CHAPTER 6

DESIGN OF TEXT RETRIEVAL SYSTEM

Document retrieval is defined as the matching of some stated user query against a set of free-text records. These records could be any type of mainly unstructured text, such as newspaper articles, real estate records or paragraphs in a manual. User queries can range from multi-sentence full descriptions of an information need to a few words. Text retrieval is a critical area of study today, since it is the fundamental basis of all internet search engines. A typical document retrieval system consists of two main components, namely, a database of documents and a classification algorithm. The input to the system is the query words, and the output is the set of relevant document that is closely similar to the input query (https://en.wikipedia.org/wiki/Document_retrieval). Thus, a document retrieval system has two main tasks as listed below.

1. Find relevant documents to user queries.
2. Sort them according to their relevance to the user query.

The most famous classical application of document retrieval is the Internet search engines. Other example applications include digital library and recommender systems. A general architecture of the document retrieval systems used by these applications is given in Figure 6.1.

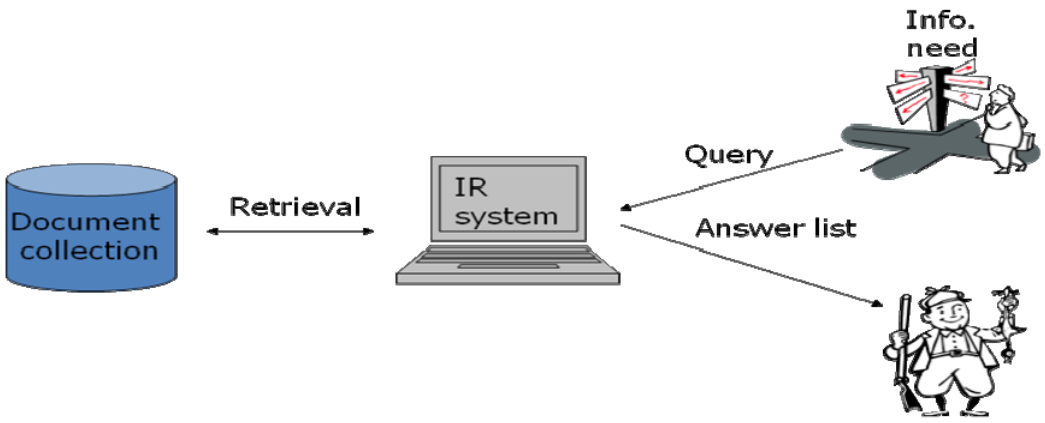

Figure 6.1 : General Architecture of a Document Retrieval System

A document collection consists of many documents containing information on various subjects or topics of interests. The user request in query form is presented to the IR system to obtain the relevant documents. In order to improve the process of document retrieval, three steps are used in Phase III. They are,

(i) Query Expansion
(ii) Document Retrieval
(iii) Ranking

This chapter presents details of the algorithms used in these steps.

6.1. QUERY EXPANSION

Query expansion (QE) is the process of reformulating a seed query to improve retrieval performance in information retrieval operations. QE techniques are widely applied for improving the efficiency of the textual information retrieval systems. These techniques help to overcome vocabulary mismatch issues, by expanding the original query with additional relevant terms, and reweighting the terms in the expanded query.

Relevance feedback is a feature of IR systems, where the results returned by a query are used to obtain information on the relevancy of the retrieved documents, using which a new query is formed. There are three types of relevance feedback techniques. They are explicit feedback, implicit feedback and blind or pseudo feedback. Explicit feedback from assessors (or users) in the form of a binary or graded value indicate the relevance of the document retrieved to the query. Implicit feedback is inferred from user behavior, such as noting which documents they do or do not select for viewing, the duration of time spent viewing a document or page browsing or scrolling actions.

Pseudo relevance automates the manual part of relevance feedback, so that the user gets improved retrieval performance without an extended interaction. This research uses this type for performing QE. The method is a well-studied query expansion technique (Efthimiadis, 1996) and performs normal retrieval to obtain an initial set of most relevant documents. From the retrieved documents, the algorithm then assumes that the top “k” ranked documents are relevant and finally does relevance feedback as before under this assumption. The general procedure of

pseudo relevance feedback technique is given below (https://en.wikipedia.org/wiki/Relevance_feedback).

(i) The top k results returned by initial query are taken as relevant results (k = 10 to 50).

(ii) Top 20-30 (indicative number) terms are selected from these documents using for instance Term Frequency-Inverse Document Frequency (TF-IDF) weights.

(iii) Query Expansion is performed by adding these terms to query and perform retrieval again to retrieve the most relevant documents.

Thus, a Pseudo-Relevance Feedback (PRF) QE assumes that the top ranking n (>0) documents of the initial retrieval are relevant and extracts expansion terms from them.

In general, however, the top retrieved documents contain noise. For example, if the precision of the top 10 documents (P@10) is 0.5, it indicates that five of them are non-relevant. This is common and even expected in all retrieval models. When combined with pseudo-relevance feedback, this noise, however, can cause the query representation to "drift" away from the original query. To solve this problem, Lee and Croft (2013) proposed a deterministic resampling method using overlapping document clusters. This research work uses this method to perform QE. The algorithm consists of the steps presented in Figure 6.2.

The study, during experiments, uses three types of queries, namely, short term title queries, descriptive title queries and narrative title queries. Out of this, the QE technique is used only for short term title queries as these queries contains only 2 or 3 words, which do not provide sufficient indications for an effective selection of relevant documents, and thus negatively affect the performance of web search in terms of both precision and recall (Kunpeng *et al.*, 2009).

- Deterministically constructing a sample space by selecting top-ranked N documents for each query based on language model from the collection of documents.
- Deterministically constructing sample units by k-NN clustering based on the similarities of documents in the sample space.
- Deterministically sampling clusters by selecting top-ranked M clusters based on the cluster-based language model. All the documents in the top M clusters are selected as feedback documents with redundancy which means one document can be selected more than twice.
- Deterministically sampling expansion terms by selecting top-ranked E terms based on the relevance model.

Figure 6.2: QE Algorithm

6.2. DOCUMENT RETREIVAL

In this research work, a machine learning classification approach is used to perform information (text) retrieval from the database. The applicability of an automatic classification technique, to retrieval of documents from text corpora, is an area of research that is very active, where methods to improve classification performance are investigated. For this purpose, two classification algorithms, namely, K-Nearest Neighbor (KNN) and Associative Rule-based Classifier (ARC) are used in Phase III. The proposed document retrieval system first improves these two classifiers, and then combines them to form a hybrid system. This hybrid system is then used to design an ensemble system, which is used to retrieve the relevant documents. This section presents details regarding the proposed document retrieval system, which is termed as EHAKNN (Ensemble Hybrid ARC and improved KNN) classifier. The design of EHAKNN consists of four main steps as listed below.

(i) Improving KNN classification algorithm

(ii) Improving ARC classification algorithm

(iii) Constructing the ensemble hybrid classification algorithm

6.2.1. Improved KNN Classification

The first classifier considered for text retrieval is based on the K-Nearest Neighbor Classifier (Cover and Hart, 2006), which has the advantage of achieving consistently high performance, without a priori assumptions about the distributions from which the training examples are drawn. The KNN classifier considers the K-Nearest points of a data point, and assigning the sign of the majority. It is common to select k small and odd to break ties (typically 1, 3 or 5). Larger K values help to reduce the effects of noisy points within the training data set and the choice of k is often performed through cross-validation. It is a non-parametric classification model, where the training dataset is used to classify each member of a "target" dataset. The process of conventional KNN classifier is described below.

Let Q be the query. The problem here is to find the K-Nearest Neighbors among the training documents and rank documents similar to the query according to a similarity score, which is estimated from its K-Nearest neighbor. If more than one KNN documents have the same similarity score, then the sum of the score is taken as the similarity score of all these categories with regard to x. The candidate category with the highest similarity score is considered as the category which is matched with the query. The decision rule of KNN is described using Equation (6.1)

$$f(x) = \arg\max score(x, C_j) = \sum_{d_i \in KNN} sim(x, d_i)\, y(d_i, C_j) \qquad (6.1)$$

where f(x) is the label assigned to the test document x; Score(x, C_j) is the score of the candidate category C_j with respect to x; sim (x, d_i) is the similarity between x and the training document d_i; y(d_i, C_j) $\in$ {0, 1} is the binary category value of the training document d_i with respect to C_j (y = 1 indicates document d_i is part of category C_j, else, y = 0). The best choice of k depends upon the data; generally, larger values of k reduce the effect of noise on the classification, but make boundaries between classes less distinct. In experiments, a value of 3 was set to 'k' (k=3).

The KNN classification algorithm is one of the simple and effective methods used for text classification and document retrieval (Bijalwan *et al.*, 2014). However, the algorithm has issues, which when handled correctly, can help to improve CLTR process (Jiang *et al.*, 2012).

The first issue is regarding the complexity involving similarity computations, which is very high. This makes the classification time high. The second issue is that its performance is affected by single training data, such as noisy data. The third issue is that as KNN is a lazy learning method and does not build classification model, it makes unsuitable for dynamic online applications.

Several researchers have analyzed techniques and offer solutions to these issues by using (i) Dimensionality reduction methods to reduce size of document feature vector (Vries *et al.*, 2002) (ii) Methods to reduce the size of training samples (Li and Hu, 2004), and (iii) Methods to speed up the process of finding K-Nearest neighbors (Aghbari, 2005; Yu and ZhengOu, 2007). This research work uses an Improved KNN (IKNN) algorithm for document classification, which is then used to retrieve relevant documents. This model combines KNN classification with a constrained one pass clustering algorithm (Jiang, 2006), based on least distance principle, to partition the training data into regions with the same radius (r). The results are then used by conventional KNN classifier to classify the documents. The steps in IKNN algorithm are presented in Figure 6.3 and are described in the following paragraph.

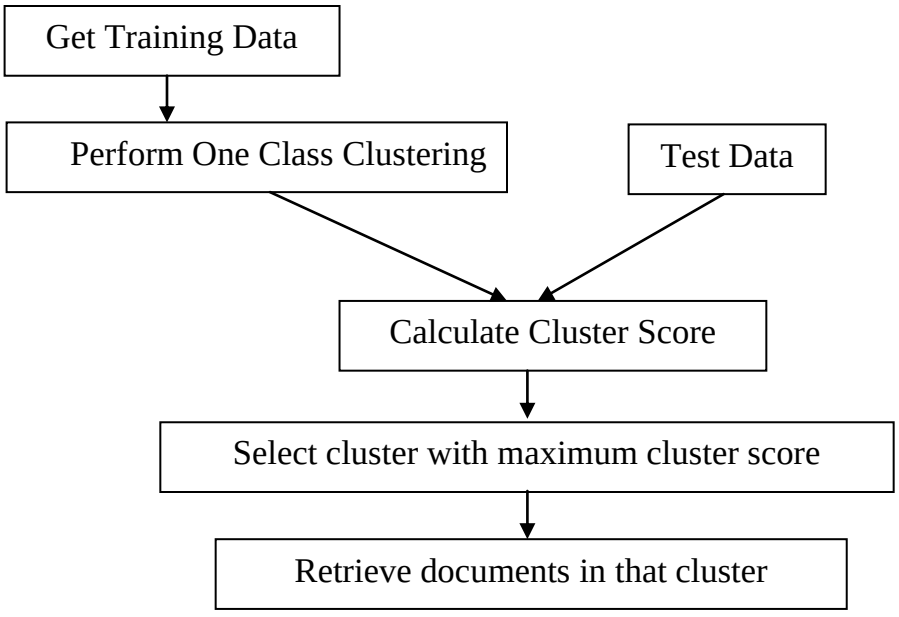

Figure 6.3: Improved KNN Classification Algorithm

In order to improve the speed of classification, the proposed IKNN algorithm works in two major stages. The first stage uses a clustering algorithm to group similar text objects, and the second stage uses the KNN classifier to identify the cluster which is close to the query.

- **Stage 1 : Single Pass Clustering**

Clustering is defined as a data mining (machine learning) technique that is used to place data elements into related groups, without advance knowledge of the group definitions (or classes). The grouping is formed in such a way that the objects in the same group (called a cluster) are more similar to each other than to those in other groups (clusters).

In general scenario, the clustering algorithms assume a sufficiently large number of data points, but in most of the available text (document) corpus, the dimensionality (that is, the number of distinct terms) is significantly larger than the number of documents to be clustered (Ertoz *et al.*, 2003; Cormack and Lynam, 2005; Keerthi and DeCoste, 2005; Yi *et al.*, 2014). At the same time, many terms in a cluster occur only in one or two documents, thus giving them a very low frequency (Term Frequency) weight in the cluster center (Sculley, 2010). In these situations, a one-pass (single pass) clustering algorithms are more useful. This is in contrast to several other clustering algorithms like K-Means (Lloyd,1982) and mixture models (Lindsey, 1996), which require going through the data set multiple times before the final centers can be determined.

Single pass clustering algorithms have been adopted by several authors (Yeom *et al.*, 2008; Forestiero *et al.*, 2013; Gnatyshak, 2015) for grouping by examining the data stream only once. According to this paradigm, as data is scanned and summaries of past data are stored to leave enough memory for processing new incoming data. These algorithms use a divide-and-conquer technique that partitions the data stream into disjoint clusters by extending the k-Median algorithm.

In IKNN, the single pass clustering is used as a pre-processing technique to discover the distribution and structure of corpus, and also to build the classification model with training documents based on the clustering result. Usage of single pass clustering provides multiple

advantages like (i) Less time consuming (ii) Non-iterative, and (iii) Requires only one scan of the corpus. The single pass clustering algorithm used is given in Figure 6.4.

In this algorithm, during clustering, each cluster is represented using a cluster vector in harmony with centroid vector for each cluster. The updation of text weights of the clusters (Step 5) is performed using Equation (6.2) (Jiang *et al.*, 2012).

$$w_{C^{\#}}^{i+1}(t) = \frac{w_{C^{\#}}^{i}(t) \times |C^{\#}| + w(t)_{New_t}}{|C^{\#}| + 1} \qquad (6.2)$$

where $w_{C^{\#}}^{i+1}(t)$ is the new updated weight of the text 't' in cluster $C^{\#}$, $w_{C^{\#}}^{i}(t)$ is the weight of word 't' in cluster $C^{\#}$, $w(t)_{New_t}$ is the weight of word 't' in text New_t and $C^{\#}$ is the number of words (texts) in cluster $C^{\#}$.

1. Initialize the set of clusters, C, as the empty set and read a new text 'New_t'.
2. Create a new cluster with the New_t; its label is regarded as the label of the new cluster.
3. If no texts are left in the text collections, go to Step 6, otherwise read next new text New_t, compute the similarities between N_t and all the clusters C_i in C (I = 1..N, N is number of clusters) using cosine similarity measure and find the cluster, $C^{\#}$, that is closest to the text New_t.
4. If similarity is less than a threshold 'T', then group text to that cluster else go to Step 2.
5. Merge text New_t into cluster $C^{\#}$ and update the weight of words of cluster $C^{\#}$; go to Step 3.
6. Stop clustering, get the clustering results, C = {C_1, …, C_N}, where C_i consists of the weighted text and cluster label and C is the classification model.

Figure 6.4 : Single Pass Clustering Algorithm

During clustering, the similarity measure used is the cosine similarity measure (Step 3). When documents are represented as term vectors, the similarity of two documents corresponds to the correlation between the vectors. This is quantified as the cosine of the angle between vectors, that is, the so-called cosine similarity. Cosine similarity is one of the most popular similarity measures applied to text documents, such as in numerous information retrieval applications (Yates and Neto, 1999) and clustering too (Larsen and Aone, 1999).

The cosine similarity between two documents is calculated using Equation (6.3).

$$SIM_C(t_a, t_b) = \frac{t_a . t_b}{|t_a| x |t_b|} \qquad (6.3)$$

Each dimension of the m-dimensional vectors, t_a and t_b, represents a term with its weight in the document, which is non-negative. As a result, the cosine similarity is non-negative and bounded between [0,1]. The threshold, T in Step 4 is automatically estimated using the procedure in Figure 6.5.

Step 1 :	Choose random pairs of texts in the corpus (N).
Step 2 :	Compute the similarities between each pair of texts.
Step 3 :	Compute the Average of Similarities (AoS) from Step 2.
Step 4 :	Set T = ε x AoS

Figure 6.5 : Automatic Threshold Estimation Procedure

In this procedure, N is the number of random pairs selected and is set to 8000. This value was obtained after several experiments, which showed that as N becomes high, the threshold value estimated is stable. The threshold value obtained using this procedure is dependent on the text corpus size and the proposed IKNN gives efficient results when ε = 3.

- **Stage 2 : Text Categorization**

The second stage performs the actual classification, using the KNN classifier. The operation is based on comparing a given new record with training records, and finding training

records that are similar to it. It searches the space for the k training records that are nearest to the new record as the new record neighbors. Here, nearest is defined in terms of a cosine metric.

The K-nearest neighbor classifier is a supervised learning algorithm, where the result of a new instance query is classified, based on majority of the K-nearest neighbor category. The training samples are described by n-dimensional numeric attributes. Each sample represents a point in an n-dimensional pattern space. In this way, all of the training samples are stored in an n-dimensional pattern space. All training samples are included as nearest neighbors, if the distance of this training sample to the query is less than or equal to the K^{th} smallest distance. In other words, the distances are sorted of all training samples to the query and determine the K^{th} minimum distance. The unknown sample is assigned the most common class among its K-Nearest neighbors. Thus, the KNN algorithm works based on minimum distance from the query instance to the training samples to determine the nearest neighbors. After gathering K-Nearest neighbors, the majority of these KNN are taken to be the prediction of the query instance. The steps are consolidated in Figure 6.6.

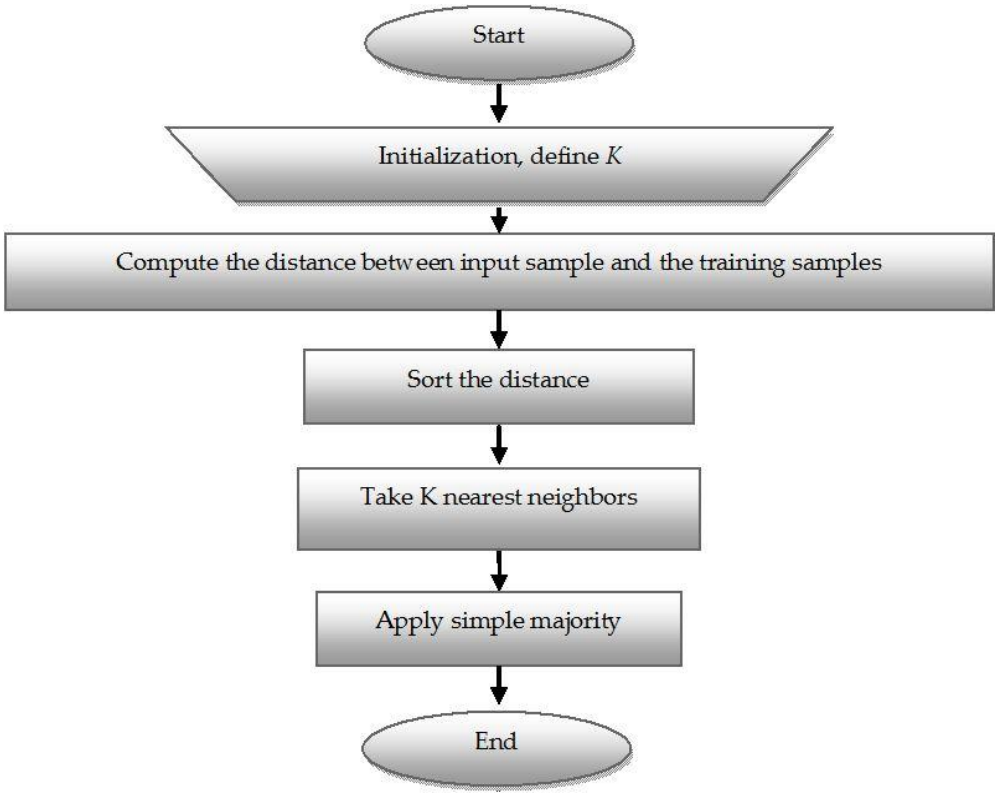

Figure 6.6 : KNN Classification

The testing process is performed, using the KNN classifier modified, to use the results from the above described single pass clustering algorithm. Given a test query 'D_t', a Cluster Score (CS) is estimated (Equation 6.4) for each cluster C_i in C.

$$F(D_t) = \arg\max CS(D_t, C_j) = \sum_{C_{i \in KNN}} sim(D_t, C_i), y(C_i, C_j) \quad (6.4)$$

where $f(D_t)$is the label assigned to the test query D_t, $CS(D_t, C_j)$ is the score of the candidate category C_j with respect to D_t, $sim(D_t, C_i)$ is the cosine similarity between D_t and the cluster C_i in model C, $y(C_i, C_j) \in \{0; 1\}$ is the cluster C_i with respect to C_j. Here, $y = 1$ indicates cluster C_i is part of class C_j else $y = 0$.

The proposed IKNN model thus requires only one scan for clustering and produces clusters, whose size is less than the whole training sample. The clustering algorithm, when integrated with KNN classifier, reduces the number of similarity computations required thus reducing the time requirement for document retrieval.

6.2.2. Improved ARC Algorithm

Constructing fast and accurate classifiers for document classification is an important task in text retrieval. There is growing evidence that, merging classification paradigm with association rule mining, can produce more efficient and accurate classification systems than traditional classification techniques (Liu *et al.,* 1998). Associative Rule Classifier (ARC) is an area of research that integrates Association Rule Mining (ARM) and classification (Nofal and Ahmad, 2010). Several proposals have proved that the classification based on ARM is a high potential approach, that constructs more predictive and accurate classification systems than traditional classification methods like decision trees (Quinlan, 1993; Yin and Han, 2003). Moreover, many of the rules found by associative classification methods cannot be found by traditional classification techniques.

The ARC systems first generate a complete set of association rules, which reveal the relationship between the features of a database item and its correlating class labels, by extracting rules that are formed with only data attributes as the antecedent and only class labels as the consequent (Thabtah, 2007). During classification, a small set of high rules are selected for

predicting the correct class for an input query. Thus, the algorithmic approach of a text classifier based on association rules consists of two fundamental tasks, as listed below:-

(i) Association Rule Mining

(ii) Classification.

One major drawback in this conventional approach is the high number of association rules generated. To solve this problem, in this research work, a pruning algorithm is included and this algorithm is referred to as Improved Associative Rule-based Classifier (IARC) in this research work and the general steps involved are presented in Figure 6.7.

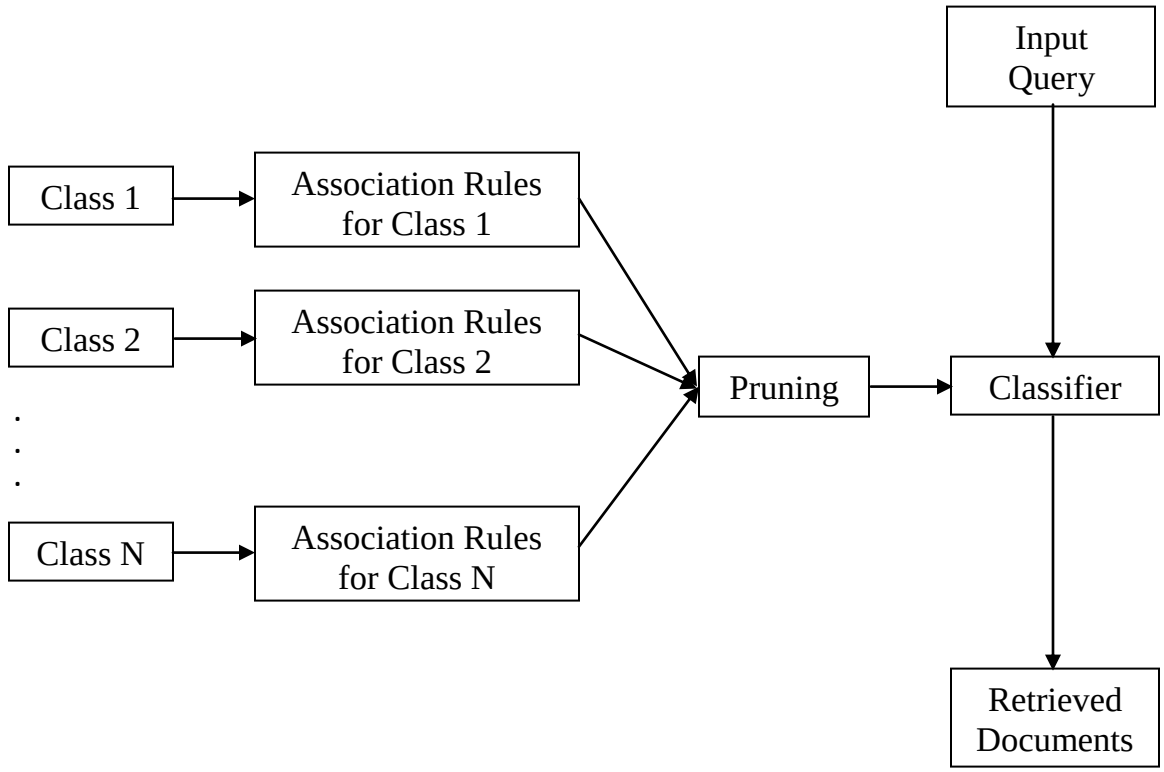

Figure 6.7: Steps in IARC Algorithm

- **Association Rule Mining**

ARM is one of the most important data mining problems (Thuan, 2010; Tseng, 2012) and is the art of discovering associative relationship among a set of items. Although ARM was initially introduced for market basket analysis (Agrawal *et al.*, 1993), it has also been proved to

be useful in many other applications like microarray data analysis, recommender systems, network intrusion and text categorization.

As mentioned earlier, the ARC operates by applying an ARM to obtain classification rules from a training set of previously classified data. The rules thus generated are influenced by the choice of two parameters, support and confidence threshold values. The support, s(X), of an association rule is defined as the ratio (in percent) of the records that contain X∪Y to the total number of records in the database, and is the statistical significance of an association rule (Equation 6.5).

$$s(X) = \frac{s(XUY)}{k} \tag{6.5}$$

where k is the total number of records in the database.

The second threshold, Confidence, is defined as the ratio (in per cent) of the number of records that contain X∪Y to the number of records that contain Y (Equation 6.6). The confidence indicates the degree of correlation in the dataset between X and Y and is a measure of a rule's strength.

$$C(x) = \frac{s(XUY)}{s(X)} \tag{6.6}$$

A low support rule normally denotes a relation that is unimportant or unprofitable transaction to an application. For this reason, support is often used to eliminate uninteresting or unimportant rules. Confidence, on the other hand, measures the reliability of the inference made by a rule. For a given rule $X \rightarrow Y$, the higher the confidence, the more likely it is for Y to be present in transactions that contain X. Confidence also provides an estimate of the conditional probability of Y given X.

In association rule mining problem, a natural approach is to compute support and confidence for every possible rule. This approach is very expensive, as the number of rules generated is very high. This problem is solved by de-coupling support and confidence. According to this solution, the support of a rule depends only on the support of its corresponding

itemset. If the itemset is not frequent, then all its candidate rules can be pruned immediately without having to compute their confidence values. This observation has made several association rule mining algorithms to follow a two step processes, frequent itemset generation step and rule generation step, as given in Figure 6.8.

- Find all sets of items which occur with a frequency that is greater than or equal to the user-specified threshold support, s.
- Generate the desired rules using the large itemsets, which have user-specified threshold confidence, α.

Figure 6.8 : Basic Association Mining Algorithm

Generally, the computational requirements for frequent itemsets generation are more expensive than those of rules generation (Li *et al.*, 2007; Kumar *et al.*, 2004). Therefore, efficient methods or algorithms are needed for generating frequent itemset and association rules. This research work proposes the use of Apriori algorithm for this purpose.

- **Apriori Algorithm**

The Apriori algorithm was initially developed for data mining and market basket analysis applications (Prathima, 2009). It has been widely used by researchers to mine information from very large databases, and this research work uses this algorithm as a method to find information from a collection of indexed documents by automatically extracting the association rules from them. The steps required to perform Apriori algorithm are given in Figure 6.9.

The Apriori algorithm, as the name of the algorithm suggests, uses prior knowledge for discovering frequent itemset in the database. The algorithm employs an iterative search and uses k-itemsets discovered to find (k+1)-itemsets. The frequent itemsets are those that have the support of higher than a minimum threshold. The idea of using Apriori behind text classification is to discover strong rules that are associated with the class labels.

C_k: Candidate itemset of size k
L_k: frequent itemset of size k

L_1= {frequent items};
for(k= 1; L_k != ϕ; k++) do begin
    C_{k+1}= candidates generated from L_k;
    for each transaction tin database do
        increment the count of all candidates in C_{k+1} that are contained in t
    L_{k+1}= candidates in C_{k+1} with min_support
end
return $\cup_k L_k$;

Figure 6.9. Apriori Algorithm

The next step is to take advantage of these patterns such that a classifier is built and the correct class for an input query can be found accurately. For this purpose, the Associative Rule Classification-By Category (ARC-BC) algorithm proposed by Antonie and Zaiane (2002) is used. The algorithm is presented in Figure 6.10. In ARC-BC model, each set of documents belonging to one category is considered as a separate text collection, to generate association rules from. If a document belongs to more than one category, then this document will be present in each set associated with the categories that the document falls into.

Step 2 of this algorithm generates the frequent 1-itemset. In steps (3-13) all the k-frequent item sets are generated and merged with the category in C_1. Steps (16-18) generate the association rules. The document space is reduced during each an iteration, by eliminating the transactions that do not contain any of the frequent item sets. This step is done by Filterable (D_{i-1}, F_{i-1}) function. During the training phase, the set of association rules are generated using this algorithm and when a query is given, the classification process searches the rule set for finding a class that is close to this input query. One solution is to relate an input query to a class which matches the rules generated mostly. This, apart from being time consuming, will also neglect the problem of multi-class. The problem of multi-class is defined as a situation, when more than two mutually exclusive classes exist, to which documents may be assigned, for example, query having famous personality, involved in both entertainment and sports. A solution to this problem is presented in Figure 6.11.

Input: A set of documents (D) of the form D_i :{ C_i, t_1, t_2, ….t_n} where C_i is the category attached to the document and t_j are the selected terms for the document, minimum support and minimum confidence

Output: A set of association rules of the form $\{t_1, t_2\} \rightarrow C_i$ where C_i is the category and t_j is a term

(1) $C_1 \leftarrow$ {Candidates 1 term-sets and their support}
(2) $F_1 \leftarrow$ {Frequent 1 term-sets and their support}
(3) for ($i \leftarrow 2$; $F_{i-1} \neq \emptyset$; $i \leftarrow i+1$) do {
(4) $C_i \leftarrow (F_{i-1} \bowtie F_{i-1})$
(5) $C_i \leftarrow C_i - \{ C \mid (i-1)$ item-set of $c \notin F_{i-1} \}$
(6) $D_i \leftarrow$ FilterTable(D_i - 1, F_{i-1})
(7) for each document d in D_i do {
(8) for each c in C_i do {
(9) C.support $\leftarrow$ C.support + Count(c, d)
(10) }
(11) }
(12) $F_i \leftarrow \{ c \in C_i \mid$ C.support > }
(13) }
(14) Sets $\leftarrow$ U { $c \in F_i \mid i > 1$}
(15) R = Ø
(16) for each itemset I in Sets do {
(17) R $\leftarrow$ R + { I => Cat}
(18) }

Figure 6.10. ARC-BC Algorithm

- **Pruning Algorithm**

The major drawback while using ARC is the huge number of rules generated. To handle this issue, this research work proposes rule reduction technique based on pruning. The main aim here is to obtain a minimized set of association rules without decreasing the classification performance. For this purpose, the database coverage pruning algorithm is used.

Database coverage pruning technique is a well known technique that was proposed by Liu *et al.* (1999) and Li *et al.* (2001) to improve the process of association rule classification.

The algorithm guarantees that every classification rule can at least classify one training instance correctly.

Further, it also guarantees that each training instance is covered by the rule with the highest ranking (according to the interestingness measure) among those rules that can cover the instance. The term 'to cover' is described as "A rule 'r' covers an instance 'd' if 'd' satisfies all conditions of the rule body of 'r' ". Therefore, if a training instance is covered by a rule, it can still be classified erroneously by the rule. A training instance'd' is removed from the training set if it is covered by λ rules, that predict this instance correctly. The resulting set is a summary of high quality rules.

```
Algorithm     Finding the class to which closely matches the query
Input         Query o, ARC, Dominance factor δ, Confidence Γ
Output        Categories attached to the input query

(1)    S ← θ  /* Set of rules that match o */
(2)    for each rule r in ARC ( the sorted set of rules)
(3)    if ( r ⊂ o)  { Count ++}
(4)    if ( count = = 1)
(5)        fr.conf ← r.conf  / * keep the first rule confidence */
(6)        S ← S ∪ r
(7)     else if ( r.conf  > fr.conf  -  Γ )
(8)        S ← S ∪ r
(9)    else exit
(10)   divide  S in subsets by category : S1, S2, …Sn
(11)   for each  subset S1, S2, ……Sn
(12)       sum the confidences of rules and divide by
           the number of rules in Sk
(13)    if it is single classification
(14)       put the new document in the class that has
           the highest confidence sum
(15)    else /* multi-class classification*/
(16)       TakeKClasses( S, δ)
(17)       assign these k classes to the new query
```

Figure 6.11 : Classification Algorithm

The procedure of database coverage is presented in Figure 6.12, where R is the set of association rules generated by Support-Confidence based algorithm (SCP) algorithm and T is the training dataset. This method tests the generated rules against the training data set, and only high

quality rules that cover at least one training instance not considered by other higher ranked rules are kept for later classification.

```
For each rule r_i in R Do
 Find all applicable data instances in T that match r_i's condition
 If r_i correctly classifies a training instance in T
    Mark r_i as a candidate rule in the classifier
    Remove all instances in T covered by r_i
 end if
 If r_i cannot correctly cover any training instance in T
       Remove r_i from R
 end if
end for
```

Figure 6.12 : Database Coverage

When the expected frequencies and the observed frequencies are different, the hypothesis that they are correlated is rejected. This method has been used in associative classification to prune negatively correlated rules. For example, a test can be done on every discovered rule, such as $r : x \rightarrow c$, to find out whether the condition x is positively correlated with the class c. If the result of the test is larger than a particular constant, there is a strong indication that x and c of r are positively correlated, and therefore r will be stored as a candidate rule in the classifier. If the test result indicates negative correlation, r will not take any part in the later prediction and is discarded.

6.2.3. Ensemble model based on Hybrid Classifier

As per the objectives formulated, the main goal is to develop an ensemble model based on a hybrid scheme that combines the above improved KNN and ARC algorithms for text classification and retrieval. The ensemble classifier is proposed to further improve the process of document retrieval using classification, and is referred to as EHAKNN classifier in this research. The methodology behind the creation of the proposed ensemble classifier is presented in Figure 6.13.

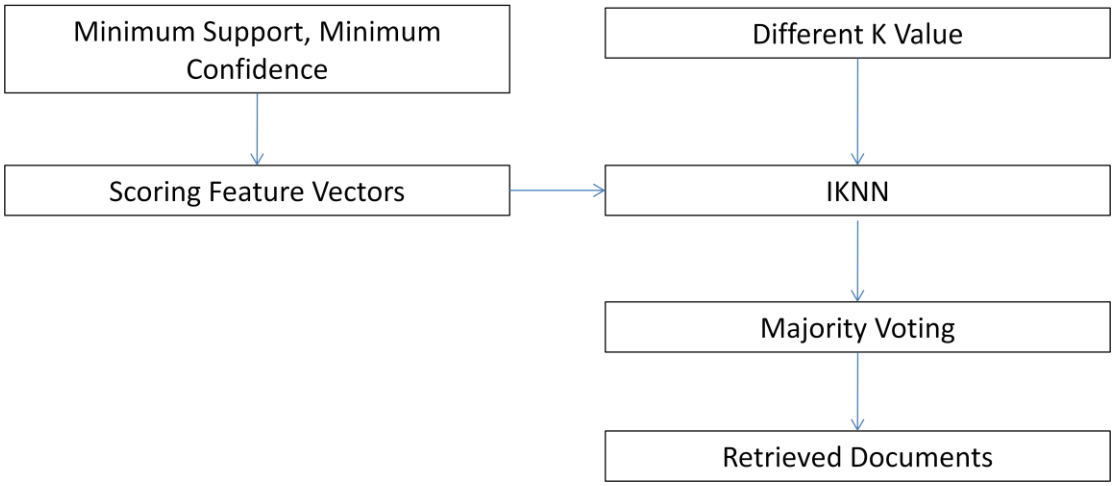

Figure 6.13 : Proposed EHAKNN Classifier

The proposed EHAKNN algorithm consists of two major steps, namely, creation of hybrid classifier and creation of ensemble model based on this hybrid classifier. In the hybrid classifier (referred to as HAKNN classifier), the score estimation of IKNN algorithm is replaced by a scoring scheme that uses the minimum support and minimum confidence thresholds of ARM. The result of the scoring scheme is treated as the weight for each term. A modified similarity measure, based on this weight, is used by IKNN during classification. The HAKNN classifier consists of two major stages as given below, and the steps performed in each of these stages are presented in Figure 6.14.

Stage 1: Calculate weights of terms using association rules

Stage 2: Classify document using IKNN algorithm

The modified weight and similarity measure is used in the procedure explained in Section 6.2.1 to classify and retrieve documents related to the given query. The weight calculated in Stage 1 is used in Equation (6.2) and the similarity measure in Equation (6.3) uses this weight measure.

Stage 1 : Weighted Term Vector Generation

Input : Minimum Support (Min_Supp), Minimum Confidence (Min_conf), Terms from each document

Output : Weighted Term Vector

Step 1 : Generate association rules using terms and its corresponding class details using IARM

Step 2 : Calculate support and confidence of each rule

Step 3 : Calculate Maximum Support (MS) and Maximum Confidence (MC) of each term

Step 4 : Repeat for all 'n' terms

```
if MS(i) < Min_Supp(i) or MC (i) < Min_Conf(i) then
    weight(i) = 0
else
    weight(i) = 1/(1-Max_Supp(i))
```

Stage 2 : Classification

Input : Term Weight Vector, K, Query

Output : Documents from Predicted Class

Step 1 : Apply IKNN Algorithm to predict the class

Step 2 : Retrieve Documents

Figure 6.14 : HAKNN Classifier

As mentioned earlier, the design of ensemble classifier consists of five factors, namely, feature vector, ensemble creation method, partition method, aggregation method and type of training. Different variants of weighted term vector (generated in Stage 1 of HAKNN) are created using different Min_supp and Min_conf values. The base classifier in the proposed ensemble system is created by varying the Min_supp (range 10 to 50 in steps of 10) and Min_conf thresholds. One major drawback in HAKNN is the choice of K which is user defined. This issue is solved again by using different K values (2, 3, 4 and 5). Thus, an ensemble system with 40 base classifiers is created by varying threshold values of ARM and K values of KNN

algorithm. Ten fold cross validation procedure is used as the partition method and majority voting scheme is used as the aggregation method. The training was performed on individual classifiers.

6.3. RANKING

Okapi Best Match (BM) 25, or BM25, is a weighting function used to rank documents according to their relevance to a given query (Robertson *et al.*, 1996). Many researchers apply the BM25 function in different corpus to retrieve relevant documents. BM25 is a probabilistic model, where the weight of a search term is assigned based on its frequency within the document and the frequency of the query term. The corresponding weighting function is defined using Equation (6.7).

$$w_i = SJ \frac{(k_1+1).freq_{id}}{k_1.[(1-b)+b.(dl/avdl)]+freq_{id}} . \frac{(k_3+1).freq_{iq}}{k_3+freq_{iq}} \qquad (6.7)$$

where $freq_{id}$ is the occurrence frequency of the term in the document (otherwise known as term frequency, tf), $freq_{iq}$ is the frequency of the term in the topic from which the query is derived, dl and avdl are, respectively, the document length and the average document length in the corpus. SJ is the Robertson Sparck Jones weight (Robertson and Jones, 1976), calculated using Equation (6.8), where 'R' is the number of documents relevant to a specific topic, 'r' is the number of relevant documents containing the term 'i', 'N' is the total number of documents in the collection and 'n' is the number of documents containing the term.

$$SJ = \log \frac{(r+0.5)/(R-r+0.5)}{(n-r+0.5)/(N-n-R+r+0.5)} \qquad (6.8)$$

In Equation (6.7), k_1, k_3 and b are user defined parameters, whose values should be adjusted based on the document collection and the type of queries (Maning *et al.*, 2008; Robertson and Zaragoza, 2009; Rivas *et al.*, 2014). In this research work, the FIRE dataset (details provided in Chapter 7) is used during experiments. After several experiments, the value of k_1 was set to 1.2 for short term title queries, 1.4 for descriptive title queries and 1.7 for narrative title queries.

Similarly, the value of k_3 was set to 1.2, 1.3 and 1.6 respectively. The parameter 'b' was set to 0.62.

6.4. CHAPTER SUMMARY

This chapter presented details regarding the algorithm used in Phase III of the research methodology. The algorithm proposed here is a classification based retrieval system, which used an ensemble hybrid classifier based on KNN and Associative Rule Mining technologies to classify the training documents and to identify the class which is close to the query words. The documents in the class identified were considered as relevant documents, which were retrieved and ranked using Okapi BM25 scoring method. The algorithms proposed in each phase of the research, which were designed with the aim of improving the overall performance of the proposed TECLTR-S and TECLTR-T systems, were evaluated using standard database and various performance metrics. The results of these experiments are presented in the next chapter, Chapter 7, **Results and Discussion**.