Recommending Music and the Audioscrobbler Data Set

S M Samin
CSE, AUST, Dhaka, Bangladesh

Tasnim Mahmud
CSE, AUST, Dhaka, Bangladesh

Anika Maisha Tasnim
CSE, AUST, Dhaka, Bangladesh

Azmain Soad
CSE, AUST, Dhaka, Bangladesh

Md. Tanzilur Rahman
CSE, AUST, Dhaka, Bangladesh

***Abstract*—Recommender systems have become an integral part of many online platforms, helping users navigate vast amounts of content by suggesting items tailored to their preferences. This paper explores the implementation and evaluation of a music recommendation system using the Alternating Least Squares (ALS) algorithm within the Apache Spark framework and Deep Neural Network with a combination of Apache Spark, TenserFlow, and Keras framework. The focus is on implicit feedback data, where user interactions with items (such as play counts) are used to generate recommendations. We discuss the underlying principles of collaborative filtering and latent factor models, emphasizing the scalability and efficiency of the ALS and DNN algorithms for large-scale data. Through a case study using a dataset of user interactions with songs, we demonstrate how the ALS and DNN algorithms can provide accurate and personalized music recommendations.**

I. Introduction

Recommender systems are ubiquitous in today's digital landscape, enhancing user experiences on platforms ranging from e-commerce to social media and music streaming services. These systems leverage machine learning techniques to analyze user behavior and suggest relevant items, thereby increasing user engagement and satisfaction. Among various approaches to recommender systems, collaborative filtering, particularly latent factor models, has proven to be highly effective.

One widely used algorithm in this domain is the Alternating Least Squares (ALS) algorithm, which is particularly suitable for large-scale datasets with implicit feedback. Implicit feedback refers to data derived from user interactions, such as clicks, views, or play counts, rather than explicit ratings. This type of data is abundant but often sparse, posing challenges for recommendation algorithms. The ALS algorithm addresses this sparsity by approximating the user-item interaction matrix through matrix factorization, uncovering latent factors that represent underlying user preferences and item characteristics.

In addition to ALS, Deep Neural Networks (DNNs) offer a powerful framework for capturing complex user-item interaction patterns. DNNs can model non-linear relationships in the data, which makes them a compelling option for personalized recommendation systems, especially when combined with collaborative filtering approaches. By leveraging neural networks to learn intricate feature representations, DNNs have the potential to outperform traditional methods, particularly on large and diverse datasets.

This paper presents a practical application of both the ALS algorithm using Apache Spark's MLlib library and a DNN model implemented with TensorFlow and Keras to build a scalable music recommender system. We explore the effectiveness of these models using a dataset of user interactions with songs, evaluating their performance in terms of accuracy and scalability. Our findings demonstrate the strengths and limitations of both approaches, highlighting ALS's efficiency in handling sparse data and the promise of DNNs for more complex data representations.

II. Related Works

Rui Lu and co-authors [1] proposed an artificial neural network architecture, namely, Triplet MatchNet, which was trained to directly detect the acoustic similarity of music. This architecture is based on residual blocks with shortcut connections. Expert assessment was used as a quantitative similarity measure of music sounding.

The authors K. Gurjar and Y. Moon[2] focus on two main areas: an overview of MIR systems and a detailed comparison of different music similarity measurement methods. They discuss the challenges in MIR, including the complexity of accessing multifaceted musical information and the need to handle diverse music representations across different cultures and user experiences.The paper categorizes similarity comparison methods into note-based and frame-based methods. Note-based methods compare sequences of musical notes, while frame-based methods use image comparison techniques on frames of musical data. The authors also cover application-specific approaches like ground truth-based methods, probability matching, and N-gram approaches.

The paper by Michaël Defferrard et al[3] introduces the Free Music Archive (FMA) dataset, which is designed to support a wide range of music information retrieval (MIR) tasks. The FMA dataset includes over 100,000 tracks with detailed metadata and pre-computed audio features, making it suitable for tasks such as genre recognition, music recommendation, and audio analysis. The dataset is organized into subsets of different sizes to accommodate various scales of research, promoting reproducibility and collaboration in the

MIR community by providing an open-access, standardized benchmark for evaluating MIR algorithms.

The paper by Thierry Bertin-Mahieux et al.[4] introduces a large-scale dataset designed to advance research in music information retrieval (MIR). The Million Song Dataset (MSD) contains audio features and metadata for one million contemporary music tracks, aiming to provide a comprehensive resource for tasks such as genre classification, music recommendation, and temporal analysis of music trends. The dataset does not include the raw audio but offers detailed features and metadata, facilitating large-scale experiments and fostering innovation in the MIR field by making a substantial amount of data freely available to researchers.

III. Motivation

Music streaming platforms have grown exponentially, presenting users with vast libraries of millions of songs and artists.The key challenge is to build systems that can effectively recommend music by learning from a user's listening history and behaviors, providing a more curated, personalized experience.Our project addresses this problem by implementing a collaborative filtering-based recommendation system using Apache Spark, a powerful distributed computing framework, allowing for efficient processing of large-scale datasets, such as those found in modern music platforms. Many music recommendation systems are limited by their ability to handle large datasets, especially when the user base and music library expand. Our system uses Apache Spark to perform distributed data processing, enabling us to handle millions of user-artist interactions efficiently. This distinguishes our work from others that may use less scalable frameworks, like single-node systems or batch processing, which struggle with real-world music recommendation tasks at scale.

Although our core approach uses collaborative filtering through ALS, our system is designed to be extended into a hybrid recommendation model, where user preferences can be combined with content-based features (e.g., genre, artist metadata) for enhanced recommendation performance.

IV. Dataset

This dataset contains profiles for around 50,000 real people. The dataset lists the artists each person listens to and a counter indicating how many times each user played each artist. This dataset contains three txt files. These are:

- **userartistdata:** The file consists of three columns: userid, artistid, and playcount. userid uniquely identifies each listener, while the artistid tracks different artists. Playcount records how many times a user has listened to a particular artist. This structure helps to analyze user preferences, allowing a recommendation system to predict and suggest new music based on past listening behavior.
- **artistdata:** The file contains two columns: artistid and artistname. artistid provides a unique identifier for each artist, while artistname stores the corresponding name of the artist. This structure allows easy mapping between artist IDs and their names, which is essential for organizing, retrieving, and displaying artist information in applications like recommendation systems or music databases.
- **artistalias:** The dataset consists of two columns: badid and goodid. The badid represents incorrectly spelled or erroneous artist IDs, while the goodid provides the correct corresponding artist ID. This file is used to correct errors in the userartistdata by mapping incorrect artist IDs to the correct ones during data processing

V. Methodology

In this section, we present a comprehensive overview of our proposed methodology for both the ALS and DNN models. This approach is visually represented in Figure 1.

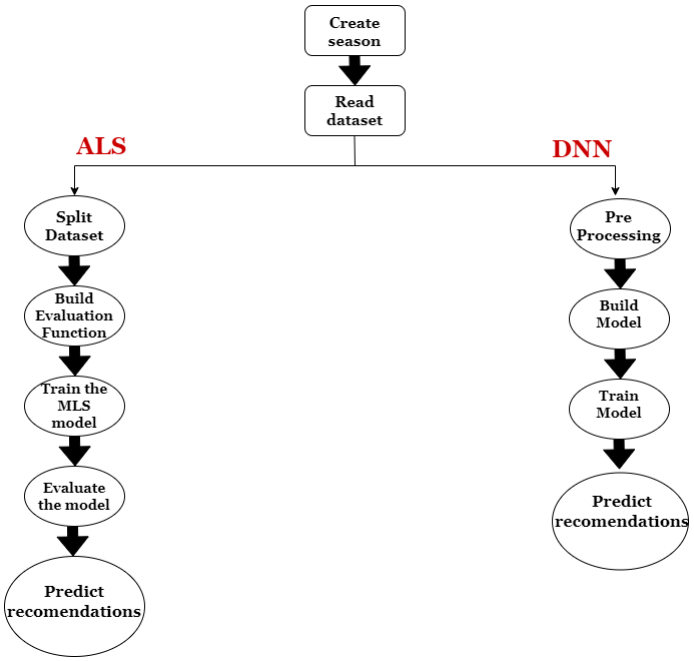

Fig. 1. Methodology

A. Collaborative Filtering using ALS Algorithm

- Create a Session: Initiate a Spark session to take advantage of its distributed computing capabilities to handle large datasets efficiently.
- Read the Dataset Files: The relevant data set files are loaded, including user-artist interaction data, artist information, and any necessary mappings to correct missing artist IDs.
- Split the Dataset: Divide the dataset into training and testing sets to ensure proper evaluation of the model's performance. This typically involves a random split or a time-based split.
- Build Evaluation Function: Develop an evaluation function to assess the model's accuracy using metrics such as Root Mean Squared Error (RMSE) or Mean Absolute Error (MAE).

- Train the ALS Model: Configure and train the Alternating Least Squares (ALS) model on the training dataset. Set appropriate parameters such as rank, max iterations, and regularization.
- Evaluate the Model: Use the evaluation function to measure the model's performance on the testing set, ensuring that it generalizes well to unseen data.
- Predict Recommendations: Generate music recommendations for users based on the trained ALS model, identifying top artists or tracks that align with user preferences.

B. Deep Neural Network (DNN) Model

- Create a Session: Set up a session using appropriate deep learning frameworks like Keras or TensorFlow to facilitate model training and evaluation.
- Read the Dataset Files: Load the dataset files, ensuring to include user-artist interaction data and any necessary artist information.
- Preprocessing: Prepare the data for input into the DNN model. This involves normalizing play counts, encoding categorical variables, and splitting the dataset into training and testing sets.
- Build Model: Design the architecture of the DNN model, which may include embedding layers for user and artist representations, followed by fully connected layers to predict user-artist interactions.
- Train Model: Train the DNN model on the training dataset, utilizing techniques such as early stopping, dropout, or regularization to enhance model performance and prevent overfitting.
- Predict Recommendations: Utilize the trained DNN model to predict user-artist interactions, generating personalized music recommendations based on user preferences and past behaviors.

VI. Result Analysis

We basically applied two models in our system, ALS(lternating Least Squares) and DNN(Deep Neural Network). The overall result of ALS model is given below:

A. User-Artist Interaction

The dataset contains user IDs, artist IDs, and play counts. Analyzing the top interactions:

1) User with Maximum Plays: User 1024631 has 6188 plays.
2) User with Minimum Plays: User 2064012 has 58 plays.
3) Artist with Maximum Plays: Artist 1034635 has 43 plays.
4) Artist with Minimum Plays: Artist 1009140 has 1 play.This is the least here.

B. Evaluation Results (on validation data)

- Rank 2: Score = 0.0864
- Rank 10: Score = 0.0960 (Best)
- Rank 20: Score = 0.0851

The best model with rank 10 achieved a test score of 0.0624. A higher rank allows the model to capture more complex patterns in the user-item interactions but can lead to overfitting if set too high, especially with limited data. Conversely, a lower rank may underfit, failing to capture enough detail about the preferences.

C. Recommendations for Users

Based on the best ALS model, 10 artist recommendations were made for user 1059637. The top 3 recommendations are:

- Something Corporate
- My Chemical Romance
- Further Seems Forever

Now, the overall summed up result of DNN model is given below:

*1) **Loss with DNN**:* A DNN model was trained with 20 epochs. The loss values increased significantly as training progressed, indicating issues with the model performance:

- Epoch 1: Loss = 7851947, Val Loss = 1286834
- Epoch 20: Loss = 3011572, Val Loss = 269451136 (indicating overfitting and poor generalization)

*2) **Recommendations with DNN**:* After training the DNN model, artist recommendations were made for user 2069337. The top artist recommendations include:

- Southworth, John
- Darlin
- Sainte Chapelle

The ALS model with rank 10 performed the best, achieving a validation score of 0.0960 and a test score of 0.0624. The DNN model, on the other hand, showed poor performance with high loss values across epochs, indicating it was not well-suited for the dataset in comparison to ALS.

VII. Conclusion

We have implemented and evaluated a music recommendation system using two distinct approaches: the Alternating Least Squares (ALS) algorithm and a Deep Neural Network (DNN) model. Both methods were applied to a large-scale dataset containing implicit feedback data, focusing on user-artist interactions. The ALS model, implemented within the Apache Spark framework, demonstrated strong performance in handling sparse datasets and efficiently generated personalized music recommendations. In contrast, the DNN model, despite being a powerful tool for capturing complex patterns, struggled with overfitting and failed to outperform ALS on this particular dataset.DNN models hold promise for more complex data representations, their application in this context requires further tuning and optimization.Our findings suggest that ALS, with its scalability and simplicity, remains a robust choice for music recommendation systems, especially when computational efficiency and real-time performance are critical.

Finally, future work may focus on improving the DNN model's architecture and exploring hybrid recommendation systems that combine collaborative filtering with content-based features to enhance recommendation accuracy.

REFERENCES

[1] R. Lu, K. Wu, Z. Duan and C. Zhang, “Deep ranking: triplet MatchNet for music learning”, 2017.

[2] K. Gurjar and Y. Moon, “A comparative analysis of music similarity measures in MIR systems”, 2018.

[3] M. Defferrard, K. Benzi, P. Vandergheynst and X. Bresson, “FMA: a dataset for music analysis”, 2017.

[4] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman and P. Lamere, “The million song dataset”, 2011.