}$ = 2/2 * log(3/1) gives 0.477

The importance of the word "hit" in entire corpus is given as

$w_{hit,D1} + w_{hit,D2} + w_{hit,D3}$ = Total Weight

(0+ 0+ 0.477) = 0.477

Similarly, the weights are estimated for all the terms in the document.

Table 3.3 Topic Index with TF-IDF Weighting

Term	TF-IDF Value	D1	D2	D3
Accident	0.26	2	0	1
car	0.17	1	1	0
cause	0.23	0	0	1
crowd	0.23	0	0	1
die	0.23	1	0	0
four	0.23	0	0	1
heavy	0.47	2	0	0
injury	0.23	0	0	1
more	0.47	0	2	0
morning	0.23	1	0	0
hit	0.47	0	0	2
quarter	0.23	0	1	0
register	0.23	0	1	0
lorry	0.23	0	0	1
vehicle	0.23	0	1	0
chennai	0	1	1	1
yesterday	0.23	1	0	0

Thus TF-IDF Values indicates the importance of infrequent terms in the document corpus.

3.2.3 Involving Hypernyms for Feature Extraction

Semantic knowledge is gained with an idea of incorporating contextual information about terms. In this approach, contextual words are identified which are not only an ordinary terms but also have high degree of relativity with candidate terms in the given query and assumed to be highly relevant. The context words are often interpreted in the way in which sentences are constructed. Here, it is assumed that terms in the contextual space are likely to be semantically related with each other and if the word is more similar to a term in the context, then it should be more informative.

Adding semantics to the word enhances the retrieval efficiency. WordNet provides the semantics of the word by Synsets, Hypernyms and Hyponyms. A Hypernym is a term in a contextual semantic space that denotes a superordinate word or superclass. For instance, the Hypernym of the term "lion" is "animal"; and "furniture" is a Hypernym of the term "chair". In this work, semantic information of each term in its noun form is treated as Hypernym.

Chair	**Honeypot**
=> seat => furniture => furnishing => instrumentality => artifact =>whole unit => object => physical entity => entity	=> protea => shrub, bush => woody plant => vascular plant => plant => organism =>living thing => object => physical entity => entity

Figure 3.2 Hypernyms of two objects: Chair and Honeypot

Hypernyms of each term vector in the document space are extracted using WordNet ontology and Hypernym word tree is created for each document to enhance the fast retrieval. A term in a document has many senses. The extracted Hpernyms are treated as semantic features and hence incorporated with the corresponding term to form a semantic space. These superordinate terms in the semantic space serves as a context vector which may be a class or topic. The term vectors in document space and Hypernym vectors in semantic space are treated as a semantic feature to train the classifier. The portion of Hypernym-Hyponym relationship among the terms in the university document is presented in the Figure 3.2. The Hypernym for the term "professor" is "faculty" and at the same time "faculty" acts as hyponym for a term "people". There are also certain transitive semantic dependencies exists among the terms, for example, the term "people" acts as a Hypernym for "professor".

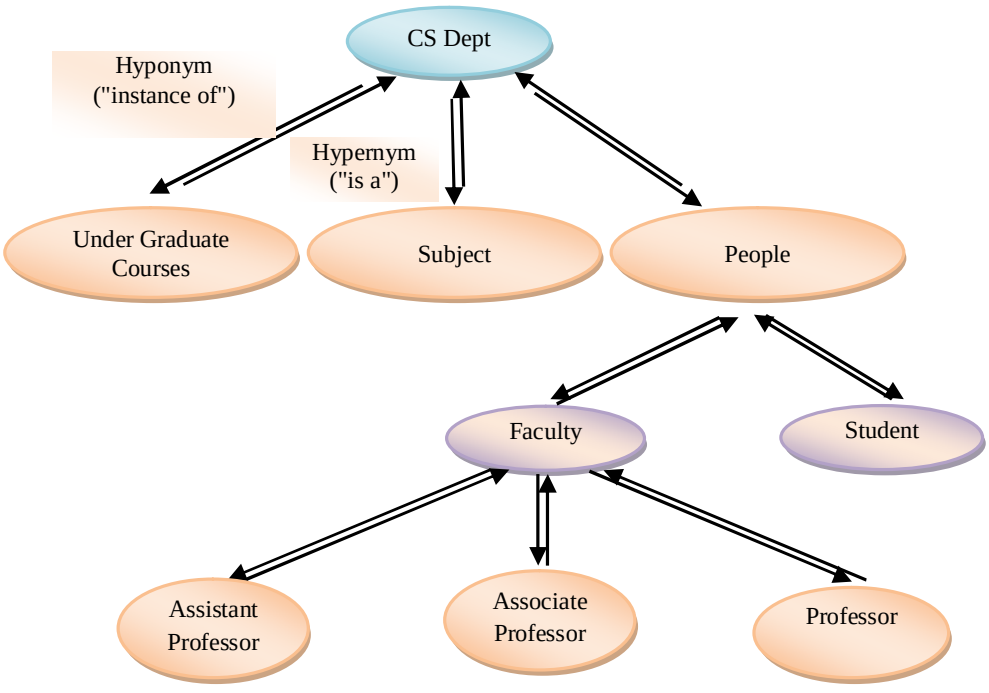

Figure 3.3 Hypernym-Hyponym Relationships Representing the Semantics

The similarity between the two Hypernyms (concept) provides a leading role in the discrimination process of the text classifier. TF-IDF is calculated for each Hypernym and stored as concept vectors. The similarity between two concept vectors is calculated using cosine similarity specified in the Equation (3.4).

$$\text{Similarity}(cv_i, cv_j) = \frac{\overrightarrow{cv_i} \cdot \overrightarrow{cv_j}}{|cv_i||cv_j|} \quad (3.4)$$

Where cv_i and cv_j denotes the concept vectors of a given term. $\overrightarrow{cv_i}$ and $\overrightarrow{cv_j}$ corresponds to TF-IDF vector values. The magnitude of vectors are represented as $|cv_i|$ and $|cv_j|$

For instance, in Figure 3.4, the terms "Postman" and "Professor" are the attributes of the two dimensional space. The context vector for the term "deliver" is intermediate between the terms "Postman" and "Professor" because the term "deliver" may be implied as “to deliver the telegraph” in the context of "Postman" or may be implied as “deliver the lecture” in the context of "Professor". The semantic context vectors for "lecture" and "student" are very close to "Professor", but they do not have a sense or meaning that is related to "Postman". The meaning for the word "subject" is interpreted as "professor deliver the lecture to the students" which includes the terms "professor", "deliver", "lecture" and "student". The gloss vectors for the term "subject" is constituted by including the context vectors of "professor", "deliver", "lecture" and "student".

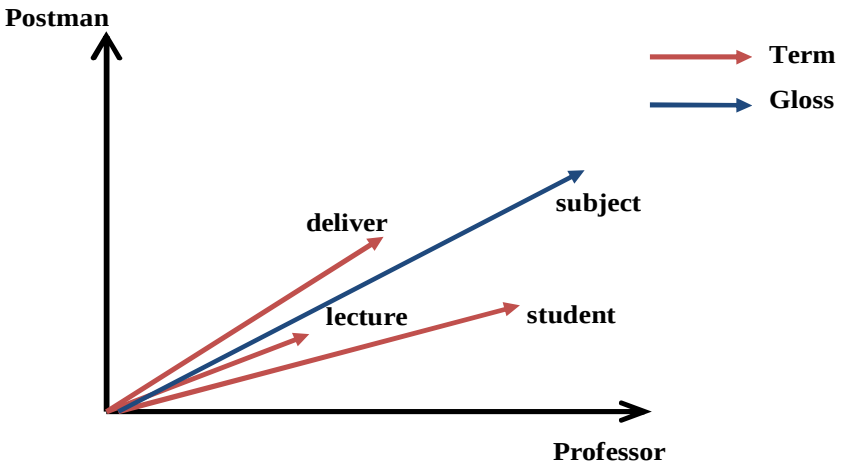

Figure 3.4 Semantic Distance Between the Hypernyms

Thus the concept "Professor", therefore, is in the same semantic space as it is associated to the concept "Subject". The semantic similarity between the Hypernyms(concepts) are calculated using Cosine Similarity metric. The figure 3.3 denotes the similarity distance between two Hypernyms "Postman" and "Professor" using Gloss vectors.

3.2.4 Hypernym Radix Trie(HRT)

In order to frequently process the document terms with Hypernyms, a trie based index structures are employed. The Hypernyms of all the words in the documents are retrieved and their word-wise permutations are represented in radix trie. Radix trie otherwise called as a prefix tree is a search tree which uses digital key representation instead of hashing/comparing key. Radix trie is not sensitive to data insertion order and so result in balanced data structures (Leis et al 2013). Radix methods are used when keys are long, which ensures that full key comparisons used by other search methods are costly. The radix method is a space-optimized data structure where each node with only one child is merged with its child. In radix methods, a key is looked at, one bit at a time or a few bits at a time. In radix search trie , all

data is stored in the leaf nodes. In this work, HRT is constructed only once and it is saved to file for further reference. Figure 3.5 shows HRT for efficient retrieval. Six different words are stored in the tree. As node number in radix tree increases, TF-IDF Hyperrnym word weightage percentage also increases.

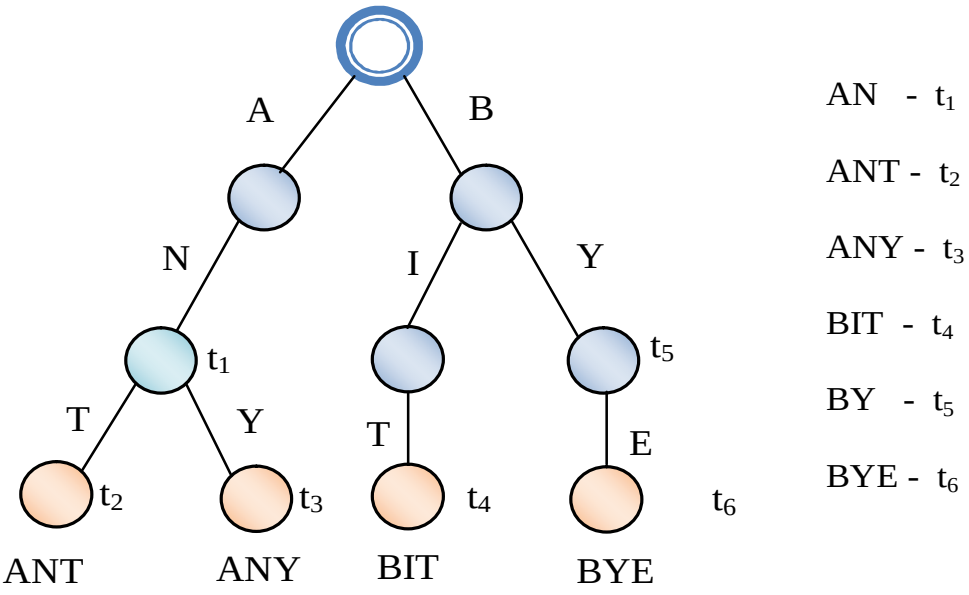

Figure 3.5 Hypernym Radix Trie for Efficient Retrieval

The Hypernyms stored in the HRT can be accessed for further processing. The TF-IDF value of the term increases when new word is inserted into the tree. Hence the weighted TF-IDF values of the Hypernyms are combined with document vectors and stored as a feature vectors. The significant feature vectors with larger TF-IDF are taken for training the classifier.

3.3 DOCUMENT CLASSIFICATION UTILIZING HYPERNYMS

The feature vectors formed through the documents features and Hypernyms(semantic features) are trained with known topic using a supervised classifier such as k-NN and Naive Bayes. The trained model is

applied to predict the category of testing documents and the performance of the classification is evaluated based on predicted category.

The k-Nearest Neighbor is labor intensive for large training sets being used in pattern recognition. To speed up k-NN, all data are stored in memory. Memory-based reasoning refers to k-NN in-memory classifier (Han et al 2006). k-NN models are also easy to understand with few predictor variables. k-NN classifiers are named for learning by analogy based where test documents are compared with similar training documents. Here, the training documents are described by d feature vector and each vector represents a point in d-dimensional space. In this work, the training features generated through term space and semantic space are stored in d-dimensional feature space. When test features are introduced, k-NN classifier searches the feature space for k training vectors that are nearer to an unknown test features vectors. These k features are designated as "nearest neighbors" of a test feature that are without class label. Cosine Similarity is used to calculate "Nearness" of features in the d dimensional space. Cosine value between any two feature vectors is given in the Equation (3.5)

$$\mathrm{Sim}(D_1, D_2) = \frac{\sum_{x \in (D_1 \cap D_2)} a_{ij} \times b_{ij}}{\|D_1\|_2 \times \|D_2\|_2} \quad (3.5)$$

where D_1 and D_2 represents the documents; x is a word shared by D_1 and D_2; a_{ij} and b_{ij} represent the weight of D_1 and D_2; $D_1 = (d_{11}, d_{12}, \ldots d_{1m})$ and $D_2 = (d_{21}, d_{22} \ldots d_{2m})$ are the feature vectors, where $\|D_1\|_2 = \sqrt{d_{11}^2, d_{12}^2, \ldots d_{1m}^2}$ and $\|D_2\|_2 = \sqrt{d_{21}^2, d_{22}^2, \ldots d_{2m}^2}$.

The algorithm for k-NN is shown in the Figure 3.6. The training vectors are represented as $\{u_1,u_2,..,u_n\}$ with the pre-defined topics as $\{t_1,t_2,...t_n\}$, testing vectors are given as $\{v_1,v_2, ...v_n\}$, k is used to specify the number of neighbors and X_k is used to denote the set of k neighbors with their corresponding topic.

k-NN Classification Algorithm

Input : Training Vectors $\{u_1,u_2,..,u_n\}$ with labeled topics $\{t_1,t_2,...t_n\}$, Testing Vectors $\{v_1,v_2, ...v_n\}$

Output : Predicted topic

1. Initialize k
2. For each testing vector v_i
3. Compute Cosine similarity between u_i and v_i using the equation (3.5)
4. Order the similarity values from lowest to highest
5. Select the k nearest training vectors to v_i along with the known topic and store it as X_k
6. Find the majority of Topic in X_k and assign to v_i
7. End

Figure 3.6 Algorithm for k-NN Classification

The next classifier used in this work is Naive Bayes technique where learning and classification methods are based on probability theory. The probability of a given document X under a topic T is calculated using the Bayes formula given in the Equation (3.6)

$$P(T|X) = \frac{P(T)P(X|T)}{P(X)} \tag{3.6}$$

where $X = (x_1,x_2,...,x_n)$ are feature vectors, T denotes the topic , P(X|T) is the conditional probability of T in X and probability of T is calculated using Equation(3.7).

$$p(T) = \frac{\text{Number of documents under T}}{\text{Total number of documents}} \tag{3.7}$$

Likewise the conditional probabilities of different words in the document are calculated. The conditional probabilities serve as a weight of the word in a document for a given topic.

Assume that a topic T_{max} yields to the maximum value for P (T|X). Naive Bayes classifier determines the topic T_{max} with the largest posterior probability using the Equation (3.8)

$$T_{max} = \text{argmax}_T P(T)P(X|T) \quad (3.8)$$

Assign the testing document to the topic which has maximum probability. The pseudo code for training and testing of Naive Bayes classifier is given in Figure 3.7.

NB-Training (Training Documents X_1, Topics T)

1. V← Extract-Vocabulary(X_1)
2. N← CountDocs(X_1)
3. for each t in T
4. do N_t ← CountDocsInClass(X_1,t)
5. prior[t] ← N_t/N
6. H ← Extract HypernymsOfAllTermsInDoc(X_1,t)
7. $text_t$ = ConcatenateTextOfAllDocsInClass(X_1,H,t)
8. For each w in V
9. do W_{ct} ← CountTokensOfWord($text_t$,w)
10. for each w in V
11. do condprob[w][t] ← $\frac{Wct+1}{\sum_{t' \epsilon V}(Wct' + 1)}$
12. return V, prior, condprob

NB-Testing (Testing Document X_2, prior, condprob, Target Topic T)

1. w ← ExtractTokensFromDoc(X_2)
2. for each t in T
3. do score[t] ← log prior[t]
4. for each w in X_2
5. do score [t] + = log condprob[w][t]
6. T_{max}← argmax_{tcT} Score[t]
7. Assign topic of w as t

Figure 3.7 Pseudo Code of Naive Bayes Algorithm

The notations such as X_1, X_2, T, w, t and W_{ct} are the training documents, testing documents, topic set, word in the given document, a specific topic in T and the number of words in a document respectively. The training and testing processes are carried out separately by using prior value and conditional probability. The input topics are indexed with 0,1,2,3 and 4. The precision, recall and accuracy of the classifier are estimated based on testing documents.

The processes involved in the training and testing modules are clearly shown in Figure 3.8.

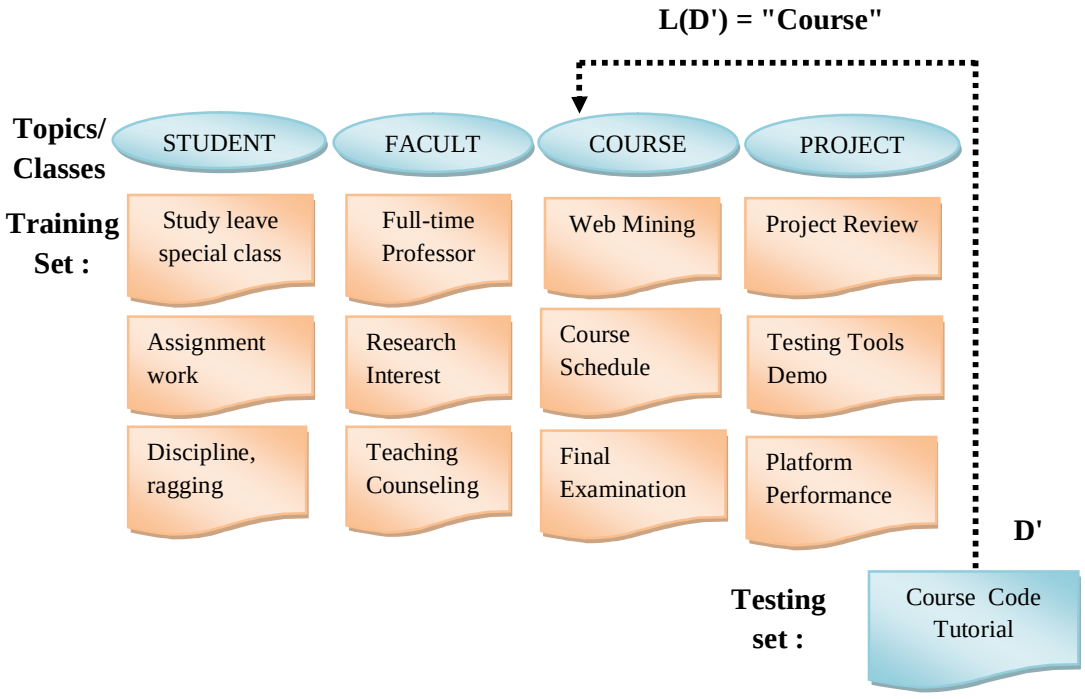

Figure 3.8 Mapping of Testing Document to Corresponding Topic

Topic classification task with set of training documents, testing document and the learning function L(D') needed for mapping testing document D' to appropriate topic are visualized in Figure 3.8. Classifier acts as a mapping function for tagging document to topic.

3.4 RESULTS AND DISCUSSIONS

The feature extracted using TF_IDF and proposed weighted TF-IDF with Hypernyms are evaluated and their effectiveness is realized with the two standard supervised classifiers such as k-NN and Naive Bayes. The implementation is carried out using Java. For incorporating semantic information, WordNet library is linked as a JAR file. The performance measures such as precision, recall and accuracy of the proposed topic classification is calculated using the Equations (1.8), (1.9) and (1.10). Web Knowledge Base (WebKB) dataset is used for analysis.

The WebKB dataset proposed by Craven et al (1998), contains WWW pages collected from Computer Science Departments of various Universities in January 1997 by the World Wide Knowledge Base project. The 8,282 pages were manually classified into the following topics : Student (1641), Faculty (1124), Staff (137), Department (182), Course (930), Project (504), Others (3764). The Table 3.4 shows the number of the training and testing documents taken for the current work.

Table 3.4 WebKB Dataset for Training and Testing

Topic (Domain)	Total Documents	No. of Documents for Training	No. of Documents for Testing
Student	120	80	40
Faculty	100	80	20
Course	100	80	20
Project	100	80	20
Others	100	80	20
Total	520	480	120

The sample training document taken under the topic "Faculty" is shown in Figure 3.9.

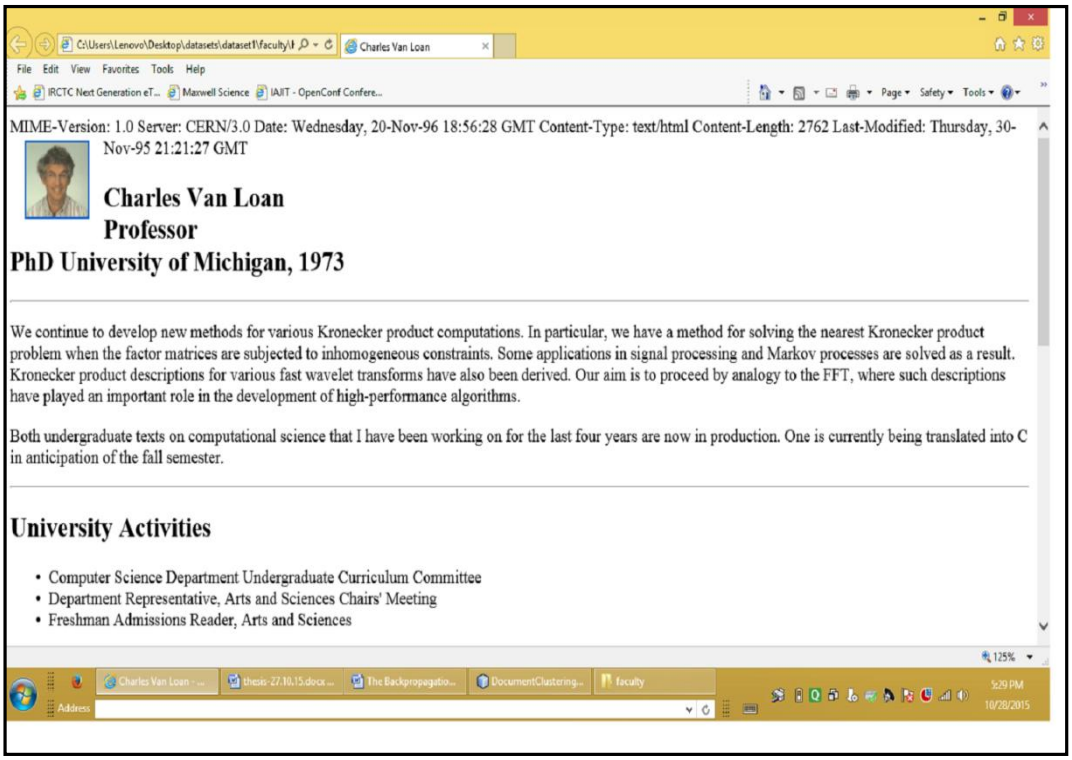

Figure 3.9 Sample Training Document of WebKB Dataset for the topic "Faculty"

The preprocessing operations such as Tokenization, Stopwords removal and Stemming are applied to the input training documents. The resultant words after preprocessing are stored in the separate file for further processing. Each term are weighted with TF-IDF measure and document vectors are created.

The Hypernyms of each word in the word space are extracted from the WordNet and weighted with TF-IDF accordingly. The similarity between the vectors is estimated using Cosine distance. These concept vectors are combined with document vectors. The weighted vectors are ranked.

The selected list of terms in the training documents are given in the Figure 3.10 and the corresponding vectors are supplied as an input to the learning algorithm.

mimeversion, cern, sunday, science, fall, cs, course, computer, hours, office, class, netscape, research, h, dl, phd, university, b, data, computing, processing, systems, applications, working, distributed, large, design, different, developed, been, algorithms, using, methods, programming, language, material, program, cornell, acm, conference, david, information, symposium, technology, please, hr, general, see, project, files, contact, previous, version, current, between, modified, click, communication, system, network, paper, order, good, building, software, implementation, technical, report, time, analysis, including, generic, variety, parallel, ieee, recent, made, techniques, operating, access, control, model, th, computation, allows, languages, support, architecture, california, doctype, public, department, ithaca, currently, engineering, performance, final, environment, written, algorithm, memory

Figure 3.10 Selected Terms from the Training Documents

The sample term vectors used for training are given as

1.0	1.0	1.0	1.0	1.0	1.0	1.0	3.0	1.0	1.0
0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
1.0	0.0	0.0	0.0	0.0	1.0	0.0	3.0	1.0	0.0
0.0	0.0	2.0	0.0	1.0	1.0	0.0	3.0	1.0	0.0
0.0	0.0	0.0	0.0	2.0	0.0	0.0	3.0	0.0	1.0

...

The vectors values of the selected terms are fed as input to the k-NN and Naive Bayes Classifiers along with pre-defined topics. An index value is assigned for each topic in such a way that [Course - 0 , Faculty - 1, Others -2, Project-3, Student-4].

The sample testing documents are given as

Testing Documents

t_doc testing1\course_http_^^www.cs.cornell.edu^Info^Courses^Current^CS280^CS280.html

t_doc testing1\course_http_^^www.cs.cornell.edu^Info^Courses^Current^CS314^

t_doc testing1\faculty_http_^^www.cs.cornell.edu^Info^Faculty^rc^rc.html

...

The training documents are used to build the model and testing documents are used for model validation.

3.4.1 Performance Comparison of k-Nearest Neighbor and Naive Bayes Algorithms with respect to Accuracy

The features extracted through TF-IDF without the semantic information and weighted TF-IDF with Hypernym semantics are trained using k-NN and Naive Bayes classifier. The performance is evaluated in terms of accuracy that is obtained through the testing documents are shown in Table 3.5.

Table 3.5 Accuracy of Representative Algorithms on WebKB Dataset

Feature Extraction(FE)	Classifier	Classification Accuracy(%)
TF-IDF	k-Nearest Neighbor	67.81
Proposed weighted TF-IDF with hypernyms	k-Nearest Neighbor	72.50
TF-IDF	Naive Bayes	70.62
Proposed weighted TF-IDF with hypernyms	Naive Bayes	76.25

The information available in Table 3.5 clearly shows that the classification accuracy for the features extracted through the proposed method with Naive Bayes is highest. From Figure 3.11, it is evidenced that classification accuracy for feature extraction using proposed technique with NB is better than feature extraction using TF-IDF with k-NN by12.45%; better than feature extraction using proposed technique with k-NN by 5.17% ; and better than FE using TF-IDF with NB by 7.97%.

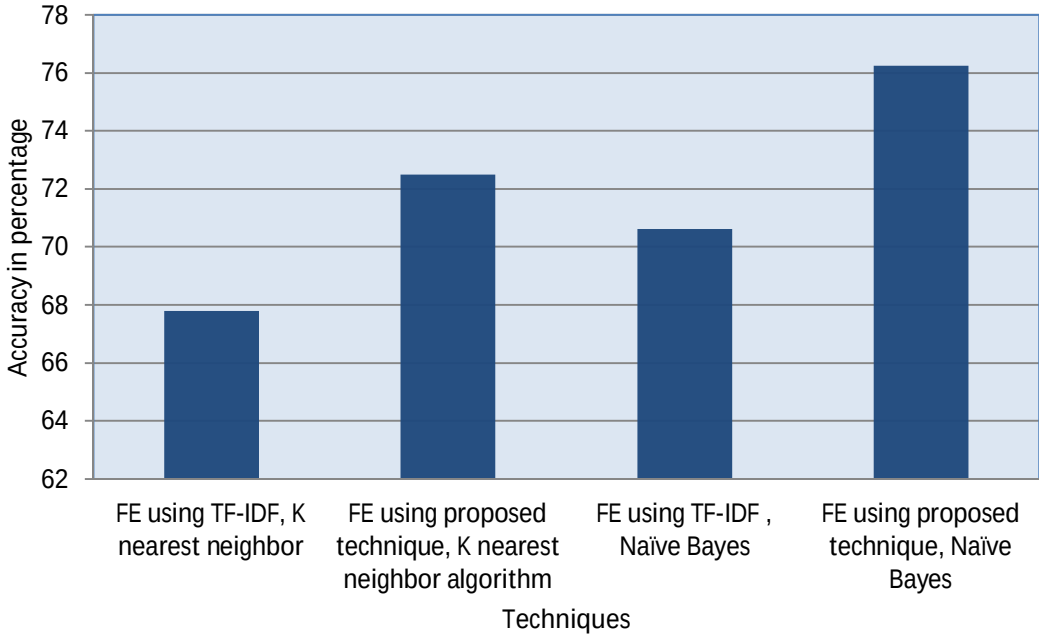

Figure 3.11 Classification Accuracy

3.4.2 Evaluation of Precision Values

The precision values are computed and represented in Table 3.6. The comparison of the values shows that precision for the topic "Course" using Naïve Bayes with features extracted through the proposed technique is highest.

Table 3.6 Precision Values of Different Techniques

Techniques Used	Precision (Course)	Precision (Faculty)	Precision (Project)	Precision (Student)
FE using TF-IDF, k-Nearest Neighbor	0.638554217	0.710144928	0.611650485	0.65
FE using proposed technique, k-Nearest Neighbor algorithm	0.7125	0.757142857	0.663265306	0.7125
FE using TF-IDF, Naïve Bayes	0.72	0.732394366	0.642857143	0.7125
FE using proposed technique, Naïve Bayes	0.766233766	0.77027027	0.712765957	0.7625

Figure 3.12 shows that precision for the topic "Course" using Naïve Bayes with FE through proposed technique is better than using k-Nearest Neighbor with FE through TF-IDF by 21.56%; better than using k-Nearest Neighbor with FE through proposed technique by 7.5% ; and better than using Naive Bayes with FE through TF-IDF by 6.4%.

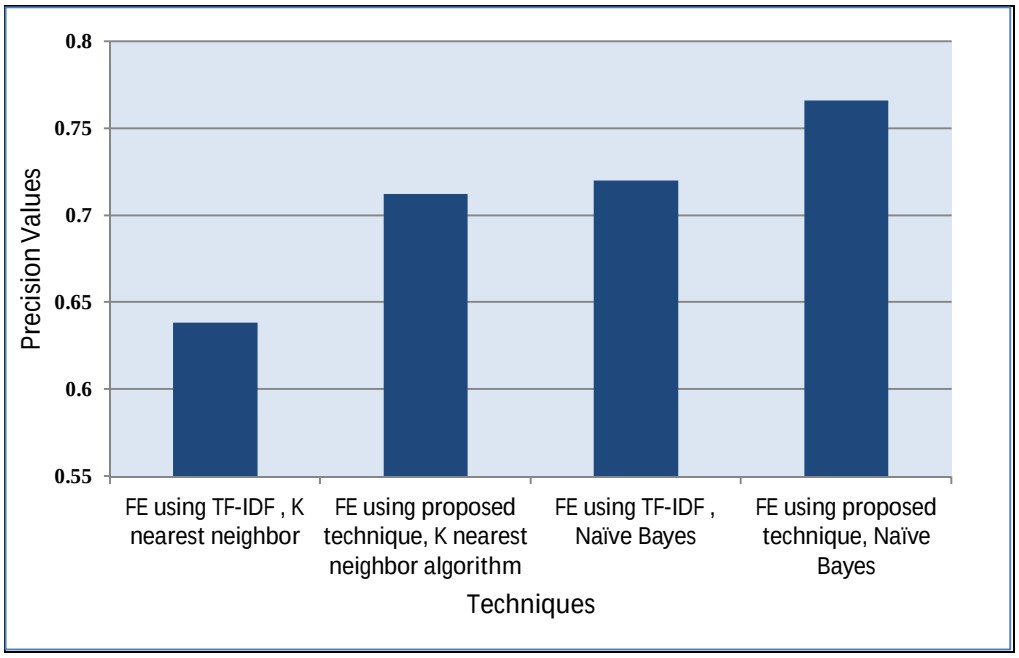

Figure 3.12 Comparison of Precision for the Topic "Course" Using different Methods

From Figure 3.13, it is evidenced that precision for the topic "Faculty" by using Naïve Bayes with FE through proposed technique is better than k -Nearest Neighbor with FE through TF-IDF by 8.4%; better than using k-Nearest Neighbor with FE through proposed technique by 1.73% ; and better than using Naïve Bayes with FE through TF-IDF by 5.1%.

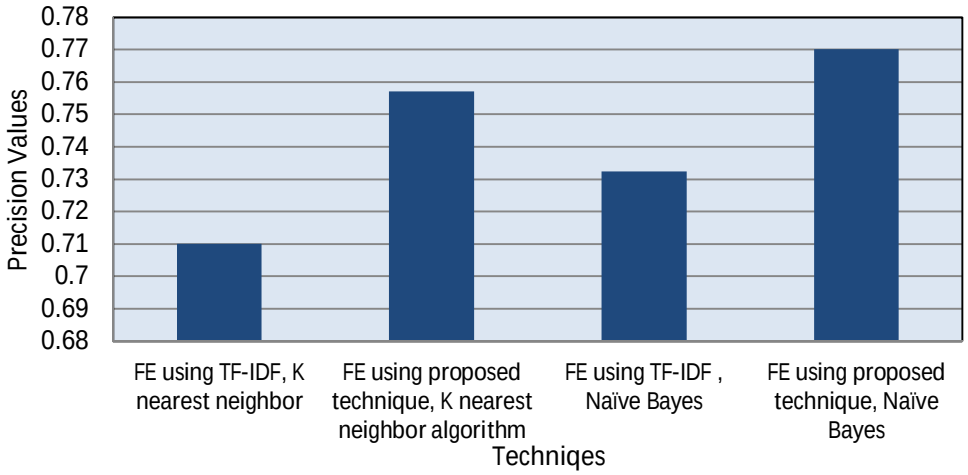

Figure 3.13 Comparison of Precision for the Topic "Faculty" Using different Methods

Figure 3.14 shows that precision for the topic "Project" using Naïve Bayes with FE through proposed technique is highest.

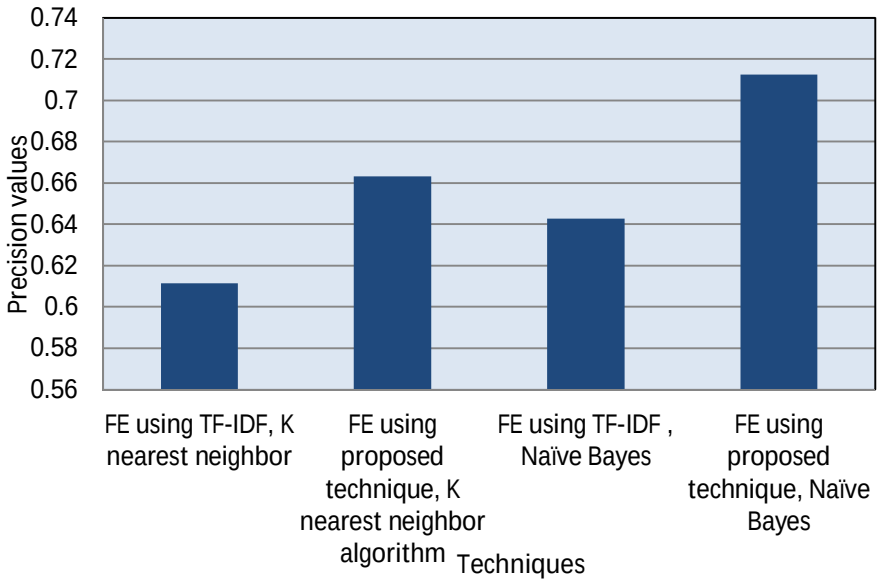

Figure 3.14 Comparison of Precision for the Topic "Project " Using different Methods

From Figure 3.14, it is evidenced that precision for the topic "Project" using Naïve Bayes with FE through the proposed technique is better than using k- Nearest Neighbor with FE through TF-IDF by 16.5%; better than using k-Nearest Neighbor with FE through proposed technique by 7.4% ; and better than using Naïve Bayes with FE through TF-IDF by 10.8%.

Figure 3.15 shows that Precision for the topic "Student" using Naïve Bayes with FE through the proposed technique is highest. Precision for FE with proposed technique, Naïve Bayes is better than FE with TF-IDF, k-Nearest Neighbor by 17.30%; better than FE with proposed technique, k-Nearest Neighbor algorithm by 7% ; and better than FE with TF-IDF , Naïve Bayes by 7%.

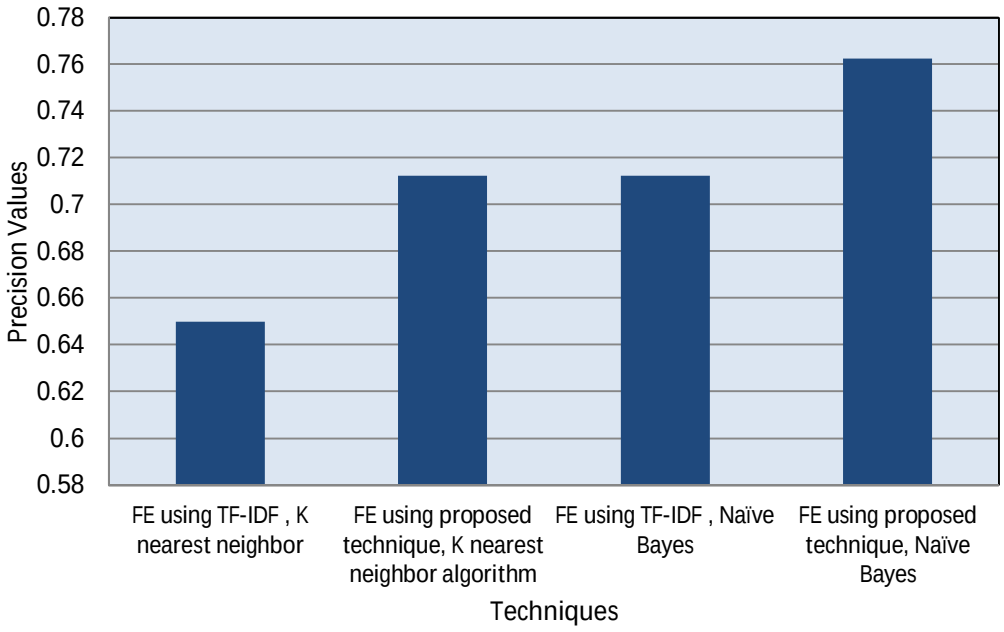

Figure 3.15 Comparison of Precision for the Topic "Student" Using different Methods

3.4.3 Evaluation of Recall

The recall values of the topics using different techniques are analyzed with the help of Table 3.7.

Table 3.7 Recall Values of Different Techniques

Techniques Used	Recall (Course)	Recall (Faculty)	Recall (Project)	Recall (Student)
FE using TF-IDF,k-Nearest Neighbor	0.6625	0.6125	0.7875	0.8
FE using proposed technique, k-Nearest Neighbor algorithm	0.7125	0.6625	0.8125	0.7917
FE using TF-IDF, Naïve Bayes	0.675	0.65	0.7875	0.75
FE using proposed technique, Naïve Bayes	0.7375	0.7125	0.8375	0.8133

Figure 3.16 shows that Recall for the topic "Course" using Naïve Bayes with FE through the proposed technique is highest. Recall of "Course" using FE with proposed technique, Naïve Bayes is better than FE with TF-IDF, k-Nearest Neighbor by 11.32%; better than FE with proposed technique, k-Nearest Neighbor algorithm by 3.5% ; and better than FE with TF-IDF , Naïve Bayes by 9.25%.

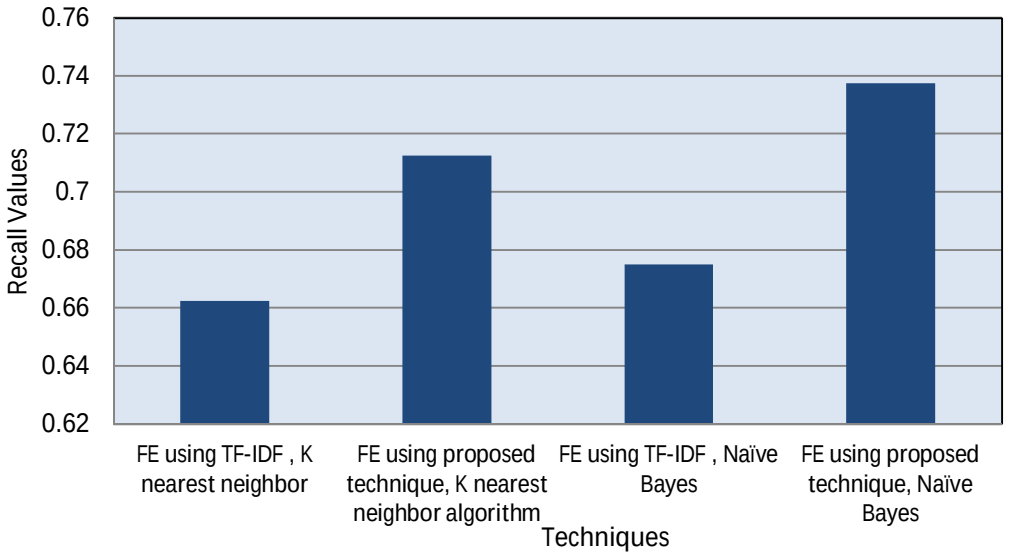

Figure 3.16 Comparison of Recall for the topic "Course" using different Methods

Figure 3.17 shows that Recall of the topic "Faculty" using Naïve Bayes with feature extraction through the proposed technique is highest. Recall for the "Faculty" using Naïve Bayes with FE through proposed technique is better than k-Nearest Neighbor with FE through TF-IDF by 16.32%; better than k-Nearest Neighbor with FE through the proposed technique by 7.5% and better than using Naïve Bayes with FE through TF-IDF, by 9.61%.

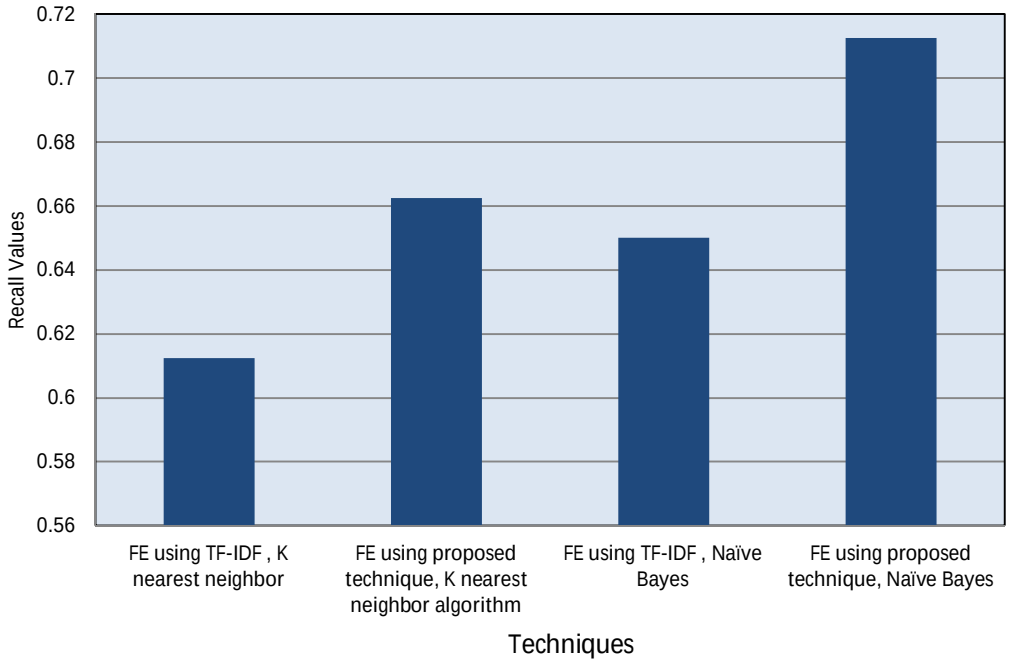

Figure 3.17 Comparison of recall for the topic "Faculty" using different Methods

Figure 3.18 shows that Recall of the topic "Project" using Naïve Bayes with feature extraction through the proposed technique is highest. Recall for "Project" using Naïve Bayes with FE through the proposed technique and is better than using k-Nearest Neighbor with FE through TF-IDF by 6.37%; better than using k-Nearest Neighbor with FE through proposed technique by 3%; and better than using Naïve Bayes with FE through TF-IDF by 6.3%.

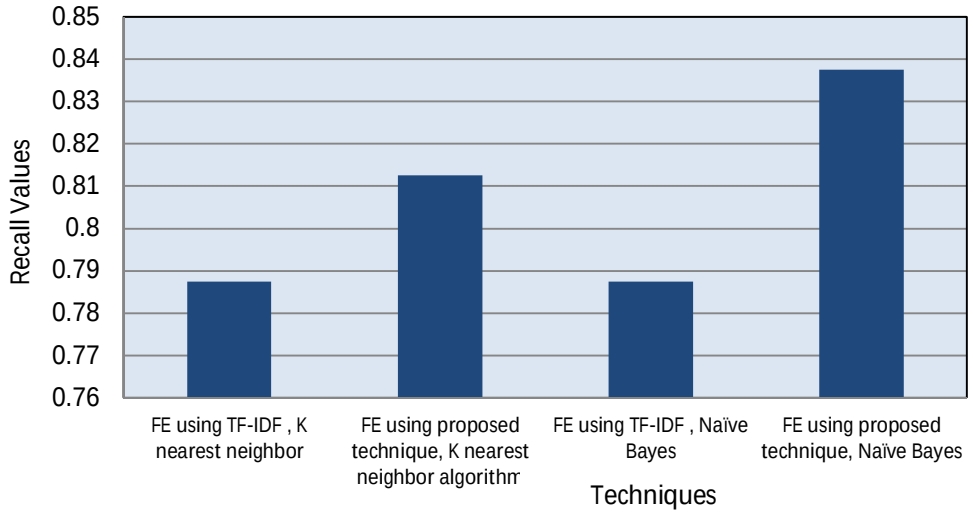

Figure 3.18 Comparison of Recall for the Topic "Project" Using Different Methods

Figure 3.19 shows that Recall for the topic "Student" using Naïve Bayes with feature extraction through the proposed technique is highest. Calculated Recall value with respect to Naïve Bayes with FE through the proposed technique is better than using k-Nearest Neighbor with FE through TF-IDF by 1.63%; better than using K- Nearest Neighbor algorithm with FE through the proposed technique by 2.7% ; and better than using Naïve Bayes with FE through TF-IDF by 8.4%.

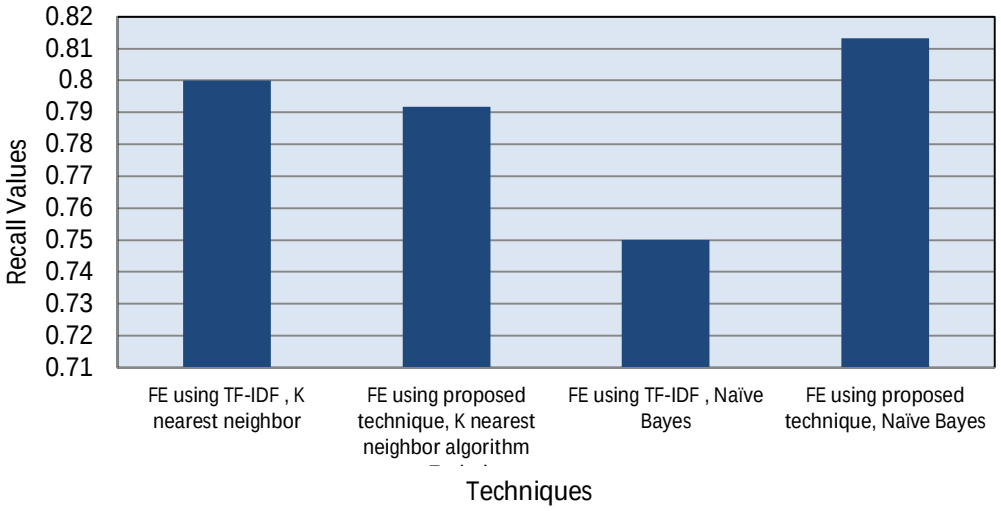

Figure 3.19 Comparison of Recall for the Topic "Student" Using Different Methods

3.5 CONCLUSION

In this work, a novel approach to combine semantic features in a form of Hypernyms in addition with document features is implemented. Extraction of semantic feature with relevant contexts performs topic categorization task efficiently. The method for identifying and relating Hypernyms using Word Net is devised and implemented. The radix trie based retrieval of Hypernyms introduced in this work provides a fast index structure for training thereby reducing the computational cost of the classifier. The terms in the document along with Hypernyms are represented as feature vectors using weighted TF-IDF and trained with k-NN and Naive Bayes classifier. The standard WebKB dataset is used for experimentation and the results show that the improved TF-IDF with Hypernyms raises text classification's precision and recall when accuracy value is evaluated correctly. The representation of document with context words provides an efficient basis for topic classification than the traditional BOW technique. This work needs further improvement in the classification accuracy. There are certain shortcomings such as lack of semantic cues and data sparseness problem because in this work, only single-word semantics are alone considered. Hence the current work needs further improvements in the methodologies for semantic feature selection and underlying learning model. This work is extended with additional approaches in the further chapters to fulfill the above mentioned shortcomings.