ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 5, Issue 7, July 2016

Relevance Feature Discovery for Text Mining by using Agglomerative Clustering and Hashing Technique

Bhavna B. Salve, Prof. N. V. Alone

***Abstract*— To guarantee the quality of discovered relevance features in text documents for describing user preferences is a big challenge because of large scale terms and patterns which faces the problem of synonymy or polysemy. To effectively use large scale patterns the new model, Relevance feature Discovery (RFD) discovers positive and negative and general patterns in text documents using clustering algorithm. RFD discovers three groups of patterns as high level features positive, negative and general and deploys them over low level features (terms). In this innovative model of RFD, we are using city-block Hashing from Locality Sensitive hash family and agglomerative clustering for classification of documents into the set of clusters of similar items. Allowing addition of new dataset dynamically makes the existing RFD model more flexible. By using the Locality sensitive hashing, the size of feature vector reduces reasonably and data mining processes like classification, clustering or association can run faster. Also the matching and discovering user preferences in discovered feature become very fast due to feat**ure hashing.

***Index Terms*— Block Hashing, Locality Sensitive Hashing, CPHC Classification algorithm, RFD, Tokenizer, PosTagging.**

I. INTRODUCTION

For describing user information needs or preferences in real world data automatic discovery of relevance features is a challenge in text mining. Over the years, there has been held the hypothesis that pattern-based methods should perform better than term-based ones in describing user preferences. But 'To effectively use large scale patterns' remains a hard problem in text mining. To make a breakthrough in this challenging issue, an innovative model of Relevance feature discovery discovers both positive and negative patterns in text documents as higher level features and deploys them over low-level features (terms). RFD also classifies terms into categories like positive, negative and general and updates term weights based on their specificity and their distributions in patterns. To find useful features available in text documents including both relevant and irrelevant ones for describing text mining results is the objective of Relevance Feature Discovery.

Salve Bhavna B., *Dept of Computer Engg., GES's Sapat College of Enggineering, Nashik, India.*
Prof. N. V. Alone, *Dept of Computer Engg, GES's Sapat College of Enggineering, Nashik, India.*

There are two challenging issues in using pattern mining technique for finding relevance features in both relevant and irrelevant documents. The First is low-support-problem, for given topic, long patterns are usually more specific for topic but they usually appear in documents with low support or frequency. If the minimum support is decreased, a lot of noisy pattern can be discovered. The second issue is misinterpretation problem, which means the measures(support, confidence) used in pattern mining turn out to be not suitable in using patterns for solving problems. For example, highly frequent pattern may be a general pattern since it can be frequently used in both relevant and irrelevant documents. Thus there is a problem of using discovered patterns to accurately weight the useful features.

There are several existing methods for solving two challenging issues in text mining, where for a given topic, term's specificity describes the extent to which the term focuses on topic that user want. However it is very difficult to measure the term's specificity because a term's specificity depends on users' perspective of their information need. In order to make breakthrough in these issues we proposed our model of RFD for text mining using Agglomerative clustering and Hashing Technique.

In this paper we continue to develop RFD model and try to prove that using the Locality Sensitive hashing, the size of feature vector reduces reasonably to find exact match of user preferences and text mining processes can run faster than early version of RFD model and the document-term classification can be effectively approximated by agglomerative clustering method using an novel (CPHC) classification by pattern-based Hierarchical classification algorithm.

- *Hashing :*

Feature hashing plays an important role in literature with increasing studies on big data, such as holding text as a data source. A usual way of text mining on big data mostly requires feature hashing, which reduces the size of feature vector. By using the feature hashing, the size of feature vector reduces reasonably and data mining processes like classification, clustering or association can run faster. Feature hashing has a major role in text mining studies. The text mining studies deal with big data same as social networks or web- mining. A generic deployment diagram of the text mining which uses feature hashing is demonstrated in *Figure1*.

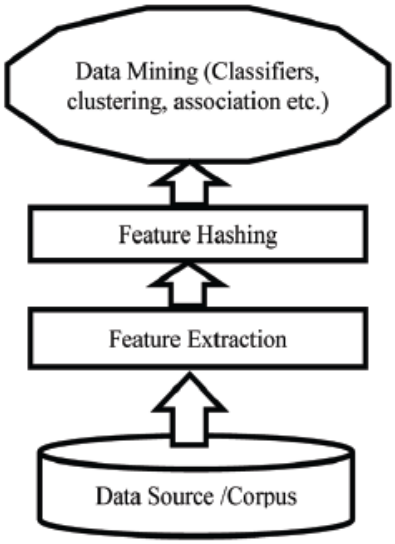


Fig. 1: Generic Deployment Diagram of Text mining

Approximate nearest neighbor (ANN) search based on hashing in huge databases has become popular because of its computational and memory efficiency. Semantic similarity is usually specified in terms of pairwise labels of samples. The supervised hashing methods can handle such semantic similarity, but they have threat of overfitting when labeled data are small or noisy. In this work, we are going to use a semi-supervised hashing (SSH) framework that reduces empirical error over the labeled set and an information theoretic regularizer over both labeled and unlabeled sets. Particularly, Locality Sensitive Hashing (LSH) is an algorithm for Solving Appropriate or exact nearest neighbor search in high dimensional space for semi-supervised hashing . It is a Randomized Algorithm, Hence provides high Probability Guarantee that it will return the correct answer for one close to it. LSH algorithm uses DOT product with random vectors to quickly find nearest neighbor and provides guarantee that it will return correct answer with improved computational performance. Therefore LSH is preferable for:

- Very large database, LSH is valuable technique for retrieving items (documents) that are similar to a query item.
- Fastest search =This technique drastically reduces computational time.

"Relevance Feature Discovery (RFD) for text mining"uses linear search which takes much time for searching. Therefore there are two solutions for this linear search is:

1.Tree based search
2.Hashing Technique.

In proposed approach, for a given query we wish to find documents as per user preferences from a very large dataset. We wish to guarantee good results with high probability(1-δ) that we return the nearest neighbor for any query point iteratively. Building a Hash table allows us to quickly map between a symbol (string) and a value (document/ term). We are going to use block hashing algorithm from LSH hash families. Block hashing gives good performance results amongst all LSH families like Euclidean Hash, Min Hash, Cosine Hash.

The general idea is to hash documents in such a way that similar documents are more likely to be hashed into the same bucket as opposed to dissimilar documents. The documents will each be hashed multiple times, with the intent of altering the probability that similar items will become candidates while dissimilar items do not. The key point is that only the documents which fall into the same bucket are considered as potentially similar. If two documents do not map to the same bucket in hashing then they are never considered as similar in this technique. Obviously, we want to minimize the number of dissimilar documents falling into the same bucket. At the same time, we hope that similar documents at least hash once into the same bucket.

- *Agglomerative Clustering*

Hierarchical clustering algorithms have also been designed in the context of text data streams. The main idea is to model the frequency of word-presence in documents with the use of a multi-poisson distribution. The parameters of this model are learned in order to assign documents to clusters. The method of agglomerative hierarchical clustering is particularly used for supporting variety of searching methods because it naturally creates a tree-like hierarchy which can be applied for the search process. In particular, the effectiveness of this method is in improving the search efficiency over a sequential scan.

The general concept of agglomerative clustering is to successively merge documents into clusters based on their similarity with one another. The main difference between these classes of methods is 'how this pairwise similarity is computed within different groups of documents'. For example, the similarity between a pair of groups can be computed as the best-case similarity, worst-case similarity, or average-case similarity between documents which are drawn from these pairs of groups. These features makes this clustering most suited for RFD model allowing dynamic insertion of new documents.

The process of performing agglomeration on the documents into successively higher levels of clusters creates a cluster hierarchy where the leaf nodes correspond to individual documents, and the internal nodes correspond to the merged groups of clusters. It can produce an ordering of the objects, which may be informative for data display. Smaller clusters are generated, which may be helpful for discovery. New data source can be easily added dynamically, which makes the model flexible.

II. LITERATURE SURVEY

Over the years, people have developed many term based techniques for ranking documents, information filtering and text classification. Text mining is the technique that helps users find useful information from a large amount of digital text documents on the Web or databases.. So we must focus on text mining techniques initially. "Text mining" by Ian H. Written [1] covers all the text mining information and Bag of words required for text mining.

"Mining Positive and Negative Patterns for Relevance Feature Discovery" [2] is a technique that discovers both positive and negative patterns in text documents as higher level features in order to accurately weight low-level features

(terms) based on their specificity and their distributions in the higher level features. This model is called as RFD1 model. For improvements over the RFD1, RFD2 model is proposed.

"Relevance Feature Discovery for text mining" [3] also refered as RFD2 model. This framework provides new definition for specificity function and used parameters to group terms into three categories: " positive, negative and general specific terms which are used for finding relevant and irrelevant documents.

"A Data Mining Framework for Relevance Feature Discovery" [4] This framework efficiently mines a training set, including relevant and non-relevant documents, for closed sequential patterns. It also introduce a new data mining technique, pattern cleaning, to refine the discovered patterns for describing the
user's topic.

"Deploying Approaches for Pattern Refinement in Text Mining" [5] Instead of the keyword-based approach which is typically used in this field, the pattern based model containing frequent sequential patterns is employed to perform the same concept of tasks. It propose two approaches based on the use of pattern deploying strategies.

Enhancing Text Clustering Using Concept-based Mining Model [6]. The process of calculating Conceptual term frequency (*ctf),Term frequency (tf), document frequency (df),* measures in a corpus is attained by the proposed algorithm which is called Concept-Based Analysis Algorithm. By doing so we cluster the web documents in an efficient way and the quality of the clusters

Fast Agglomerative Clustering for Rendering [7] This paper show that agglomerative clustering can be done efficiently even for very large data sets. It introduce a novel locally-ordered algorithm that is faster than traditional heap-based agglomerative clustering

A Survey of Text clustering Algorithms [8] provide a detailed survey of the problem of text clustering, which study the key challenges, advantages and key methods of the clustering problem.

Hash-Based Approach to Data Mining [9] focuses on the hash-based method to improve performance of finding association rules in the transaction databases and use the PHS (perfect hashing and data shrinking) algorithm to build a system

Locality-Sensitive Hashing Scheme Based on p-Stable Distributions [10] presents a novel Locality-Sensitive Hashing scheme for the Approximate Nearest Neighbor Problem under lp norm, based on pstable distributions to improves the running time of the earlier algorithm Unlike earlier schemes, LSH scheme works directly on points in the Euclidean space without embeddings.

Semi-Supervised Hashing for Large-Scale Search [11] propose a semi-supervised hashing (SSH) framework that minimizes empirical error over the labeled set and an information theoretic regularize over both labeled and unlabeled sets. This framework presents three different semi-supervised hashing methods

A Novel Feature Hashing for Text Mining[12] proposes a new approach which is mainly built on the substitution boxes , which is in the core of all Feistel Networks and processes the text faster than the other implementations.

III. PROPOSED SYSTEM

The proposed system consist of three main parts first Training part, where we are going to tokenize the documents for stop words removal then we will parse the document for finding frequent terms using minimum support count and their frequencies using pos tagging from each document in XML file. In second part which is hashing part, we are going to calculate hash values for each relevant documents and store them after indexing in term-document matrix. In third part, we are going to classify the documents based on their semantic similarity and performing agglomerative clustering. Finally we are going to test proposed system over different sizes of dataset from Reuters 21578 which will help us in analysis of the system. The Figure 2 shows the system architecture of RFD using Hashing and agglomerative clustering where Text file of XML documents is taken as input to the system. Then that documents are tokenized and frequent item set with their frequencies is calculated using postagging. Block hashing the termset with their frequencies and documentId we can calculate the final hash value of all the documents. The resulting Documents-term matrix with their hash values is used for performing Agglomerative clustering. Finally we are having all the documents classified such that all similar documents are grouped into different clusters.

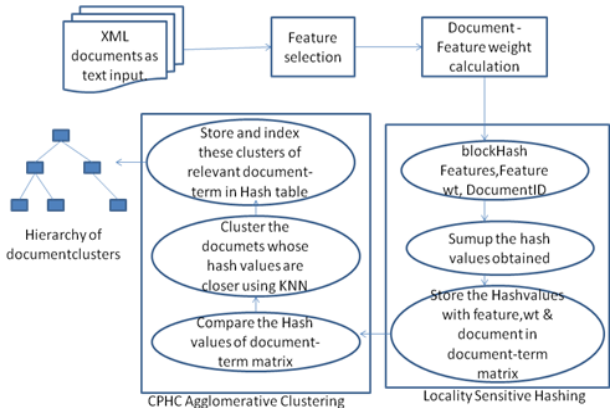


Figure 2: System Architecture of RFD by using Agglomerative clustering and Hashing technique.

Mathematical model

Let S be a system

S = {I, T, P, O, H, C}

Where,
I = {D+,D-} , set of relevant as well as non relevant documents in training set D.

T = Set of extracted features from input text documents.

P = Set of frequent item set following minimum support using pos tagging.

O = Extracted features (T+, G, T-) with dozens of

characteristics mapped to relatively small index.

H = {Hft, Hf, Hdoc}, set of hash values of frequent terms, frequency of that term, and document in which that term belongs.

C= (c1, c2,…, cn) set of clusters in which similar type of documents are grouped together under similar HashValues applying KNN.

Rules = Let Fn(i) be the rule of i into T to give-

F1(I) → T.

F2(T) → P

F3(P) → O

F4(O) → H

F5(H) → C

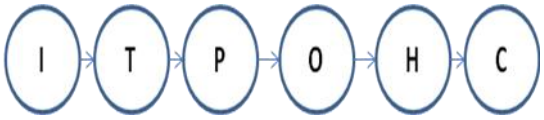


Figure 3. State transition diagram

All relations have one-to-one mapping.

- **Implementation:**

1. The system takes different set of documents from xml file as positive and negative input. Whenever the new data set is given it tokenizes the document then performs feature selection, feature extraction using postagging.

Let SP1, SP2;….; SP|D+| be the sets of discovered closed sequential patterns for all documents di ϵ D+(i = 1; . . . ; n), where n = |D+|. For a given term t, its d_support (deploying support, called weight) in discovered patterns can be described as follows:

$$d\_sup(t, D^+) = \sum_{i=1}^{n} sup_i(t) = \sum_{i=1}^{n} \frac{|\{p|p \in SP_i, t \in p\}|}{\sum_{p \in SP_i} |p|},$$

where |p| is the number of terms in p.

Once the weights are calculated we can classify the terms as positive terms (T+), negative terms (T-)and General terms G as follows.

$$w(t) = \begin{cases} d\_sup(t, D^+)(1 + spe(t)) & t \in T^+ \\ d\_sup(t, D^+) & t \in G \\ d\_sup(t, D^+)(1 - |spe(t)|) & t \in T_1 \\ -d\_sup(t, D^-)(1 + |spe(t)|) & \text{otherwise,} \end{cases}$$

After the deploying supports of terms have been computed from the training set, let w(t) = d_sup(t, D+), the following rank function is used to decide the relevance of document d:

$$rank(d) = \sum_{t \in T} w(t)\tau(t, d),$$

where τ(t, d+) = 1 if t ϵ d; otherwise τ(t, d+) = 0.

2. Once we get the frequent termset with their frequencies, we are going to perform block hashing from Locality Sensitive hash family. The hash functions h(.) from the LSH family satisfy the following elegant locality preserving property:

$$P\{h(\mathbf{x}) = h(\mathbf{y})\} = sim(\mathbf{x}, \mathbf{y}),$$

where the similarity measure can be directly linked to the distance function d, for example,

$$sim(\mathbf{x}, \mathbf{y}) = \exp\{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\sigma^2}\}$$

A typical category of LSH functions consists of random projections and thresholds as:

$$h(x) = sign(\mathbf{w}^\top \mathbf{x} + b),$$

where w is a random hyperplane and b is a random intercept. Clearly, the random vector w is data independent, which is usually constructed by sampling each component of w randomly from a p-stable distributionThe Self-tuning indexing technique, called LSH aims at improving the performance without additional storage and query overhead.

- **Block Hashing:**

In Block hashing the input file is divided into non-overlapped N number of blocks in which N is block number equal to length of the final hash bit string. For example in our approach we devide xml file into different documents, thus each document is one block.

Let us assume that now we have computed document-term matrix for the set of documents in previous section. We are going to partition the rows of this matrix into *b* = *n/r* bands, where each band is made of *r*-consecutive rows. We'll assume that *r* divides *n*. For each band we define a hash function *h* : R^r → Z, which takes a column vector of length *r* and maps it to an integer (i.e.a bucket). If we want we can even choose the same hash function for all the bands, but the buckets are kept distinct for each band. Now if two vectors of length *r* in any one of the bands hash to the same bucket, we declare that the corresponding documents are potentially similar.

- **Agglomerative Clustering :**

Then the process of agglomerating documents into successively higher levels of clusters creates a cluster hierarchy (or dendogram) for which the leaf nodes correspond to individual documents, and the intere nal nodes correspond to the merged groups of clusters. When two groups are merged, a new node is created in this tree corresponding to this larger merged group. The two children of this node correspond to the two groups of documents which have been merged to it.

In our example now as we are having all the

documents with their hash values, while classifying test documents we are going to compare the hash values of test documents with hash value of clustered one. Applying KNN we will find that the hash value of the document matches with which cluster's hash value. Finally the document is inserted into the cluster whose hash value is closer to the hash value of the test document. For performing this classification we are going to use Classification by Pattern based Hierarchical clustering (CPHC) algorithm.

IV. ALGORITHM

In Proposed System three algorithms are used for classifying the documents as per the relevant features. Those are featureSelection() for filtering out the relevant and irrelevant terms, BlockHAshing() for hashing document-tern with termfrequencies, CPHC() for classifying the training set into cluster Hierarchy. The algorithm for relevance Feature Discovery in text mining using Hashing and Agglomerative Clustering is as follows-

Let D be a dataset, $I = \{i1, i2, i3, \ldots, in\}$ be the complete set of distinct items (i.e.,binary attributes) in D, and C = $\{c1, c2, c3, \ldots, cm\}$ be the complete set of distinct class labels. An instance X is denoted as a triple $<id, L, Y>$ such that id is an identifier that uniquely identifies X, $L \subseteq C$ represents the set of class labels associated with X ($L=\Phi$ if X represents a test instance), and $Y \subseteq I$ represents the set of items in X. A pattern $P = \{p1, p2, p3, \ldots, pn\}$ is a subset of I. The set of data that contains P is denoted as $Dp = \{(id, L, Y) \in D | P \subseteq Y\}$. The *support* of a pattern P is defined as:

$$Support(P) = \frac{|D_P|}{|D|}$$

AlgoRFDUsingH&Aclustering()

1 : Select Frequent Features
Input : training instances trn1, trn2,…..,trnn
//positive & negative dataset
test instances tst1, tst2, …..,tstm
Apply AlgofeatureSelection()
Output : trn'1,trn'2,…...,trn'n and
tst'1,tst'2,…..,tst'm with reduced feature

2 : Hashing Documents and frequent features
Input : training instances trn'1,trn'2,….,trn'n
Test instances tst'1,tst'2,…..,tst'm
Apply AlgoBlockHAshing()
Output : Hash table of feature, frequencies and DocumentID Hash Values.

3 : Obtain a cluster Hierarchy of training and test instances.
Input : Hash table of features, their frequencies and documentIDs.
Apply AlgoCPHC()
Output : Cluster Hierarchy h.

AlgoFeatureSelection()

1 : Calculate j

$$j = \begin{cases} f & f < n \\ n + \left(n \times \log \frac{f}{n} \right) & otherwise \end{cases}$$

Where n = number of training instances(documents) f = total number of available features. This formula ensures reasonable base amount for low dimensional dataset while moderately growing this number for high dimensional dataset.

2 : Select Globally significant features
Select features that exist in more than minimum_support percent instances.Further refine these features by first sorting them in decreasing order of their information gain values.Then add the result top j features in S set of selected features.

3 : Ensure Local coverage of training instances.
For each instance X in the training set represented as triple $<id, L, Y>$, check if $|Y \cap S| \geq t$ (where t is user defined). If the condition is not met, sort all features in the current instance in the decreasing order of their (TF * *Information Gain*), where TF = local term frequency count in the instance. This "balances" the local significance (i.e., TF) and the global significance (i.e., *Information Gain*). Finally, add the resulting top (t - $|Y \cap S|$) features to set S.

4 : Ensure local coverage of test instance.
For each instance X in the test set represented as triple $<id, L, Y>$, check if $|Y \cap S| \geq t$. If the condition is not met, sort all features in the current instance in the decreasing order of their Term Frequency values. Finally, add the resulting top (t - $|Y \cap S|$) features to set S.

AlgoBlockHashing( )

1: Define Feature Vector V[i] and generate random projection

Once we find the S set of frequent feature we define a feature vector of random values of range W. The size of this vector will be 128. We convert the feature into its ASCII values and stores using random vector V[i] using random projection.

2: Apply Block- Hash function

Then for a complete block of vector we apply following Hash function. Therefore,

For i=0 to 128
{
Hash[i]= $\frac{V[i] - Random\ Values}{W}$
BlockHashValue = $\sum$ Hash[i]
}
HashValue[document-feature] = BlockHashValue / 128

Means here we are ging to find out hash value of Frequent feature, its frequency and the DocumentID. Finally to get a single hash value we are going to sum up the hash values of these feature, frequency, DocumentID.

AlgoCPHC( )

1 : Given a test instance *t*, and hierarchy *h*, first initialize scores for all classes. Next, traverse *h* from root to leaves, identifying the set *S* of nodes that contain *t*.

2: For each node *e* in *S*, compute *w* such that:

w = node-pattern-length * node-interestingness

3: For each class *c* represented by at least one training instance in *e* (considering all instances in the node as well as instances in all child nodes, as usual), add *x* to the score of *c* such that:

$$x = w \times \frac{|\text{training instances with label} = c \text{ in } e|}{|\text{training instances in } e|}$$

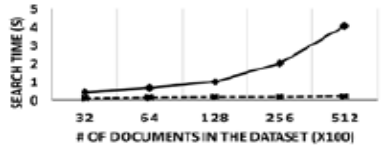
4: For single-label classification problems, select the label of the class with the highest score. For multi-label problems, select multiple classes using the "weighted dominant factor", except replacing all uses of confidence with the selected interestingness measure.

V. PERFORMANCE ANALYSIS

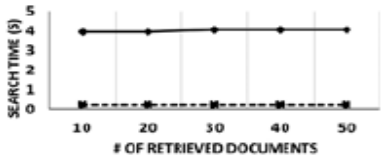
In order to test the performance of proposed system on real dataset we built an experimental platform to test search efficiency, search precision and accuracy. As we are focusing on Text mining, we are going to use Reuters-21578 data collection which is widely used collection for text mining. The data is originally collected and labeled by Carnegie Group, Inc. and Reuters, Ltd. In course of developing the CONSTRUE text categorization System. The documents are described in XML. All documents are treated as plaintext documents by preprocessing. Here n denotes the dictionary size, k denotes the number of top-k documents, m denotes the number of documents in the data set, and w denotes the number of keywords in the users query.

Fig. 4 is used to describe search efficiency with different conditions. Fig. 4a describes search efficiency using the different size of document set with unchanged dictionary size, number of retrieved documents and number of query keywords. In Fig. 4b, we adjust the value of k with unchanged dictionary size, document set size and number of query keywords.

From fig 4a we observe that with exponential growth of document set size search time of existing RFD system increases exponentially while Search time of Proposed System increases linearly. Fig. 4b shows that search time is stable with increase of retrieved documents.

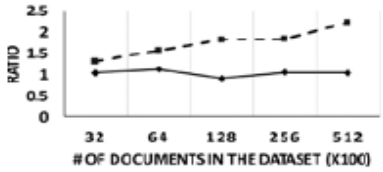


(a) Search time with the increasing documents

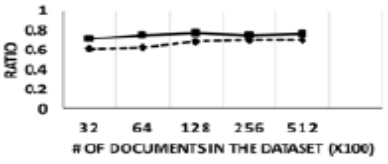


(b) Search time with the increasing number of retrieved documents

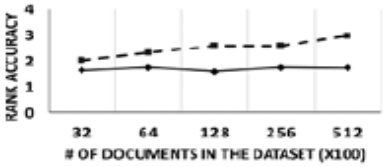
Figure 4. Search Efficiency



(a) Relevance of documents



(b) Relevance between documents and query



(c) Overall evaluation

Fig. 5: Search Precision and Accuracy.

Fig. 5 describes the search accuracy of over plaintext search. From fig. 5a we observe that relevance of retrieved documents in proposed system is almost twice as compared to existing RFD technique. Fig. 5b shows the relevance between query and documents improves as data size increases. Fig 5c shows rank accuracy according to equation-

$$Acc = aP_q + P_d,$$

Where,

$$P_q = \sum_{i=1}^{k'} S(qw, d_i) / \left(\sum_{i=1}^{k} S(qw, d_i) \right).$$

Defines the relevance between retrieved documents and query.

$$P_d = \sum_{j=1}^{k'} \sum_{i=1}^{k'} S(d_j, d_i) / \left(\sum_{j=1}^{k} \sum_{i=1}^{k} S(d_j, d_i) \right).$$

Defines relevance of different retrieved documents.

The trade off parameter a is set to 1, which means there is no bias towards relevance of documents or relevance between documents and query. From these results we conclude that RFD in text mining by using agglomerative clustering and Hashing technique is better than existing RFD model.

VI. Applicability

Concept relating to proposed RFD scheme is started, when a method for finding similar files from large file system was needed. Fingerprint matching typically requires comparison of several features of the fingerprint pattern.
Fingerprint matching falls into an application of RFD using Hashing. Image similarity can also be found using this approach. One of the obvious applications of similarity is in web search. However most search engines rely on information gathered from surrounding text on the web page along with any image meta-data. Therefore web search is a typical application of this scheme.

VII. Conclusion

This research proposes an alternative approach for relevance feature discovery in text mining. Here, we continued to develop the RFD model by applying Block hashing algorithm from Locality Sensitive Hash family. Applying Hashing allows us to quickly find nearest neighbor and provide guarantee that it will return correct answer with improved computational performance. This technique drastically reduces computational time. Building a Hash table allows us to quickly map between a symbol (string) and a value (document/ term). Well designed Hash function, separates two symbols that are closed together into different buckets. This helps to find exact matches efficiently.

The classification can be effectively approximated by a feature clustering method called agglomerative clustering. Using CPHC algorithm for clustering allows insertion of new dataset dynamically making the system flexible. This semi-supervised approach clusters both training and test sets and then uses these hierarchy as direct mean for classification. This eliminate need to train a classifier on training set. This method uses parameters that are robust across dataset across varying characteristics.

In the existing RFD algorithm feature matching part uses the most time ($O(|T|^2)$) for finding the initial value of T^-. The feature matching takes $O(|T|)$ time if a hash function is used for the containment test. This proves the effectiveness of proposed System

Acknowledgment

I take this opportunity to thank Prof. N.V.Alone , My guide and Head of the Computer Engineering department, for his encouragement and guidance and continuous help and enormous assistance. He helped me in a broad range of issues from giving me direction, helping to find solutions to problems, outlining requirements and always having the time to guide me. I also extend sincere thanks to our PG co-ordinator Prof. A.S. Vaidya and all the staff members for their valuable assistance. Last but not least I am very thankful to my class-mates for all their valuable suggestions and support.

REFERENCES

[1] Ian H. Witten " Text Mining", Computer Science, University of Waikato, Hamilton, New Zealand.

[2] Yuefeng Li, Abdulmohsen Algarni, Ning Zhong, "Mining Positive and Negative Patterns for Relevance Feature Discovery"ACM, KDD'10, July 25–28, 2010, Washington, DC, USA.

[3] Yuefeng Li, Abdulmohsen Algarni, Mubarak Albathan, Yan Shen, and Moch Arif Bijaksana, " Relevance Feature Discovery for text mining" IEEE transactions on knowledge and data engineering Vol. 27, No. 6, June 2015.

[4] Luepol Pipanmaekaporn "A Data Mining Framework for Relevance Feature Discovery" Queensland University of Technology.

[5] Sheng-Tang Wu Yuefeng Li Yue Xu, "Deploying Approaches for Pattern Refinement in Text Mining", IEEE Proceedings of the Sixth International Conference on Data Mining (ICDM'06)

[6] Lincy Liptha R.1, Raja K.2 , G.Tholkappia Arasu3, "Enhancing Text Clustering Using Concept-based Mining Model", International Journal of Electronics and Computer Science Engineering ISSN: 2277-1956,

[7] Bruce Walter, Kavita Bala, Milind Kulkarni, Keshav Pingali "Fast Agglomerative Clustering for Rendering",

[8] Charu C. Aggarwal, "A survey of Text Clustering Algorithms"

[9] Lê Kim Thu, "HASH-Based Approach to Data Mining",

[10] Wei Qiao, Hongdong Huaqing Liang, "Locality-Sensitive Hashing Scheme Based on p-Stable Distributions", International Conference on Measuring Technology and Mechatronics automation ,2009 IEEE

[11] Jun Wang, Member, IEEE, Sanjiv Kumar, Member, IEEE, and Shih-Fu Chang, Fellow, IEEE, "Semi-Supervised Hashing for Large-Scale Search", IEEE transactions on knowledge and data engineering Vol. 13, No. 12, December 2012.

[12] Cihan Mert, Sadi Evren Seker, "A Novel Feature Hashing for Text Mining" Journal of Technical Science and Technologies; ISSN 2298-0032

[13] Hassan H. Malik, John R. Kender,"Classification by Pattern based Hierarchical Clustering" , Columbia University.