IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 16, Issue 5, Ver. VIII (Sep – Oct. 2014), PP 24-30
www.iosrjournals.org

Text Clustering in Distributed Networks with Enhanced File Security

Teena Susan Chandy

Abstract: *Text Clustering technique is widely used in both centralized and distributed environments for information retrieval. Centralized approaches are used in traditional text clustering algorithms. In such approaches, clustering is performed on a dedicated node and also they are not suitable for deployment in large distributed networks. This centralized approach require high processing time and retrieving time during searching due to scalability of users. To overcome this, the Probabilistic text clustering for peer to peer(PCP2P) algorithm was introduced. It provides improved scalability by using a probabilistic approach for assigning documents to clusters. Only most relevant clusters are considered for comparison with each document. PCP2P alone does not provide any security for the system. It only ensures efficient information retrieval of text data by assigning document to most relevant cluster. This project is implemented within a medical research environment. Here, the system is implemented using this PCP2P algorithm and an enhanced version of a security related technique called LZW. Enhanced LZW technique can ensure secure transmission within the system. This technique includes hiding information directly in compression codes. Lempel -Ziv-Welch (LZW) coding is a well known lossless compression algorithm is simple and does not require prior analysis of the source or send extra information to the decoder. This introduces a new input table which contains 255 contents. This method, reversibly embeds data in LZW compression codes by modifying the value of the compression codes. The value of the LZW code either remains unchanged or is changed to the original value of the LZW code plus the LZW dictionary size according to the data to be embedded.*

Index Terms: Text clustering, K-means, P2P network, Distributed network, DHT, Lossless data compression, LZW code

I. Introduction

Text clustering is an established technique that is widely employed in most networks for improving the quality of information retrieval. It avoids the problem of information overflow by structuring large document collections into clusters and by enabling cluster-based information retrieval. Text clustering includes grouping of similar data objects into clusters based on their textual contents, such that objects in the same cluster are similar, and those in different clusters are dissimilar.

Most traditional text clustering algorithms are designed for centralized environments. In such approaches, clustering is performed on a dedicated node and also they are not suitable for deployment in large distributed networks. This centralized approach require high processing time and retrieving time during searching due to scalability of users. It also increases load on the network. Therefore, specialized algorithms such as [1], [2], [3] are designed for distributed and P2P clustering. But as these approaches are limited to only small number of nodes or as they focus only on low dimensional data, they will not work out efficiently in distributed peer to peer environments.

In distributed environments, data sources are distributed over a large,dynamic nework. Clustering in such networks is comparitively difficult because of some reasons such as: (1) data is widely distributed and no participant in the network has the capacity to collect and process all data. (2) availability of content and of computational nodes is affected because of high churn rate. As a result, a P2P algorithm that can perform text clustering in a decentralized manner without overloading any of the participant peers was required. Thus PCP2P algorithm was introduced. By using a probabilistic approach, this algorithm reduces the number of required comparisons between the document and the cluster. Network traffic is highly reduced in these system by reducing the number of required comparisons between document and clusters. As the peers are distributed in nature, searching and retrieval of informations are relatively fast. With each document, only the most few relevant clusters are taken into consideration which in effect highly reduces the retrieving time of informations. In this existing PCP2P system, no security feature is provided. So in the proposed system, this algorithm called PCP2P is implemented along with some security measures which provides file protection. Here, an enhanced version of Lempel-Ziv-Welch (LZW) technique is used to provide security for the files transferred. When compared to the LZW, this enhanced version of LZW reduces the size of dictionary table/input table.

II. Related Work

K-Means algorithm is widely used for document clustering because of its low complexity and high clustering quality. Its direct distribution is not suitable for text clustering in large networks. As a result a distributed version of K Means are used.K-Means algorithm can be summarized as: (1) Select k random starting points as initial centroids forthe k clusters. (2) Assign each document to the cluster with the nearest centroid. (3) Recompute the centroid of each cluster as the mean of all cluster documents. (4) Repeat steps 2-3 until a stopping criterion is met, e.g.,no documents change clusters anymore.

There are several works which focus on P2P clustering [1], [2], [3]. One of the first P2P clustering algorithm was proposed by Eisenhardt et al. [1]. By broadcasting the centroid information to all peers, this algorithm distributes K-Means computation. This approach makes use of two algorithms, K-Means clustering algorithm is used for the categorization of documents and a probe/echo mechanism is used to broadcast the task through the P2P network and to propagate the results back to the initiator of the clustering. Because of this centroid broadcasting, it imposes heavy load and congestion in the network. As a result, it does not scale to large networks. Hsaio and King [2] avoid broadcasting by using a DHT to index all clusters using manually selected terms. This approach requires extensive human interaction for selecting the terms, and the algorithm cannot adapt to new topics.

Hammuda and Kamel [3] propose a hierarchical topology for distributing K-Means. Clustering starts at the lowest level of the hierarchy, and the local solutions are aggregated until the root peer is reached. This algorithm has the disadvantage that clustering quality decreases noticeably for each aggregation level, because of the random grouping of peers at each level. There-fore, quality decreases significantly for large networks. Already for a network of 65 nodes organized in three levels, the authors report a drastic drop in quality.

A frequently used technique which focus on constructing an index over a distributed hash table(DHT) that maps terms to documents, and enables locating the most similar documents for each term [4]. Structured peer to peer networks support keyword search by building a distributed index over the collective content shared by all peers. DHT is used for the distribution of the inverted index over all participating peers. Each peer analyzes its own collection, and extracts a set of terms, normally after stemming and stopword removal processes. For each extracted term, the peer executes a DHT lookup to locate the relevant peer in the network, and then posts the details with the term and its appropriate term score.

The centralized execution of text clustering imposes high network traffic and relatively requires high processing and retrieval time during information retrieval. As a result PCP2P [5], a decentralized probabilistic text clustering algorithm was introduced to cluster highly dynamic and distributed text collections without imposing any overload on its participating peers. This approach makes use of probabilistic pruning in which, instead of considering all clusters, only few most relevant clusters are taken into consideration. A peer can undertake three roles in PCP2P,i.e., it takes the role of a document holder. Second, it can be a DHT participant and third, it can be a cluster holder. PCP2P reduces the number of required comparisons between the document and the cluster and as a result provides faster information retrieval.

III. Proposed System Description

In my proposed system, the existing PCP2P algorithm is implemented for fast information retrieval of text datas.Also, an enhanced version of additional security technique called Lempel-Ziv-Welch(LZW) is implemented which provides protection for the file transfer within the system. The system is implemented within a medical research environment. As there are 1000s of records, for the fast and easy access of records, the Doctor/Medical researcher can make use of this PCP2P algorithm implementation within this system. The Doctor can easily search and retrieve the required information.In this system, user can search and retieve data. The Researcher will upload all necessary data in this system. Here, employee is used to upload data. Any other Doctor/Researcher who needs information can search and download the required details easily. Also, another security feature is incorporated in the system using an enhanced version of LZW technique. LZW is used in the secret file transfer between the Doctors/Researchers. The Doctor uses LZW hiding and can hide any confidential data regarding patients or hospitals which he/she wish to hide from outside world. Then it can be send to the required Doctor/Researcher in an encoded form whenever necessrary . He/ She can then decode the information and can view it using LZW decoding. This provides security within the system in addition to the fast retrieval of informations. This system can also be implemented in various other applications such as military, banking etc.

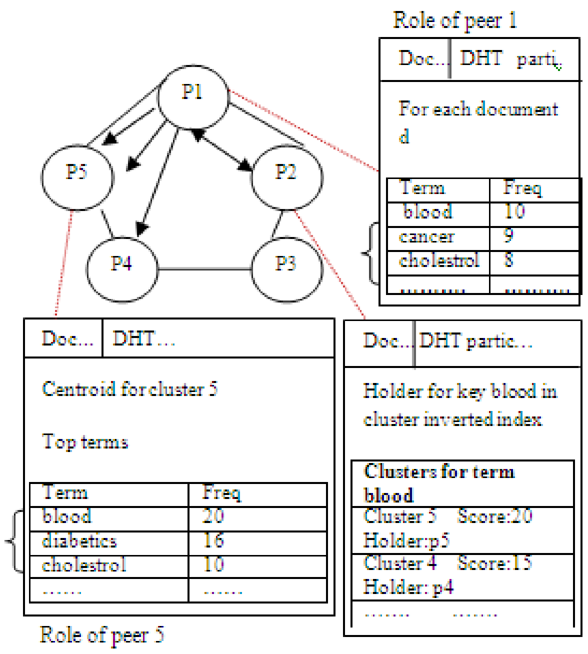

Fig 1. PCP2P System overview

3.1 PCP2P

Steps:
1. DHT lookup for top terms of d (in Fig. p1->p2)
2 .Retrieve all relevant clusters (in Fig. p2->p1)
3. Compare d with top relevant clusters (in Fig, p1>p4, p1->p5)
4. Assign d to best cluster (in Fig. p1->p5)

```
ALGORITHM 1: PCP2P
1.      for Document d in documents do
PRESELECTION STEP:
2.          CandClusters ← CandidateClustersFromDHT()
FULLCOMPARISON STEP:
3.      RemainingClusters ⟵ FilterOut(CandClusters)
4.          for Cluster c in Remaining Clusters do
5.          Send termVector(d) to ClusterHolder(c)
6.          Sim[c] ⟵ Retrieve similarity(d,c)
7.      end for
8.      Assign(d, cluster with maximum similarity)
9.      end for
```

In PCP2P , a peer can act in three different roles. First, as a document holder which takes care of clustering its documents. Second, as a DHT participant which participates in the underlying DHT by holding part of the distributed index. Third, a peer can become a cluster holder which includes the centroid and document assignments for one cluster. This approach also includes two activities in parallel which are ‘cluster indexing’ and ‘document assignment to clusters’.

A Cluster Indexing

The peer which is the cluster holder will perform cluster indexing. Each cluster holder, will recompute the cluster centroid periodically ,using the documents which are assigned to the cluster at the time. These peers makes cluster summaries and index them in the underlying DHT, using the most frequent cluster terms as keys. The set of terms are denoted as top terms. This helps peers to identify the relevant clusters for the documents.

B Document assignment to clusters

The document assignment to clusters includes two steps such as ‘preselection’ and ‘full comparison’. When a text document is uploaded, it undergoes preprocessing which includes stopwords removal and stemming processes. In stopword removal, eg. is, was, and, a, “, when, where etc, are removed. In stemming, all words within the text document are converted to standard forms. For eg. words such as relation, relating, relates etc can be taken as single 'relate' word which is the standard form. The top terms of d thus includes no stopwords. Next, the frequency for each term in document d is calculated. After this, look up on the Distributed Hash Table(DHT) for top terms of d. Next in the preselection step (Alg. Line 2) , it filters out most of the clusters. The peer holding the document d retrieves selected cluster summaries from the DHT index , to identify most relevant clusters.The preselection list is denoted as Cpre. In the full comparison step , comparison of 'd' with each cluster is done by measuring the similarity, ie., similarity score estimates are calculated for d by the peer using the retrieved cluster summaries (Alg. Line 3). Here after computing similarity, the clusters with low similarity are filtered out . The document is then sent to the few remaining cluster holders for full similarity computation and retrieves the comparison results which is illustrated in Alg. Lines 4-7. To avoid sending the document to all clusters holders for comparison, the peer p uses the cluster summaries contained in Cpre to eliminate the clusters which are not required for the document at hand. Finally, document d is assigned to cluster with highest similarity(Alg Line 8).

C Filtering Strategy

Filtering is done by computing the cosine similarity between a document and each of the clusters.The cosine similarity between a document and cluster centroid is expressed as,

$\mathrm{Cos}(d,c) = \sum_{t \in d} \frac{TF(t,\ d) \times TF(t,\ c)}{|d| \times |c|}$ where |d| and |c| are length of document and cluster centroid respectively which are in L2-Norms and TF denotes the term frequency of the term in the document/cluster.

L2 Norm is denoted as $|d| = \sqrt{\sum_{t \in d} TF^2(t, d)}$

The filtering method works as follows: Peer p sends the document vector to first cluster holder in Cpre, denoted as Cselected, and then retrieves the actual cosine similarity Cos(d, Cselected). Then the summary of Cselected is removed from Cpre list. This process is continued until Cpre list is empty. Finally, the document will only be assigned to the cluster with maximum similarity which will help in the easy retrieval of informations during searching.

3.2 Lempel –Ziv-Welch (LZW) Technique

This technique is used within the system for providing secure file transmissions. LZW technique [6] hides data in compression codes**.** This process includes both LZW compression and LZW decompression processes.The technique makes use of an INPUT table (dictionary) for compression and a Receiver table for decompression. Once a new symbol is added into the dictionary, that symbol can be used to hide a secret. The data–hiding phase modifies the value of the LZW compression code to hide secrets, except for the initial 256 symbols in the dictionary. Every embeddable symbol can be used to hide one secret bit. Before a new symbol is added to the dictionary, the LZW encoder modifies the value of the output LZW code according to the secret value. If the secret bit is '0', the output is the original LZW code; and if the secret bit is '1', the output is the sum of the value of the LZW code and the size of the LZW dictionary. When the secret bit is '1', the value added to LZW code can vary from 256 onwards. At first if 256 is added, then for next bit, 257 will be added and so on. Each 'space' in the input text is assigned value 32. The proposed method has new input table containing frequently used English grammar words.

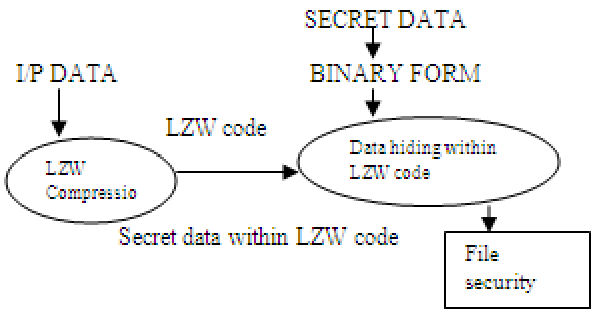

Fig 2. Data hiding within compression codes

Before a new symbol is added to the dictionary, the encoder modifies the value of the output LZW code according to the secret value. LZW Compression is performed by taking input text file and with the help of new dictionary. The binary format of the secret file is taken.Then hide the secret file within the LZW code using the algorithm.

ALGORITHM 2:

The compression algorithm is summarized as follows:

Input : Text file and Secret file
Output :LZW codes
Step 1: Scan first 2 letters in the input text. Each time substring of 2 letters is taken
Step 2: Check whether the DICTIONARY table contains that substring
If Yes then pick the corresponding INDEX value goto step 3.
else goto step 5.
Step 3: scan that complete input text and obtain the corresponding INDEX values for each 2 letter substring in the entire input text and convert it into respective compression codes.
Step4: Each space in input text is assigned value 32.
Step 5: else extract each letter in that substring and obtain their corresponding INDEX values for each letter as the compression code for that substring. Then ADD that particular substring as the next WORD in the DICTIONARY table after INDEX 255. For the next occurrence of the same substring, this INDEX value from dictionary can be used. Each time after processing, the newly added words/substrings from the input and receiver table will be cleared
Step 6: For hiding LZW code within the secret data , When b= '0', no change in the code. If b = '1' Set C = C + Size , where Size is the size of dictionary and 'b' is each bit in the secret data. C is the Compression code. The Size can vary. If 256 is first added with C for first bit, then for next bit 257 will be added and so on.
Step 7: Continue Step 1 to 6 ,till the input file is completed,

From the ASCII table, most frequently used characters are selected. An INPUT/ DICTIONARY Table is generated which contains the most frequently used words. For example, if the input file is 'haai ',split it as substrings ha and ai. Then first check if 'ha' is present or not in the DICTIONARY TABLE. If present, use the index of ha as code. Else, find the INDEX values for each 'h', 'a', separately from DICTIONARY and use that as code. Then add that 'ha' word as new word in the DICTIONARY. For further occurrences of that same word, this can be used. Space has value 32. In the same way, the process is repeated until the entire input text is completed. When hiding the compression code within the secret data, if the bit of the secret data is '1' ,then add the code with the dictionary size . As the bit position changes, the added value can be 256, 257 and so on.and if the bit is '0', then no change for the code.
ALGORITHM 3:
The decompression process is summarized as follows :
Input : Compression codes
Output: Source file and Secret file
Step 1: Read a compression code a. If a > Size ,where Size is the dictionary size(256) .
Set a =a- Size
Secret bit=1
Else
Secret bit=0
Output s, where s is the symbol or word of 'a ' in the INPUT / DICTIONARY Table
Step 2: Get the next compression code C
If C>Size +1
Set C=C- (Size +1)
Secret bit=1
Else
Secret bit=0

Output s, where s is the symbol or word of 'C ' in the INPUT/DICTIONARY Table. The sender makes use of input/dictionary table for compression and receiver makes use of receiver table for decompression respectively. The process is repeated until the entire INPUT TEXT FILE is correctly obtained back.

Table 1sample Input Table- Part 1

INDEX	WORDS	INDEX	WORDS
0	are	16	among
1	as	17	across
2	at	18	already
3	about	19	ago
4	also	20	by
5	any	21	But
6	after	22	Before
7	around	23	Between
8	always	24	Best
9	away	25	Become

10	actually	26	both
11	against	27	Back
12	along	28	Better
13	again	29	Bad
14	almost	30	Below
15	above	31	Behind

Table 2 Sample Input Table- Part 2

INDEX	WORDS	INDEX	WORDS
215	since	237	throughout
216	sure	238	Up
217	small	239	Usually
218	sometimes	240	Under
219	simply	241	Until
220	short	242	Very
221	straight	243	With
222	the	244	What
223	to	245	Which
224	this	246	When
225	there	247	Well
226	these	248	Would
227	through	249	Where
228	then	250	Want
229	take	251	Why
230	too	252	Without
231	though	253	Within
232	tonight	254	Wide
233	together	255	Wrong
234	today		
235	therefore		
236	thus		

So if C is larger than Size, then the extracted secret bit is '1; else the secret bit is'0'. And if the extracted secret is '1, then the original LZW code is the difference between C and Size, if the extracted secret is '0', then the original LZW code is C.

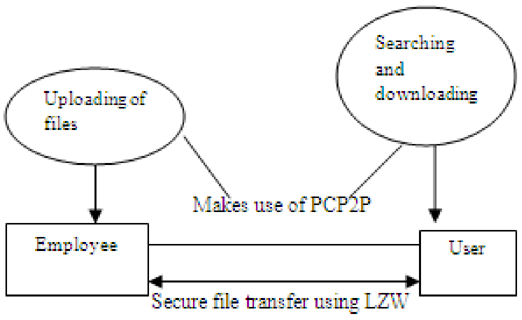

Fig 3. Proposed System

IV. Performance Analysis

Table 3 Comparison With Existing Approaches

System	Search time	Security
Text Clustering in Centralized system	Relatively high search time required for information retrieval.	No Security feature adopted in the system.
PCP2P	Efficient method for information retrieval. Requires less searching time.	Focusses only on providing easy access for informations but does not include any security feature for the system.
Text Clustering in Distributed Networks with Enhanced File Security	Requires less search time for information retrieval even with large document collections	Provides secure file transmissions within the system in addition to easy information retrieval.

		Includes PCP2P and LZW.

This section includes performance comparison of the proposed system with other traditional approaches. By comparing with existing methods, it is well understood that the proposed system is more advantageous. It not only provides easy data access but also provides secure file transmission within the system.

V. Conclusion & Future Work

The system 'Text Clustering in Distributed Networks with Enhanced File Security', provides an efficient searching method for fast information retrieval along with secure file transmission within the system. By using PCP2P for the system, it provides easy information access of text data. The system is appropriate for text collections with wide range of characteristics. With enhanced LZW hiding process, it uses only the basic arithmetic calculations and also the contents of newly created dictionary will reduce the extra time to add the commonly used grammar words into the dictionary. With LZW, the size of encoded data is less when compared to normal encryption techniques. LZW is feasible and there is no unauthorized keys in this process. Since hidden codes are based on lossless compression , it is suitable for any kind of media, including text and image data .

This work only concentrates on the textual data clustering. As future enhancement, PDF and WORD documents can be taken into consideration with suitable system configurations. Also, more advanced encryption methods such as AES can be used to further encrypt the compressed data and to give more security.

References

[1] M. Eisenhardt, W. M ¨ uller, and A. Henrich, "Classifying documents by distributed P2P clustering." in INFORMATIK, 2003.

[2] H.-C. Hsiao and C.-T. King, "Similarity discovery in structured P2P overlays," in ICPP, 2003.

[3] K. M. Hammouda and M. S. Kamel, "Hierarchically distributed peer-to-peer document clustering and cluster summarization,"IEEE Trans. Knowl. Data Eng., vol. 21, no. 5, pp. 681–698, 2009

[4] L. T. Nguyen, W. G. Yee, and O. Frieder, "Adaptive distributed indexing for structured peer-to-peer networks," in CIKM, 2008, pp. 1241–1250.

[5] Odysseas Papapetrou, Wolf Siberski, and Norbert Fuhr,"DecentralizedProbabilisticText Clustering" IEEE Trans. Knowl. Data Eng., vol..24, no.10, 2012

[6] T.A.Welch, A technique for high performance data compression, IEEE Computer,vol .17,no.6,pp.8- 19,1984.