International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys.14, No.1 (Jul-23)

http://dx.doi.org/10.12785/ijcds/140121

Data Mining for Non-Redundant Big Data Using Dynamic KMEAN Clustering

Saja Taha Ahmed[1]

[1]*Ministry of Education, Vocational Education Department, Baghdad, Iraq*
[1]*https://orcid.org/0000-0003-3069-7251, sajataha@ymail.com, drsajataha@gmail.com)*

Received 13 Sep. 2022, Revised 30 Apr. 2023, Accepted 14 May. 2023, Published 1 Jul. 2023

Abstract: There is an increasing demand for techniques that can process and collect valuable information from huge data in the Big Data era. Duplicates can seriously influence data processing and data mining, so the major challenge is finding as many duplicate records as possible. Data deduplication (or Redundancy Removal) removes redundant data and stores only one copy, promoting single instance storage. The main idea suggests using K-Means clustering for big data deduplication. K-Means Clustering, a localized optimization approach, is vulnerable to the starting point chosen from the cluster's center. The K-Means Clustering technique will produce more errors and bad cluster outcomes if the center of a defective cluster is used as the starting point. The suggested deduplication solution is based on the numeric conversion of the dataset and pre-processing them to extract useful information utilized by Dynamic K-Mean clustering (DKMEAN) to categorize replicated chunks. The proposed system greatly improves dataset quality and ultimately reduces resource consumption. It outperformed Traditional K-Means (TKMEAN) in terms of the number of detected redundant chunks, accuracy, the number of iterations, and efficiency.

Keywords: Data Mining, Data Deduplication, Clustering , Kmean algorithm, Big Data

1. INTRODUCTION

In the current digital era, the size of data has expanded substantially in numerous areas such as industry, business, science, and online applications. Enormous and powerful data servers have expanded rapidly in response to the massive advancement and development of the internet and online world technologies with the rise of data, sharing websites like Facebook, Flickr, and YouTube. Facebook, for example, reports 2.5 billion content every day, 105 terabytes of data every half hour, 300 million photographs, and 4 million movies. Twitter has a user base of 651 million people and sends out over 6000 tweets each second [1]. These data originate from a range of sources, some of which may have data quality issues. Duplicates are a fundamental issue that emerges from integrating different data [2].

Deduplication eliminates data redundancy by maintaining only one physical copy and referring to previously copied records [3]. Deduplication at the chunk level is more prevalent than deduplication at the file level as it identifies and prevents duplication at a higher specification. Separating input (a data stream) into many segments is the essential component of data deduplication. Chunking algorithms are usually divided into two types: Fixed-length chunking and variable-length chunking [4] [5]. Deduplication methods based on Content Dynamic Chunking can disclose 10-20% more duplication than fixed-size chunking [6].

Data de-duplication can be considered a clustering problem. The purpose is to group chunks that refer to the same physical entity together while isolating chunks that refer to other entities into distinct clusters. Clustering for de-duplication has a variety of different properties that set it apart from other clustering tasks. Many prominent clustering algorithms, such as k-means and k-median, use the number of clusters to output as an input. In de-duplication applications, this information is unknown [7] [8].

The unsupervised learning approach k-Means, which is partition-based, is noted for its ease of use and simplicity. However, the k-Mean algorithm has the following severe flaw [9] [10]:
1- The performance of the K-mean algorithm is substantially determined by the initial centroids, which are chosen at random, and the resulting clusters differ between runs for the same input dataset (i.e., converge to Local minima).
2- The traditional K-means algorithm's distance calculation procedure takes a long time to converge clustering results because it calculates the distance from each data object to each cluster's centroids in each iteration.

Given the various flaws in the traditional k-mean ap-

E-mail address: sajataha@ymail.com, drsajataha@gmail.com

proach. In this paper, we proposed a redundancy removal system with the following contributions:
1. The suggested system pre-processes big datasets to extract numeric information from them.
2. The statistical analysis is implemented based on certain criteria (in this work, chunk size) to automatically determine the number of clusters that can group chunks according to them. Since the number of clusters of big data is unknown.
3. The TKMEAN randomly assigns the points into clusters, our proposed DKMEAN equally assigns the clusters via sorting numeric records according to chunks' size which can take less time to converge and avoid local minima.
4. The TKMEAN computes the distance between all points and centroids of all clusters at each iteration which affects the performance of the algorithm. There is no need to calculate the distance each time in DKMEAN as the resulting clusters contain some data objects that remain in the same cluster after several iterations.

In brief, we develop a de-duplication technique based on complete data numeric conversion. The proposed DKMEAN used chunks' numeric records to match records inside a cluster to locate and eliminate duplicated records, reducing the number of record comparisons, and improving the algorithm's performance.

The remainder of the paper is organized as follows: In Section 2, we take a look at related works on data deduplication and KMEAN clustering. We describe our proposed system in Section 3. In Section 4, we look at the experiments and discuss the results, before concluding Section 5.

2. RELATED WORK

Researchers have made several efforts to raise the k-means algorithm's effectiveness and efficiency. All of the mentioned methods in this study address the same k means algorithm concerns, such as clustering huge datasets, the number of iterations in the algorithm, specifying the number of clusters, and selecting the initial cluster center.

Bide et al [11] proposed a new document clustering approach that didn't require a predetermined k value. Rather, it employed cluster labels as input and a cosine similarity metric to group comparable texts into the appropriate number of clusters. The experimental findings showed that when compared to a similar algorithm that was currently in use and was based on the F-measure, the new method exhibited a high level of accuracy.

Gopalani et al. [12] presented a distributed computing framework based on Spark and used it to implement the parallel K-means-based text clustering method in order to address the issue of the K-means clustering algorithm requiring too many iterations. Based on the k-means algorithm's shortcomings of excessive iterations and poor execution efficiency, this was achieved. Gupta et al [13] used k-means clustering to study an outlier problem. They aim to minimize variation among sample points inside the same cluster while discarding a small group of samples that could be regarded as outliers. The k-means clustering strategy with an outlier was suggested by the researchers. The method is adaptable to large datasets. In the experiments, the proposed algorithm was found to be accurate.

However, using k-means to choose the initial center at random will result in a local optimum. Values are vulnerable, the process is iterative and time-consuming. To increase sampling efficiency and discover the starting center point, Sardar and Ansari [14] used a MapReduce computing framework in association with the K-selection sorting approach for parallel sampling and a sample-based pre-processing strategy, resulting in a higher accuracy rate.

Based on optimal partitioning, a k-means initial grouping core selection method was proposed by Jin et al. [15]. The technique breaks the data samples first, then identifies the first cluster centers based on the sample distribution's properties. This approach enhances clustering accuracy and efficiency. The approach, on the other hand, will increase the number of recursions for high-dimensional sample spaces, increasing the complexity of the calculations and diminishing efficiency.

K-means has shown the selection of the K-center value and lower the iterations [16] through [21]. Due to difficulties with local optimization and the randomness in the choice of the initial center, the cluster number must be predetermined. We investigated more precise and effective k-means to achieve a high degree of accuracy.

3. THE PROPOSED SYSTEM

There are three stages to any deduplication system [26]: chunking, hashing and indexing, and matching. To process the data, the proposed deduplication systems work in two stages:
1- Convert file chunks into numeric information.
2- Apply the proposed DKMEAN approach to recognize redundant chunks.

The suggested deduplication systems partition files into chunks and find duplicated chunks by comparing their chunk information after clustering them into clusters using DKMEAN. A variety of features are utilized to maintain the relationships between files and chunks, which necessitate additional resources besides the deduplicated data. The chunk index holds the chunk information for the chunks that have been saved. Every deduplication system has a persistent index that contains the data required to recreate file contents using file templates. A file template contains a list of chunk identifiers. Each of these chunk identifiers is unique. The uniquely identifiable chunked data can be used to reconstruct the original file contents (referred to as logical data). To rebuild the logical data, the chunk identifiers are read, and their relevant data chunks are loaded and concatenated in a specific order.

A. NUMERIC INFORMATION GENERATION

Chunking is the first and most essential phase in the data deduplication process, in which a file or data stream is divided into small, isolated pieces so that chunking locations so that the chunking positions can be recognized to save time when duplicate chunks arise in the future. The chunking algorithm consists of two key steps:
1. For each byte sequence of a specified length, compute the code (i.e., hash value).
2. To use as a divisor, find the code values in the file that have the most appearances bytes.

The deduplication method splits files according to the most frequent patterns in the file, utilizing the option of determining the length of breakpoints or divisors. In other words, cut points indicate some of the files' most fundamental attributes. The selected code is utilized to define the breakpoints rather than using judges based on a criterion value at every rollover hash computation (i.e., standard Rabin fingerprint). Figure 1 shows the chunking process.

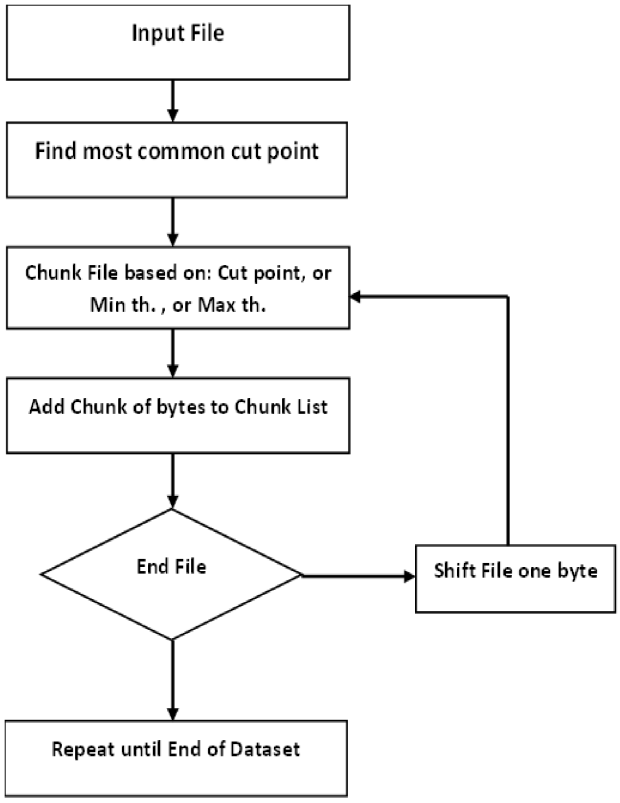

Figure 1. The Chunking Process.

Consequently, each file in the dataset is divided into chunks, with three codes for each chunk. Algorithm 1 showed the functions that generate these codes for each chunk. At this stage, file chunks will be converted into numeric information containing the chunk's cut point, size, code1, code2, and code3.

Algorithm 1. Chunks' Codes Generation

Objective: Generate three codes for each chunk.
Input: chunks as an array of bytes.
Output: three codes for each chunk.
Step1: Load chunks from a chunk list.
step2: Generate three random number arrays one for each code defined as R1, R2, and R3.
Step3: Set I to 0.
Step4: while not end of the chunk
Step5:Set sum1, sum2, sum3 to 0.
Step6: Compute
code1=sum1+chunk[i]*R1[i].
code2=sum2+chunk[i]*R2[i].
code3=sum3+chunk[i]*R3[i].
Step7: Put code1,code2,code3 in chunk information record.
Step8: Repeat Steps 1-7 until the end of the chunks.

Long codes (i.e., hashes) reduce the chance of a hash collision, but they take more time to execute and require more overhead information in the matching step, therefore the hash function chosen is heavily influenced by application accuracy. Common functions used in the redundancy elimination system are MD5 and SHA-1. The likelihood of hash collisions is often determined by the strength of the hashing method. The number of bits needed to store the hash value in SHA-1 is 160 bits and 128 bits for the MD5 signature, while the proposed method required 48 bits. Since hashing uses a lot of computational resources, employing the suggested hash algorithms drastically cuts down on the time of comparisons by effectively handling the collisions.

Non-redundant chunks are contained in the chunk bucket (or index table) in the suggested system; consequently, every segment generated by the system has its own id Number, which represents the segment index and accepts an integer number between 0 and M, with M increasing indefinitely for each segment. Table 1 shows the contents of the index table. Algorithm 2 describes the full conversion of big data to numerical information.

TABLE I. The Chunk Bucket Structure

Chunk ID	Cut Point Code	Size (Bytes)	code1	code2	code3
0	16320	627	55434	57737	21156
1	20348	159	54678	35617	33431
2	28590	512	12010	11164	45946

B. THE PROPOSED DKMEAN APPROACH

After passing the previous phases (chunking and hashing), the system must identify and delete duplicated chunks in the deduplication matching steps when a new file is entered. To begin, the traditional approach compares the chunks' code values; if they are the same, the algorithm checks the two chunks byte for byte; if they are equal, the system deletes the new one and adds a logical reference to the location of the old one. Otherwise, there's a collision;

Algorithm 2. Numeric Information Generation

Objective: convert files of a dataset to numeric information.
Input: File of different sizes and types as an array of bytes.
Output: A index table of chunks of information.
Step1: Load file from a dataset
Step2: Compute file Divisor based on most common bytes of predetermined length.
Step3: Break the file into chunks according to File Divisor.
Step4: Compute code1, code2 & code3 for each chunk.
Step5: Put chunk identifier, chunk size, file Divisor, code1, code2 & code3 into Index Table.
Step6: Repeat Steps 1-5 until the end of the dataset

the chunks aren't the same, thus the system would save the new one as a new chunk. This procedure takes a long time and places a significant amount of load on the system.

As a result, a novel technique based on the DKMEAN matching process will be employed to improve efficiency, i.e., reduce running time and resource consumption. This paper proposes employing chunks' information to enhance the comparison procedure using the size, cut point, and triple code values already obtained in the hashing step. This information serves as an identifier for each chunk. We introduced dynamic k-mean clustering to address the drawbacks of traditional K-mean clustering discussed earlier in this paper. The main idea is to calculate the number of clusters K relying on the size criteria for chunks that contain duplicated portions, then evenly set the initial clusters and compute the initial centroid for each group. By using these mechanisms, we avoid assigning random centroids for initial clusters. Because the divide-and-conquer strategy of isolating unique chunks from later clustering computations can have a substantial convergence effect. Additionally, removing redundant chunks as clusters emerge can help speed up the process of identifying duplicated chunks. The greatest chance of having identical chunks since they are either from the same file with the same code or from various files of the same size. Finally, three codes are checked by applying the similarity measure to them. Because it is preferable to group the chunks by their sizes and then compare three codes rather than byte-by-byte compare all of the relevant chunks, the test result reveals a significant improvement in the time and deduplication ratio.

Because of the excessive amount of numbers between the variables, the information of chunks would not be processed immediately. Determining the difference in distance or the amount of this value throughout the clustering process can be difficult. Normalization using equation 1 is one of the solutions for lowering the number of variables [22]:

$$Norm(X_i) = dfracX_i - X_{min}X_{max} - X_{min} \quad (1)$$

The variables' values are standardized to a range of 0 to 1. Before the calculation process, values on each variable are normalized such that the parameter's centroid value does not take precedence in calculating the distance between data [23]. Clustering often aims for high intra-cluster similarity and low intercluster similarity. To put it another way, we want chunks within a cluster to be as similar as possible, but chunks from separate clusters to be as diverse as possible. As a result, it's desired that all chunks of the same size are grouped in a single cluster. When defining clusters, we wanted to minimize the overall intra-cluster variations that measure the compactness of the clustering [24][25].

Let S and T be the clusters created by partitioning U. The distance between two items x and y belonging to S and T is given by d (x, y). The distance d (x, y) is calculated using well-known distance calculation methods like Euclidean. inter and intracluster distance are represented using equations 2 & 3, respectively. The proposed DKMEAN is stated in algorithm3.

$$Inter(S,T) = Min(d(x,y)x \in s, y \in T) \quad (2)$$

$$Intra(S,T) = Max(d(x,y)x, y \in s) \quad (3)$$

Algorithm 3. Dynamic K-Mean Clustering

Input: Raw data as an array of 0…N, 0…4,
where N is the number of chunks,and Raw data represents chunks of Numeric information
Output: A set of deduplicated clusters.
Step1: Compute frequency-based chunk size
and exclude unique chunks, put duplicated records in
Chunks_info array.
Set intcluster= number of chunks size have frequent chunks.
Normalize Raw data using Eq1.
Sort the raw data based on size.
Step2: Initialize cluster(s) equally based
on a number of initial clusters.
Step3. Repeat
(re)assign each chunk to the cluster to which the chunks are most similar,
based on the mean value of the Chunk_info in the cluster.
Update the cluster means, i.e., calculate the mean value of the objects
for each cluster. until no change.
Step4: Remove duplicated chunks from each cluster
and update the cluster.
Eq.2 is used to calculate the inter-cluster range.
Eq.3 can be used to calculate the intra-cluster range.
If the new intra-cluster value is less than the old intra-cluster value and
the new inter-cluster value is more than the old inter-cluster value,
go to step 3; otherwise, go to step 5.
Step5: End

It should be noted that clusters cannot be known in advance, but the data preprocessing and relying on a certain characteristic (i.e., size), the cluster number K is predetermined from the data rather than being fixed a priori as in TKMEAN. It can also help distinguish outliers from the data. When compared to TKMEAN, this added flexibility does not involve a large processing penalty, and DKMEAN convergence is often obtained faster. The figure 2 illustrated the proposed system procedure.

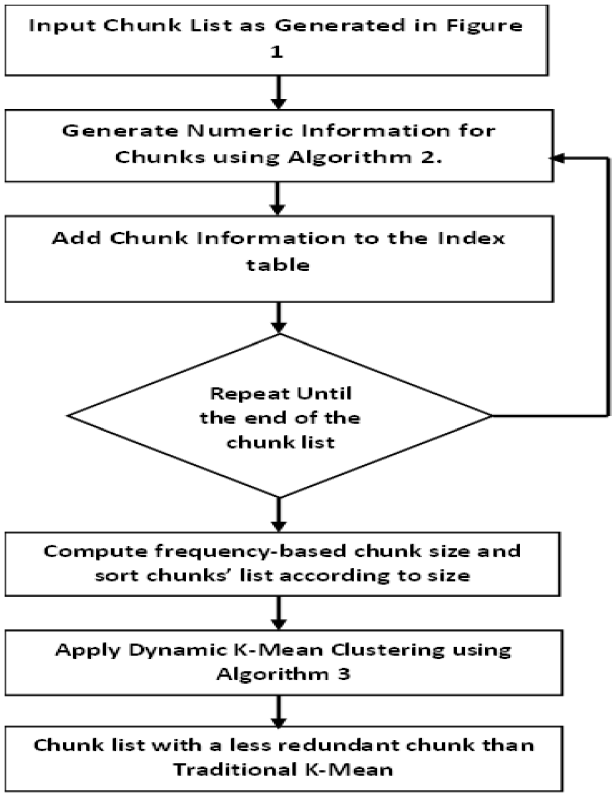

Figure 2. The Proposed System Flow.

4. THE EXPERMINTAL RESULTS

The input data for the recommended system comprises several files of varying sizes and types. The system processes each of these files separately, one at once. The suggested method was built and evaluated on three Linux file system dataset of 2,32 GB [26]. To conduct fair and reliable evaluations, the results are evaluated using the following metrics:

1- Accuracy: This metric describes the degree to which the predicted labels match the actual labels. The expected labels are the same as the class labels used to classify new instances. The accuracy is calculated by Equation (4).

Accuracy=((Correctly identified object)/(Total number of object))*100 (4)

2- Redundancy Removal: the number of nonredundant clustered records divided into total data, which measures the effectiveness of the deduplication process.

3- A number of iterations: the number of iterations taken by the K-mean clustering algorithm.

In order to set up the suggested system, C# and Visual Studio 2015 were used. The tests are carried out to examine and analyze the proposed system's behavior utilizing the chunk size. Furthermore, the generated chunks' numeric information is approximately 11,896,717 records, to demonstrate the effects of the proposed system and since it is more difficult to depict these chunks in a figure and make the clustering process more understandable, we selected around 37 records of them as minor test data to prove the impacts of the proposed system. Raw test data, as shown in Table 2 and Figure 3, has 20 duplicate chunks and 17 unique chunks, totaling 37 chunks.

TABLE II. Raw Test Chunks Information Distribution

Chunks No.	Chunk Size (byte)	Chunk Divisor	code1	code2	code3
0	138	6520466	57643	38208	18795
1	497	6520466	50499	40670	31049
2	337	6520466	26793	47199	2209
3	138	6520466	57643	38208	18795
4	497	6520466	50499	40670	31049
5	355	17069388	43313	4708	31816
6	221	17069388	49085	23553	63353
7	177	17069388	10546	1972	58887
8	196	17069388	30034	21779	13507
9	139	17069388	13180	40869	2986
10	134	17069388	4324	12169	20027
11	133	17069388	14465	37019	59646
12	179	17069388	12543	5786	64704
13	143	17069388	26181	11200	61779
14	153	17069388	44052	2353	26380
15	153	17069388	44052	2353	26380
16	156	17069388	50390	20999	57202
17	190	17069388	32028	13400	60469
18	355	17069388	43313	4708	31816
19	221	17069388	49085	23553	63353
20	177	17069388	10546	1972	58887
21	196	17069388	30034	21779	13507
22	140	17069388	22145	62147	36788
23	183	17069388	20736	35571	50575
24	141	17069388	41519	41904	42310
25	143	17069388	26181	11200	61779
26	153	17069388	44052	2353	26380
27	156	17069388	50390	20999	57202
28	190	17069388	32028	13400	60469
29	140	17069388	22145	62147	36788
30	183	17069388	20736	35571	50575
31	141	17069388	41519	41904	42310
32	146	17069388	34784	36527	38330
33	179	17069388	24926	40199	55736
34	146	17069388	39165	43762	48519
35	141	17069388	41519	41904	42310
36	156	17069388	50390	20999	57202

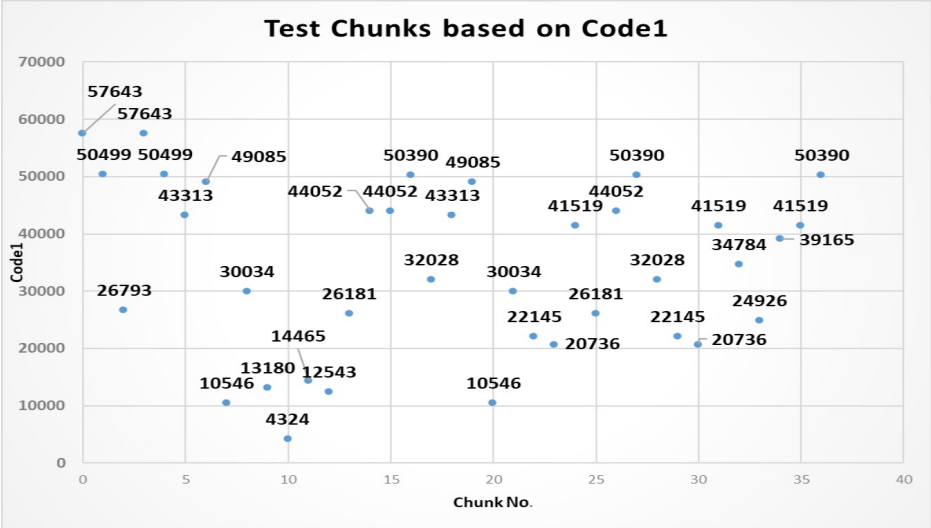

Figure 3. Raw Test Chunks Information Distribution Based On Code 1 Attribute.

TABLE III. DKMEAN Clustering Of Chunk Information

Chunk NO.	Cluster	Chunk Size	Divisor	code1	code2	code3	Redundant Chunks
0	1	138	6520466	57643	38208	18795	5
2		138	6520466	57643	38208	18795	
9		143	17069388	26181	11200	61779	
18		140	17069388	22145	62147	36788	
20		141	17069388	41519	41904	42310	
21		143	17069388	26181	11200	61779	
25		140	17069388	22145	62147	36788	
27		141	17069388	41519	41904	42310	
28		146	17069388	34784	36527	38330	
30		146	17069388	39165	43762	48519	
30		146	17069388	39165	43762	48519	
31		141	17069388	41519	41904	42310	
1	2	497	6520466	50499	40670	31049	2
3		497	6520466	50499	40670	31049	
4		355	17069388	43313	4708	31816	
14		355	17069388	43313	4708	31816	
10	3	153	17069388	44052	2353	26380	4
11		153	17069388	44052	2353	26380	
12		156	17069388	50390	20999	57202	
22		153	17069388	44052	2353	26380	
23		156	17069388	50390	20999	57202	
32		156	17069388	50390	20999	57202	
5	4	221	17069388	49085	23553	63353	1
15		221	17069388	49085	23553	63353	
6	5	177	17069388	10546	1972	58887	4
7		196	17069388	30034	21779	13507	
8		179	17069388	12543	5786	64704	
13		190	17069388	32028	13400	60469	
16		177	17069388	10546	1972	58887	
17		196	17069388	30034	21779	13507	
19		183	17069388	20736	35571	50575	
24		190	17069388	32028	13400	60469	
26		183	17069388	20736	35571	50575	
29		179	17069388	24926	40199	55736	

If the number of clusters is limited, there is a probability that dissimilar chunks will be grouped, but if the number of clusters is big, more similar chunks will be grouped in different clusters. Since the goal is to group redundant chunks in the same cluster, the proposed system preprocesses and refines raw data based on chunk size and finds 4 unique chunks that are transmitted directly to a deduplicated list of chunks. These chunks are considered an outlier that can affect an algorithm decision. The remaining chunks are aggregated into five clusters using DKMEAN, which can correctly identify all redundant chunks. Table 3 depicted all duplicated chunks belonging to a specific cluster. For instance, Cluster 1 has 5 duplicated chunks.

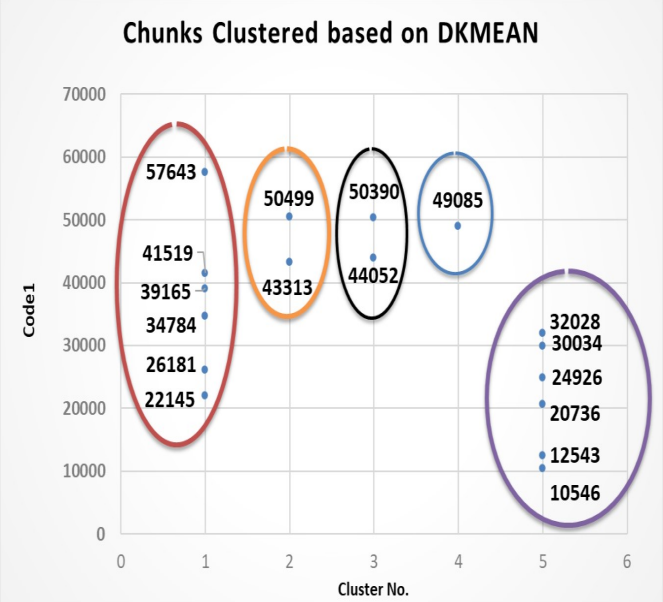

Figure 4. The Test Data After Applying DKMEAN Attribute.

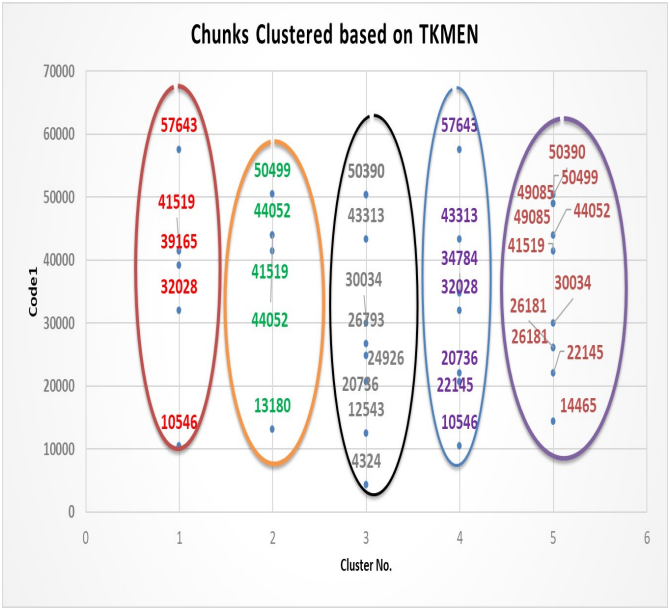

Figure 5. The Test Data After Applying TKMEAN

TABLE IV. Deduplicated Chunks clustered in five groups based on Code1.

code1	Clusters' No
57643	1
26181	1
22145	1
34784	1
39165	1
41519	1
50499	2
43313	2
44052	3
50390	3
49085	4
10546	5
12543	5
30034	5
32028	5
20736	5
24926	5

Figure 4 and Table 4 show the chunks after duplication has been removed, it appears that the dataset's 33 records were reduced to 17 records clustered into five groups, resulting in the removal of 16 redundant chunks.

The performance of the traditional KMEAN has experimented on the same dataset in Table 2, the conducted results showed a low effect of grouping duplicated chunks together in the same clusters. Table 5 shows the TKMEAN can detect 4 duplicated chunks of 16 ones. In comparison with DKMEAN can detect all duplication in chunks. Figure 5 shows TKEAM chunk clustering.

The big dataset of 11,896,717 chunks, all files chunked with a minimum threshold of 128 bytes and a maximum threshold of 1024 bytes based on 3 bytes divisor. The duplicated chunks detected by the proposed DKMEAN are 7825426 chunks and the unique chunks transmitted to deduplicated list of a dataset are 4071291. The inferred truth is the proposed system reduced the dataset from 2.32 GB to 802 MB. Thus, redundancy removal of the proposed system is about 65.78 percent. In contrast to the traditional Kmean detected 1595710 chunks of 11,896,717, thus the output of TKMEAN is 10,301,007. The redundancy removal is 13.5 percent.

The dataset is divided into two categories: 70% training and 30% testing. The improved strategy outperforms TK-MEAN in terms of the number of iterations and accuracy since DKMEAN clusters chunks according to their sizes and establishes clusters evenly by the initial value. Table 6 compares the accuracy and number of iterations of the traditional k-means algorithm and the enhanced approach. The modified algorithm considerably surpasses the original k-means algorithm in terms of accuracy, number of iterations, and efficiency, as shown in the above experiments.

One more significant result is drawn when implementing the TKMEAN shows different performance measures for the same dataset even fixing the number of clusters, TKMEAN algorithm provides different results in different runs because of its nature of randomly selecting initial centers even value of k is fixed, but this problem is overcoming by the proposed DKMEAN as it fixes the way initial centers are calculated.

TABLE V. Traditional KMEAN Clustering of Chunk Information

Cluster No	Size	Divisor	code1	code2	code3	Redundant
1	138	6520466	57643	38208	18795	
	177	17069388	10546	1972	58887	
	190	17069388	32028	13400	60469	
	146	17069388	39165	43762	48519	
	141	17069388	41519	41904	42310	
2	497	6520466	50499	40670	31049	1
	139	17069388	13180	40869	2986	
	153	17069388	44052	2353	26380	
	153	17069388	44052	2353	26380	
	141	17069388	41519	41904	42310	
3	337	6520466	26793	47199	2209	1
	196	17069388	30034	21779	13507	
	134	17069388	4324	12169	20027	
	179	17069388	12543	5786	64704	
	156	17069388	50390	20999	57202	
	355	17069388	43313	4708	31816	
	183	17069388	20736	35571	50575	
	179	17069388	24926	40199	55736	
	156	17069388	50390	20999	57202	
4	138	6520466	57643	38208	18795	
	355	17069388	43313	4708	31816	
	177	17069388	10546	1972	58887	
	190	17069388	32028	13400	60469	
	183	17069388	20736	35571	50575	
	140	17069388	22145	62147	36788	
	146	17069388	34784	36527	38330	
5	497	6520466	50499	40670	31049	2
	221	17069388	49085	23553	63353	
	133	17069388	14465	37019	59646	
	143	17069388	26181	11200	61779	
	221	17069388	49085	23553	63353	
	196	17069388	30034	21779	13507	
	140	17069388	22145	62147	36788	
	143	17069388	26181	11200	61779	
	153	17069388	44052	2353	26380	
	156	17069388	50390	20999	57202	
	141	17069388	41519	41904	42310	

TABLE VI. Performance Evaluation of TKMEAN and DKMEAN

Metrics	TKMEAN	DKMEAN
Accuracy	60%	95%
No. Iteration	115	25
Time	905	175

Any researcher using large data already has to deal with these restrictions:
1. The use of larger data implemented on basic computer hardware specifications is the restriction of the big data study.
2. Utilizing huge data sets makes visual data mining and visual-exploratory analysis unfeasible.

5. Conclusions

The traditional Kmean has clustering-related problems since it relies on arbitrary center selection and is vulnerable to the local optimization issue. The proposed approach solves this problem by estimating the number of clusters based on chunk size and preprocessing the dataset to determine the best number of clusters. Furthermore, the clusters are initially selected in the same way, based on evenly distributed initial clusters of sorted data, to achieve stable and quick convergence and avoid random selection. The findings reveal that, the experiments on the large Linux dataset demonstrate the proposed algorithm has remarkable clustering performance when compared to the traditional algorithm in terms of the number of redundant chunks, accuracy, number of iterations, and time. Since the chunks of duplicated files are divided based on the same divisor, further research can focus on how chunks clustered using another criterion like the file divisor.

References

[1] Fahad, SK Ahammad, and Md Mahbub Alam. "A modified K-means algorithm for big data clustering." International Journal of Science, Engineering and Computer Technology 6, no. 4 (2016): 129. DOI: 10.13140/RG.2.2.33836.31360

[2] Udechukwu, Ajumobi, Christie Ezeife, and Ken Barker. "Independent de-duplication in data cleaning." Journal of Information and Organizational Sciences 29, no. 2 (2005): 53-68. DOI: https://doi.org/10.1007/978-0-387-39940-9_596

[3] Reddy, B. Tirapathi, and MVP Chandra Sekhara Rao. "Filter based data deduplication in cloud storage using dynamic perfect hash functions." International Journal of Simulation Systems, Science & Technology (2018). DOI 10.5013/IJSSST.a.19.04.08

[4] Periasamy, J. K., and B. Latha. "Efficient hash function–based duplication detection algorithm for data Deduplication deduction and reduction." Concurrency and Computation: Practice and Experience (2019): e5213. DOI: 10.1002/cpe.5213

[5] Xia, W., Zou, X., Jiang, H., Zhou, Y., Liu, C., Feng, D., Zhang, Y., 2020. The design of fast content-defined chunking for data deduplication based storage systems. IEEE Trans. Parallel Distrib. Syst. 31 (9), 2017–2031. DOI: 10.1109/TPDS.2020.2984632.

[6] Xia, Wen, Dan Feng, Hong Jiang, Yucheng Zhang, Victor Chang, and Xiangyu Zou. "Accelerating content-defined-chunking based data deduplication by exploiting parallelism." Future Generation Computer Systems 98 (2019): 406-418. DOI: 10.1016/j.future.2019.02.008.

[7] Kushagra, Shrinu, Hemant Saxena, Ihab F. Ilyas, and Shai Ben-David. "A semi-supervised framework of clustering selection for de-duplication." In 2019 IEEE 35th International Conference on Data Engineering (ICDE), pp. 208-219. IEEE, 2019. DOI: 10.1109/ICDE.2019.00027.

[8] Manghi, Paolo, Claudio Atzori, Michele De Bonis, and Alessia Bardi. "Entity deduplication in big data graphs for scholarly communication." Data Technologies and Applications (2020). DOI: 10.1108/DTA-09-2019-0163.

[9] Katara, Juhi, and Naveen Choudhary. "A modified version of the K-means clustering algorithm." Global Journal of Computer Science and Technology (2015). DOI: 10.1109/34.927466.

[10] Bhatia, M. P. S., and Deepika Khurana. "Experimental study of Data clustering using k-Means and modified algorithms." International Journal of Data Mining & Knowledge Management Process 3, no. 3 (2013): 17. DOI: 10.5121/ijdkp.2013.3302.

[11] Bide, P.; Shedge, R. Improved Document Clustering using k-means algorithm. In Proceedings of the 2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), Coimbatore, India, 5–7 March 2015; pp. 1–5. DOI: 10.1109/ICECCT.2015.7226065.

[12] S. Gopalani, R. Arora, and S. Gopalani, "Comparing Apache Spark and MapReduce with performance analysis using Kmeans," International Journal of Computer Applications, vol. 113, no. 1, pp. 8–11, 2015. DOI: 10.5120/19788-0531.

[13] Gupta, S.; Kumar, R.; Lu, K.; Moseley, B.; Vassilvitskii, S. Local search methods for k-means with outliers. Proc. VLDB Endow. 2017, 10, 757–768. https://doi.org/10.14778/3067421.3067425.

[14] T. H. Sardar and Z. Ansari, "An analysis of distributed document clustering using MapReduce based K-means algorithm," Journal of Ae Institution of Engineers (India) Series B, vol. 101, no. 2, pp. 1–10, 2020. DOI: 10.1007/s40031-020-00485-2.

[15] T. K. Jin, J. Song, and C. S. Ah, "Optically readable waveguide integrated electrochromic artificial synaptic device for photonic neuromorphic systems," ACS Applied Electronic Materials, vol. 2, no. 7, pp. 2057–2063, 2020. Doi: 10.1021/acsaelm.0c00314.

[16] A. A. Aldino, D. Darwis, and A. T. Prastowo, "Implementation of K-means algorithm for clustering corn planting feasibility area in south Lampung regency," Journal of Physics: Conference Series, IOP Publishing, vol. 1751, no. 1, pp. 012– 038, 2021. DOI: 10.1088/1742-6596/1751/1/012038.

[17] J. Rejito, A. Atthariq, and A. S. Abdullah, "Application of text mining employing k-means algorithms for clustering tweets of Tokopedia," Journal of Physics: Conference Series, IOP Publishing, vol. 1722, no. 1, pp. 012–019, 2021. DOI: 10.1088/1742-6596/1722/1/012019.

[18] C. Li, F. Kulwa, J. Zhang, Z. Li, H. Xu, and X. Zhao, "A review of clustering methods in microorganism image analysis," Advances in Intelligent Systems and Computing, pp. 13–25, 2021. DOI: 10.1007/978-3-030-49666-1_2.

[19] X. Chen and Y. Yang, "Hanson–Wright inequality in Hilbert spaces with application to *K*-means clustering for non-Euclidean data," Bernoulli, vol. 27, no. 1, pp. 586–614, 2021. https://doi.org/10.3150/20-BEJ1251.

[20] M. B. Gesicho, M. C. Were, and A. Babic, "Evaluating performance of health care facilities at meeting HIV-indicator reporting requirements in Kenya: An application of K-means clustering algorithm," BMC Medical Informatics and Decision Making, vol. 21, no. 1, pp. 1–18, 2021. https://doi.org/10.1186/s12911-020-01367-9.

[21] M. Ghadiri, S. Samadi, and S. Vempala, "Socially fair k-means clustering," in Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, pp. 438–448, Toronto, ON, Canada, March 2021.

[22] Syakur, M. A., B. K. Khotimah, E. M. S. Rochman, and Budi Dwi Satoto. "Integration k-means clustering method and elbow method for identification of the best customer profile cluster." In IOP Conference Series: Materials Science and Engineering, vol. 336, no. 1, p. 012017. IOP Publishing, 2018. doi:10.1088/1757-899X/336/1/012017.

[23] Virmani, Deepali, Shweta Taneja, and Geetika Malhotra. "Normalization based K means Clustering Algorithm." arXiv preprint arXiv:1503.00900 (2015).

[24] Alsuhaim, Amjad F., Aqil M. Azmi, and Muhammad Hussain. "Improving the Retrieval of Arabic Web Search Results Using Enhanced k-Means Clustering Algorithm." Entropy 23, no. 4 (2021): 449. DOI: 10.3390/e23040449.

[25] Sreedhar, Chowdam, Nagulapally Kasiviswanath, and Pakanti Chenna Reddy. "Clustering large datasets using K-means modified inter and intra clustering (KM-I2C) in Hadoop." Journal of Big Data 4, no. 1 (2017): 1-19, https://doi.org/10.1186/s40537-017-0087-2.

[26] Saja Taha Ahmed, Loay E. George, Lightweight hash-based de-duplication system using the self detection of most repeated patterns as chunks divisors, Journal of King Saud University - Computer and Information Sciences, Volume 34, Issue 7, 2022, Pages 4669-4678, ISSN 1319-1578, https://doi.org/10.1016/j.jksuci.2021.04.005.

References

DR. Saja Taha Ahmed Ph.D. holder, University of Information Technology and Communication, Informatics Institute of Postgraduate Studies. MSC degree from the College of Science, Department of Computing, University of Baghdad. Higher Diploma in computer science. I am a lecturer with the vocational education department in the ministry of education.