Hierarchical Clustering Based Improved Data Partitioning using Hybrid Similarity Measurement Approach

Kiranjit Kaur, Vijay Laxmi

Abstract: Data partitioning improves the convenience of data utilization. It is complicated to extract information from a collection of data in a fast manner, so this research proposed a hierarchical clustering based improved data partitioning approach. General cloud architectures support and enforce partitioning to offer fast search results. The purpose of this study is to build up a partitioning method based on the amalgamation of cosine and soft cosine similarities to improve the data partitioning performance with better portioning speed and accuracy. Threshold-based partitioning methods are considered to have vertical and horizontal partitioning, where the basis is cosine and soft cosine similarity. The main focus of this research is to develop a hybrid approach for data partitioning in vertical as well as in horizontal manner. To assess the proposed work, Quality of Service (QoS) parameters such as accuracy, recall, true positive, false positive, true negative, and false negative are calculated and compared to data partitioning using a cosine-like algorithm. A comparison has been drawn between H. Guo et al. and K. Korjus et al. with the proposed work. 2.28% enhancement of recall and 39.94 % of enhancement of precision has been noticed with H. Guo et al. and with accuracy, 16.98% improvement has been shown with K. Korjus.

Index Terms: Data Partitioning, Clustering, Cosine Similarity, Soft Cosine Similarity, QoS measures

I. INTRODUCTION

Data partitioning is a distribution of logical data or its related elements in different sovereign parts. Data partitioning is considered for performance, manage ability and for availability reasons in load balancing. Data partitioning could be utilized in varied tasks like the integration of same data (news, tweets, images etc.) and for the examination of employee's response, detection of significant inherent subjects for each response, detection of consequential subjects for each response [1]. In case of huge scale solutions, the data has been categorized into varied partitions that may be accessed and managed differently. The strategy of data partitioning could be selected cautiously for the increment in pros for the minimization of unpleasant effects [2].Data partitioning help in improving scalability, reducing contention, and for optimizing the performance. Partitioning could even provide a method for the division of data in pattern manner. For example; a user can archive existing, less active (cold) data in cheaper data storage.

Data partition optimizes the scalability, security, and performance of cloud search. Cloud has gained more attention in the last couple of years and has maximum transitions in terms of database servers. The external layer of the cloud receives the query from a user and passes it to the inner layer of the cloud [7]. The Complexity of Search (CoS) at a cloud layer is defined as

$$\mathrm{Cos} = \left(\frac{\mathrm{Stc}}{\mathrm{Pt}}\right)^{n} \qquad (1)$$

Where Stc is the complexity of the search term, Pt is the time complexity and n is the total relevant term[8]. Higher the search time, higher is the complexity. Figure 1 demonstrates the calculation of CoS.

As shown in figure 1, Tt is the total true sample; Tst is the total selected true term [9]. Wt is the wrong term by means of search query and Wst is the wrong selected terms.

$$\mathrm{N} = \mathrm{Wst} + \mathrm{Tst} \qquad (2)$$

$$\mathrm{Pt} = \mathrm{Tstt} + \mathrm{Wstt} \qquad (3)$$

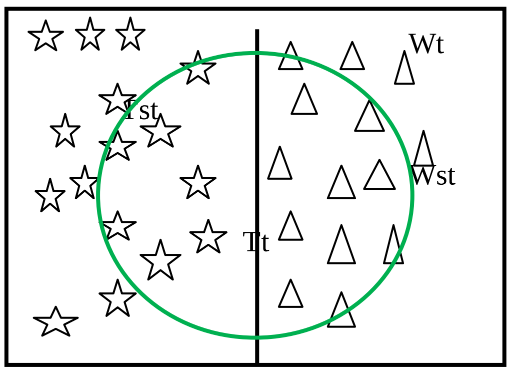

Fig.1 Data Structure

As shown in equation (3), Tstt and Wstt is the time to list down true and false elements. The performance of partitioned data depends upon how effectively and tightly the data is bonded. The bond between the data elements is based on related factors [9]. This paper utilizes a combination of cosine and soft cosine similarity as a correlation between the data elements for partitioning.

Clustering basically integrates the entities in varied classes as Clusters with similar objects [11]. A cluster is a collection of entities that are similar and belongs to the same class. In the process of clustering, firstly the set of data is partitioned in groups as data similarity and afterwards, labels are given to those groups. Clustering is generally used in several applications such as data analysis, pattern recognition, image processing, etc.

Clustering helps the marketers to determine different groups in the customer base by which they can distinguish their customer's group based on the buying history. Clustering is very useful in detecting credit card fraud. It also assists in the classification of documents for the purpose of information finding in the web. Clustering is sometimes by mistake referred to as automatic classification; however, this is inexact, since the clusters found are not known prior to processing whereas in case of classification the classes are pre-defined. With the concept of clustering for text documents, the documents contribute the similar topic are amalgamated. The user may choose the group being relevant while the clusters are being returned that assist in making the search engine more effective and reliable. The aim of a good document is to lessen the intra cluster distance amongst the documents by enlarging the distance among inter cluster. A distance measure relies on the document clustering. Because the task of collecting is subjective, the resources that could be utilized for achieving this aim are much. Each method uses varied rules to define the similarity between data. In fact, there are more than 100 known algorithms.

In this research work, clustering based data portioning has been proposed with similarity measurement technique, termed as, Cosine Similarity. Initially, the similarity of data is calculated as per one and another data and after that soft cosine similarity is applied to the similarity index of cosine. The matching process is initiated after that which depicts the matching of the groups and according to which the portioning of the data is done. The manuscript is structured in a way that the related work is listed in section 2. The proposed algorithms and architectures are provided in section 3. The results are evaluated and presented in section 4. The paper is concluded in section 5 and a hint of future aspects is provided in the same section.

II. RELATED WORK

This section elaborates the existing work of data partitioning using numerous similarity measures and at the end of this section; a problem has been formulated according to which the proposed mechanism has been drawn. Gao Xiaanan Gao and Sen Wu (2018) have proposed a binary data hierarchical clustering algorithm based on cosine similarity (HABOC). The clustering algorithm of binary data is a challenging problem in the field of data mining and machine learning. Although some efforts have been made to deal with the clustering of binary data, they lack an effective method to balance the quality and efficiency of clustering. First, the researchers have used cosine similarity to evaluate the similarity between data objects with binary attributes. Then, the cosine similarity of the set is defined to calculate the similarity of the set containing the plurality of objects. Based on CSS, we propose the cosine feature vector of the set and the additivity of CFVS to compress the data and directly merge the two clusters. Hierarchical clustering methods are used to implement clustering to avoid sensitivity to the order of data objects and algorithm parameters. Experimental results of several UCI datasets have shown that HABOC is superior to existing binary data clustering algorithms.Xiang et al. (2018) have conducted a comprehensive experimental study of a test suite consisting of twenty-four benchmark functions. ABC (Artificial bee colony) is a very popular and powerful optimization tool. However, ABC still has a lack of convergence. In order to further improve the convergence speed of ABC, a new type of ABC (CosABC) based on cosine similarity is proposed to select better neighbours. Under the guidance of selected neighbours, a new solution search equation was introduced to reduce the weakness of ABC undirected search. In addition, in the bee phase employed, a solution search equation with global best individual guidance is also integrated, and the frequency of parameter perturbations is also used to further increase information sharing between different individuals. In the beekeeper bee stage, ABC / rand / 1 / is used to enhance development capabilities, while the opposite learning technique is also used to balance the development of ABC / rand / 1. All of these modifications together with ABC constitute the proposed Cos-ABC algorithm. More importantly, it is further compared with some of the most advanced algorithms to verify the superiority of Cos-ABC. The relevant comparison results show that Cos-ABC is effective and competitive. Tao Zhiqiang et al. (2017) have proposed a new clustering algorithm called saliency guided constrained clustering method with cosine similarity (SGC3) is used for image co-segmentation tasks, where the common foreground is extracted by one-step clustering process. In the method, the unsupervised significant prior is used to guide the partition-level auxiliary information of the clustering process. In order to ensure the robustness of the noise and outliers in a given prior, the similarity between the instance level and the partition level is used for joint calculation. Specifically, the researchers have used the cosine distance to calculate the feature similarity between the data points and their cluster centroids and have introduced the cosine utility function to measure the similarity between the clustering results and the auxiliary information. Based on the cosine similarity, it is able to capture the intrinsic structure of the data, especially for non-spherical cluster structures. Finally, similar K-means optimization aims to solve the objective function in an efficient way. Experimental results of two widely used data sets indicate that the proposed approach has achieved competitive performance over the most advanced distribution methods.(Gu, X., Zhang et.al 2018) proposed deep neural network such as CODEnn (Code-Description Embedding Neural Network). CODE does not match the textual similarity but includes code snippets and natural language descriptions into the high-dimensional vector area, as well as the code fragment and its corresponding description have similar vectors. With associated vector representation, code snippets associated with a natural language query can be accessed in accordance with their vectors. Semantically, the work could even be identified with the keywords in the queries that could be managed. In this work, the researchers have not considered control structures of source code to better symbolize and the deep neural network is used for the specific benefit of software engineering problems and it is limited.

Published By:
Blue Eyes Intelligence Engineering & Sciences Publication

Wang Jingdong et al. (2014) has presented a survey of one of the main solutions, hashing, which has been extensively studied since the local sensitive hash of groundbreaking work. Similarity search (nearest neighbour search) is the problem of tracking the smallest distance data item from a large database. Various methods have been developed to solve this problem, and many efforts have recently been made to perform an approximate search. Researchers divide hash algorithms into two broad categories: local-sensitive hashes, which design hash functions without exploring data distribution and learning hashes, learning hash functions based on data distribution, and examining them from various aspects, including hashing. Function design and distance measurement and search schemes in the hash coding space. Rama Krishnan M. et al. (2014) discussed the modified version of the k-mean algorithm for formulating clusters. Modified k-means algorithms could execute the clustering very efficiently on Categorical data sets. The modified k-mean use pruning methods by this number of iterations that are needed for calculating distance are reduced. This technique is applicable for definite data sets and also improving the efficiency of the k-mean clustering algorithm and reduces the overall computational time. K-mean is efficient for small dataset only and if noise is present it cannot work efficiently. In this work, the optimization technique is absent and there is a need to add better distance minimization because the number of distance calculation is more. Verma A. et al. (2014) as k-methods is a less popular algorithm for clustering because it's too complex and costly than that of k-mean. The proposed algorithm uses standard deviation that reduces the total time to formulate the cluster by simple k-mean. The proposed method divides the square root distance with standard deviation. This enhanced k-mean perform much better than that of k-methods and k-mean. It takes less time to formulate the clusters. But still neither k-means nor k-methods work on very large scale data. The authors have not utilized any types of similarity measurement technique for distance calculation between different types of data that results in low clustering performance and there is a need to add it with k-means using optimization approaches. The study of various researchers regarding data partitioning has been presented in the above section. It has been concluded that the researchers are not emphasizing on data partitioning efficiency of proposed work is less and improvement is needed. Because of unsupervised techniques for data partitioning, this problems occurs. The major drawbacks of existing algorithms are:Some researchers have used clustering techniques but due to the nature of unsupervised, the accuracy is lower and need to use similarity concept with clustering.

i. Due to the usage of cosine similarity, research can only calculate the similarity amongst two non-zero vectors computes the cosine of the angle linking them.
ii. The researchers have used the concept of soft cosine similarity and have calculated the similarity based on the feature value.
iii. The concepts of existing techniques are good but not applicable for each case, so, hybridization of algorithms is the best option.

In this research work, cosine similarity with the concept of soft cosine similarity is used to achieve better performance of proposed work.

III. PROPOSED WORK

The proposed solution of data partition is divided into two sections.

A. *Similarity Calculation*

The similarity calculation between data elements returns the closeness between the data elements. The text-based similarity is never precise and cannot be always trusted. The proposed solution uses Cosine and Soft Cosine similarity for closeness.

a) Cosine similarity

It represents the cos angle between two data values A and B.

$$A \cdot B = \| A \| \| B \| \cos \theta \qquad (3)$$

$$\text{Cosine Similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \qquad (4)$$

Algorithm 1 demonstrates the working of Cosine Similarity.

Algorithm 1: Cosine Similarity Relation

Input: *repository //repository is the input*
Output: *Cosine_data_sim // Cosine_data_sim is the output*

```
1.Cosine_data_sim = [ ]; // to store similarity value of data
2.Similarity_count = 0; // used to increment in array
3.For I = 0 → repository.length
4.current_data_file = repository (I);
5.For p = I +1 → repository.length
6.L=|Cosine_data(current_data_file)-cos(repository(n))|;
7.Cosine_data_sim[relation_count,0]= current_data_file;
8.Cosine_data_sim[relation_count,1]= repository(p);
9.Cosine_data_sim[relation_count,2]=L;
10.Similarity_count = Similarity_count + 1;
11.End for
12.End for
13. Return: Cosine_data_sim as output
14.End function;
```

The similarity values are calculated as follows as per Algorithm 1.

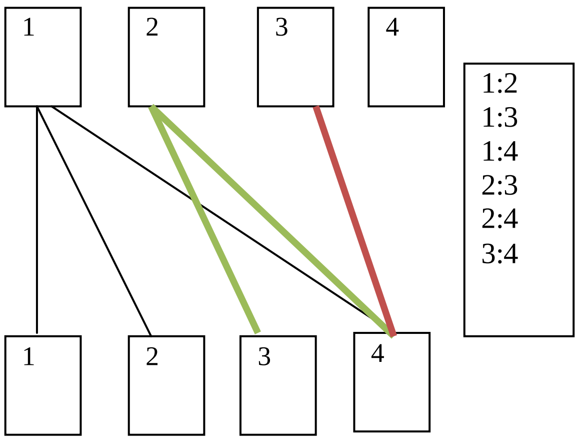

Fig.2 Evaluation of Relation between data

For every data file in the list, a relation is established which is shown in the right corner box.The relation value of 1→2 will be the same as that of 2→1, so no need to calculate the similarity between 2 and 1.

After the similarity calculation between documents, the structure of the storage is as follows

Cell 1 →	Main Data File
Cell 2 →	Connecting Data File
Cell 3 →	Relation Value

Where, Cell is the array of the matrix which contains three columns named as Cell 1, Cell 2, and Cell 3.

b) Soft Cosine Similarity

Soft cosine similarity is an extended similarity index based on cosine similarity [6-7].

$$\text{Soft Cosine}(a,b) = \frac{\sum_{i,j}^{N} \text{sim}_{ij} A_i B_j}{\sqrt{\sum_{i,j}^{N} \text{sim}_{ij} A_i A_j}\sqrt{\sum_{i,j}^{N} \text{sim}_{ij} B_i B_j}} \quad (5)$$

Where the sim is the cosine similarity value of A and B data. The proposed mechanism utilizes both Cosine and Soft Cosine Similarity as follows [10].

Algorithm 2: Combined Soft Cosine

```
Input:Enhanced_Sim // Improved similarity using cosine similarity technique is the input
Output:Create_Combined_Soft_Cos // Create_Combined_Soft_Cos is the output
1. For i =0 to Total_Similarity_Values // i is number of loops
2. Csr=Cosine_Relation(i); // Csr is cosine similarity values of documents
3. Scr=Soft_Cosine_Relation(i) // Scr is soft cosine similarity values of documents
4. Improved_Similarity=Csr/Scr;
5. Combined_Similarity(i,1)=Main_Data;
6. Combined_Similarity (i,2)=Connecting_Data;
7.Combined_Similarity (i,3)=Improved_similarity; // Where Combined_Similarity is an array in which similarity value is stored according to the documents
8. EndFor
9.Return Scr;
10. EndAlgorithm
```

The combined similarity value takes cosine similarity as the numerator and soft cosine as the denominator. The storage structure will be the same as algorithm 1 inthe form of cells. Because of the combination of cosine and soft cosine, the outcome of combined similarity is improved.

B. Partitioning

The partitioning method takes the combined relation value as input and applies the following algorithm to perform partition.

Algorithm 3: Partition Formation

```
Input: Relationvalues // It is the input of algorithm
Output: Partition1, Partition2 // Partitioned data is the output
1. [Partition1, Partition2] = function createinitialPartitions(Relationvalues)
2. Partition1 = []; // initially both the Partitions would be empty
3. Partition2 = []; // Partition 2 is also empty
4. Partitionelements1 = 0; // Total number of elements in Partition1
5. Partitionelements2 = 0; // Total number of elements in Partition 2
6. 1streference=Relation values (1, 1) // Taking 1st element as the first reference value, every other document will be referred by the connection this reference
7. For each connection in Relation values //Finding the same document in the relation values
8.Similar_Connection=Find(Relationvalues(:,1)==1streference); // Looking for the first reference in the i.
        //entire list
// Finding the other connection of 1st reference
9.Similar_Connection_Value=Sum(All_Similar_Connection)/Total number of Connections
 .      // Finding out the average connection value
10. IfAvg_Value of Relation>Similar_Connection // If the other connection values has greater
//reference value then it would shift to the first //partition
11.Partion1[partitionelements1]=CurentDocument;
12. partitionelements1=partitionelements1+1;
13. Else
14. Partion2 [partitionelements2] =CurentDocument;
15. partitionelements2=partitionelements2+1;
16.End for each
17.End Algorithm
```

This algorithm performs the partition in the data elements based on similarity measurements using algorithm 2. For the first time, the first document will be set as the main reference value. Table 1 represents the sample relation value.

Table 1Sample relation value

Main Data	Connecting Data	Relation Value
1	2	0.23
1	3	0.41
1	4	0.26
1	5	0.11
2	3	0.24
2	4	0.28

Table 1 shows that Main Data (MD) 1 is connected to Connecting Data (CD) with a Relation Value (RV). Here 1→2 stands 0.23. Hence, 0.23 is the reference value for partition 1. Now MD1 is connected to MD2. Other MD2 connections are 2→3 and 2→4. The average other connection stands an average value of (0.24+0.28)/2--→0.26. The other connection value 0.26 is greater than the reference connection value 0.23 and hence 1 and 2 won't be in a similar cluster and the reference value will be 0.26 now.The proposed partition algorithm is evaluated on the base of True Positive, False Positive, Tue Negative, False Negative, Precision, and Recall. The evaluated results are established in section IV.

IV. RESULT AND DISCUSSION

This fragment explains the results obtained after the evaluation of the proposed work. For the evaluation, parameters like;True Positive, False Positive, True Negative, False Negative, Precision,and Recall are computed.

$$\text{True Positive } (T_p) = \frac{\text{Total true selected elements}}{\text{Total sample size}} \quad (6)$$

$$\text{False Positive } (F_p) = \frac{\text{False selected elements}}{\text{Total sample size}} \quad (7)$$

$$\text{True Negative } (T_n) = \frac{\text{True leftt samples}}{\text{Total sample size}} \quad (8)$$

$$\text{False Negative } (F_n) = \frac{\text{False left samples}}{\text{Total sample size}} \quad (9)$$

The high true positive rate represents a good partition value. If the Tpis high then it is clear that the data is kept in a good partition cluster and no issue occurs in retrieving the data. The proposed algorithm passed around 1000 queries to the partition and the Tp stood a good value of 0.68 on an average.

Table 2Evaluation of Tp, Fp, Tn and Fn

Total passed Query	10	100	200	500	1000
Tp	0.65	0.66	0.67	0.68	0.69
Fp	0.31	0.32	0.33	0.35	0.37
Tn	0.34	0.33	0.34	0.32	0.34
Fn	0.15	0.18	0.12	0.15	0.18

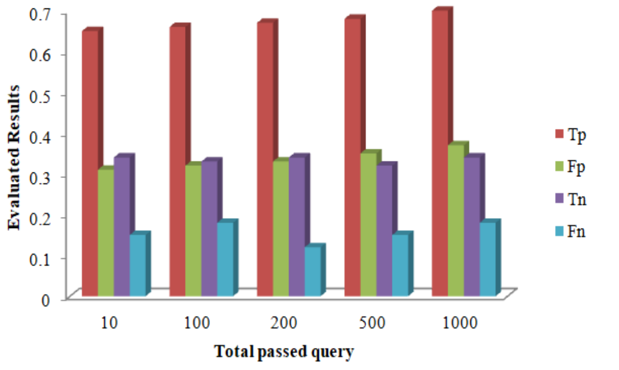

Fig.3True positive-False positive-True negative-False negative computation

In paradox to Tp, Fp is the result of bad selection from the partition. High Fp value represents that the elements were not placed in appropriate clusters and as a result, the reverted results against the passed queries were also false. The average Fp value for the proposed algorithm lies between 0.31 to 0.37 which is not bad at all. Tn is a total left bad sample. Tn is a controversial parameter. If Tn is high, it means that the search result is good as a lot of bad samples in relation to the query are left but it also denotes that there are a lot of irrelevant data in the partition. The Tn of the proposed structure is close to Fp and lies in between 0.32 -0.35. The evaluation for the same is shown in figure 3 and Table 2.Based on the Tp, Fp, Tn, and Fn, precision and recall are also calculated.

$$\text{Precision} = \frac{\text{Tp}}{\text{Tp+Fp}} \quad (10)$$

$$\text{Recall} = \frac{\text{Tp}}{\text{Tp+Tn}} \quad (11)$$

Table 3Evaluation of Precision and Recall

Total passed Query	10	100	200	500	1000
Precision	0.67	0.66	0.629	0.68	0.69
Recall	0.66	0.62	0.618	0.62	0.68

The precision and recall are consistent throughout for every range of queries as shown in Figure 4 and Table 3.

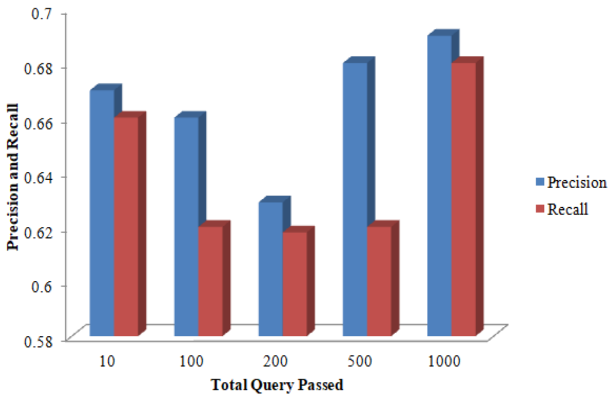

Fig.4 Precision and Recall

The maximum attained precision is 0.6925 for the query count of 1000. The recall value for same query count is 0.68. The precise structure of the proposed mechanism leads to a consistent precision and recall value.

Table 4 Evaluation of Precision for Cosine and Cosine+ Soft Cosine.

Total passed Query	10	100	200	500	1000
Cosine	0.54	0.59	0.60	060	0.59
Cosine and Soft Cosine	0.68	0.67	0.63	0.69	0.70

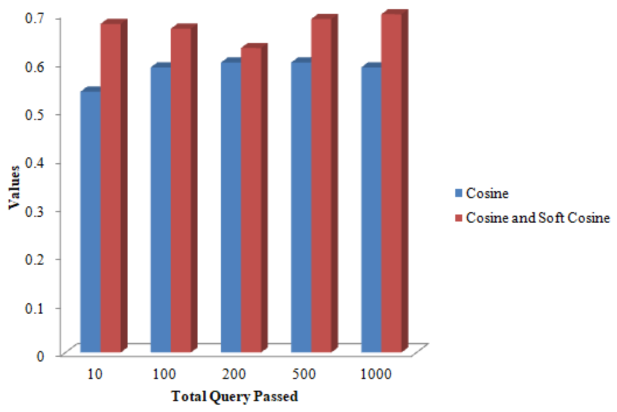

Fig.5 Comparison of Precision

Table 5Evaluation of Recall for Cosine and Cosine+ Soft Cosine

Total passed Query	10	100	200	500	1000
Cosine	0.51	0.54	0.53	0.57	0.59
Cosine and Soft Cosine	0.60	0.62	0.68	0.60	0.69

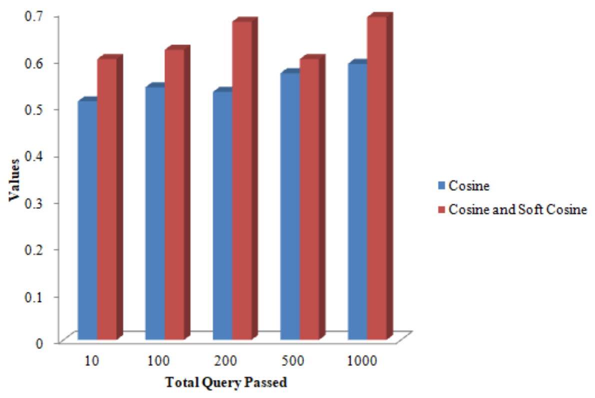

Fig.6 Comparison of Recall

Fig.5 and 6 are showing the comparison of Precision and Recall. Improvement is seen in both the figures by means of precision and recall when the hybridization of cosine and soft cosine are considered.

V. COMPARATIVE ANALYSIS OF PROPOSED AND EXISTING WORK

Table 6.Comparison of Precision and Recall of proposed work and existing work

Parameters	Huaping Guo et al. [20]	Proposed Work
Precision	0.4786	0.6925
Recall	0.5490	0.678

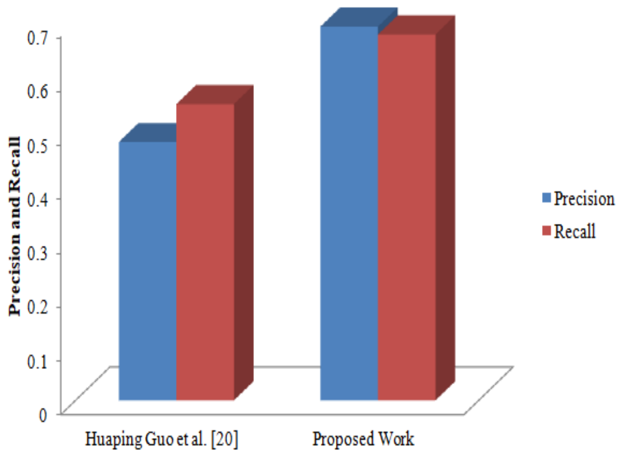

Fig.7 Comparison of proposed work with existing work for precision and recall

Fig.7 depicts the comparison of proposed work with the existing work [20] for Precision and Recall as well. In [20], the researchers have utilized k-means based cosine similarity measurement approach to compute the feature similarity among the cluster centroids and the data points for measuring the similarity among clustering result and the side information. [20] Has proposed an algorithm of clustering for data partitioning using the concept of a learning approach. The main drawback of the existing work with the consideration of proposed work is that in [20], only cosine similarity based k-means has been used whereas, we have used a hybridization of cosine and soft cosine as an enhanced mechanism. Accordingly, with this hybridization scheme, the values of Precision and Recall came out to be enhanced as contrasted with the existing work. There is an enhancement of proposed work in case of precision 2.28% and in case of a recall, there is an enhancement of 39.94%

Table 7Comparison of Accuracy of proposed work and existing work

Parameters	Kristjan Korjus et al. [25]	Proposed Work
Accuracy	61.1	73.6

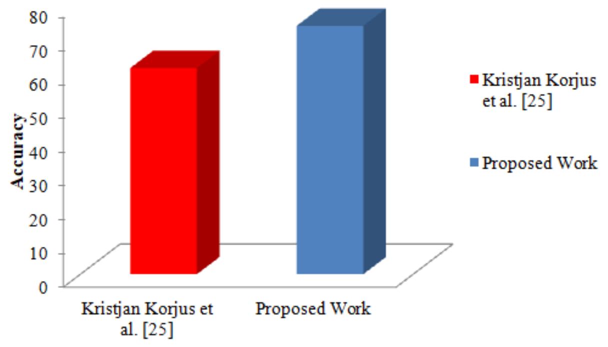

Fig.8 Comparison of proposed work with existing work for accuracy

Fig.8 and table 6 represents the comparison of proposed accuracy and achieved accuracy by Kristjan Korjus [25]. From the observation, it has been observed accuracy of proposed work is better than [25] and it has been improved by 16.98% by using the concept of hybrid similarity measurement technique for data partitioning.

VI. CONCLUSION AND FUTURE SCOPE

This paper has utilized the concept of the partition using a hybrid similarity index of cosine and soft cosine similarity. The proposed algorithm used the hybrid similarity to list the data as per the reference value of the connection. The proposed algorithm is evaluated using Tp, Fp, Tn, and Fn. Tn of the proposed mechanism is high and stood a value of .68 which indicates the high performance of the proposed algorithm. The proposed algorithm represents a low Fp value ranging from 0.32 to 0.38.Precision and Recall is also evaluated. Maximum precision of 0.6925 is attained for 1000 queries. For the same set of query set, a total recall value if 0.678 is attained. A comparison has been also been made to show the effectiveness of the research work. A comparison has been drawn between precision, recall and accuracy with [20] and [25].There is an enhancement of proposed work in case of precision 2.28% and in case of a recall, there is an enhancement of 39.94% with [20] and 16.98% improvement is observed of proposed work with [25]. The main reason of parameters improvement is hybridization of cosine similarity measurement technique with soft cosine for data partitioning. The current research work widely opens a lot of futuristic approaches. Cross-validation using machine learning would be interesting to see in this contrast.

REFERENCES

1. S. A. Anazi, H. A. Mahmoud and I. A.Turaiki, "Finding similar documents using different clustering techniques",*Procedia Computer Science*, Vol. 82, pp.28-34, 2016.
2. S.Bhatia, S.Tuarob, P. Mitra and C. L.Giles, "An algorithm search engine for software developers", In Proceedings of the 3rd International Workshop on Search-Driven Development: Users, Infrastructure, Tools, and Evaluation, ACM, pp. 13-16, 2011.
3. R. Daniel and A. K. Shukla, "Improving Text Search Process using Text Document Clustering Approach", *International Journal of Science and Research (IJSR)*, Vol. 3,No. 5, pp. 1424,2014.
4. V.C. Garcia , E. S. de Almeida, L.B. Lisboa, A. C. Martins, S. R. L. Meira, D. Lucredio and R. P. de M. Fortes, "Toward a code search engine based on the state-of-art and practice", *Software Engineering Conference*, 13th Asia Pacific, IEEE, pp. 61-70,2016.
5. X. Gu, H. Zhang and S. Kim, "Deep code search", In Proceedings of the 40th *International Conference on Software Engineering*, ACM,pp. 933-944,2018.
6. M. N. Najat, and A. M. Abdulazeez, "Evaluation of Partitioning Around Medoids Algorithm with Various Distances on Microarray Data", In the Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), *IEEE International Conference*,pp. 1011-1016,2017.
7. D. T. Nguyen,L. Chen and C. K.Chan, "Clustering with a multi viewpoint-based similarity measure", *IEEE transactions on knowledge and data engineering*, Vol. 24, No. 6, pp. 988-1001,2012.
8. S.C.Punithaa and M. Punithavalli, "Performance evaluation of semantic-based and ontology based text document clustering techniques", *Procedia Engineering*, Vol.30, pp.100-106, 2012.
9. M.S.B.Phridviraj, V. R. Krishna, C.Srinivas, C.V.GuruRao, "A novel Gaussian based similarity measure for clustering customer transactions using transaction sequence vector", *Procedia Technology*, Vol.19, pp.880-887,2015.
10. M.Ramakrishnan and D.Tennyson Jayaraj, "Modified k-Means algorithm for effective clustering of categorical data sets", *International Journal of Computer Applications*, Vol. 89, No. 7, 2014
11. Z. Tao, H. Liu and Y. Fu, "Simultaneous Clustering and Ensemble", AAAI,pp. 1546-1552, 2017.
12. Singh, C. Dabas and J. P. Gupta, "Cosine Similarity with Centroid Implication for Text Clustering of Document Files", *Indian Journal of Science and Technology*, Vol.9, No. 48, 2016.
13. Turcu, R. Palmieri, B. Ravindran and S. Hirve, "Automated data partitioning for highly scalable and strongly consistent transactions", *In Proceedings of International Conference on Systems and Storage*, ACM,pp. 1-11, 2014.
14. Z.Tao, H. Liu, H. Fu and Y.Fu, "Image Co-segmentation via Saliency-Guided Constrained Clustering with Cosine Similarity", AAAI,pp. 4285-4291,2017.
15. T. A. Vanderlei, F. A. Durão,A. C. Martins,V. C. Garcia, E. S. Almeida and S. R. de L. Meira, "A cooperative classification mechanism for search and retrieval software components*", In Proceedings of the ACM symposium on Applied computing*, pp. 866-871,2007.
16. Verma and A. Kumar, "Performance Enhancement of K-Means Clustering Algorithms for High Dimensional Data sets", *International Journal of Advanced Research in Computer Science and Software Engineering*, Vol. 4, No. 1,pp.5-9, 2014.
17. Z. Wang, J. Paul, B. He and W. Zhang, "Multikernel Data Partitioning With Channel on OpenCL-Based FPGAs",*IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 25, No. 6, pp.1906-1918, 2017.
18. Y. Xun, J. Zhang, X. Qin and X. Zhao, "FiDoop-DP: data partitioning in frequent itemset mining on Hadoop clusters", *IEEE Transactions on Parallel and Distributed Systems*,Vol. 28, No. 1, pp.101-114, 2017.
19. Y. Zhaoand and G. Karypis, "Soft clustering criterion functions for partitional document clustering: a summary of results",*In Proceedings of the thirteenth ACM international conference on Information and knowledge management*,pp. 246-247,2004.
20. H. Guo, J. Zhou and C.A. Wu, "Imbalanced Learning Based on Data-Partition and SMOTE", Information, Vol. 9, No. 9, pp. 238, 2018.
21. X. Gao and S. Wu, "Hierarchical Clustering Algorithm for Binary Data Based on Cosine Similarity", 8th International Conference on Logistics, Informatics and Service Sciences (LISS), Toronto, ON, pp. 1-6,2018.
22. W. L Xiang, Y. Z. Li, R. C. He, M.X. Gao, M.Q An, "A novel artificial bee colony algorithm based on the cosine similarity", Computers & Industrial Engineering, Vol. 115, pp.54-68, 2018.
23. Z. Tao, H.Liu, H.Fu and Yun Fu, "Image Cosegmentation via Saliency-Guided Constrained Clustering with Cosine Similarity", Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, pp. 4285-4291, 2017.
24. J. Wang, H. T. Shen, J. Song, J. Ji, "Hashing for Similarity Search: A Survey", arXiv preprint arXiv:1408.292, pp.1-29,2014.
25. K. Korjus, M. N. Hebart and R. Vicente, "An Efficient Data Partitioning to Improve Classification Performance While Keeping Parameters Interpretable", PloS one , Vol.11, No. 8, pp. 1-16, 2016.