AN EFFECTIVE CONCEPT BASED PATTERN DISCOVERY FOR TEXT MINING

Muthuvalli.A.R[1], Manikandan.M[2]

PG Scholar [1], Assistant Professor[2],Department of Computer Science and Engineering,
Adhiyamaan college of Engineering, Hosur (India)

ABSTRACT

Text Mining is the technique that helps users find useful information from a large amount of text documents on the web or database. Most of the existing text mining methods have adopted term-based approaches. This approaches are suffered from the problems of polysemy and synonymy. Over the years pattern based methods should perform better than term based methods. But pattern based methods also discovered two challenging issues like low-support-problem and misinterpretation problem. The proposed system uses clustering to make a breakthrough in this challenging issues. The proposed algorithm works for finding the frequent terms or words from the documents.

***Keywords*: Apriori algorithm, *Clustering, Frequent itemset,* Pattern mining,*Text mining*.**

I. INTRODUCTION

Data Mining is used to analyze the data from various perspectives and giving it in meaningful information. The meaningful data can be used for enhance the growth, profit, cut costs or both. In data mining, text mining is the discovery of interesting knowledge in text documents. Text mining is a progressively more significant research field since the requirement of attaining knowledge from the massive amount of text documents. In order to mine large document collections, it is vital to pre-process the text documents and save the data in a data structure, which is suitable for processing it further than a plain text file. Typically, text preprocessing includes, word stemming and the application of a stop words removal technique. The main goal of this paper is to finding the relevant documents by using the Apriori algorithm. This algorithm works in two steps. 1) Finding the support count value 2) Finding the threshold value. First, finding the support count value, if the minimum support value is decreased that value should be eliminated. Second, finding the threshold value, which means finding the average value and compared with the high value. Finally compared with all other document files and finding the relevant documents by using the level by level searching.

The remainder of the paper is organized as follows. In Section 2 shows the overview of the related work. In section 3 shows the methodologies. In Section 4 shows the efficient algorithm. In Section 5 shows the Experiment and Result. And Section 6 shows the conclusion.

II. RELATED WORK

Relevance Feature Discovery for Text Mining proposed FClustering and WFeature algorithms are used. Algorithm FClustering describes the process of feature clustering, where DP^+ is the set of discovered patterns of D^+ and DP^- is the set of discovered patterns of D^-. Algorithm WFeature is used to calculate term weights after terms are classified using Algorithm FClustering.

Effective Pattern Discovery for Text Mining proposed D-Pattern Mining Algorithm are used. It describes the training process of finding the set of d-patterns. The main focus of this paper is the deploying process, which consists of the d-pattern discovery and term support evaluation. It discovered all patterns in a positive document are composed into a d-pattern.

Mining Positive and Negative Patterns for Relevance Feature Discovery proposed two algorithms are used that is Mining and Revision algorithms. The algorithm calls twice one for positive documents and one for negative documents. The process of the revision firstly finds features in the positive documents in the training set, including higher level positive patterns and low-level terms. It then selects top-K negative samples in the training set according to the positive features. It also discovers negative patterns and terms from selected negative documents using the same pattern mining technique that we used for the feature discovery in positive documents. In addition, the process revises the initial features and obtains a revised weight function. The former finds higher level positive features, selects top-K negative samples, discovers higher level negative features, and composes the set of terms.

An Efficient Concept-Based Mining Model for Enhancing Text Clustering proposed Concept Based Analysis Algorithm are used. The concept-based analysis algorithm describes the concepts in the documents. Each concept in the verb argument structures, which represents the semantic structures of the sentence, is processed sequentially. Each concept in the current document is matched with the other concepts in the previously processed documents.

A Topic based Document Relevance Ranking Model proposed Pattern Enhanced Topic Model (PETM) are used. PETM determine document relevance based on topics distribution and maximum matched patterns. LDA (Latent Dirichlet Allocation) is one of the most popular probabilistic text modelling techniques. It can discover the hidden topics in collections of documents with the appearing words. PETM pattern mining is used to discover semantically meaningful and efficient patterns to represent topics and documents are implemented in two steps. Firstly, construct a new transactional dataset from the LDA outcomes of the document collection; secondly, generate pattern based representations from the transactional dataset to represent user needs.

On Similarity Preserving Feature Selection proposed SPFS-NES (Similarity Preserving Feature Selection-Nesterov’s method) are used. Feature selection is to choose a subset of the original features according to a selection criterion. It selects a small set of the original features. The original features, feature selection improves the interpretability of learning models and it is a learning process, and use the objective function of the learning model to guide searching for relevant features.

High Dimensional Data Clustering using Fast Cluster Based Feature Selection proposed Fast clustering based feature Selection algorithm (FAST) are used. Based on the MST method the FAST algorithm works in two steps. In the first step, features are divided into clusters by using graph-theoretic clustering methods. In the second step, the most representative feature that is strongly related to target classes is selected from each cluster to form the final subset of features.

A Two-stage Information Filtering Based on Rough Decision Rule and Pattern Mining proposed two stage information filtering are used. In first filtering stage is supported by a novel rough analysis model which efficiently removes a large number of irrelevant documents, thereby addressing the overload problem. In second filtering stage is empowered by a semantically rich pattern taxonomy mining model which effectively fetches incoming documents according to the specific information needs of a user, thereby addressing the mismatch problem.

III.METHODOLOGY

In this paper we introduce the RFD (Relevance Feature Discovery) model. It describes the relevant features in relation to three groups: positive specific terms, general terms and negative specific terms based on their appearances in a training set. The goal of relevance feature discovery in text documents is to find a set of useful features, including patterns, terms and their weights, in a training set D, which consists of a set of relevant documents, D^+, and a set of irrelevant documents, D^-.

3.1 Frequent and Closed Patterns

Let $T_1=\{t_{1,}t_{2,\dots.}t_m\}$ be a set of terms (or words) which are extracted from D^+, and termset X be a set of terms. For a given document d, *coverset(X)* is called the covering set of X in d, which includes all paragraphs $dp \in PS(d)$ such that $X \text{€}\ dp$, i.e., $coverset(X)= \{dp/dp \in PS(d),\ X \text{€}\ dp\}$. Its absolute support is the number of occurences of X in $PS(d)$, that is $sup_a(X)=|coverset\ (X)|$. Itss relative support is the fraction of the paragraphs that contain the pattern, that is $sup_r(X)=\frac{|coverset(X)|}{|PS(d)|}$. A termset X is called a *frequent pattern* if its sup_a *(or* sup_r*)* $\geq min\_sup$, a given minimum support.

It is obvious that a termset X can be mapped to a set of paragraphs *Coverset(X)*. We can also map a set of paragraphs $Y \text{€}\ PS(d)$ to a termset, which satisfies

$$termset(Y)=\{t|\forall dp \in Y \rightarrow t \in dp\}.$$

A pattern X (also a termset) is called if and only if $X= termset(coverset(X))$. Let X be a closed pattern. We have $sup_a(X_1)<sup_a(X)$ for all patterns $X_1 \supset X$.

3.2 Closed Sequential Patterns

A sequential pattern $s=< t_1 \dots t_r >$ $(t_i \in T_1)$ is an ordered list of items. A sequence $s_1=< x_1 \dots . x_i >$ is called a subsequence of another sequence $s_2= < y_1, \dots .. y_j >$, denoted by $s_1 \text{€} s_2$, iff $\exists j_1, \dots, j_i$ such that $1 \leq j_1 < j_2 \dots < j_i \leq j$ and $x_1 = y_{j1}$, $x_2 = y_{j2}$,...., $x_i = y_{ji}$. A sequential pattern X in document d, *coverset(X)* is still used to describe the covering set of X , which includes all paragraphs $ps \in PS(d)$ such that $X \text{€}\ ps$, i.e., $coverset(X) = \{ps|ps \in PS(d)\ ,\ X \text{€}\ ps\}$. Its *absolute support* and *relative support* are defined the same as for the normal patterns. A sequential pattern X is called a *frequent pattern* if its relative support $\geq min\_sup$. The property of closed patterns is used to defined closed sequential patterns. A frequent sequential pattern X is closed is $sup_a(X_1) \neq sup_a(X)$ for any super-pattern X_1 of X.

3.3 Deploying Higher Level Patterns on Low- Level Terms

To improve the efficiency of the pattern taxonomy mining, an algorithm, *SPMining*(D^+, min_sup) to find closed sequential patterns for all documents $\in D^+$, which used the well-known *Apriori* property to reduce the searching space. For all relevant documents $d_i \in D^+$, the *SPMining* algorithm discovers all closed sequential patterns, SP_i, based on a given *min_sup*. Let $SP_1, SP_2, \ldots, SP|_D{}^+|$ be the sets of discovered closed sequential patterns for all documents $d_i \in D^+$ $(i = 1, \ldots, n)$, where $n = |D^+|$. 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:

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

where $\tau(t,d) = 1$ *if* $t \in d$; otherwise $\tau(t,d) = 0$.

IV. EFFICIENT ALGORITHM

Apriori algorithm is used for mining the frequent itemsets. The mined frequent itemsets are then used for obtaining the partition, where the documents are initially clustered without overlapping. Furthermore, the clusters are effectively obtained by grouping the documents within the partition by means of derived keywords. The devised approach consists of the following major steps.

1) Preprocessing
2) Mining of frequent itemsets
3) Clustering process
4) Apriori algorithm for frequent itemset

4.1 Preprocessing

Let D be a set of text documents represented as $D=\{d_1\ d_2\ d_3 \ldots\ d_n\}$ □ □ □≤□ □≤□n where , n is the number documents in the text dataset$D$. The text document set D is converted from unstructured format into some common representation using the text preprocessing techniques, in which the words or terms are extracted (tokenization). The input data set D(text documents) are preprocessed using the techniques namely, removing stop words and stemming algorithm.

a) *Stop word Removal:* Removes the stop (linking) words like “have”, “then”, "it", "can", “need", "but", "they", "from", "was", "the", "to", "also" from the document.

b) *Stemming algorithm:* Removes the prefixes and suffixes of each word.

4.2 Mining of frequent itemsets

This sub-section describes the mining of frequent itemsets from the preprocessed text documents D. For every document d_i , the frequency of the extracted words or terms from the preprocessing step is computed and the top-*p* frequent words from each document d_i are taken out.

$$K_w = \{ d_i \mid p(d_i);\ \forall d_i \in D \}$$

$$Where,\ p(d_i)=T_{wj};\ 1 \leq j \leq p$$

From the set of top- p frequent words, the binary database B is formed by obtaining the unique words. Let B_T be a binary database consisting of n number of transactions (documents) T and q number of attributes (unique words) $U=[u_1,u_{2,\ldots,}u_q]$. Binary database B_T consists of binary data that represents whether the unique words are presented or not in the documents d_i . Then, the binary database B_T is given to the Apriori algorithm for mining the frequent itemsets (words/terms) F_s .

4.3 Clustering Process

In clustering process, group of similar data formed into single dataset. This Clustering technique uses grouping the similar words or terms and calculates the weight of an each term and finding the relevant documents.

4.4 Apriori algorithm for frequent itemset

Apriori is a conventional algorithm that was first introduced mining association rules. The two steps used for mining association rules are as follows. (1) Identifying frequent itemsets (2) Generating association rules from the frequent itemsets. Frequent itemsets can be mined in two steps. At first, candidate itemsets are generated and afterwards frequent itemsets are mined with the help of these candidate itemsets. Frequent itemsets are nothing but the itemsets whose support is greater than the minimum support specified by the user. In the proposed approach, we have used only the frequent itemsets for further processing so that, we undergone only the first step (generation of frequent itemsets) of the Apriori algorithm. The pseudo code corresponding to the Apriori algorithm is,

4.5 Pseudo Code

```
Procedure Apriori (T, minSupport) {
    //T is the database and minSupport is    the minimum support
        L1= { frequent items };
        for (k=2; $L_{k-1}$ !=∅; k++) {
          $C_k$ = candidates generated from $L_{k-1}$
//that is Cartesian product $L_{k-1}$ x $L_{k-1}$ and eliminating any k-1 size itemset that is not frequent
        for each transaction t in database do {
# increment the count of all candidates in $C_k$ that are contained in t
 $L_k$= candidates in $C_k$ with minSupport
} //end for each
} //end for
return $U_k L_k$;
}
```

V. EXPERIMENT AND RESULT

The architecture diagram of the system is defined below. The devised approach consists of the following major steps.

1) Term Frequency (TF)

2) Inverse Document Frequency (IDF)

3) Similarity measure

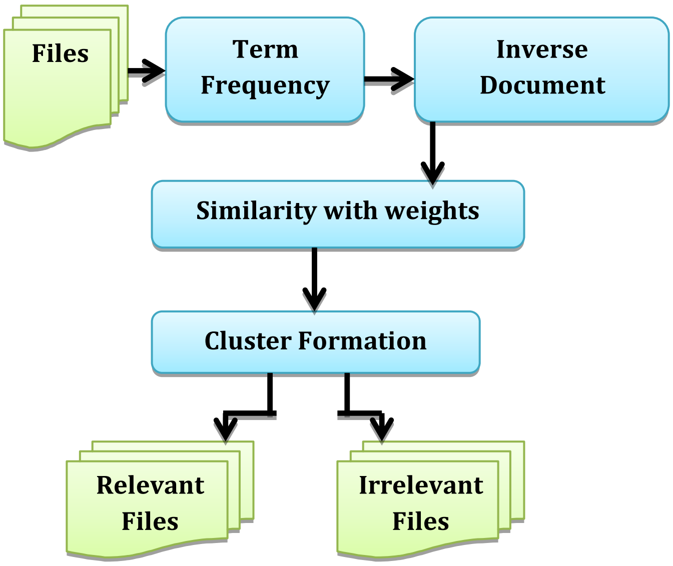
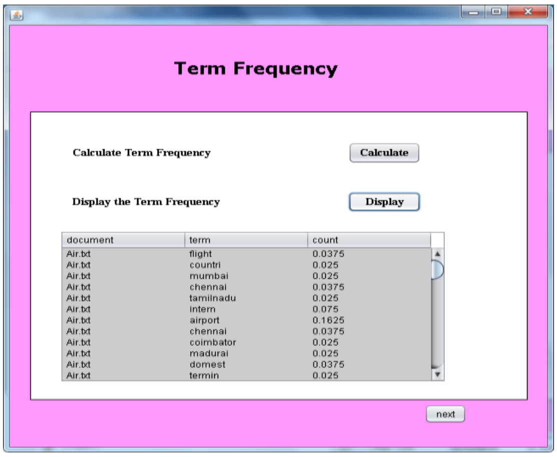


Fig 5. Architecture Diagram

5.1 Term Frequency (TF)

Term Frequency (TF) which measures how frequently a term occurs in a document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus the term frequency is often divided by the document length (the total number of terms in the documents) as a way of normalization.

TF(t)=(Number of times term t appears in a document) / (Total number of terms in the document).



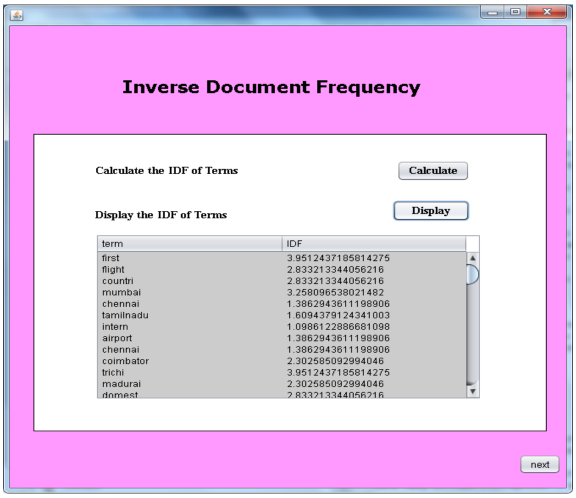
document	term	count
Air.txt	flight	0.0375
Air.txt	countri	0.025
Air.txt	mumbai	0.025
Air.txt	chennai	0.0375
Air.txt	tamilnadu	0.025
Air.txt	intern	0.075
Air.txt	airport	0.1625
Air.txt	chennai	0.0375
Air.txt	coimbator	0.025
Air.txt	madurai	0.025
Air.txt	domest	0.0375
Air.txt	termin	0.025

Fig 5.1. Term Frequency

5.2 Inverse Document Frequency (IDF)

Inverse Document Frequency (IDF) which measures how important a term is, while computing TF, all terms are considered equally important. However it is known that certain terms, such as “is”, “of”, and “that”, may appear a lot of times but have little importance. Thus we need to weight down the frequent terms while scale up the rare ones, by computing the following:

IDF(t) = log_e(total number of documents / Number of documents with term t in t).



Inverse Document Frequency

Calculate the IDF of Terms

Display the IDF of Terms

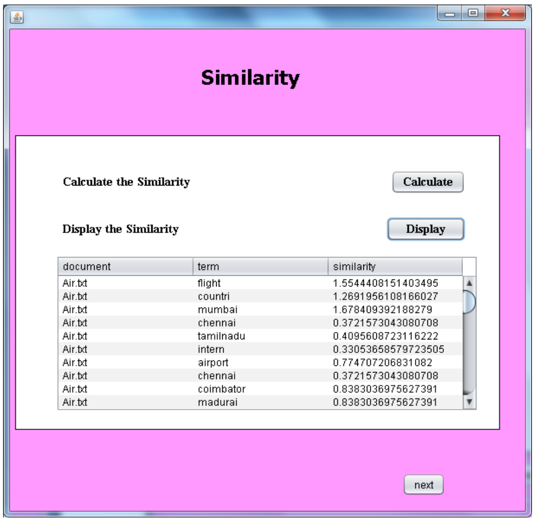
term	IDF
first	3.9512437185814275
flight	2.833213344056216
countri	2.833213344056216
mumbai	3.258096538021482
chennai	1.3862943611198906
tamilnadu	1.6094379124341003
intern	1.0986122886681098
airport	1.3862943611198906
chennai	1.3862943611198906
coimbator	2.302585092994046
trichi	3.9512437185814275
madurai	2.302585092994046
domest	2.833213344056216

Fig 5.2. Inverse Document Frequency

5.3 Similarity measure

In similarity measure we can find out the similarity between any two documents.

$$Cosine\ similarity(d_1,d_2) = dot\ product(d_1,\ d_2) / ||d_1|| * ||d_2||.$$



Similarity

Calculate the Similarity

Display the Similarity

document	term	similarity
Air.txt	flight	1.5544408151403495
Air.txt	countri	1.2691956108166027
Air.txt	mumbai	1.678409392188279
Air.txt	chennai	0.3721573043080708
Air.txt	tamilnadu	0.4095608723116222
Air.txt	intern	0.33053658579723505
Air.txt	airport	0.774707206831082
Air.txt	chennai	0.3721573043080708
Air.txt	coimbator	0.8383036975627391
Air.txt	madurai	0.8383036975627391

Fig 5.3. Similarity Measure

5.4 Performance measure

This below figure shows that how execution time is decreased when compared to the existing system. The performance measures indicate that the proposed method clusters the data in a more accurate manner.

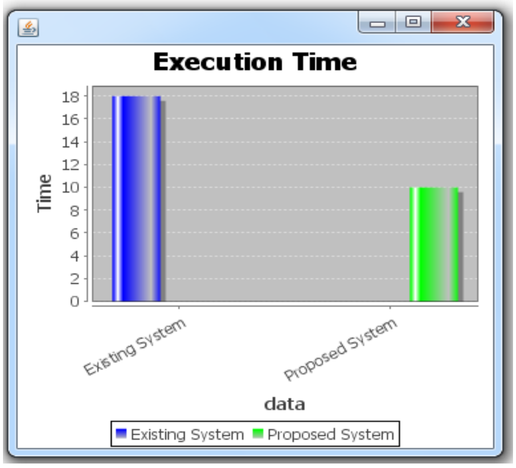


Fig 5.4. Performance measures

VI. CONCLUSION

In this paper Apriori algorithm are used. This algorithm reduces the execution time for finding the frequent terms or words from documents. It searching the words or terms level by level by using filtering method. The main issue in using irrelevant documents is how to select a suitable set of irrelevant documents since a very large set of negative samples is typically obtained. For example, a Google Search can return millions of documents; however, only a few of those documents may be of interest to a Web user. Obviously, it is not efficient to use all of the irrelevant documents. This algorithm can reducing the irrelevant documents by using level by level searching. In this paper clustering technique uses grouping the similar words or terms and calculates the weight of an each term and finding the relevant documents and it can achieve the better accuracy.

VII. ACKNOWLEDGEMENT

This is a great pleasure and immense satisfaction to express my deepest sense of gratitude and thank to everyone who has directly or indirectly helped me in this project work successfully.

REFERENCES

[1] M. Aghdam, N. Ghasem-Aghaee, and M. Basiri, “Text feature selection using ant colony optimization,” in Expert Syst. Appl. , vol. 36, pp. 6843–6853, 2009.

[2] A. Algarni and Y. Li, “Mining specific features for acquiring user information needs,” in Proc. Pacific Asia Knowl. Discovery Data Mining, 2013, pp. 532–543.

[3] N. Azam and J. Yao, “Comparison of term frequency and document frequency based feature selection metrics in text categorization,” Expert Syst. Appl., vol.3 9, no. 5, pp. 4760–4 768, 2012.

[4] R. Bekkerman and M. Gavish, “High-precision phrase-based document classification on a modern scale,” in Proc. 11th ACM SIGKDD Knowl. Discovery Data Mining, 2011, pp. 231–239.

[5] A. Blum and P. Langley, "Selection of relevant features and examples in machine learning," Artif. Intell. , vol. 97, nos. 1/2, pp. 245–271, 1997.

[6] C. Buckley, G. Salton, and J. Allan, "The effect of adding relevance information in a relevance feedback environment," in Proc. Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, 1994, pp. 292–300.

[7] N.Zhong, Y.Li, and S.-T. Wu, "Effective pattern discovery for text mining," in IEEE Trans. Knowl. Data Eng., vol. 24, no. 1, pp. 30–44, Jan. 2012.

[8] Z. Zhao, L. Wang, H. Liu, and J. Ye, "On similarity preserving feature selection," in IEEE Trans. Knowl. Data Eng., vol. 25, no. 3, pp. 619–632, Mar. 2013.

[19] Y. Li, X. Zhou, P. Bruza, Y. Xu, and R. Y. Lau, "A two-stage text mining model for information filtering," in Proc. 17th ACM Conf. Inf. Knowl. Manage., 2008, pp. 1023–1032.

[10] Y. Li, A. Algarni, and N. Zhong, "Mining positive and negative patterns for relevance feature discovery," in Proc. ACM SIGKDD Knowl. Discovery Data Mining, 2010, pp. 753–762.