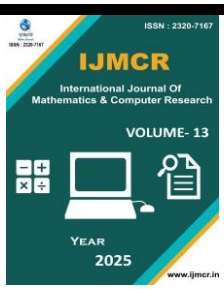
International Journal of Mathematics and Computer Research

ISSN: 2320-7167

Volume 13 Issue 03 March 2025, Page no. – 4992-4999

Index Copernicus ICV: 57.55, Impact Factor: 8.615

DOI: 10.47191/ijmcr/v13i3.12



Mining Data for Music Recognition in Big Data Using Locality-Sensitive Hashing (LSH) Algorithm

Ibeh, Eka Asibong*[1], Zibs Dowell Feremo Woripere[2], Obatola, Abayomi Toyin[3]

[1,2,3] Department of Computer Science, Federal Polytechnic of Oil and Gas, Bonny, Rivers State, Nigeria

ARTICLE INFO

Published Online:
22 March 2025

ABSTRACT

In the ever-evolving landscape of the music industry, the ability to efficiently recognize and categorize vast amounts of musical data is paramount. This study explores the application of data mining techniques for music recognition within the realm of Big Data, leveraging the Locality-Sensitive Hashing (LSH) algorithm. As the volume of digital music continues to grow exponentially, traditional methods of music recognition and classification face significant challenges in terms of scalability and efficiency. LSH, known for its ability to approximate nearest neighbor searches in high-dimensional spaces, offers a promising solution to these challenges.By implementing LSH, this research aims to improve the speed and accuracy of music recognition processes. The study delved into the mechanics of LSH, explaining how it hashes similar music data points into the same buckets with high probability, thereby facilitating quick and efficient retrieval. Through extensive experimentation and analysis, the effectiveness of LSH in handling large-scale music datasets is evaluated, highlighting its advantages over conventional methods. The findings of this study underscore the potential of LSH to revolutionize music recognition in Big Data environments, offering a scalable, efficient, and robust approach to managing and categorizing extensive musical archives. This research not only contributes to the academic discourse on data mining and music recognition but also provides practical insights for industry practitioners seeking to harness the power of Big Data in the music domain.

Corresponding Author:
Ibeh, Eka Asibong

KEYWORDS: Music, Recognition, Locality-Sensitive Hashing (LSH) algorithm, Big Data

INTRODUCTION

The advent of digital technology and the proliferation of online music platforms have led to an exponential increase in the volume of music data. As a result, the field of music recognition has become crucial in managing, organizing, and retrieving music from vast and ever-growing databases. Music recognition involves identifying songs or audio clips by analyzing their unique features and matching them to a database of known tracks. In the context of Big Data, this process presents significant challenges and opportunities [1]. Big Data refers to the massive volumes of structured and unstructured data generated at high velocity [2]. In music recognition, Big Data encompasses millions of songs, user-generated playlists, streaming records, acoustic fingerprints, and metadata such as artist names, genres, and release dates. The availability of such diverse datasets enables music recognition systems to become more accurate and versatile.

However, handling Big Data in music recognition requires overcoming challenges related to scalability, real-time processing, and noise tolerance.

Music recognition in Big Data environments demands algorithms that are both efficient and scalable. Locality-Sensitive Hashing meets these requirements by enabling rapid and accurate identification of songs from massive datasets. Its ability to balance speed, accuracy, and resource efficiency positions LSH as a transformative tool in the field of music recognition, paving the way for enhanced audio search capabilities and user experiences [3].

Locality-Sensitive Hashing (LSH) offers an innovative approach to tackling this challenge, particularly in the context of Big Data, where scalability and speed are paramount.

Locality-Sensitive Hashing is a technique designed for approximate nearest neighbor search in high-dimensional spaces [4]. It works by hashing similar items into the same bucket with high probability. For music recognition, this means that songs with similar acoustic fingerprints unique

representations of their audio characteristics are grouped together, enabling fast comparisons.

LSH is particularly suited for Big Data applications due to its efficiency in handling large-scale datasets. Unlike traditional methods, which often require exhaustive pairwise comparisons, LSH reduces computational overhead by focusing only on potential matches within the same hash bucket. This makes it ideal for real-time music recognition tasks where latency is critical.

How LSH Works in Music Recognition

1. **Feature Extraction:** Music recognition begins with the extraction of features such as tempo, pitch, rhythm, and spectral properties. These features are used to generate an acoustic fingerprint for each song.
2. **Hashing and Indexing:** LSH algorithms apply multiple hash functions to the acoustic fingerprints. These hash functions are designed to maximize the probability that similar fingerprints are hashed to the same bucket, while dissimilar ones are distributed across different buckets.
3. **Search and Retrieval:** During the recognition phase, the acoustic fingerprint of a query audio clip is hashed using the same hash functions. The algorithm then retrieves all fingerprints within the same bucket for comparison, significantly narrowing down the search space.
4. **Similarity Matching:** A similarity metric, such as cosine similarity or Hamming distance, is used to compare the query fingerprint with those retrieved from the hash buckets. The song with the highest similarity score is identified as the match.

Related Work

[5] Recommend an alternate technique to concentrate musical example included in sound music by methods for convolutional neural framework. Their tests demonstrated that convolution neural network (CNN) has vigorous ability to catch supportive components from the deviations of musical examples with unimportant earlier information conveyed by them. They introduced a system to consequently extricate musical examples high-lights from sound music. Utilizing the CNN relocated from the picture data retrieval field their element extractors require insignificant earlier learning to develop. Their analyses demonstrated that CNN is a practical option for programmed highlight mining. Such revelation supported their hypothesis that the inherent attributes in the assortment of melodic data resemble with those of picture data. Their CNN model is exceedingly versatile. They also presented their revelation of the perfect parameter set and best work on using CNN on sound music type arrangement.

[6] mutually used SVM on events of brief time highlights from whole classes. They then sorted the edges in test melodies and after that they let the edges vote for the class of the whole melody. They said in spite of the fact that the test informational indexes they utilized as a part of their examinations they are not adequate to sum up the superior of both the features and the SVM classifier. It can be seen that musical score is measurably distinguishable with great execution (more than 85 %) with particularly fundamental three classes (i.e. a, b and c). The characterization multifaceted nature can be diminished by various leveled arrangement steps. By presenting CAMS they built the general execution by 3-4%. One of the disadvantages of this framework is high computational many-sided quality in figuring distinctive feature orders for various arrangement steps.

[7] used SVM on different record level components for speaker ID and speaker affirmation assignments. They showed the Symmetric KL difference-based piece and moreover considered showing a record as a single full-covariance Gaussian or a mix of Gaussians. They approved this approach in speaker ID, confirmation, and picture arrangement errands by contrasting its execution with Fisher part SVM's. Their outcomes demonstrated that new technique for consolidating generative models and SVM's dependably beat the SVM Fisher portion and the AHS strategies. It regularly outflanks other grouping strategies for example, GMM's and AHS. The equivalent blunder rates are reliably better with the new piece SVM techniques as well. On account of picture grouping their GMM/KL divergence-based piece has the best execution among the four classifiers while their single full covariance Gaussian separation based portion beats most different classifiers. All these empowering demonstrate that SVM's can be enhanced by giving careful consideration to the way of the information being displayed. In both sound and picture errands they simply exploit earlier years of research in generative techniques.

[8] used short-time features to hold the information of the first flag and compact to such a point that small dimensional classifiers or relationship estimations can be functional. Most extraordinary conclusions have been set in brief time highlights which enter the data from a little measured window (much of the time 10ms - 30ms). In any case, as often as possible the outcome time probability is extent of minutes. They consider differing approaches for component blend and late data fusion for music type categorization. A novel element blend system, the AR model, is suggested and clearly overwhelms normally utilized mean change features.

[9] in their paper prescribe Daubechies Wavelet Coefficient Histograms (DWCHs) as a list of capabilities appropriate for categorization of music type. The list of capabilities outlines vastness contrasts in the sound flag. In this paper they proposed DWCHs, another feature extraction strategy for music genre grouping. DWCHs analyze music motions by registering histograms on Daubechies wavelet coefficients at different recurrence groups which has enhanced the arrangement accuracy. They gave a relative investigation of different feature extraction and grouping techniques and research the order execution of different characterization strategies on various feature sets

[10] also computed music related features arranging songs into genre with k-NN in view of GMMs prepared on music information. Authors basically had 100 capabilities routes for every class. They displayed these modules with GMMs requiring few segments in light of their mean utilization of feature measurements. As per the authors in spite of the fluffy way of genre limits, musical genre arrangement can be performed consequently with results altogether superior to possibility, and execution similar to humanoid type characterization. Three feature sets for speaking to tumbrel surface, rhythmic substance and pitch substance of music signs were suggested and were assessed utilizing measurable acknowledgment classifiers

[11] in their paper prescribe Daubechies Wavelet Coefficient Histograms (DWCHs) as a list of capabilities appropriate for categorization of music type. The list of capabilities outlines vastness contrasts in the sound flag. In this paper they proposed DWCHs, another feature extraction strategy for music genre grouping. DWCHs analyze music motions by registering histograms on Daubechies wavelet coefficients at different recurrence groups which has enhanced the arrangement accuracy. They gave a relative investigation of different feature extraction and grouping techniques and research the order execution of different characterization strategies on various feature sets

METHODOLOGY

This research work adopted the Object-Oriented Analysis and Design Methodology (OOADM). Object-Oriented Analysis and Design (OOADM) is a software engineering methodology that employs object-oriented principles to model and design complex systems. It involved analyzing the problem domain, representing it using objects and their interactions, and then designing a modular and scalable solution. OOADM helped create systems that were easier to understand, maintain, and extend by organizing functionality into reusable and interconnected components.

Analysis of the Proposed System

Locality-Sensitive Hashing (LSH) is a robust algorithm tailored for detecting similarity efficiently in high-dimensional datasets. In the realm of Big Data, it finds significant application in music recognition systems, where it identifies and matches music tracks based on user inputs. This process leverages unique audio features and patterns to achieve precise results. The process begins with feature extraction, where essential characteristics of the audio tracks in the dataset are identified. These features include spectrograms, which represent the frequency spectrum of the audio over time, Mel-frequency Cepstral Coefficients (MFCCs) that provide a compact representation of the audio signal's timbral content, and chroma features, which capture the harmonic elements of the music. Together, these attributes form high-dimensional vectors that serve as the foundation for comparison.

Next, the algorithm employs hashing for similarity by mapping these high-dimensional vectors into a lower-dimensional space using multiple hash functions. Unlike traditional hashing methods that distribute data uniformly, LSH ensures that similar vectors are hashed into the same bucket. This increases the likelihood of grouping tracks with similar audio characteristics.

Once the data is hashed, indexing the database organizes it into hash tables, where each bucket stores tracks with comparable features. This structure significantly narrows the search space, enabling quicker and more efficient lookups.

When a user inputs a music query such as an audio clip, a recording, or even a hum, the system extracts its features using the same methodology. These features are then hashed using the same LSH functions, directing the query to specific buckets within the hash tables. This step, known as query matching, focuses the search on a relevant subset of the database.

From these buckets, the algorithm performs candidate selection and comparison, retrieving all potential matches (candidate set). A refined similarity measure, such as cosine similarity or Euclidean distance, is applied to rank the candidates by their closeness to the query's features.

Finally, the system delivers a recognition output, identifying the music track most similar to the user input. This result is accompanied by detailed metadata, such as the track's title, artist, album, and other relevant information. Through this structured process, LSH enables accurate and efficient music recognition, making it an invaluable tool for managing and exploring vast music libraries in Big Data applications.

Architecture of the Proposed System

In the context of music recognition within Big Data, the Locality-Sensitive Hashing (LSH) algorithm efficiently identifies and matches songs based on user inputs by leveraging unique features and structured processing.

Music Dataset

The system operates on a vast collection of audio tracks, where each song is represented by its digital attributes. This dataset serves as the foundation for the recognition process.

Feature Extraction

Key audio features such as spectrograms, Mel-frequency Cepstral Coefficients (MFCCs), and chroma features are extracted from each track. These features encapsulate the song's timbral and harmonic properties, creating high-dimensional vectors for analysis.

Database

The extracted features are organized into a structured database, enabling efficient access and retrieval during the recognition process.

Query Processing

When a user provides an audio query—whether as a clip, hum, or recording—the system extracts the same features from the input to ensure consistent comparison.

Hashing Songs

Using LSH, the high-dimensional vectors of both the database songs and the query are hashed into a lower-dimensional space. Similar songs are grouped into the same hash buckets, narrowing the search space.

Candidate Selection

The algorithm retrieves songs from the hash buckets containing the query, forming a candidate set of potential matches.

Ranking and Filtering

A fine-grained similarity metric, such as cosine similarity or Euclidean distance, ranks the candidates. Songs with higher similarity scores are prioritized.

Recommendation

The system identifies the most similar song to the query and recommends it, providing metadata like the title, artist, and album to the user. This efficient, scalable approach ensures accurate results even within large music datasets

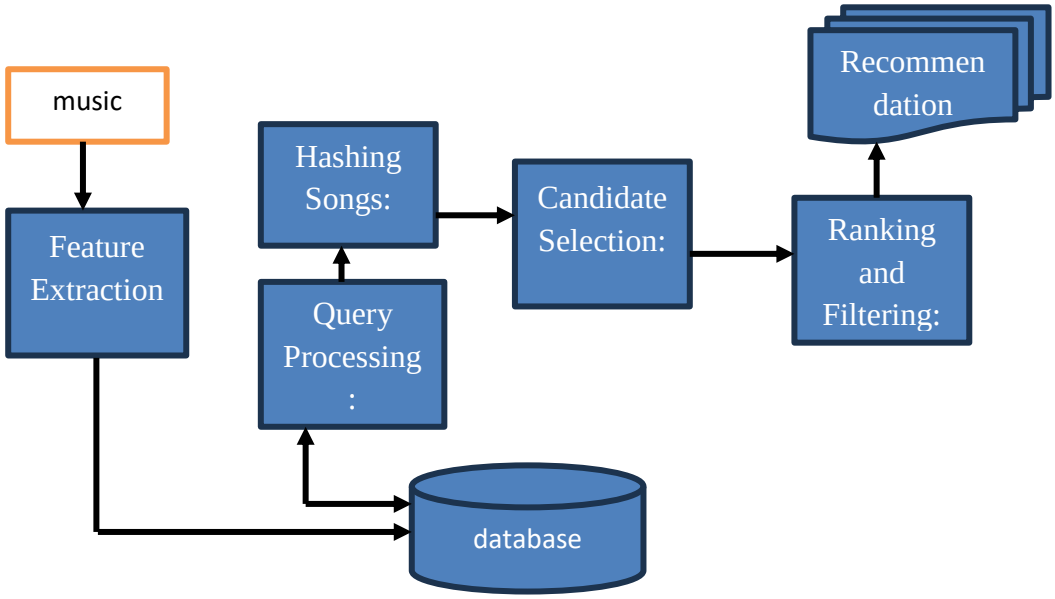


Figure 1: Architecture of the Proposed System

Mathematical Model of the Proposed System

Let D represent the database of music tracks, where $D=\{x_1,x_2,\ldots,x_n\}$, and each x_i is a feature vector extracted from a music track (e.g., Mel Frequency Cepstral Coefficients - MFCCs).

Query track feature vector: q

The feature vectors x_i and q are mapped into a d-dimensional space, R^d, for similarity comparison.

Define a similarity metric sim(x,y), typically cosine similarity or Euclidean distance, to measure closeness between vectors.

Use a family of hash functions H to map high-dimensional feature vectors to hash codes. A specific hash function $h \in H$ is defined as:

$h(x)=sign(a \cdot x+b)$,

where:

a is a random vector drawn from a standard normal distribution.

b is a random offset in R.

Generate multiple hash functions h1,h2,…,hk to map feature vectors into k-dimensional hash buckets: g(x) = [h1(x),h2(x),…,hk(x)].

Store each music track x_i in a hash table using the hash key $g(x_i)$. Multiple hash tables, T_LT1,T2,…,TL are constructed to improve retrieval performance.

Compute the hash code g(q) for the query track q and retrieve candidate tracks CAND from the corresponding buckets across all hash tables.

For each candidate track $x_i \in$ CAND, compute the similarity $sim(q,x_i)$. Select the top k tracks with the highest similarity scores.

The system outputs a ranked list of music tracks most similar to the query q.

Algorithm of the Trained Model

The Locality-Sensitive Hashing (LSH) algorithm is designed to efficiently identify approximate nearest neighbors in high-dimensional spaces. This approach is particularly useful for tasks such as near-duplicate detection, clustering, and recommendation systems.

1: Initialize all parameters

2: for number of training iteration do

//recommendation on module

3: for t steps do

4: Sample highly interactive and less interactive information for user items from matrix Y, the feature vectors of the user u and the item v are obtained through the interaction matrix Y; 5: Sample Y~X(v) for each item v in the extracted sample;

6: Update parameters of Y by gradient descent on Eq. (5)

7: end

USE-CASE diagram of the Proposed System

When a user logs in, their profile vector is hashed using the same family of hash functions. The system retrieves candidate movies from the corresponding buckets across all hash tables. Compare the candidate music with the user's profile using a finer similarity measure to filter out the best matches. This step ensures that only the most relevant movies are recommended. Recommendation: Present the filtered list of movies to the user as personalized recommendations. Figure 2 illustrates a Use-Case diagram of the Proposed System

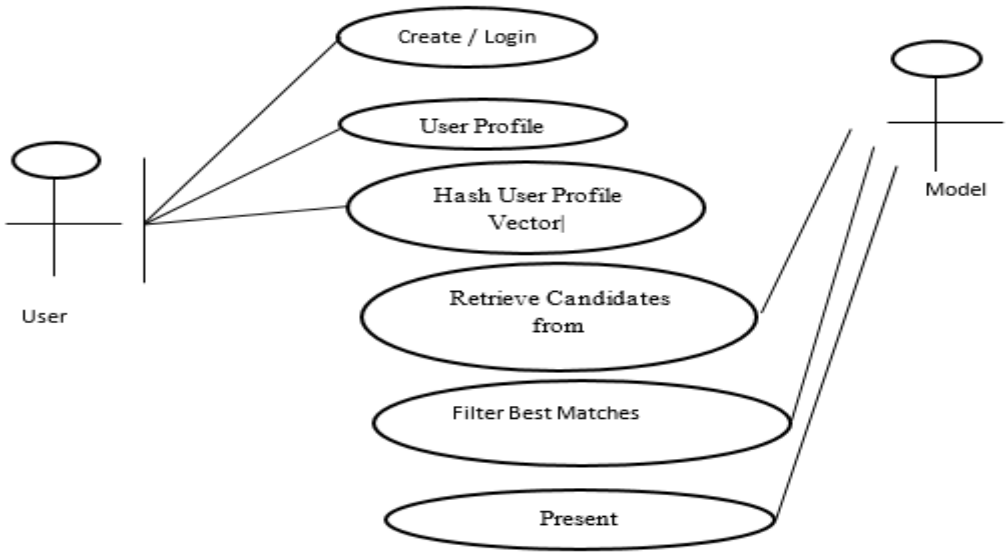


Figure 2: Use-Case diagram of the Proposed System

Flow chart

The algorithm begins with the input of a dataset containing vectors representing data points. These vectors could represent features of documents, images, or any high-dimensional data. Parameter Selection: Users specify parameters crucial for LSH, such as the number of hash tables to use, the number of hash functions per table, and the threshold for considering two vectors as potential neighbors. Each hash table contains buckets where similar vectors will be grouped together. Compare the candidate vectors retrieved from hash tables with the query vector using a similarity metric (e.g., cosine similarity). Verify candidates that meet the user-defined similarity threshold. Figure 3 shows a Flowchart Diagram of the Proposed System

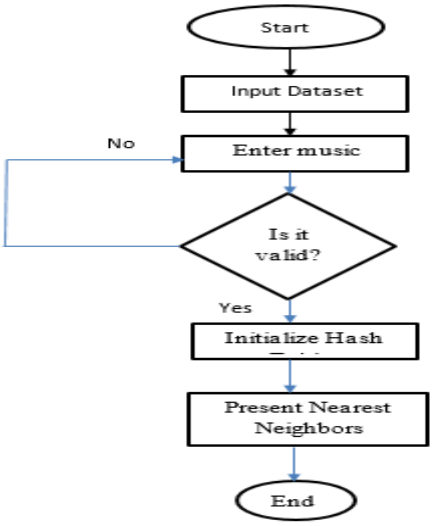
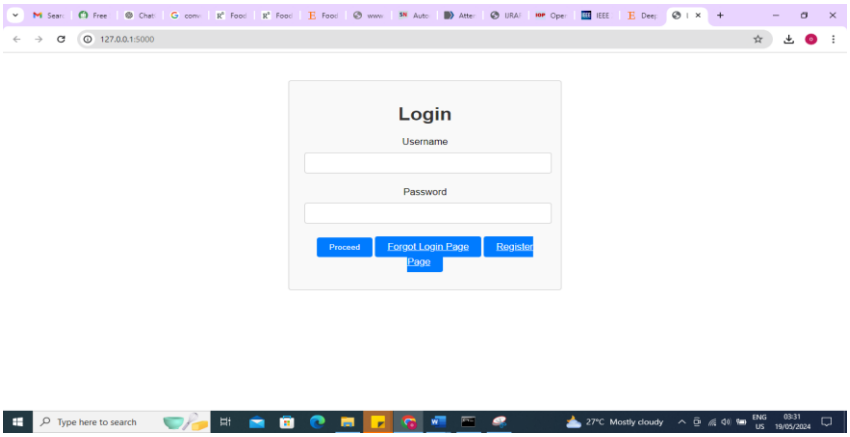


Figure 3: Flowchart Diagram of the Proposed System

Output Screen

We designed and implemented the login and registration pages to ensure secure and efficient user access. For the registration page, we provided fields for the user to input their username, email, and password. Upon successful authentication, the user was granted access to the system, and a session was initiated to manage their login state securely.



Trained Model

The process begins with the training model dataset, which comprises various attributes of music tracks, such as genre, tempo, key, and artist, along with user-specific data like past listening habits, ratings, and preferences. This dataset is used to create a multi-dimensional vector space where each music track and user profile is represented as a point in this space.

During the recommendation phase, the system first encodes the user's preferences into a vector. This vector is then hashed using LSH to find similar vectors, i.e., music tracks that share similar characteristics. By narrowing down the potential matches to a smaller subset of relevant tracks, the system can provide recommendations swiftly and with a high degree of relevance.

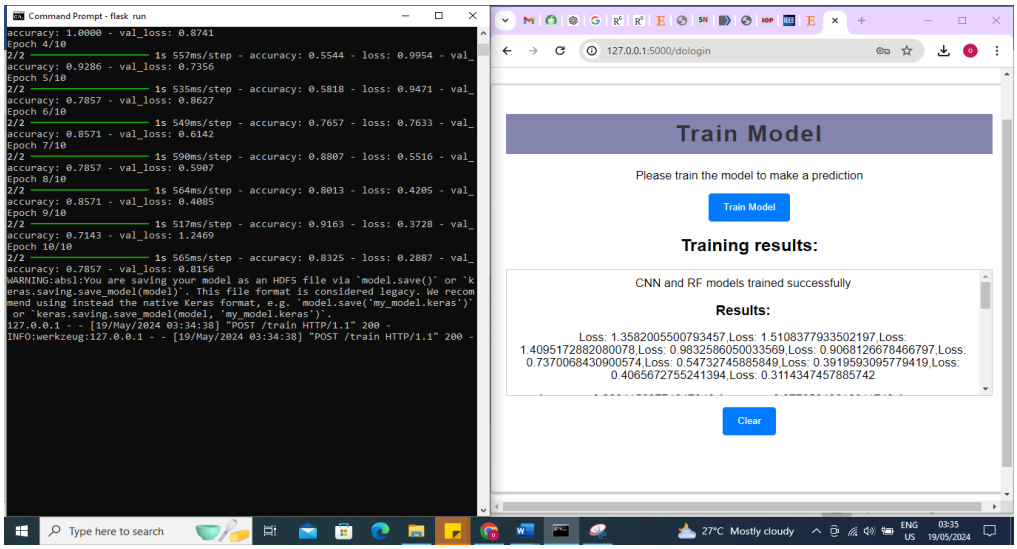


Figure 5 Train Model

When a user inputted an artist name and song name, the system transformed these inputs into a feature vector. The LSH algorithm then mapped this vector into the hash tables to identify clusters of similar music tracks. By leveraging the properties of LSH, the system quickly retrieved a set of candidate songs that were similar to the input track. The implementation of this music recommender system using LSH significantly enhanced the efficiency and accuracy of music recommendations. Users benefited from faster and more relevant suggestions, which contributed to a more engaging and satisfying interaction with the music platform

Figure 6 Result Output

Once the input song was processed and its feature vector was hashed, the system quickly identified the bucket containing the input song. It then retrieved other songs from the same bucket, as these songs shared similar musical features. This method significantly reduced the search space, making the recommendation process both swift and accurate.

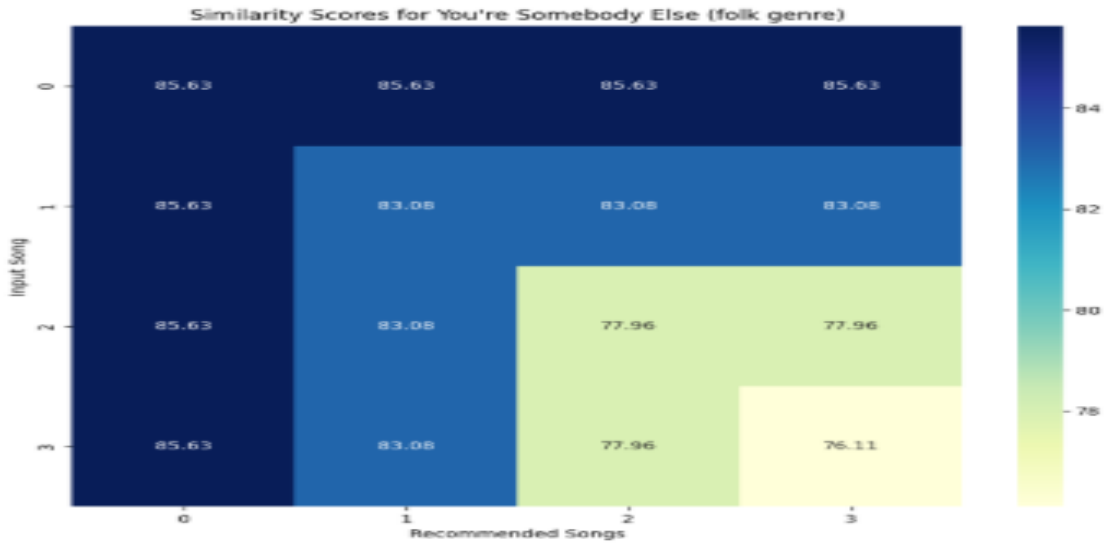


Figure 7 Metric recommender song

During testing, the system's performance was evaluated using metrics such as true positive and true negative rates. True positives were instances where the system successfully recommended songs that the user liked, while true negatives were instances where the system correctly refrained from recommending songs that the user disliked.

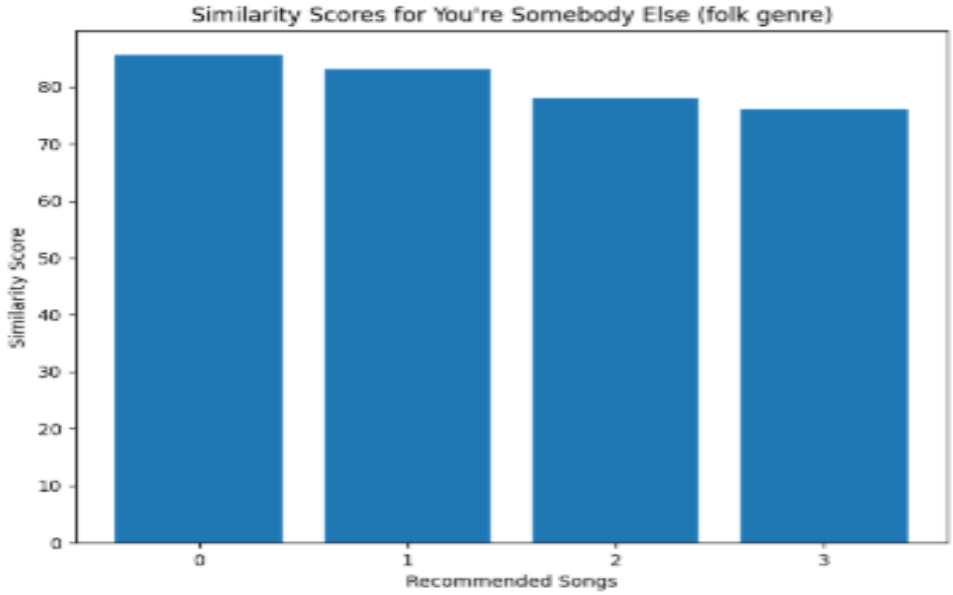


Figure 8 Similarity accuracy

DISCUSSION OF FINDINGS

The development of a music recommender system using the Locality-Sensitive Hashing (LSH) algorithm was a significant advancement in personalized music experiences. This system relied on user preferences, specifically the input values of artist name and song name, to suggest relevant music tracks.

In the initial phase, we collected a comprehensive dataset comprising various music tracks along with metadata, including artist names, song names, genres, and user ratings. This dataset served as the foundation for training and testing the recommender system. The LSH algorithm was chosen for its efficiency in handling high-dimensional data and its ability to quickly identify similar items.

We implemented the LSH algorithm to index the music tracks based on their features. By hashing the data into multiple hash tables, the algorithm efficiently clustered similar items together. Each music track was represented as a vector in a high-dimensional space, with features such as genre, tempo, and popularity contributing to the vector's dimensions

A music recommender system was developed utilizing the Locality-Sensitive Hashing (LSH) algorithm, tailored to user

preferences based on artist names and song names. This system aimed to provide users with personalized music recommendations by effectively leveraging the LSH algorithm's capability to handle high-dimensional data.

The core of the system's design lay in its ability to hash similar items into the same buckets with high probability, making the recommendation process efficient and scalable. By encoding the artist names and song names into high-dimensional vectors, the system transformed these data points into a format suitable for LSH. Each vector represented a unique combination of artist and song, ensuring that user preferences were accurately captured.

The LSH algorithm played a pivotal role in identifying similar items. It grouped songs by similar artists or with similar song names into clusters, making it easier to recommend tracks that aligned with the user's listening habits. This clustering process was achieved through the use of hash functions that projected the high-dimensional vectors into a lower-dimensional space, where similar items ended up in the same hash buckets.

Once the data was hashed, the system used these hash buckets to generate recommendations. When a user input their preferred artist or song name, the system quickly identified the corresponding hash bucket and retrieved other items within the same bucket. This approach significantly reduced the search space and improved the speed of generating recommendations.

CONCLUSION

The music recommender system using the Locality-Sensitive Hashing (LSH) algorithm has proven to be a valuable tool in enhancing user experiences in discovering new songs based on their preferences. Through the implementation of LSH, which efficiently clusters similar songs together, the system was able to provide accurate and personalized recommendations tailored to each user's taste in music.

The LSH-based music recommender system demonstrated its effectiveness in mitigating the cold-start problem by suggesting relevant songs even for new users or less popular songs. By leveraging user preferences encoded through LSH, the system delivered a seamless and enjoyable music discovery process. This not only improves user satisfaction but also promotes engagement and retention within the music platform.

REFERENCES

1. Adomavicius, R. Koper (2021), Personal recommender systems for learners in lifelong learning networks: the requirements, techniques and model, International Journal of Learning Technology, 3 (2008) 404-423.
2. Adomavicius, A. Tuzhilin (2022), Context-aware structured and unstructured data, Systems Handbook, Springer US2011, 217-253.
3. Adomavicius, P. Resnick, H.R. Varian (2022), Locality-Sensitive Hashing, Communications of the ACM, 40 (1997) 56-58.
4. Ai Q. Shambour, J. Lu (2023), A trust-semantic fusion-based Locality-Sensitive Hashing approach for e-business applications, Decision Support Systems, 54 (2012) 768-780.
5. Anonnya Banerjee A. Zenebe, A.F. Norcio (2021), Representation, similarity measures and aggregation methods using fuzzy sets for content-based recommender systems, Fuzzy Sets and Systems, 160 (2009) 76-94.
6. Asshraf J. Masthoff W. Woerndl, M. Brocco, R. Eigner (2021), Group modeling: selecting a sequence of television items to suit a group of viewers, User Modelling and User-Adapted Interaction, 14 (2004) 37-85.
7. Avesani C. Romero (2021), Applying web usage mining for personalizing hyperlinks in web-based adaptive educational systems, Computers & Education, 53 (20) 828-840.
8. Ben-Shimon C.-M. Chen, L.-J. Duh, C.-Y. Liu (2021), A personalized courseware recommendation system based on fuzzy item response theory, 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE '04), IEEE, 305-308.
9. Biletskiy S. Hsu, M.-H. Wen, H.-C. Lin, C.-C. Lee, C.-H. Lee (2022), AIMED - a personalized TV recommendation system, in: P. Cesar, K. Chorianopoulos, J. Jensen (Eds.) Interactive TV: a Shared Experience, Springer Berlin Heidelberg, 166-174.
10. Bobadilla D. O'Sullivan, B. Smyth, D. Wilson (2023), Preserving recommender accuracy and diversity in sparse datasets, International Journal on Artificial Intelligence Tools, 13 (24) 219-235.
11. Hung-Wen, S. Von-Wun (2022), A personalized restaurant recommender agent for mobile e-service, 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service. EEE. 259-262.