Real-Time Movie Recommendation System using Locality-Sensitive Hashing

Baramkula Vishnu Pavan[1], D V S Mihir[2], Yogen Aralaguppi[3] and Gayathri Ramasamy[4]*
{ bl.en.u4cse22108@bl.students.amrita.edu[1], bl.en.u4cse22110@bl.students.amrita.edu[2], bl.en.u4cse22168@bl.students.amrita.edu[3], gayathri_r@bl.amrita.edu[4] }

Department of Computer Science and Engineering, Amrita School of Computing, Amrita Vishwa Vidyapeetham, Bengaluru, Karnataka, India[1, 2, 3, 4]

Abstract. The increasing volume of online movie content and the need to categorize and rate them requires an efficient recommendation system. To achieve this, Locality-Sensitive Hashing (LSH) and fuzzy matching (FM) are used in the recommendation system. The method of LSH is described, focusing on both user-based and item-based approaches. To achieve this, LSH schemes MinHash and SimHash are used, which use cosine similarity and resemblance similarity to compare data elements. Moreover, fuzzy matching techniques are used to accommodate the uncertainty in user ratings and preferences which improves the accuracy of recommendations. The movie recommendation system is trained on a real-world dataset. LSH and fuzzy matching algorithm have their advantages over other methods such as KNN and therefore are ideal for building real-time recommendation systems that can effectively address the challenges of big data and user preference dynamics.

Keywords: Locality-Sensitive Hashing (LSH), Movie Recommendation System, Collaborative Filtering, fuzzy matching, MinHash,SimHash, cosine similarity, resemblance similarity, user preference dynamics, collaborative filtering, KNN alternative.

1 Introduction

There is a great demand for personalized recommendations, especially in areas like movies and e-commerce websites. A recommendation system is used to meet these demands, which usually uses KNN. However, there are several problems faced by recommendation systems such as the cold start problem, data sparsity, and the over-specialization problem. There exists a need to create a better recommendation system. We achieve this using LSH or Locality-Sensitive Hashing combined with fuzzy matching.

LSH offers a highly efficient approach to approximate nearest neighbor search, a crucial step in Collaborative Filtering algorithms, which can be categorized into user-based and item-based. By mapping similar items or users to the same hash buckets, LSH drastically reduces the search space in comparison to KNN. We also see various LSH schemes such as SimHash and MinHash and their applications in recommendation systems. LSH has the increased benefit of privacy because the user data can be anatomized and processed through hash functions without revealing any sensitive information.

Furthermore, to handle the uncertainties in user ratings and preferences, we try to incorporate fuzzy matching techniques into our movie recommendation system. The project is developed using Java.

ICITSM-Part I 2025, April 28-29, Tiruchengode, India

DOI 10.4108/eai.28-4-2025.2357924

2 Related works

Qi et al. [1] discuss perfecting the trade-off between data privacy and recommendation accuracy without violating the Quality of Service. It improves privacy by creating less sensitive user indices for neighbor searches while still delivering accurate and tailored suggestions.

Ayeteki et al. [2] introduce the concept of LSH to address scalability issues observed in neighborhood-based collaborative filtering in real-time applications. LSH effectively decreases the search space while allowing for real-time processing, further improving speed. Positive outlooks include better real-time efficiency, improved scalability with bigger datasets, and more consistent recommendation accuracy. Limitations include parameter sensitivity, the impact of data sparsity, and the possibility of overlooking recommendation novelty and diversity.

Anwar et al. [3] dives into a comparison between two recommendation algorithms that use CF and KNN, addressing cold start problems and sparsity issues in recommender systems, followed by a detailed analysis. Upon addressing these problems, [3] offers selection recommendations validated by experiments on the MovieTrust dataset.

Li et al. [4] talk about IKNN, an enhanced version of the KNN algorithm, and how recommendation systems using this algorithm tend to reduce data sparsity issues. IKNN offers better precision and accuracy in its recommendation process by using global effect factors and compression, validated by several experiments. Although it's easy to use and interpretable, it's deficient in scalability, and its evaluation is dataset-specific.

Hu et al. [5] suggest privacy-protecting methods for federated recommendation systems. Scalability and efficiency for similarity computing in large-scale systems are guaranteed when employing LSH. However, it lacks emphasis on data freshness in large-scale systems.

Lin et al. [6] explain the LSH methodology based on the collaborative recommendation method responsible for AI-driven recommender systems. Both privacy protection and calculation efficiency can be achieved. [6] mainly uses DisRecLSH, a type of LSH, and executes it in three phases: feature extraction, user indexing, and online recommendation. It enhances computational efficiency and provides collaboration among distributed data sources. However, issues might arise for sparse user-item interaction.

Farahani et al. [7] use AUPRSLA, a method for adaptive personalized recommender systems. It uses learning automata for accurate recommendations, capturing user preferences. However, this might impact accuracy due to reliability and completeness issues. Additionally, scalability issues arise with an increase in new data, suggesting the need for parallel or distributed computing.

Paleti et al. [8] introduce an ALS approach for mitigating the cold start problem in recommender systems, combining Louvain’s algorithm with alternating square factorization for effective recommendations. It helped improve prediction accuracy by reducing errors between actual and predicted data. However, it might result in poor performance due to missing item interactions, suggesting a need for collaborative filtering methods. Complexity can increase with an increasing number of users.

Kumar et al. [9] use the DRWMR approach to enhance recommender systems by tackling data sparsity and the cold start problem through a two-phase process: neighbor formation and recommendation. DRWMR efficiently enhances recommendation precision and accuracy compared to other methods. Challenges remain with increasing items and data.

Xu et al. [10] introduce an enhanced collaborative recommendation algorithm combining locality-sensitive hashing (LSH) and matrix factorization to improve efficiency and save user privacy. [10] use LSH for neighbor search, which leads to privacy by using hash values instead of plain data. It also predicts missing ratings to address data sparsity and uses neighborhood-based collaborative filtering methods. The advantage of using the LSH method is that it speeds up the neighbor search and results in faster recommendations and increased accuracy.

Shrivastava and Li [11] provide a comprehensive analysis of MinHash and SimHash and states why MinHash reigns supreme in neighbor search in binary data, especially in search applications. It demonstrates the higher retrieval accuracy and efficiency of MinHash over SimHash, particularly in high-similarity regions.

Heidari et al. [12] explores recommender systems and categorizes them into content-based and collaborative filtering with their respective challenges and advantages. Matrix Factorization (MF), a model-based method, ensures accuracy but still faces challenges such as the cold start and sparsity. Existing methods lack addressing long-distance dependencies in data sequences. [12] propose using a deep learning-based method addressing sparsity and cold-start problems by integrating side information using self-attention with matrix factorization. The complexity of deep learning-based methods might be difficult in real-time practical implementation, specifically in resource-constrained environments.

Harper et al. [13] The MovieLens dataset have been downloaded more than 140,000 times and have been cited in more than 7,500 scholarly articles MovieLens datasets are widely accepted as valid and reliable resources for scholarly research. Each dataset captures more than 17 years of data, allowing findings to include long-term trends and patterns in user behavior and recommendations. Within the context of recommendation systems, the datasets can be used to solve problems that are of interest to data scientists and data engineers alike, such as data summarization, discovery of patterns and relationships, and data visualization and exploration. As with all human assessments, ratings on movies are inherently subjective, which makes the MovieLens datasets ideal for the evaluation and enhancement of technologies that personalize. The datasets have had a large impact on teaching, research, and industry, including the creation of high-quality tools for teaching and learning, software systems, and commercial ventures (start-ups).

Mao et al. [14] use fuzzy content matching to improve recommendation accuracy. It models user ratings and preferences using fuzzy numbers to handle uncertainty in user ratings, establishes relationships between item descriptors to help complete user profiles, and shows better performance on Yelp and MovieLens datasets.

Jahanvi et al. [15] work compares Convolutional Neural Networks (CNNs) for the classification of skin cancer based on images from the HAM10000 dataset. It compares custom CNN and MobileNetV2 architectures with various optimization algorithms (Adam, Adagrad, SGD, Adadelta). Experiments indicate that MobileNetV2 with SGD has the highest accuracy, and this demonstrates deep learning's ability to enhance dermatological diagnosis.

Murthy and Kavitha [16] studies on secure fuzzy keyword search on encrypted cloud data has been investigated recently using cryptographic indexing, encrypted data structures, similarity measures, and verifiability mechanisms for maintaining privacy. The research considers efficient search improvement, multi-keyword query support, and addressing security issues. The systems to be proposed need to achieve tradeoffs between security and performance with supporting accurate privacy-preserving search over sensitive information stored in clouds.

Ramasamy et al. [17] provide a framework for pest identification in the Coconut Leaf Dataset using cost-sensitive learning, multi-class classification, and under sampling to address class imbalances. It uses ResNet-50 for the accurate detection of pests and evaluates the performance with metrics such as accuracy, precision, and F1 score. Future research should be on advanced architectures, ensemble learning, and real-world validation for enhanced pest identification in agriculture.

Shaik et al. [18] explore using deep learning models such as EfficientNetB2 and ResNet for the early detection of Alzheimer's disease based on MRI scans. SMOTE and data augmentation helped to handle class imbalance to improve accuracy. The paper concludes that, in complexity and performance, EfficientNetB2 is the most efficient model and recommends more research into larger, uniform datasets that could generalize these methods in neurodegenerative disorder diagnosis.

3 Methodology

The movie recommendation system developed in this paper combines features of both collaborative filtering and locality-sensitive hashing algorithms to provide accurate movie recommendations with minimal search time. The process starts with data loading and data pre-processing which involves transforming and cleaning the data. The data set containing the user IDs, movie IDs, and the corresponding rating given is loaded from the CSV files and is stored in the Hashmap. Specifically, it makes it easier to review data and changes as the need arises in the standard format. Then the movie names are read from the other CSV file into another HashMap so that the movie names can be quickly accessed by their IDs.

MovieLens dataset is a complete set of user ratings and movie attributes and descriptions needed to create a recommendation system. Containing more than 20 million ratings from 138000 users on 27000 movies, it represents a large data set that can be used to develop efficient and scalable recommendation algorithms.

The dataset comprises two main components:

ratings.csv: It contains user-movie ratings with the fields: unique user and movie ids, rating given by the user id and the time the rating was given in Unix timestamp format. The ratings provide the foundation for building the user-item matrix that is used in collaborative filtering approaches.

movies.csv: Here, each movie is identified by the unique ID, the movie title, and the list of genres separated by commas. In this way, using movie IDs for further referencing and matching the IDs with movie titles and genres makes it easier for the user to navigate the recommendations in the context of movie titles.

Key characteristics of the dataset include:

Sparsity: Because the number of movies is so large and the rating data is dense which means that users generally rate only a small subset of the total set of movies. The sparsity reduces the amount of empty space in the matrix and improves both storage and computation.

Temporal Aspect: Despite the fact that timestamps enable time-related analysis and recommendation strategies called temporal strategies, including trends or seasonal recommendations, this feature is not employed in our current work. But it opens the possibilities for the improvement of the recommendation systems in the future.

MoviesLens data set was used to generate this user-item matrix. The 2-tuple or matrix is a user-item matrix where the cell represents what value user U has given to item i, thus allowing us to make a finer analysis of how users interact with the items.

The basis of the recommendation system is the sparse user-item matrix based on previous users' interactions with items. This matrix focuses more on the users, as it reflects on the ratings given by the users with respect to the movies. While generating recommendations, the first step involves creating LSH signatures for the users.

Computing similarities for high-dimensional data is an optimization problem, here LSH is used as an optimal method to estimate similarities between the two data points. In the given user-item matrix, multiple hash functions are applied to derive totally different signatures for various users so that the ratings of similar users can be distinguished.

For time series data, different approaches for address matching are accounted for variations and noise in the user entry. This means that irrespective of the number of characters the user fails to type, or the typo mistakes the user may make in a movie title, the database can accurately highlight the right movie. After selecting a movie, the system finds the other individuals who have watched the same movie and rates it above a given rating point, after which it gives the user the list of recommended movies based on the individual's preferences.

Recommendations derived are based on the data of movies other users with similar interests to the given movie liked. From these ratings, the system determines scores of corresponding movies to help in sorting and generating the preferable recommendations. This form of collaborative filtering works on the following principle:

- \\ If two users A and B both liked the same movie, A would like what B likes, or in this case, A would rate other movies that B rated highly.

Finally, the obtained results are presented to the user who receives the list of movies that could be considered relevant for him.

3.1 System Architecture

The system architecture is depicted in fig 1:

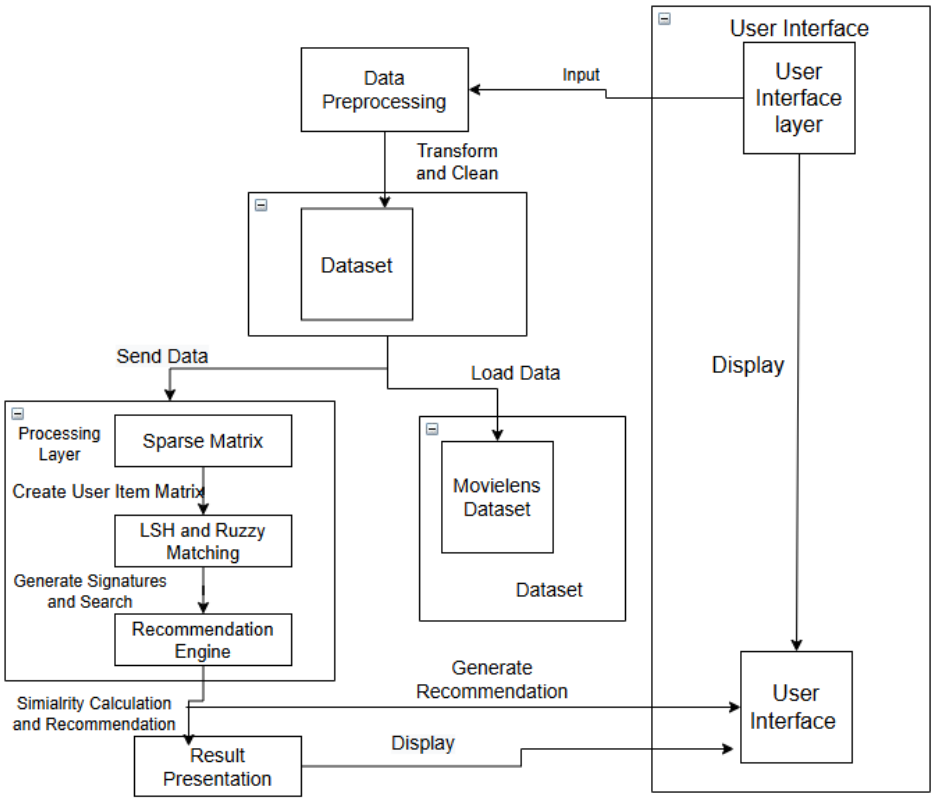

Fig. 1. Schematic Flow.

3.1.1 Data Loading and Preprocessing:

The most effective solution to read the data input is to use the CSVReader from the OpenCSV library to read the ratings and the movie titles from the CSV files. The actual ratings are stored in a HashMap along with the unique ID for the user and the IDs of the movie it relates to; the keys are arrays containing the unique user ID for each entry and the actual ratings are the value that make it easier to manipulate the data for access. Movies for a catalogue are stored in a HashMap format improving the retrieval of a specific movie name given the movie ID.

3.1.2 User-Item Matrix:

A 2-dimensional array is used to capture the user ratings on movies which is in the form of a sparse matrix. It consists of a cross-tabulation with the users as the first dimension and movies as the second dimension (where the latter highlights the ratings that different users have assigned to different movies). While building the matrix, across the rows of the matrix, it first becomes

a nested HashMap where the key is a user number and the value is another HashMap where the keys are the movie number and the value is the rating.

3.1.3 Locality-Sensitive Hashing (LSH):

This is done for each user to create its specific set of different LSH signatures with respect to each hash function applied to the user-item matrix in order to increase the likelihood of collision between similar users or items. These signatures help in identifying other similar-minded users or those who share the same preferences within a short span of time. These signatures are stored in HashMap having the user ID as the keys and the list of hash values as the value of that key.

3.1.4 Fuzzy Matching

As for the search terms for movie titles, the Fuzzy Search library is used as a tool for handling minor changes in the titles. This is helpful to search for a movie and connect it to the prospective listener even in the case when entering of the name is incorrect or contains only a part of it. Another class named FuzzyMatcher is added, which is the main body to find out the optimal match for the given input and to do the main fuzzy matching.

3.1.5 Recommendation Engine

In the event that a specific movie fades from the memory of the system at the fuzzy matching stage of the system, then highly rated users who have seeded this specific picture are also found. After that, it gathers a list containing the movies that these users liked and calculates similarity based on the provided ratings.

If there are many more movies to be rated, the system maintains a pending movies list and recommends the movies according to the highest value of the score achieved here in the context of the top recommended movies ranking.

3.2 Framework

The project is implemented in Java and IDE used is Eclipse.

The main libraries and Tools are:

Open CSV: It is used for reading and writing CSV files which simples the process of loading ratings and movie titles from file.

Fuzzy Wuzzy: This library is used for fuzzy string matching to account for variations and inaccuracies in user input while searching for movie titles.

Java Collection Framework: Used extensively for data storage and manipulation in particular through Hash Maps and Array Lists for efficient data retrieval and storage.

It is modular and is extensible in which integration of additional features is easy. The use of interfaces and abstract classes enables flexibility and reusability of code components. To visualize through bar graphs and plots, Matplotlib and Seaborn has been used, extracting data from a Pandas Data frame.

Matplotlib is an important library of data visualization in Python and it is used in various research and industry applications.

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics, which can visualize relationships and distributions and can work well with Pandas data frame.

3.3 LSH Signature Generation

Let u be the set of users. M be the set of Movies. $R_{u, m}$ be the rating given by user u to movie m. H be the number of hash functions. s_u be the LSH signature for user u.

3.3.1 Initialize Signature

For each user u € U initialize their signature s_u as a list of H elements, each initialized to the maximum possible integer value:

$$s_u = [\text{MAX\_INT}, \text{MAX\_INT}, \ldots \text{MAX\_INT}] \quad (1)$$

3.3.2 Hashing

For each user u € U and each movie m € M where $R_{u, m}$ is not null:

Generate a random hash value $h_{u, m}$ using a hash function h_i (where i is the index of the hash function in the signature):

$$h_{u,\ m} = h_i(u, m) \quad (2)$$

Update the corresponding element in the signature s_u with the minimum value between the current value and the hash value:

$$s_u[i] = min(s_u[i], h_{m,n}) \quad (3)$$

3.3.3 Final Signature

After iterating through all users and movies, the final signature s_u for each user u will contain the minimum hash values for all movies rated.

4 Results and Discussion

4.1 Analysis of MovieLens Dataset

The distribution is described as a long tail, where some movies are rated often while many films are rated rarely because a number of people tend to give limited ratings to them.

Top-rated films: Some movies may have as many as 120,000 ratings, possibly due to being box office hits or indie movies with a huge following.

Long tail: Most movies receive low ratings, indicating a large population of relatively unknown or specific films. This distribution implies that a small percentage of content attracts most consumers, a trend observed in digital platforms.

Power users: A few users rate a large number of movies, with the most active user having rated over 12,000 movies. These users contribute significantly to the system's data.

Casual users: Many users contribute minimally, with only a few ratings. However, their contributions collectively affect the overall volume of ratings.

Movie rating: scores provide an idea of a movie's quality.

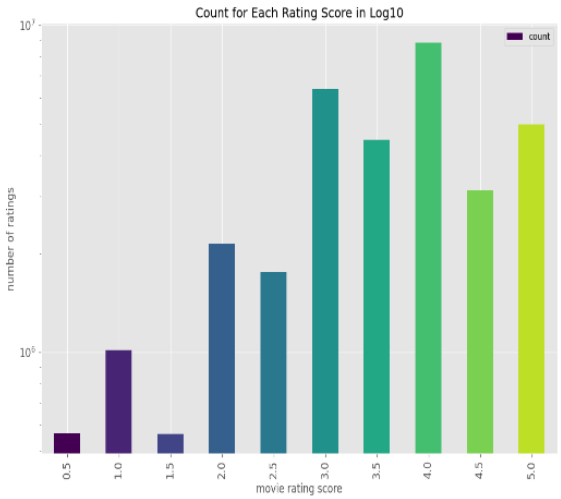

Fig. 2. Distribution of the rating score in logarithmic scale.

Bimodal distribution: Ratings are distributed with two major humps, indicating two groups of active and positive respondents as shown in fig 2. The highest frequency is at 4.0, suggesting a positive skew in ratings. Low extreme ratings of 0.5,1 and 1.5 are less frequent, suggesting users are reluctant to give the lowest scores unless thoroughly dissatisfied.

4.2 Comparison between LSH and KNN + CF for Recommendation Systems

4.2.1 Scalability and Real-time Performance

KNN and CF suffer from scalability issues as computational complexity increases linearly with the increase in the size of the dataset, making it impractical for large datasets.

LSH addresses the scalability problem by hashing items into buckets such that similar items have a higher probability of being mapped to the same bucket, which significantly reduces the search space.

4.2.2 Accuracy and Diversity

KNN and CF provide accurate recommendations; however, the movies they recommend might lack diversity. While this might be important in recommendation engines that back up e-commerce and dating apps, this might hinder a user from discovering new items.

LSH's hashing techniques balance accuracy and diversity, ensuring that a wider variety of items are recommended across all users, keeping the recommendations fresh and exciting.

4.2.3 Computational Efficiency

CF + KNN involve extensive similarity computation, making them less efficient for real-time recommendations.

LSH uses hashing techniques to approximate nearest neighbors, reducing the need for extensive similarity computations, making it more efficient and faster than CF-based algorithms, and thus more suitable for real-time applications.

4.3. Our Approach

The findings of the effectiveness of the proposed Movie recommendation system appeared quite encouraging in terms of accuracy and time complexity. The fuzzy matching algorithm was also apt in operation when interpreting appropriate user inputs with slight errors as it would ensure the appropriate movie was chosen amidst similar movie titles. This made the use easier because the input from the users could be in any format and the program would have to adapt to that format.

The system achieved satisfactory results and guaranteed fast generation of user signatures and subsequent similarity searches by employing the LSH technique, thus decreasing the real-time computational process many times compared to conventional approaches. In this manner, the recommendation process was made very fast but yet very efficient and accurate. It has also been deduced that the recommendation produced by the system is quite appropriate and in proximity to the user's tastes, thereby revealing how efficient collaborative filtering is alongside LSH. Fig 3 show the Recommendation for Dune (1984)

From the rating data, the user-item matrix allowed the system to study the user's similarities and combine the presented ratings to optimize these recommendations. This was ensured as it allowed for the assessment of user's behavior and their preference in an all-encompassing manner. However, the recommender system uses a large portion of resources.

In terms of the similarity search, we employed LSH and, thus, reduced the computation time regarding recommendations. So, it can be concluded it is feasible to provide real-time prediction and recommendation even if a large amount of data is involved. The output of our approach is shown in fig 3.

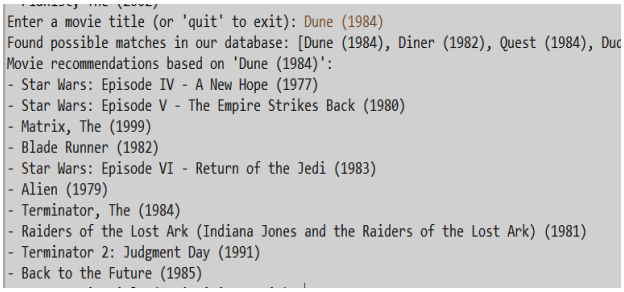

Fig. 3. Recommendation for Dune (1984).

5 Conclusion

In this paper, it is observed that the application of fuzzy matching, collaborative filtering, and LSH, the recommendation system for the movie has proved to be an effective way of recommending movies. The applied methodology proved suitable for the management of the large amount of data, for the integration of the variations of the user input, and for the real-time context of the recommendations. This type of fuzzy matching used in the system's recommendation algorithm helped identify movies correctly based on inputs from the users.

The application of the LSH for the similarity searches showed an increase in the operation's scalability and efficiency where it made it possible to recommend large amounts of data and a large number of users. The use of the above techniques helped in the formulation of the recommendation system that was at the same time optimal in terms of accuracy and performance since it offered appropriate recommendations based on user's behaviors and choices of movies.

References

[1] L. Qi, X. Wang, X. Xu, W. Dou, and S. Li, “Privacy-aware cross platform service recommendation based on enhanced locality-sensitive hashing,” IEEE Transactions on Network Science and Engineering,vol. 8, no. 2, pp. 1145–1153, 2020.

[2] A. M. Aytekin and T. Aytekin, “Real-time recommendation with locality sensitive hashing,” Journal of Intelligent Information Systems, vol. 53, pp. 1–26, 2019.

[3] T. Anwar, V. Uma, M. I. Hussain, and M. Pantula, “Collaborative filtering and knn based recommendation to overcome cold start and sparsity issues: A comparative analysis,” Multimedia tools and applications,vol. 81, no. 25, pp. 35693–35711, 2022.

[4] B. Li, S. Wan, H. Xia, and F. Qian, “The research for recommendation system based on improved knn algorithm,” in 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA), pp. 796–798, IEEE, 2020.

[5] H. Hu, G. Dobbie, Z. Salcic, M. Liu, J. Zhang, and X. Zhang, “A locality sensitive hashing-based approach for federated recommender system,” in 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID), pp. 836–842, IEEE, 2020.

[6] W. Lin, X. Zhou, L. Sun, L. Qi, S.-B. Tsai, Y. Yang, H. Liu, H. Kou, and L. Kong, “A locality-sensitive hashing based collaborative recommendation method for responsible ai driven recommender systems,” IEEE Transactions on Artificial Intelligence, 2024.

[7] M. G. Farahani, J. A. Torkestani, and M. Rahmani, “Dynamic user profile for adaptive personalized recommender system using learning automata,” Multimedia Tools and Applications, vol. 83, no. 17, pp. 49905–49925, 2024.

[8] L. Paleti, P. Radha Krishna, and J. Murthy, “Approaching the cold-start problem using community detection based alternating least square factorization in recommendation systems,” Evolutionary Intelligence,vol. 14, pp. 835–849, 2021.
[9] S. Kumar, V. K. Chauhan, D. Upadhyay, S. Singh, and P. Tripathi,“Enhancing recommender systems to alleviate data sparsity and the cold start problem,” in 2023 12th International Conference on System Modeling & Advancement in Research Trends (SMART), pp. 486–491,IEEE, 2023.
[10] J. Xu, X. Li, H. Wang, H.-N. Dai, and S. Meng, “Lsh-based collaborative recommendation method with privacy-preservation,” in 2020 IEEE 13th International Conference on Cloud Computing (CLOUD), pp. 566–573, IEEE, 2020.
[11] A. Shrivastava and P. Li, “In defense of minhash over simhash,” in Artificial intelligence and statistics, pp. 886–894, PMLR, 2014.
[12] N. Heidari, P. Moradi, and A. Koochari, “An attention-based deep learning method for solving the cold-start and sparsity issues of recommender systems,” Knowledge-Based Systems, vol. 256, p. 109835, 2022.
[13] F. M. Harper and J. A. Konstan, “The movielens datasets: History and context,” Acm transactions on interactive intelligent systems (tiis), vol. 5, no. 4, pp. 1–19, 2015.
[14] M. Mao, J. Lu, G. Zhang, and J. Zhang, “A fuzzy content matching-based e-commerce recommendation approach,” in 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pp. 1–8, 2015.
[15] S. Murthy and C. Kavitha, “Preserving data privacy in cloud using homomorphic encryption,” in 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA), pp. 1131–1135, IEEE, 2019.
[16] M. Jahnavi, K. Haritha, R. Gowtham, and S. M. Rajgopal, “Secure fuzzy keyword search in cloud computing using levenshtein distance and cryptographic indexing,” in 2024 5th International Conference on Data Intelligence and Cognitive Informatics (ICDICI), pp. 41–45, IEEE,2024.
[17] G. Ramasamy, M. Gurupriya, C. S. Vasavi, and B. Karthikeyan, “A cost-sensitive learning approach with multi-class classification and undersampling techniques for pest identification in the coconut leaf dataset,” in 2024 3rd International Conference on Artificial Intelligence For Internet of Things (AIIoT), pp. 1–5, IEEE, May 2024.
[18] B. A. Shaik, M. V. Narayani, A. R. Manikanta, N. R. Durupudi,T. V. Reddy, and G. Ramaswamy, “Comparative analysis of multi-class classification deep network for mri images using ml-based augmentation technique,” in Challenges in Information, Communication and Computing Technology, pp. 443–448, CRC Press, 2025.