MongoDB Clustering using K-means for Real-Time Song Recognition

Murtadha Arif Bin Sahbudin, Marco Scarpa and Salvatore Serrano
Department of Engineering, University of Messina
Messina, C.da Di Dio - Villaggio S. Agata - 98166 Messina Italy
Email: (msahbudin, mscarpa, sserrano)@unime.it

***Abstract*—Recently, the increased competition in song recognition has led to the necessity to identify songs within very huge databases compared to previous years. Therefore, information retrieval technique requires a more efficient and scalable data storage framework. In this work, we propose an approach exploiting K-means clustering and describe strategies for improving accuracy and speed. In collaboration with an audio expert company providing us with 2.4 billion fingerprints data, we evaluated the performance of the proposed clustering and recognition algorithm.**

I. Introduction

Song recognition is a process of identifying a song segment either from a digital or an analogue audio source. There are different applications of such system; song rankings based on radio/TV broadcasting or streaming, protection of song copyright, or automatic recognition of songs a person wishes to identify while listening to it. Important information such as song title, artist name, and album title can be provided in an instant. Given the high demand for such an application, several approaches have already been studied based on songs fingerprinting recognition such in [1], [2], [3], [4], and [5]. Nowadays, the state of the art of recognition techniques are those developed by Shazam [6], [7] and SoundHound [8], [9]. These services are widely known from their mobile device applications.

Recently, there has been a huge increase in the number of songs in the music industry [10]. With large datasets, it has becoming more difficult to manage and identify songs using a traditional relational database management system. Common linear search technique which checks the presence of every fingerprint in an array one at a time has a noticeable decrease in performance for large datasets [11]. Therefore, the information stored requires a scalable database framework to satisfy the execution time, memory use and computational resources for retrieval purpose [12].

K-means is one of the most common algorithms used in clustering data. Clustering is a method of grouping data into adjacent partitions so that objects with similar features to each other will collate into the same clusters [13]. Once the groups are defined, any new data can be easily assigned to the correct group [14]. Based on the clusters, fingerprints recognition will be narrowed into a smaller search space which benefits much faster response.

In this research, we proposed a K-means clustering for song fingerprint collections in MongoDB. We also provide the recognition algorithm for song fingerprints from the clusters and next, we evaluate the performance and accuracy of our recognition approach using a real-time input stream audio. However, we do not provide any audio fingerprint generation; instead, we focus on the analysis of data distribution and classification of song fingerprint database in order to improve the recognition method.

The paper is organized as follows: Section II, summarizes the current work and systems from a project collaboration with a company working in the audio processing field. In Section III, we have analyzed the songs fingerprints database provided by the company. In Section IV, we described the technique for deploying K-means clustering in MongoDB. The songs information retrieval and recognition method are described in Section V; whereas the evaluation of our implemented work is presented in Section VI. Finally, Section VII proposed some future work and planning.

II. Current Work For Song Database Platform

This work borns in the context of a collaboration with a company specialized in real time parallel songs recognition. The fingerprint database collection provided by the company was used in this research. The main objectives of the collaboration were to gain insight into the industrial needs and problem in this domain. However, the company is anonymous due to legal privacy and confidentiality agreement which is still in the midst of negotiation. In the next section, we discuss the current framework and challenges.

A. Song Fingerprint Database

MongoDB is a Non Structured Query Language (NoSQL) database structure, which allows data collection to be stored in an array or JavaScript Object Notation (JSON) like structure [15]. Therefore, each record will have more information stored without constrains of a pre-defined field such as a relational database.

Currently, we acquired a collection of fingerprints used in actual songs recognition. At present, we are not able to reveal the technique used for generating these fingerprints. The representation values of the fingerprints are as below:

- Each song is represented by a sequence of fingerprints.
- Each fingerprint is an integer value.
- Each fingerprint represents a chunk of real audio playtime; we denote its time length with δ.

978-1-5386-9223-3/19/$31.00 ©2019 IEEE

- The database contains hundred thousand songs which are associated with approximately 2.4 billion fingerprints.

This large number of fingerprints are stored in a single collection and sorted accordingly by the song number (key). Here, the main issue during retrieval is attributed to non-classification of boundaries for the fingerprints values which would result in an exhaustive search; this is due to the fact that we want to recognize songs from a continuous songs streaming without any information on their boundaries.

B. Recognition Approach in Real-Time Broadcast Stream

For a successful song information retrieval, it is essential to have robust and fast recognition. Major clients such as music labels, producers, promoter, radio stations require information on trending songs, airtime schedule, and song versions. Hence, they demand an application that is capable of producing a fast and accurate information. For this reason, the company has developed an application that monitors and listen to several radio broadcast channels in real-time (on air FM/AM and in online streaming). These sources may include noise distortion, tempo and pitch shift.

The legacy solution was by pre-loading all fingerprints data into central memory for faster recognition. Currently, they implemented two version; one with a 32-bit system which runs on seven 1.5 GB of central memory per instances. Meanwhile, the 64-bit system run on a single instance with the usage of 12 GB of central memory. However, the drawback of this method is that it could not accommodate the increasing number of fingerprints in the future. To overcome it, we have proposed a more scalable big data framework using MongoDB K-means clustering. In addition, a new recognition algorithm also was required for the new clustered collection. We also compared the performance from both the legacy system (non-clustered) and the new clustered database.

III. Fingerprints Distribution Analysis

We performed a fundamental analysis of data using histograms. The purpose of this analysis was to have a representation of the fingerprint value and range. Apart from that, this analysis provided justification for using the K-means method.

Sample No.	Total fingerprints	Selection Segments
1	$2 \cdot 10^4$	Single random song
2	$2 \cdot 10^5$	Sorted by song keys
3	$2 \cdot 10^6$	Sorted by song keys
4	$2 \cdot 10^6$	Random
5	$2 \cdot 10^7$	Random
6	$2 \cdot 10^9$	Complete

TABLE I: Songs fingerprints datasets

The first approach consisted of extracting datasets samples in ranges such as TABLE I. The result shown in Fig. 1 is based on the overall 2.4 billion samples. The *X-axis* describes the fingerprint value range, while *Y-axis* describes the frequency of fingerprints falling into equally spaced bins between 512 and $4.2 \cdot 10^9$ and steps of 1024. We obtained a near identical frequency distribution of fingerprints for all the samples from

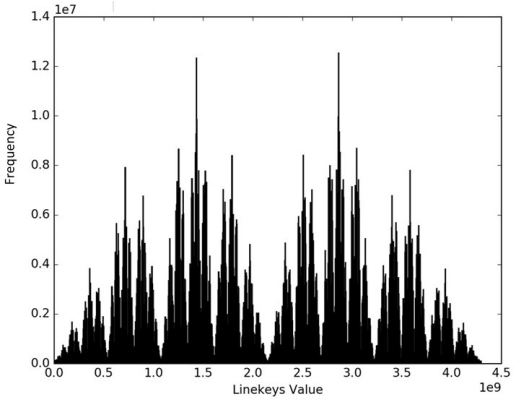

Fig. 1: $2.4 \cdot 10^9$ Fingerprint distribution

TABLE I. We concluded for this particular datasets, that at any given sample it would have similar fingerprint distribution. This is based on the evidence of similar pattern characteristic:

- A mirror axis between the range $2.0 \cdot 10^9$ and $2.5 \cdot 10^9$ approximately half position of the overall dataset.
- Two maximum peaks at near range $1.5 \cdot 10^9$ and $3.0 \cdot 10^9$.
- The maximum and minimum boundaries value ranges between 0 and $3.0 \cdot 10^9$.

Fingerprint frequency diagram suggests us that fingerprint distribution is not uniform and it could be modelled by a multimodal distribution, thus the K-means clustering approach could be effective.

Focusing on this findings, we were able to perform the K-means computation to obtain a clustered fingerprint collection associated with centroids value. In the next section, we provided requirements and implementation of the clustering method.

IV. MongoDB and K-means Clustering

K-means clustering is used when we have unclassified data. It performs an unsupervised algorithm for a large volume of data. Working with the fingerprint collections that we have, K-means classification organizes the fingerprints into subgroups. As such, those objects in the same group (clusters) are more similar to each other. Although K-means is mostly used for high dimensional data classification, in this research, we take advantage of the centroid value as distance point when performing a nearest computation. The basis for using K-means is that we require to produce clusters of relatively uniform datasets size.

K-means clustering algorithm partitions n data into k clusters in which each data is associated to the cluster with the nearest mean intra-cluster distance. Thus, it produces many different distinct clusters. The main purpose of K-means is thus to minimize intra-cluster distances. This is obtained by defining the index J as follows

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n_j} ||x_i^{(j)} - C_j||^2 \,, \tag{1}$$

where C_j, with $1 \leq j \leq k$ is the mean value of the j-th cluster, and $x_i^{(j)}$ represents a fingerprint falling in the cluster j. Usually the C_j are named *centroid*s. The K-means algorithm discovers the sets of k clusters, $k+1$ boundaries and k centroids, minimizing the optimization index J. More formally, it computes the set of fingerprints partitions $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k\}$ resulting from

$$\underset{\mathcal{P}}{\operatorname{argmin}}\, J \tag{2}$$

The centroids C_j of the clusters are used as reference points. However, we are required to define the number of clusters k prior computation.

Fig. 2, shows the overview of the clustering process in our implementation. We started with the original collection of the fingerprints stored in MongoDB. Based on the fingerprint

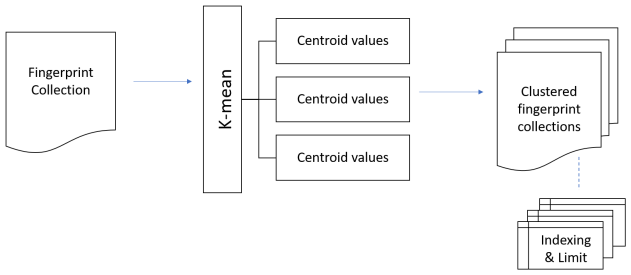

Fig. 2: K-means clustering implementation overview

distribution analysis, we select a reduced data sample of $2 \cdot 10^6$ (random segment) from the overall collection. There is no specific limitation of data can be performed by the K-means. However, this depends on the computing resources, time constraint, and other hardware specification.

Further on, once the computation has completed, centroids values are produced and they represent a key value for the cluster collection in MongoDB. We then perform a computation of fingerprints nearest distance to the centroids value and migrate the data to subgroup collections.

A. Stepwise Implementation of K-means

In our experimentation, we aimed to produce a number of $k = 10,000$ clusters with key centroids values as reference. In the next phase, we distributed the song fingerprints into the new clustered collections.

1) Batch Data Processing:
In managing a significant amount of data for K-mean computation, the initial approach was to divide the first instances into blocks by applying a segmentation of data. We then dumped the model from the computation using the *pickle* data structure in python. Subsequently, we used the previous model values and incrementally processed them until all the fingerprint datasets were processed. Code Listing 1 shows the steps of implemented algorithm using python *scikit-learn* library for K-means [16].

1) The algorithm selects $10,000$ points as first elements of the *n_cluster* centres.
2) We *chunk* $100,000$ of fingerprints and load them into memory and, using these data, we feed to the K-means computation. Each cluster centre was recomputed as the average of the points in that cluster.
3) computation at item 2) is repeated until no further changes in the assignment of fingerprints to clusters is experimented.

```
def dumpFitKmean(n_clusters,i,chunk,chunksize):
    if i==1:
        X = chunk
        mbk = KMeans(init='K-means++',n_clusters=int(n_clusters), n_init=1,
                random_state=42)
    else:
        X = chunk
        mbk = getFitKmean(i-1)
        mbk.fit(X)
    pickle.dump(mbk,open("pickelDump"+str(i)+".p", "wb"))

def getFitKmean(i):
    dumpFile = "pickelDump"+str(i)+".p"
    mbk = pickle.load(open(dumpFile, "rb"))
    return mbk

if __name__ == "__main__":
    i=1
    chunksize = 100000
    for chunk in pd.read_csv(filename, chunksize=chunksize, header=None):
    dumpFitKmean(n_clusters,i,chunk,chunksize)
    i=i+1
```

Listing 1: Python k-mean computation

2) Building Cluster Segmentation in MongoDB:
Once the centroid values were produced, we computed the distance of each fingerprint $f^{(i)}$ to each centroid to find its partition association by evaluating the following optimization problem:

$$p^{(i)} = \underset{j=1}{\overset{k}{\operatorname{argmin}}} ||f^{(i)} - C_j|| \tag{3}$$

In obtaining the distance for a fingerprint $f^{(i)}$, we perform an Euclidean distance between $f^{(i)}$ and every centroid value C_j. From the distance function list, we applied argmin to get the minimum distance. Thus, we would acquire the fingerprint association $p^{(i)}$ with its matching cluster $\mathcal{P}_{p^{(i)}}$.

V. Song Recognition and Information Retrieval

In the recognition phase, we need to identify the fingerprints sequence in our clustered fingerprints database. The algorithm performs a sequential window search in two stages. First, we associate each fingerprint in the query sequence to the nearest centroid value. Then, we perform an in-depth search within the associated cluster values and we obtain a set of candidates. Here, we apply the similar algorithm as the clustering process to obtain an acceptable accuracy in results.

The advantage of the adopted cluster organization is that it is possible to apply a binary search technique [11]. It first finds the position of a particular fingerprint query value within a sorted collection. In each step, this method examines and search the nearest key value with respect to the middle centroids key value of the given sorted fingerprint.

A. Real-Time Slide Window

Fig. 3 depicts the approach we implemented for identifying a song based on a stream of input fingerprints. As there shown, we chunk the fingerprint query stream into sizable windows

Fig. 3: Fingerprint Windows Slides Recognition

segments. This method of sliding window is a standard technique in information retrieval for detecting matching sequence. Each window represents a segment of time in the current song. As earlier mentioned, each fingerprint represents δ time instant of an audio source. Configurable fingerprints windows sizes, e.g., 500, 1000, 2000, 3000, 5000 or 6000 which translate to the actual time segment are important. We suggest using a window size not exceeding 6000 fingerprints such as this represents about 60 seconds of actual audio.

Each fingerprint is not unique; thus it could appear in several songs and then it will yield a set of song candidates. However, by means of occurrences scoring, we will eliminate such candidates and obtain a winner. We segment a sub-window of recognition for acquiring fingerprint search from database cluster. This sub-window contains $3/4$ of the fingerprints included in a window. The purpose of this sub-segment window is to reduce the recognition time. Moreover, the selected fraction size weight is sufficient to represent the overall fingerprints in the window.

B. Fingerprint Cluster Identification and Recognition

In order to reach an accurate result in the search, we set two stages in the recognition process for each fingerprint query. Firstly, by using the following eq. (4)

$$p^{(i)} = \underset{j=1}{\overset{k}{\operatorname{argmin}}} ||q^{(i)} - C_j||\,, \tag{4}$$

we compute the cluster $p^{(i)}$ which contains the nearest centroid. There, the fingerprint values $q^{(i)}$ is compared with every centroid value C_j and it evaluates to the index $p^{(i)}$ of cluster with the nearest centroid to $q^{(i)}$. When the cluster $p^{(i)}$ is identified, we select the nearest fingerprint to $q^{(i)}$ inside it by evaluating

$$wf^{(i)} = \underset{j}{\operatorname{argmin}} ||q^{(i)} - f_j^{p^{(i)}}||, \tag{5}$$

thus selecting a single fingerprint value $wf^{(i)}$. A set of songs stored in the collection and containing $wf^{(i)}$ in their sequence will be associated to each winner fingerprint $wf^{(i)}$. For these songs, we set to "1" an entry in a column array having an entry for each song to recognize. A sequence of N arrays is collected into the matrix $\mathbf{S}$. Thus a generic element $\mathbf{S}_{k,i}$ is 1 whether the song k is a found candidate at the i-th position inside the N long window, or 0 otherwise.

C. Candidates Scoring

In the previous step, we have obtained a list of candidate songs from a single fingerprint query. However, we need to evaluate the overall result within the fingerprint sequence in the specified window. To this purpose, we compute the frequency $F_k^{(n)}$ of each song found in a window (indexed n) of N fingerprints by evaluating

$$F_k^{(n)} = \sum_{i=1}^{N} \mathbf{S}_{k,i}. \tag{6}$$

The highest value of song frequency in the n-th windows will denote the winning candidates in that window segment. We evaluate it solving the following problem:

$$ws^{(n)} = \underset{k}{\operatorname{argmax}}\, F_k^{(n)} \tag{7}$$

While the winning song $ws^{(n)} = ws^{(n-1)}$ we increment the value of a specific counter. As soon as the winning song $ws^{(n)} \neq ws^{(n-1)}$, we can evaluate the number W of subsequent windows with the same winner and reset the counter to 1. As a consequence, the airtime song duration is evaluated as $d = N \cdot W \cdot \delta$.

VI. Evaluation and Performance

Given the previous recognition result of non-clustered approach, we set out to evaluate the performance of the new clustered database recognition. Theoretically, the performance in terms of memory occupancy should improve as the objective was to reduce the search space within the database. However, we should prove we haven't a degradation in terms of recognition accuracy.

A. Recognition Accuracy

To evaluate the accuracy, we tested our approach on 11 minutes of a mix stream audio. The streaming audio is then converted into a sequence of fingerprints with no indication of changes in the song. This set of a query will test the robustness of the recognition algorithm.

Windows size	Precision	Recall	Airtime accuracy
$Non-cluster$	80%	20%	80%
500	80%	20%	80%
1000	78%	30%	78%
2000	78%	30%	75%
3000	77%	50%	72%
5000	75%	60%	68%
6000	75%	80%	65%

TABLE II: Precision, Recall and Airtime Accuracy Results

For the song recognition, we had to make a trade-off between retrieving a high recall and avoiding false positives. In TABLE II, the window size selection determines the recall percentage as the larger window size, the higher the recall obtained. Nevertheless, we were able to maintain the precision as similar to the non-clustered database. Though 80% of the same accuracy, this validates the result match. Also, we achieved a good result of airtime accuracy. The remaining 20% unidentified result was due to the pitch shifting audio query which is another interesting research area within this field.

B. Recognition Speed

Once we have established a positive result of the recognition, the most distinct performance compared to the non-clustered is the recognition speed. These findings prove that a similar positive result could be achieved in a shorter period of time. We measure the capability of the algorithm with different sets of windows sizes configuration.

1) Windows Performance: In Fig. 4, we observed the performance in different windows sliding segments. Each window segment will yield a result of the final candidates' songs. The changes in song result for each window perform the best with $windowsize = 6000$. The completion time for each song result superseded others window sizes. However, as mentioned earlier the threshold of windows size is important as it would affect the precision and recall results.

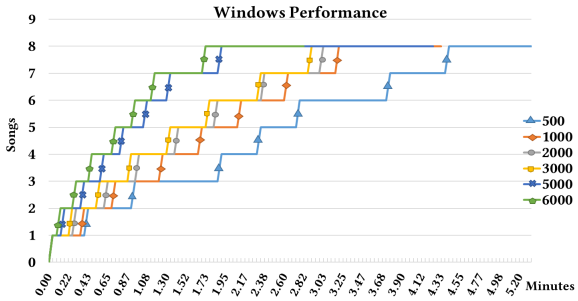

Fig. 4: Windows Sizes Performance Comparison

2) Overall Performance: In Fig. 5 shows the overall time completion for the 11 minutes audio stream. We achieved an outstanding performance for the 6000 window size. The total time of recognition is 2.81 minutes. There is a 94.6% improvement in speed from the non-cluster recognition of 52 minutes.

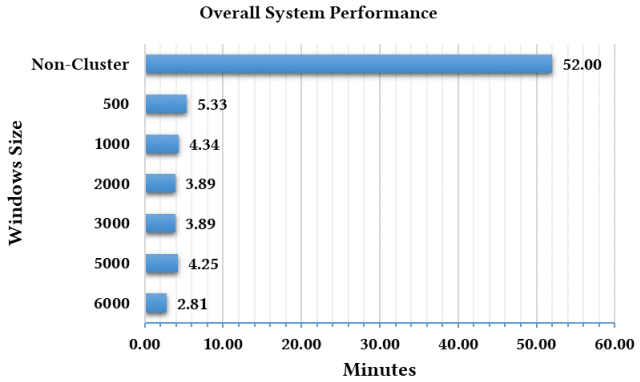

Fig. 5: Speed comparison using different window sizes and non-cluster recognition

VII. Conclusions

We are well aware of findings within song recognition and fingerprint from other researchers which are established such as Shazam [7] and SoundHound [8]. We acknowledge the implementation of clustering using K-means for the experiment conducted with actual fingerprint database with 2.4 billion fingerprint collection. We would like to emphasize that a database of this size rarely appear in the scientific literature although there is evidence of its existence.

The next important part was the performance of the recognition implementation. Although the initial K-means computation for the collection is resource demanding, we achieved a significant speed performance in the end. Besides, the configuration and algorithm proposed can contribute to a new perspective within the song recognition field.

Part of the extension plan of this research consists of clusters collection distribution in a cloud platform using *Sharding* in MongoDB [15]. A clustered collection can take advantage of storage distribution for a hybrid storage configuration setup for faster retrieval using SSD and cost saving using HDD storage.

We have a plan also on evaluating our performance with other states of the art song recognition method such as Shazam [7] and Labrosa [3].The discussion we provided would facilitate better understanding of the topic as well as improving the design and use of such systems.

Acknowledgment

We would like to thanks MIUR as the funder of the Industrial PhD program. We are also very grateful for the continuous support from research team members at MDSLabs, University of Messina.

References

[1] C. Bellettini and G. Mazzini, "A framework for robust audio fingerprinting." *JCM*, vol. 5, no. 5, pp. 409–424, 2010.
[2] J. Deng, W. Wan, X. Yu, and W. Yang, "Audio fingerprinting based on spectral energy structure and nmf," in *Communication Technology (ICCT), 2011 IEEE 13th International Conference on*. IEEE, 2011, pp. 1103–1106.
[3] D. Ellis, "The 2014 labrosa audio fingerprint system'." ISMIR, 2014.
[4] M. Malekesmaeili and R. K. Ward, "A local fingerprinting approach for audio copy detection," *Signal Processing*, vol. 98, pp. 308–321, 2014.
[5] Y. Shustef, "Music recognition method and system based on socialized music server," Jun. 30 2015, uS Patent 9,069,771.
[6] A. L.-C. Wang *et al.*, "An industrial strength audio search algorithm." in *Ismir*, vol. 2003. Washington, DC, 2003, pp. 7–13.
[7] A. L.-C. Wang and J. O. Smith III, "Systems and methods for recognizing sound and music signals in high noise and distortion," Feb. 20 2018, uS Patent 9,899,030.
[8] B. Mont-Reynaud, A. Master, T. P. Stonehocker, and K. Mohajer, "System and methods for continuous audio matching," 2016, uS Patent 9,390,167.
[9] A. Master and T. P. Stonehocker, "System and method for identifying original music," Jun. 10 2010, uS Patent App. 12/629,821.
[10] Y. Murthy and S. G. Koolagudi, "Content-based music information retrieval (cb-mir) and its applications toward the music industry: A review," *ACM Computing Surveys (CSUR)*, vol. 51, no. 3, p. 45, 2018.
[11] B. Saini, V. Singh, and S. Kumar, "Information retrieval models and searching methodologies: Survey," *Information Retrieval*, vol. 1, no. 2, p. 20, 2014.
[12] C. Sreedhar, N. Kasiviswanath, and P. C. Reddy, "Clustering large datasets using k-means modified inter and intra clustering (km-i2c) in hadoop," *Journal of Big Data*, vol. 4, no. 1, p. 27, 2017.
[13] P.-N. Tan, M. Steinbach, A. Karpatne, and V. Kumar, *Introduction to data mining*, 2005.
[14] Z. Huang, "Extensions to the k-means algorithm for clustering large data sets with categorical values," *Data mining and knowledge discovery*, vol. 2, no. 3, pp. 283–304, 1998.
[15] https://www.mongodb.com/what-is-mongodb.
[16] http://scikit-learn.org/stable/modules/clustering.html/k-means.