IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

ISO 3297:2007 Certified ⁂ Impact Factor 8.066 ⁂ Peer-reviewed / Refereed journal ⁂ Vol. 10, Issue 7, July 2023

DOI: 10.17148/IARJSET.2023.10754

Optimization of an Smart Media-Playing System Based on Network Communication

Prof. C S Swetha[1*], Shamanth B Patil[2]

Assistant Professor, Department of Master of Computer Application, BIT Bengaluru[1]

PG Student, Department of Master of Computer Application, BIT Bengaluru[2]

Abstract: In the process of the rapid development of mobile networks, music recommendation systems (MRSs) have experienced considerable success in recent years. Conventional music recommendation systems are, however, in general based on the simple user–track relationships or the content of songs and recommend songs according to intrinsic factors. Furthermore, they do not consider the users' contextual factors towards providing them with a more interpretable, efficient and smart recommendation experience. To address these issues, we propose anovel *H*eterogeneous *I*nformation *N*etwork-based *M*usic *R*ecommendation *S*ystem (HIN-MRS). By considering the extrinsic factors, such as contextual factors, internal factors, such as the user's personalized preference, and the heterogeneous relationship between items of song information, this method can perceive the user's music selection from multiple aspects, automatically maintain the user's playlist and improve the user's music experience. First we used the obtained textual data to extract the user's music preference to provide the topic which is usually related to the contextual factors, by means of which an HIN-MRS can realize the perception of the mobile environment. Second, after determining the topics, we built a small-scale HIN of songs according to topics and used a graph-based algorithm to generate recommendations. The recommendation method based on an HIN renders the recommendation process more efficient and the recommendation results more accurate and increases the user's satisfaction. The results of our final experiments also prove the significant advantages of the proposed model over the conventional approaches.

Keywords: MRS, HIN, SVD, PMF, LDA, SDA, CNN, LRU

I. INTRODUCTION

In today's Internet and Big Data era, cell phone APPs are mushrooming, people create exponentially more data, and the amount of knowledge available on the Internet far exceeds what a person can absorb in a lifetime. Due to the information overload, it is now urgent to figure out how to properly collect the necessary information. On the one hand, the cost for information producers to have their information found by users in the massive information flow is rising daily; on the other hand, the amount of information that users can actually access is a very small percentage of the enormous amount of resources on the network. There have been many methods put out to address this issue, including the recently created portals, search engines, and recommendation systems. On portals, users can browse material using the categories, which are usually chosen by professionals. Consumer classifications of goods and information may differ from expert classifications, and objects may appear to fit into only one category but in reality their classifications are typically unclear and overlap numerous categories. Compared to the suggestion system, the search engine's flaws are far more visible. To get the intended results in the aforementioned scenario, the user's keywords must properly reflect their intentions. They should also refrain from using information about their own or other people's past behaviour.

It is a typical property of these domains that the majority of sites where recommendation systems are regularly employed contain a large number of things that are challenging for users to explore and filter. The majority of customers also don't necessarily need the items, and it can be challenging for users to convey their preferences in a clear and unambiguous way. Several of these industries, including music, books, photographs, short films, television shows, news, and e-commerce, are referred to as "pan-entertainment." Recommendation systems are appropriate for online music services. First off, it's challenging for users to find their favorite songs because millions, if not tens of millions, of new songs are added to well-known music platforms on the Internet every day. Only 20% of digital music listeners can find good songs with ease, according to "QQ Music 2018 Listening Data," which found that 88% of users are willing to find new good songs. Second, customer needs are frequently vague because they just need their favourite music to control the ambiance in their homes or places of employment. Online media is more inexpensive, takes less time to listen to, and has a higher rate of repeat play than movies and books. The appeal of prospective music may also rise with a great music recommendation system. Many online music platforms, such as NetEase Cloud Music, QQ Music, and Spotify, use music recommendation systems nowadays to enhance user experience and quality [2]. There are still a lot of challenges in this field despite the abundance of answers. Users of popular music streaming services can listen to more than a dozen songs in an hour [4], with peak listening rates of hundreds of thousands of records per second [3]. It is also difficult to maintain the recommendation model up to date in light of the continuous changes because the area of music recommendation has changed from being based on explicit feedback (rating data) to being dominated by implicit feedback (play records) [4]. Traditional music recommendation systems must first train their models on static data before regularly retraining them on all data as it becomes available because the majority of them are offline. Retraining the models frequently requires several hours, consumes a lot of memory and processing resources, and does not take current changes in customer preferences into consideration. Every day or perhaps more frequently, these offline recommendation systems update their models [5]. This issue will likely worsen as interactive data generated by music websites grows at an exponential rate and offline recommendation algorithms lose efficiency in comparison to simple online algorithms. It is anticipated that streaming recommendation algorithms would be widely employed due to the frequent updates of online music libraries and the subsequent fast changes in user interests [6–10]. The majority of systems now in use have poor user experiences because they are memory-intensive, inefficient, or difficult to grasp why particular suggestions should be provided. Numerous mobile applications have been developed as a result of the phenomenal proliferation of smartphones. Smartphone music players are quickly gaining popularity because it's a good method to unwind and reduce stress [11].

Smartphone music players certainly have advantages over conventional music playback devices (recorders, MP3, etc.). First of all, transporting conventional music players is difficult. When using smartphones to download music playback software to achieve music playing, users do not need to carry a cell phone at the same time as an additional independent music playback device, which generates easy circumstances for the carrying of equipment. Traditional music playing devices normally only allow for the playback of music, in contrast to smartphone music players that also serve as regular cell phones and callers. On the other hand, the usual music player really only enables local playing. Recorders can only play songs on the tapes they own, giving customers a smaller selection of music, in contrast to smartphone music players that can download music and play it from the internet. The biggest problem is that traditional music players don't have a suggestion feature, which makes it very difficult for users to choose songs from a huge selection of recordings they might find interesting. For the purpose of effectively resolving the aforementioned problems, this paper develops a mobile music playback system based on a joint track recommendation algorithm. There are many useful applications for the study of real-time music recommendation systems. Users may now tag songs and artists thanks to the development of social tagging, which makes it simpler for them to locate the music they want and gives the recommendation system relevant data. The song's concept and tone are heavily alluded to in the lyrics. How to increase the effectiveness of suggestions by including music auxiliary data, such as tags, into streaming music recommendation systems, is another intriguing field of research. The requirements are looked at, the general music recommendation system is created using the software engineering process, each functional module is installed utilizing the most recent theories and technologies, and the system is then put to the test for performance and functionality.

II. RELATED WORK

Recommender systems have grown in popularity over the past 20 years as a hot area of study in both business and academia. Its proactive nature gives it an advantage because it can automatically gather data on user behavior and create models to discover acceptable products to recommend to system users. Some of the more well-liked recommendation strategies and current research fads are included in the list that follows.

The main goal of content-based recommendations is to identify products that are comparable to the user's preferred products by gathering enough information about an item's features, comparing them, and then suggesting them. It is constrained by objects and strongly reliant on the universe of suggested objects. The method used to assess item similarity differs significantly depending on the topic, and it might be challenging to extract the attributes for multimedia materials. It frequently lacks innovation as well.

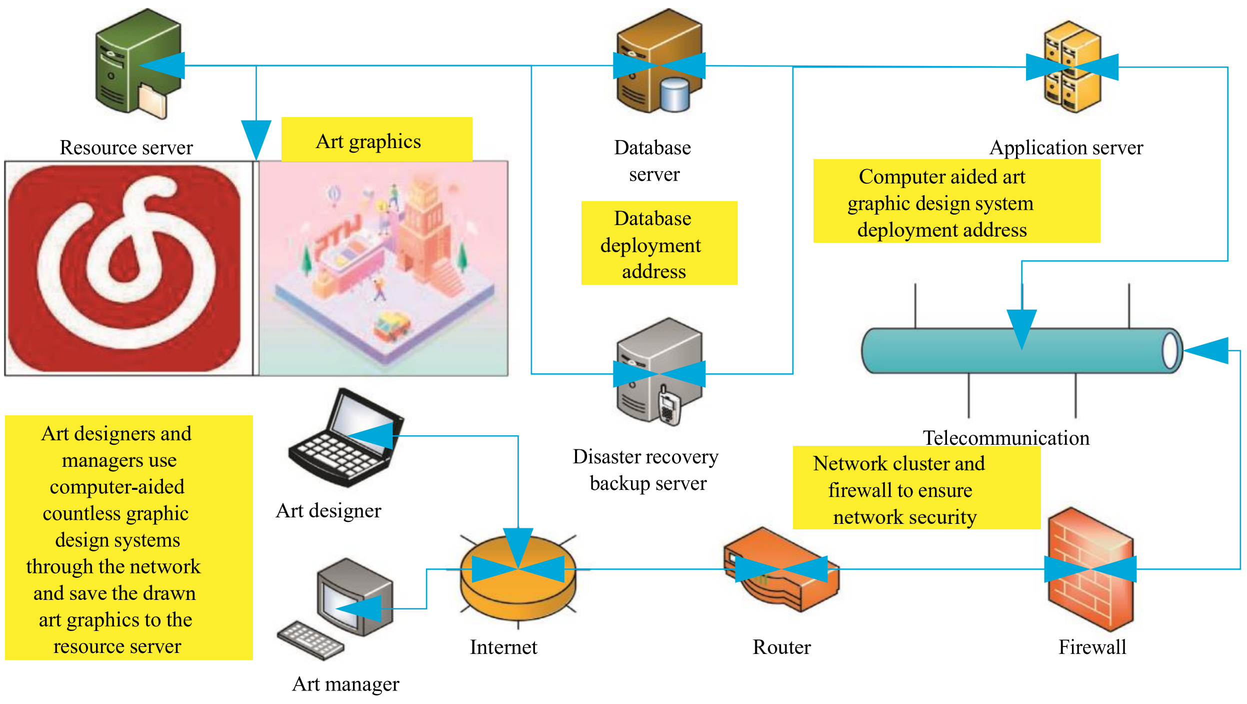
There are two primary directions in the field of recommendation systems unrelated to collaborative filtering. The first method, closest neighbour-based collaborative filtering, compares users or things based on feedback data from previous users. Euclidean distance, cosine similarity, Pearson correlation coefficient, and other metrics are the main ones used to measure similarity. The second trend is model-based collaborative filtering, which focuses comparisons mostly on mathematical models.

The collaborative filtering method based on matrix decomposition acquired particular popularity after the earlier SVD-based recommendation system won the 2006 Netflix competition, beating out several other algorithms [12]. Positive Matrix Factorization (PMF)'s addition of a probabilistic model strengthens the SVD decomposition procedure even further. Overall, the model-based collaborative filtering method outperforms the closest neighbour-based strategy despite

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
ISO 3297:2007 Certified ⁜ Impact Factor 8.066 ⁜ Peer-reviewed / Refereed journal ⁜ Vol. 10, Issue 7, July 2023
DOI: 10.17148/IARJSET.2023.10754

being more difficult to understand. A collaborative filtering technique based on label weighting with time decay was created and used by the authors of the literature [13].

Hybrid recommendation systems: Hybrid recommendation systems allow for the productive collaboration of multiple recommendation systems, which can increase novelty and partially alleviate the cold start issue. Ancillary data of objects, mostly textual data like articles, were the main source of support for earlier research in this field.


Resource server
Art graphics
Database server
Database deployment address
Application server
Computer aided art graphic design system deployment address
Telecommunication
Art designers and managers use computer-aided countless graphic design systems through the network and save the drawn art graphics to the resource server
Art designer
Disaster recovery backup server
Network cluster and firewall to ensure network security
Art manager
Internet
Router
Firewall

reviews, item descriptions, and abstracts [14]. Hybrid recommender systems use a variety of methods, including Latent Dirichlet Allocation (LDA), Stacked Denoising Autoencoders (SDA), and Convolutional Neural Networks (CNN) [15]. The visual aspects of images can also be modeled using algorithms [16].

Online and streaming recommender systems [17]: Most hybrid recommender systems are created for batch processing, which means that the initial model is constructed from static data and the model is rebuilt on all data at regular intervals when new data is introduced. To maintain the calibre of recommendations, it is essential to keep the recommendation model current with the advent of fresh data [18]. Recommendation systems can be abstracted as data flow issues in real-world applications. Due to limited computational resources, it might be challenging to store all the data and recompute all the data for online recommendation systems that must update models in real-time from a continuous stream of data. It needs to immediately respond to data changes and continuously learn from the data stream. Recent research have shown that less complex online algorithms can produce more accurate suggestions than more complex offline algorithms that are updated regularly. As with incremental neighbour-based algorithms and incremental matrix decomposition [19] (based on stochastic gradient descent or alternating least squares [20]), most online recommender systems rely on incremental learning. A collaborative item filtering-based streaming recommendation system called Stream Rec was suggested in the literature [21]. It created a movie recommendation system by segmenting the similarity computation and gradually upgrading the similarity utilizing a cloud computing platform designed for profit. The literature used a scalable updating mechanism to address a number of challenges with relation to real-world business applications. For the positive feedback, the matrix decomposition literature [22] advised moderate changes.

Platforms for big data-based recommendation systems: big data processing frameworks are frequently employed in recommendation systems since distributed computing platforms like Hadoop and Spark are so well-liked [23–26]. Spark, in contrast to Hadoop, tries to store intermediate results in internal storage as opposed to HDFS, which reduces the amount of disk IO operations and considerably speeds up processing. A collaborative filtering-based recommendation system is created on the Spark platform and put to the test using datasets like Movie Lens. In this paper, a massive data processing and storage architecture is investigated while Spark is used to develop a real-time recommendation system. A

DOI: 10.17148/IARJSET.2023.10754

recommendation system is constructed on the Spark platform, and matrix decomposition is used in the study to solve the problem of information loss.

Documents have a topic vector in topic models, which view them as unordered collections of words and usually just take word occurrences into account rather than word order. The Hidden Dirichlet Distribution model, which sees the topic as a Multivariate Distribution with parameters produced by the Dirichlet distribution, is the most significant topic model. The popularity of LDA is due to how easy it is to understand. LDA variations that include a temporal component can deal with shifting distributions over time.

III. DETAILED SYSTEM DESIGN AND IMPLEMENTATION

3.1. Designing a system architecture. This system, which is designed to store and manage massive music, review, and user behavior data streams, is based on the existing big data and recommendation system architecture. The system's overall architectural diagram is displayed in Figure 1.

(1) Spring Boot is used to build the B/S architecture-based user interface module.
(2) To build discrete streams for pushing to Spark Streaming, Flume, and Kafka, which both have higher fault tolerance methods to assure dependability, the Kafka cluster load-balances and buffers the data received. User-generated logs are gathered and immediately sent to the Kafka cluster using the Flume platform.
(3) On the user click stream that Spark Streaming receives from Kafka in real time, it computes statistical metrics like the number of music plays in the previous day, the daily plays, and the number of users playing while training online collaborative theme models, incremental matrix decomposition, and other models on the data stream. After that, the metrics are stored in the Mongo DB database.
(4) The Redis in-memory database primarily holds the number of times the user most recently played music machines and the user's most recent suggestion list in order to take advantage of Redis' capabilities and respond to recommendation requests rapidly. When Redis' memory is full, keys are deleted according to the Least Recently Used (LRU) policy.
(5) The business database in this study is Mongo DB, which is quick to input and get data from, great for storing a lot of data, has strong high availability, and can be readily connected with Spark SQL. It integrates with Elastic Search to enable efficient music retrieval by tags and titles.

The functional division of this system, which is Derived from functional needs, is depicted in Figure 2. The system is broken fall into different functional modules, including those for music browsing, data gathering and statistics, recommendation engines, personal homepages, and other features. Each functional module is further divided into more submodules, and each module is in charge of carrying out distinct tasks and may rely on other functional modules. The music browsing module enables exploring music and artists in a variety of ways, including browsing by daily and real-time music charts, browsing by artist categories, browsing by artist information, and browsing by popular singles. Users may also add, see, and like comments under the music, and it keeps track of when they play and move between songs in a log. The module that implements suggestions is called the recommendation engine. The submodules for personal account login, registration, information updating, and viewing recent personal plays are all part of the personal homepage module.

3.2. Implementation of the music browsing module. This module carries out the tasks of searching for popular singles by singers, playing music, and exploring singers and singles by category. On the system's home page, three tabs labelled "Recommend," "Music Gallery," and "My" display various pages, respectively. There are three subtabs available under the "Music Gallery" tab: "Artist," "Chart," and "Category." There are three subtabs under the "Music Gallery" tab: "Artists," "Charts," and "Categories." These are the primary functional submodules of this module.

3.2.1. *Listen to music by artist*. This module allows several combinations of search criteria, including area, gender, and style, to let users easily locate their favourite singers. A list of singers who fit the criteria is shown below, along with the singer's photo and name. The singer tab in the music library offers three columns of filtering criteria, which may be limited to the region, gender, and style of the singer. When the user changes the filter criterion, the list of singers is updated. The singer information page, which displays the singer photo and hit songs and albums they are a part of, may be accessed by clicking on a singer's name in the singer list. The song will start playing when you click on it in the singer's work list, and the log will be recorded in the background. Figure 3 depicts the singer's process of listening to and browsing songs.

DOI: 10.17148/IARJSET.2023.10754

3.2.2. *Search for Music by Genre.* Users choose the category subtab in the music library. The system is segmented into different sections based on music themes, situations, and moods to show common category tags. By selecting a category, you may browse a list of the most popular songs that fall under that category.

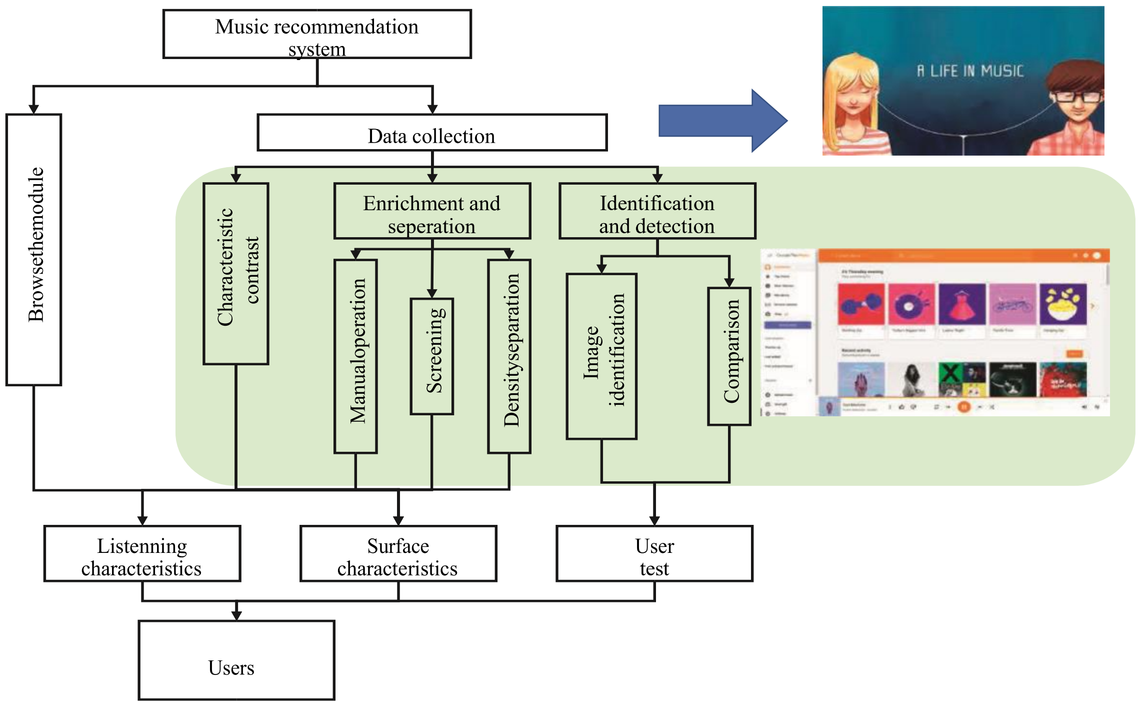


Figure 2: System functional structure diagram.

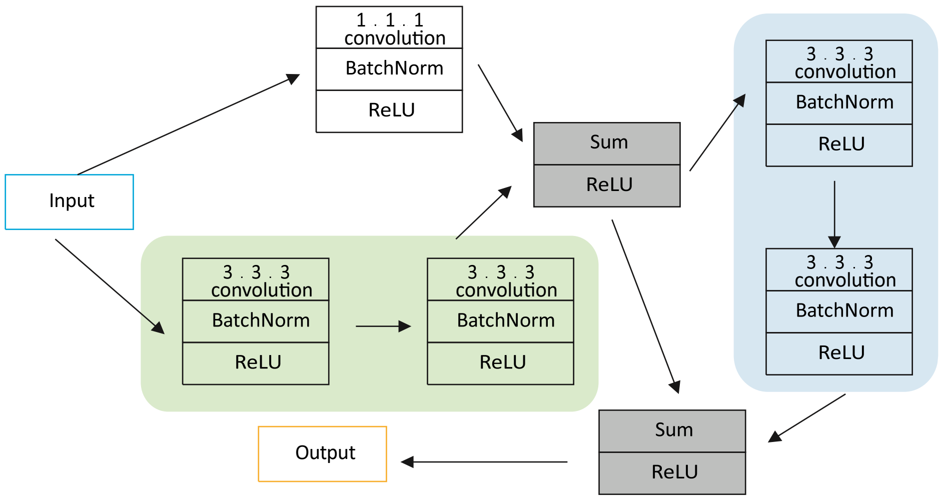


Figure 3: Browse music timeline by artist.

3.2.3. *Music Charts.* When users choose the Charts subtab in the Music Gallery, a live chart of the last 24 hours worth of playing music is displayed. Users may switch between the charts for the current day and the current week, and within those charts, they can select whether to sort by the quantity of plays or the quantity of users playing.

DOI: 10.17148/IARJSET.2023.10754

3.2.4. *Personal Recently Played Music.* On the system home page, users may pick the My tab to access their own home page, where they can view their account avatar and nickname, the most recently played music is listed beneath their account, and they can choose to sort by playing times or recent playing time.

Figure 4 displays the class diagram for the music browsing module. The controller layer is in charge of accepting and responding to requests, the service layer is in charge of handling particular business logic, and the data access layer is in charge of gaining access to databases. Artist Controller, Track Controller, and User Controller all derive from Base Controller in the controller layer, which contains fundamental functions like writing log, getting list, and getting single

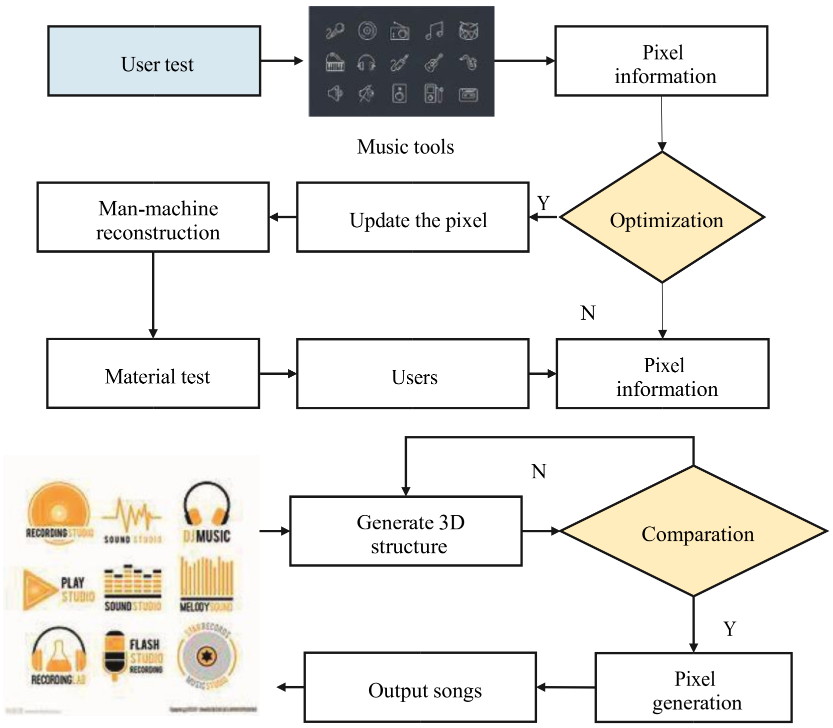


Figure 4: Music browsing class diagram

record by id. Artist Controller provides methods to get artist details, search popular artists list by region and style, etc. Track Controller offers ways to search songs by lyrics, query real-time popular music lists and daily charts, query popular singles by artist, and more. The techniques offered by the user controller include login and registration, asking about recent play history, etc. The Artist DAO, Track DAO, and User DAO classes in the data access layer handle database actions pertaining to musicians, artists, and users, respectively. They all derive from the Base DAO class, which performs fundamental operations like adding and removing single pieces of data and querying the list. The Redis Template and Mongo Template are encapsulated in the data access layer classes, which use them to accomplish different database operations. The most recent query is kept in the Redis cache by this module for the majority of requests, and when processing the query, it first checks the cache before querying Mango DB data base and storing the response in the cache when it cannot be retrieved.

3.3. Statistics Module and Data collection. The Statistics Module and data collection provides training data streams for recommendation engine while collecting user activity real time data and performing common statistics. Although this module is not a front-end function module from the user's perspective, it serves as the foundation for the entire system and is responsible for functions like music real-time charts, hot charts display, and recommendation, so making it independent can significantly increase code reuse, as shown in Figure 5. This module employs Flume to keep track of changes to user behaviour logs, transmit those updates to Kafka and Spark Streaming to calculate trends in music popularity, and store the results in database. These are the layers into which it may be separated. This section provides examples of how to query the suggestion list, an artist list, and a daily music library to compare the music recommendation system's resemblance to other interfaces.

(1) Log collecting layer: Tomcat, which is included in Spring Boot, is used in these studies. Tomcat gathers data about user activity, such as listening to music, and writes it until the end of the log collecting layer.

Producer layer: Flume is deployed on each server node. Flume uses the tail -f command to track the new content at the end of the logs to monitor and collect server logs and send them to the Kafka cluster in the specified file size.

(2) Kafka cluster layer The data arriving from the producer layer is load-balanced and buffered in this layer. It prevents message loss and offers improved dependability. Following the collection of log updates by Kafka, DStream is created through the Kafka Stream program and sent to Spark Streaming. RDD ["user ID t music ID t operation type t date and time"] is what makes up a DStream.

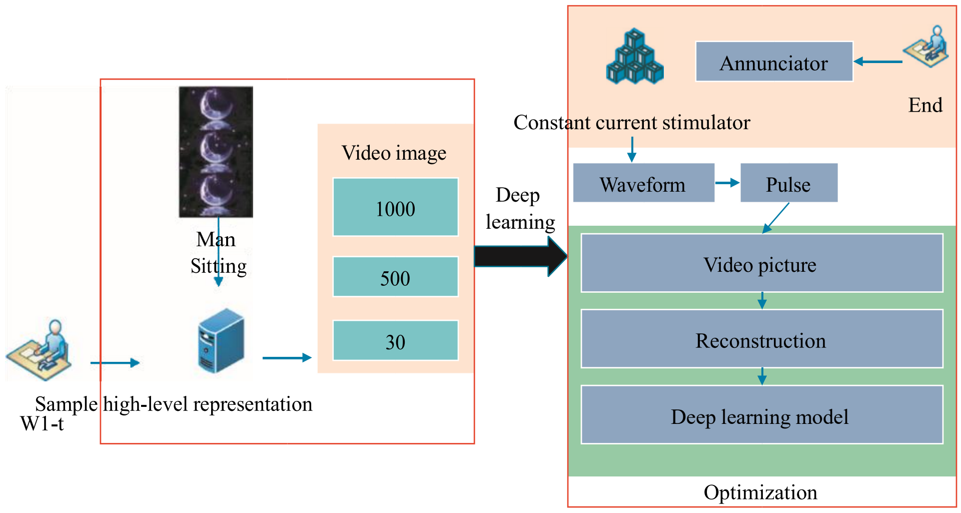


Figure 5: Statistics Module and data collection hierarchy.

(3) Stream processing layer: Spark Streaming is used to write statistical data to Mongo DB, count Kafka data streams, train online recommendation algorithms, and determine how many people listen to music daily.

(4) Storage layer: the log statistics are stored in Mongo DB, including the number of users playing and the number of daily music plays.

This module's statistical duty is to tally those number of real-time plays (i.e., plays during the last 24 hours) for each song, as well as the total number of plays and users playing that day. By counting the occurrences of each key in a time frame, Spark Streaming's count By Value And frame function, which counts the number of plays in a time window, may be used. In this study, we utilize Sorted Set to keep track of the number of music plays over the last 24 hours after exporting the findings to Redis.

An ordered set's components have a score, which is the quantity of music performed, and the components are kept in order according to the score. The Sorted Set's ZRANGE operation may rapidly query the pieces of music that have received the most plays in real time. It can also query components whose scores fall within a specific interval. Simply sum up the number of times the music ID occurs in the microbatch and the number of songs played on the same day in Redis to get the total number of songs played on a given day Reduce By Key and group By Key, as well as other conversion operations, are applied to each batch interval RDD and cannot use the previous RDD data, which is obviously not possible; window operations, such as reduce By Key And Window, are also challenging to handle and have a significant overhead due to the deduplication of users.

Therefore, The key in this article is a string of the form "date: music ID," and it is used to keep track of the users who play each song each day using the Redis set data structure. We can save all of the musical recordings from that day in this paper by setting the key's expiry period to one day. At most, we simply need to store the users who played every song from today and yesterday. Since the set action in Redis is an atomic operation, thread safety is not a concern in this tutorial.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

ISO 3297:2007 Certified ⁜ Impact Factor 8.066 ⁜ Peer-reviewed / Refereed journal ⁜ Vol. 10, Issue 7, July 2023

DOI: 10.17148/IARJSET.2023.10754

IV. SYSTEM TESTING

The online collaborative topic model (COLDA), incremental matrix decomposition (Inc MF), and enhanced item/user-based incremental collaborative filtering (Knni++) are all developed in this study predominantly using Spark. When the data volume is high, the incremental matrix decomposition recommendation engine is preferred; otherwise, the COLDA recommendation engine is preferred. Because the registration information filled out by users is likely to have more missing values or inaccuracies due to privacy considerations, this system does not use the registration information to make recommendations for new users instead uses the method of letting new users select the tags of interest to make initial recommendations.

When receiving data from microbatch and a request for a user recommendation list, we first initialize the model parameters and then update it incrementally. If the user recommendation list is in the cache and hasn't expired, the cache is returned; if not, a recommendation list is calculated. The user-item hidden feature vector is represented by Dense Vector under the org. Apache.spark.ml.linalg package in the paper's actual implementation, which employs small-batch gradient descent on each batch interval. A user playing a record is represented by the Record class The hidden feature dimension, learning rate, and user-item regularity are all included in the parameters class. The item's determined recommendation score for the user is represented by the Rating class. In this paper, the hidden feature vectors of users and items, referred to as user features and item features, respectively, are stored in the RDD [(id: Long, features: Dense Vector)] key-value pair collection, and the parameters of the user features and item features are enclosed in the Streaming MFModel class along with the parameters.

in this paper, We start by removing categories with fewer than five songs and users with fewer than ten listenings. 11,000 tags are still visible. Figure 6 displays the Recall and F1 variations of a few of the system's recommendation engines when the suggestion list length N is changed. It is clear that the incremental matrix decomposition effect and the optimized item-based collaborative filtering (Knni++) both outperform the original item-based collaborative filtering algorithm (Knni), with the COLDA recommendation engine being the best performer among the algorithms. This suggests that adding auxiliary data, such lyrics and tags, can increase the suggestion accuracy. The number of things favoured by users in the test set remains constant as the recommendation list grows, and items liked by users in the test set are more likely to appear in the recommendation list, therefore the recall of all algorithms rises as a result. Additionally, the majority of algorithms' F1 scores rise and subsequently fall, with N 20 serving as the best number. The outcomes of the COLDA algorithm are also shown to be affected by altering the topic model's number of subjects. When the other parameters are left at their default settings and the number of suggested items N and the number of subjects K are modified, the accuracy, recall, and F1 score of the COLDA method are shown in 0(a) and (b). The accuracy of the COLDA algorithm often declines as N rises, which is to be anticipated. The Recall grows gradually as K reaches 20 and thereafter climbs, and the number of subjects K has minimal impact on the metrics overall.

In this paper We used the Load Runner virtual user mechanism to create virtual users for the test and then executed the prerecorded script to simulate the real users of the system to send requests to the system. We let the number of virtual users increase from 50, 100, to 200, all the way up to more than 500 and, at the same time, send out requests to a specific page. This allowed us to test the response time delay and concurrency on the APP side. Then, in order to obtain the concurrency data for the system's response time, we log the average time it takes for each user to receive a response and for the page to load. This section uses the requests for a suggestion list, an artist category list query, and a music daily list query to demonstrate the concurrent test of the music recommendation system. The test results for different interfaces are comparable. The typical response time for requests for suggestion lists from various concurrent users is displayed in Figure 7. When the number of concurrent users does not exceed 300, it can be seen that the response time delay is often minimal and all requests are fulfilled. The average response time stays within 2 seconds even when the number of concurrent users reaches 500, and the request failure rate doesn't go beyond 2%.

In this article, The real-world Last.fm dataset and the Movie Lens dataset, both of which contain labelled objects and time-stamped interaction records, are used to assess the efficacy of the system recommendation algorithm.

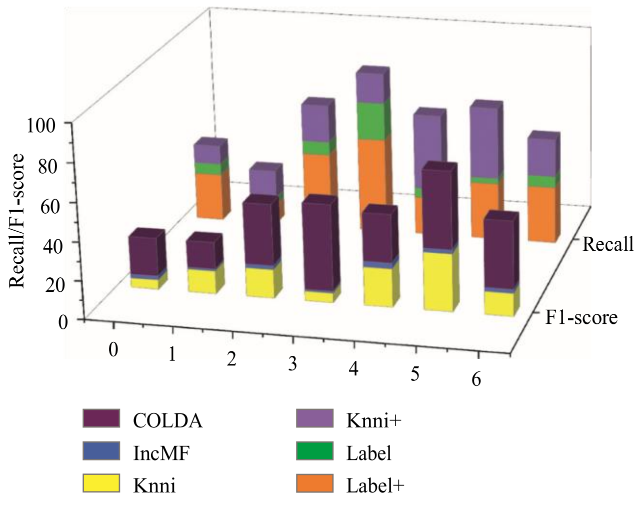


Figure 6: Comparison of datasets, Recall.

The two datasets are first statistically analysed below. The two datasets are summarized in Figure 8. This dataset is a record of music played by users of the Last.fm website, which has 992 different users and 16.98 million play records involving 180,478 different music tracks. Each play record is time-stamped (to the second) and spans the period from February 2005 to September 2013. In this paper, we crawled the tags of this music using the Last.fm API and ended up with 68,756 different tags. For each tag, the paper calculated the amount of music tagged by it and the total number of times it was used. The analysis shows that there is a clear long-tail distribution of tag usage (Power Law Distribution), with the most tagged songs being "rock," tagged a total of 53,181 songs. In contrast, 13,132 tags were used for only one song, 336 tags were used for more than 1000 songs, and 31 tags were used for more than 10000 songs. The scatter plot (logarithmic coordinates) of the amount of music used by tags versus the number of tags approximates a straight line, which shows that they follow a long-tailed distribution. The distribution of the total number of tags used is similar to the long-tail distribution but is more balanced than the long-tail distribution. The Recall of all algorithms increases as the recommendation list increases because the number of items preferred by users in the test set remains the same as the recommendation list increases, while the items preferred by users in the test set are more likely to be included in the recommendation list, as shown in Table 1.

For both datasets, the interaction record data are sorted by time stamp in this article. The first 50% of the recorded data is used it for training, and the second 50% is used as the test set. In this paper, we compare the effectiveness of three recommendation engines, including the system implementation of Collaborative Online Topic Model (COLDA), Incremental Matrix Decomposition (Inc MF), and Improved Incremental Item-Based Collaborative Filtering (Knni++), to find the best parameters using grid search, and the unoptimized item-Based collaborative filtering (Knni) was used as a benchmark. For the online collaborative topic

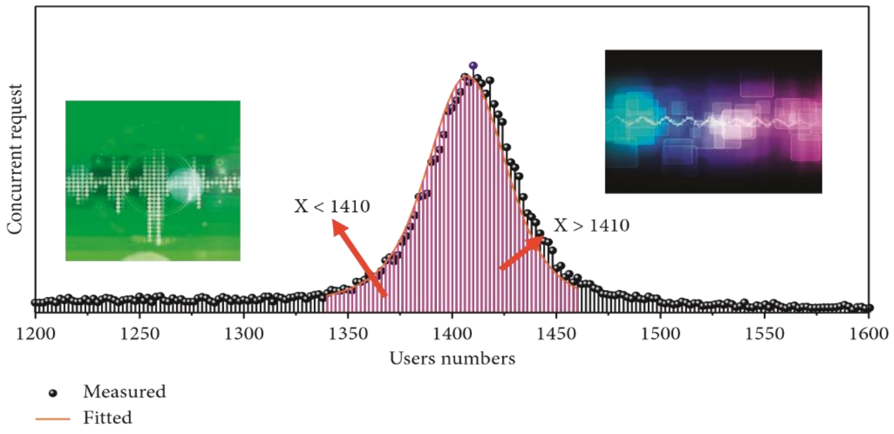


Figure 7: Concurrent request recommendation list test results.

DOI: 10.17148/IARJSET.2023.10754

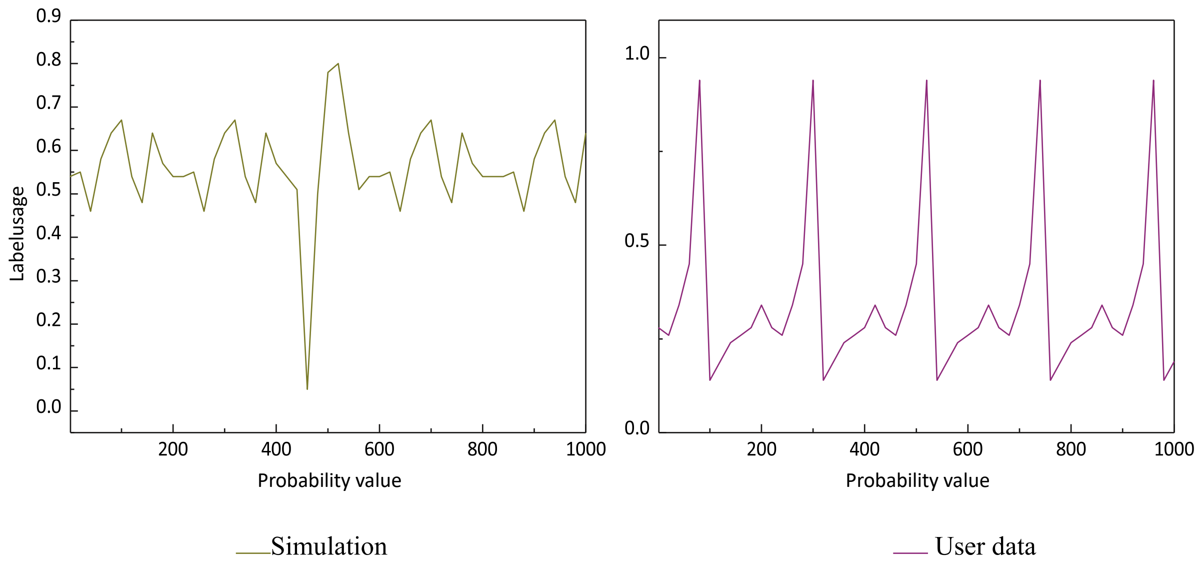


Figure 8: Distribution of dataset label usage.

Table 1: Recall of all algorithms.

Algorithms	List number	Time set/s	Recall time/s
COLDA	2	2	1.7
Inc MF	3	2	1.9
Knni++	2	2	2.4
Knni	1	2	2.5
Blank	4	2	2.7

model (COLDA) algorithm, the default parameters are set as the number of topics K 20, the learning rate x 0.05, the user regularization factor u 0.01, and the item regularization factor i 10. For the Incremental Matrix Decomposition (Inc MF) algorithm, the feature dimension is K 50, the learning rate x 0.03, the user regularization factor u 0.02, and the item regularization factor i 0.02. For the improved item-based collaborative filtering, the similarity weight based on user behaviour is α 0.4, the similarity weight based on song tags β 0.4, the similarity weight based on song lyrics c 0.2, and the number of item similar neighbours M 200. For all algorithms, the default number of recommended items for each user is N 20.

V. CONCLUSION

This paper first describes how the test environment is set up, then tests the recommended module on actual datasets like Last.fm and Movie Lens. It then compares the collaborative online model topics, Additionally, enhanced item-based collaborative filtering techniques and incremental matrix decomposition, and finds that the COLDA algorithm is more accurate and superior to the others. It is demonstrated that the online approach outperforms the incremental matrix decomposition and item-based collaborative filtering algorithms by providing labels beneath the topics acquired by the collaborative topic model.

The system is then subjected to a thorough black-box test, a large number of test cases are created for each function, the system's primary user interface is displayed, and lastly a nonfunctional test is conducted. The incremental matrix decomposition on data streams, online collaborative topic model, and enhanced item-based collaborative filtering algorithm were all implemented in the recommendation engine module of this study using Spark. The system testing portion assesses the system's functionality and performance as well as the correctness of the recommendation engine using real datasets. A thorough analysis of the test results is provided in this part, along with samples of the detected musical themes. This study fixes problems with the offline recommendation system, improves user experience, and continuously updates the recommendation model in response to shifting user interests. Additionally, it offers rankings and real-time data on musical popularity trends.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

ISO 3297:2007 Certified ⁜ Impact Factor 8.066 ⁜ Peer-reviewed / Refereed journal ⁜ Vol. 10, Issue 7, July 2023

DOI: 10.17148/IARJSET.2023.10754

REFERENCES

[1] M. Mohammadi, M. Lakestani, and M. H. Mohamed, “Intelligent parameter optimization of savonius rotor using artificial neural network and genetic algorithm,” *Energy*, vol. 143, pp. 56–68, 2018.
[2] M. Qiu, S.-Y. Kung, and K. Gai, “Intelligent security and optimization in edge/fog computing,” *Future Generation Computer Systems*, vol. 107, pp. 1140–1142, 2020.
[3] Q. Wu and R. Zhang, “Beamforming optimization for wireless network aided by intelligent reflecting surface with discrete phase shifts,” *IEEE Transactions On Communications*, vol. 68, no. 3, pp. 1838–1851, 2020.
[4] J. H. Lee, J.-Y. Song, D.-W. Kim, J.-W. Kim, Y.-J. Kim, and S.-Y. Jung, “Particle swarm optimization algorithm with intelligent particle number control for optimal design of electric machines,” *IEEE Transactions On Industrial Electronics*, vol. 65, no. 2, pp. 1791–1798, 2018.
[5] T. Liu, D. Zhang, H. Dai, and T. Wu, “Intelligent modeling and optimization for smart energy hub,” *IEEE Transactions On Industrial Electronics*, vol. 66, no. 12, pp. 9898–9908, 2019.
[6] B. Gao, L. Guo, Q. Zheng, B. Huang, and H. Chen, “Acceleration speed optimization of intelligent EVs in consideration of battery aging,” *IEEE Transactions On Vehicular Technology*, vol. 67, no. 9, pp. 8009–8018, 2018.
[7] Y. Jia, C. Ye, and Y. Cui, “Analysis and optimization of an intelligent reflecting surface-assisted system with interference,” *IEEE Transactions On Wireless Communications*, vol.19, no.12, pp. 8068–8082, 2020.
[8] V. D. P. Souto, R. D. Souza, B. F. Uchoa-Filho, A. Li, and Y. Li, “Beamforming optimization for intelligent reflecting surfaces without csi[J],” *IEEE Wireless Communications Letters*, vol. 9, no. 9, pp. 1476–1480, 2020.
[9] R. Xiao, J. Li, and T. Chen, “Modeling and intelligent optimization of social collective behavior with online public opinion synchronization,” *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 8, pp. 1979–1996, 2019.
[10] X. Jin, “Research on optimization of intelligent warehousing business of state grid based on genetic algorithm,” *Journal of Computers*, vol. 13, pp. 1164–1170, 2018.
[11] N. Thakur, Y. K. Awasthi, M. Hooda, and A. S. Siddiqui, “Adaptive whale optimization for intelligent multi-constraints power quality improvement under deregulated environment,” *Journal of Engineering, Design and Technology*, vol. 17, no. 3, pp. 490–514, 2019.
[12] L. Chen, M. Ma, and L. Sun, “Heuristic swarm intelligent optimization algorithm for path planning of agricultural product logistics distribution,” *Journal of Intelligent & Fuzzy Systems*, vol. 37, no. 4, pp. 4697–4703, 2019.
[13] H. Hu, J. Yang, and C. Tian, “Research on intelligent optimization of parameters of deurring process with fluid-impact to automobile master cylinder cross hole,” *Journal of Intelligent & Fuzzy Systems*, vol. 35, no. 1, pp. 315–323, 2018.
[14] K. Anjaria and A. Mishra, “Thread scheduling using ant colony optimization: an intelligent scheduling approach towards minimal information leakage,” *Karbala International Journal of Modern Science*, vol. 3, no. 4, pp. 241–258, 2017.
[15] Z. Wang and X. Luo, “Modeling study of nonlinear dynamic soft sensors and robust parameter identification using swarm intelligent optimization CS-NLJ,” *Journal of Process Control*, vol. 58, pp. 33–45, 2017.
[16] S. K. Tamang and M. Chandrasekaran, “Integrated optimization methodology for intelligent machining of inconel 825 and its shop-floor application,” *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 39, no. 3, pp. 865–877, 2017.
[17] W. Chen, Z. Chen, X. Ma, Y. Chi, and Z. Li, “Secrecy rate optimization for intelligent reflecting surface aided multiinput-single-output terahertz communication,” *Microwave and Optical Technology Letters*, vol. 62, no. 8, pp. 2760–2765, 2020.
[18] F. Zhao, G. Li, R. Zhang et al., “Swarm-based intelligent optimization approach for layout problem,” *Multimedia Tools and Applications*, vol. 76, no. 19, pp. 19445–19461, 2017.
[19] C. Shao, Z.-J. Zhang, X. Ye, Y.-J. Zhao, and H.-C. Sun, “Modular design and optimization for intelligent assembly system,” *Procedia Cirp*, vol. 76, pp. 67–72, 2018.
[20] M. Kovalsky´ and B. Micieta, “Support planning and opti-ˇ mization of intelligent logistics systems,” *Procedia Engineering*, vol. 192, pp. 451–456, 2017.
[21] C. Lv, H. Wang, B. Zhao et al., “Cyber-physical system based optimization framework for intelligent powertrain control,” *Sae International Journal of Commercial Vehicles*, vol. 10, no. 1, pp. 254–264, 2017.
[22] X. Wang, W. Sun, E. Li, and X. Song, “Energy-minimum optimization of the intelligent excavating process for large cable shovel through trajectory planning,” *Structural and Multidisciplinary Optimization*, vol. 58, no. 5, pp. 2219–2237, 2018.
[23] S. Qi, Y. Lu, W. Wei, and X. Chen, “Efficient data access control with fine-grained data protection in cloud-assisted IIoT,” *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2886–2899, 2021.
[24] A. Zielonka, A. Sikora, M. Wozniak, W. Wei, Q. Ke, and Z. Bai, “Intelligent Internet of things system for smart home optimal convection,” *IEEE Transactions on Industrial Informatics*, vol. 17, no. 6, pp. 4308–4317, 2021.
[25] W. Wang, Z. Gong, J. Ren et al., “Venue topic model–enhanced joint graph modelling for citation recommendation in scholarly big data,” *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, vol. 20, no. 1, pp. 1–15, 2020.
[26] H.-x. Hu, Z.-w. Jiang, Y.-f. Zhao, Y. Zhang, H. Wang, and W. Wang, “Network representation learning-enhanced multisource information fusion model for POI recommendation in smart city,” *IEEE Internet of Things Journal*, vol. 1, 2020.