



Article

Enhancing Sequence Movie Recommendation System Using Deep Learning and KMeans

Sophort Siet ¹, Sony Peng ¹ , Sadriddinov Ilkhomjon ¹, Misun Kang ^{1,*} and Doo-Soon Park ^{1,2,*} 

¹ Department of Software Convergence, Soonchunhyang University, Asan-si 31538, Republic of Korea; siet.sophort60@gmail.com (S.S.); peng.sony61@gmail.com (S.P.); ilkhomjon.sadriddinov@gmail.com (S.I.)

² Department of Computer Software Engineering, Soonchunhyang University, Asan-si 31538, Republic of Korea

* Correspondence: ms.kang@sch.ac.kr (M.K.); parkds@sch.ac.kr (D.-S.P.)

Abstract: A flood of information has occurred, making it challenging for people to find and filter their favorite items. Recommendation systems (RSs) have emerged as a solution to this problem; however, traditional Appenrecommendation systems, including collaborative filtering, and content-based filtering, face significant challenges such as data scalability, data scarcity, and the cold-start problem, all of which require advanced solutions. Therefore, we propose a ranking and enhancing sequence movie recommendation system that utilizes the combination model of deep learning to resolve the existing issues. To mitigate these challenges, we design an RSs model that utilizes user information (age, gender, occupation) to analyze new users and match them with others who have similar preferences. Initially, we construct sequences of user behavior to effectively predict the potential next target movie of users. We then incorporate user information and movie sequence embeddings as input features to reduce the dimensionality, before feeding them into a transformer architecture and multilayer perceptron (MLP). Our model integrates a transformer layer with positional encoding for user behavior sequences and multi-head attention mechanisms to enhance prediction accuracy. Furthermore, the system applies KMeans clustering to movie genre embeddings, grouping similar movies and integrating this clustering information with predicted ratings to ensure diversity in the personalized recommendations for target users. Evaluating our model on two MovieLens datasets (100 K and 1 M) demonstrated significant improvements, achieving RMSE, MAE, precision, recall, and F1 scores of 1.0756, 0.8741, 0.5516, 0.3260, and 0.4098 for the 100 K dataset, and 0.9927, 0.8007, 0.5838, 0.4723, and 0.5222 for the 1 M dataset, respectively. This approach not only effectively mitigates cold-start and scalability issues but also surpasses baseline techniques in Top-N item recommendations, highlighting its efficacy in the contemporary environment of abundant data.

Keywords: recommender systems; collaborative filtering; enhancing sequence movie recommendation system; deep learning; transformer architecture; KMeans



Citation: Siet, S.; Peng, S.; Ilkhomjon, S.; Kang, M.; Park, D.-S. Enhancing Sequence Movie Recommendation System Using Deep Learning and KMeans. *Appl. Sci.* **2024**, *14*, 2505. <https://doi.org/10.3390/app14062505>

Academic Editor: Chilukuri K. Mohan

Received: 31 January 2024

Revised: 10 March 2024

Accepted: 12 March 2024

Published: 15 March 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Internet technology and artificial intelligence have attracted researchers, industries, and markets to the Internet of Things (IoT) in recent decades. IoT connects heterogeneous physical objects to collect data from smart gadget sensors [1]. Smartphones, smartwatches, laptops, sensors, and their internet connectivity generate big data or structured and unstructured data. Big data dominates research in business, healthcare, monitoring systems, transportation, smart homes, and more [2–4]. Patient monitoring systems use IoT devices to capture real-time data for personalized therapy and early disease detection. Businesses use big data to improve their decision-making, streamline operations, and customize consumer experiences, enhancing performance and competitiveness. IoT gadgets in smart homes use data for convenience, security, and energy management, demonstrating how big data can improve daily life.

However, the increasing volume of big data introduces the challenge of information overload [4]. The vast expanse of data complicates searches for specific information, making it increasingly difficult to provide the optimal relevant items. Previously, people would manually filter through traditional data, row by row, to find what they needed—a process that was both time-consuming and inefficient.

RSs have proved to be a valuable solution, assisting users in solving time-consuming problems [5]. They are utilized for decision-making by generating customized lists of products, resources, and information tailored to users' interests and past experiences. Additionally, RSs use various technologies to filter out satisfactory results, thereby reducing the searching time and quickly providing users with the information they need. Recent advancements in RSs have seen their application broaden significantly, touching on a variety of sectors, including entertainment, e-commerce, scientific research, news, healthcare, tourism, education, social media, and more [6]. These advancements highlight the increasing significance and practicality of RSs in assisting people in going through the extensive range of choices that are accessible in these fields.

For instance, in healthcare, RSs are transforming patient care by offering personalized recommendations for food, drugs, physical activities, healthcare professionals, and hospital locations [7]. This level of personalization in healthcare recommendations is instrumental in improving patient outcomes and satisfaction by catering to users' health needs and preferences [8]. In e-commerce and entertainment, giants like Amazon, Netflix, YouTube, Facebook, and Google have pioneered the development of sophisticated recommendation models to serve their diverse user bases [9–12]. These models are adept at navigating the vast amounts of data generated by users to offer personalized suggestions. By analyzing past search histories, user demographics, and individual preferences (such as likes, dislikes, and ratings), these platforms can curate content, products, or services that resonate with each user. This personalized approach not only enhances user engagement but also boosts the platforms' ability to cross-sell and upsell, thereby increasing their value proposition to users.

YouTube and Netflix, for instance, utilize RSs algorithms to offer personalized video suggestions and content recommendations. These recommendations are mainly based on social links, user behavior patterns, and content analyses. The systems take advantage of the similarities in friend connections and listening histories to recommend content that aligns with users' interests and past interactions. This method ensures that users continually discover new and engaging content that matches their preferences, keeping them hooked to the platform. Similarly, social networking and content-sharing companies like LinkedIn, Facebook, and Instagram employ RS algorithms to boost user engagement. By evaluating social connections, user activity, and content interactions, these systems can suggest new connections, tailor material feeds, and offer content/items that are likely to interest the user. This enhances the user experience by making the platforms more relevant and interesting, and it also promotes deeper user participation and interaction.

Traditional recommendation systems use collaborative filtering, content-based filtering, and hybrid approaches [13]. Content-based filtering creates a recommendation model from item content. It proposes goods based on the user's profile and past decisions. This content-based strategy adapts quickly to new profiles. Its constraints include the need for a well-organized user profile to make a suggestion [14]. Collaborative filtering, another old method, relies on user preferences to make suggestions. The model assesses user similarity and groups users as neighbors [15]. Items are recommended to people with similar tastes. Finally, the hybrid approach uses collaborative, content-based filtering, and other strategies to improve its suggestion quality [16]. These algorithms usually work well, although data sparsity, scalability, cold starts, and other issues arise [17]. Collaborative filtering algorithms, which use an object's average rating to compare user ratings and make new suggestions, are the most popular. For analyzing big item sets, data sparsity is crucial, but there are various drawbacks. Another constraint is the closest neighbor prediction difficulty. The third barrier is scalability, which becomes a difficulty as the numbers of users

and items increase. A fourth constraint is the cold start problem, caused by a lack of user relationships and information [18].

Many academics have used deep learning, a cutting-edge method for complex learning and predicting tasks, to overcome these issues. We designed a deep learning-based movie recommendation system to alter CF in this study. CF has two learning models, influenced by model-based and memory-based techniques [19]. This study uses a succession of movies and user data to improve recommendation system models. It addresses the issues discussed earlier. Our goal is to provide users with a personalized experience and enhance the accuracy of the system. To make a good RSs, we use the advanced deep learning model's strong point—the transformer architecture, which is good at working with sequential data—to pull out temporal dependencies by looking at a user's past movie ratings and adding demographic information to look for correlations for people who are new to the system. This method provides users with personalized movie ideas and accurate recommendation tailor to their preferences. It achieves this by employing a mix of data mining and advanced techniques to make a list of movies similar to those they enjoy. The KMeans clustering technique helps to find hidden trends in movie features that are not always obvious. It also improves the user experience, makes recommendations more diverse, and increases efficiency. The main points of our manuscript are as follows:

- We interpret the recommendation model using the sequential temporal of user interactions (movie ratings) to deliver a dynamic and contextual comprehension of user preferences based on the Movielens dataset. Our approach adopts a Transformer architecture, integrating multi-head attention with user demographic and movie embeddings, which allows the model to weigh various aspects of a user's movie-watching history differently when making predictions for the next target movie. Sequential recommendations are referred to as advanced model-based CF, as this is more effective in tackling the issues with existing traditional techniques.
- Then, the model contains a KMeans clustering post process to group movies into clusters depending on their embeddings, which aids in diversifying recommendations. It also integrates movie genres as extra attributes, boosting the model's capacity to represent varied movie qualities. The algorithm is designed to anticipate Top-N recommendations for users, employing clustering to ensure a mix of genres and preferences. The evaluation measures expand beyond typical loss functions to include precision, recall, and F1 scores, offering a comprehensive view of the model's performance.

The structures of the paper are arranged as follows: Section 2 illustrates related works regarding movie recommendations. Section 3 pertains to our proposed framework; we present the system architecture and a theoretical transformer model. Section 4 shows the experimental results and data analysis and compares other baseline methods. Finally, Section 5 is described in the conclusion of the work.

2. Related Work

This section introduces the traditional-based recommendation systems and recent advance-based recommendation systems, which are the major core of our model. Currently, RSs have been widely investigated in various disciplines of artificial intelligence, including different perspectives of machine learning techniques, to solve specific difficulties.

2.1. Traditional-Based Recommendation Systems

Several studies have investigated the evolution of recommendation systems, which traditionally comprise two parts: content-based filtering and collaborative filtering [20]. Content-based filtering (CBF) has focused on recommending new items that are similar to those a user liked in the past, relying on item features and user profiles. Pazzani, Michael J and Daniel Billsus, [21] emphasized the importance of item features and user profile creation for effective recommendations. The strength of CBF lies in its ability to recommend items similar to those a user preferred in the past, utilizing detailed descriptions and metadata of items. However, it often suffers from a lack of diversity in its recommendations

and struggles with the new item (cold start) problem, where new items have not yet been rated sufficiently to be recommended.

On the other hand, collaborative filtering (CF), the more prevalent approach, relies on both implicit and explicit user–item interactions [22]. CF utilized the user–item interaction matrix, introducing foundational algorithms like matrix factorization. Despite their effectiveness, challenges such as cold start, scalability, and sparsity issues persist. Sarwar, Badrul et al. [23] addressed scalability with “item-based collaborative filtering”, proposing algorithms that improved the recommendation efficiency and performance. Takács, Gábor et al. [24] furthered CF with matrix factorization techniques, enhancing its ability to deal with large datasets and improving recommendation accuracy. Barathy, R, and P. Chitra [25] proposed matrix factorization (MF) and Singular Value Decomposition (SVD) and decomposed the user–item interaction matrix into latent factors, capturing underlying patterns in user preferences and item characteristics to improve recommendation accuracy. Rendle, Steffen et al. [26] utilized Bayesian Personalized Ranking (BPR) to optimize personalized ranking item predictions by extracting latent features representing user preferences and factorizing the user–item interaction matrix into a lower dimension based on explicit or implicit feedback. To overcome the limitations inherent in CBF and CF, researchers explored hybrid models. Sun, Chang et al. [27] presented hybrid news recommendation algorithms that combine content-based methods (using TF-IDF and K-means clustering) with SVD-based collaborative filtering to improve overall performance. Further, Patoulia, Agori Argyro et al. [28] conducted a comparative study of collaborative filtering in product recommendation, which demonstrated that the LightFM library outperforms the surprise library in handling foodservice transactional data, emphasizing LightFM’s ability to deal with sparse data and establish personalized recommendations. This paper emphasizes the importance of hyperparameter tuning for optimal algorithm performance and represents a significant development in collaborative filtering techniques.

Initially, recommendation systems relied heavily on content-based and collaborative filtering techniques. Although they are effective, they are limited in their ability to handle complex patterns and large-scale data, particularly when the rating matrix is sparse.

2.2. Advanced-Based Recommendation Systems

The advent of deep learning has led to a paradigm shift in recommendation systems. These models excel at capturing complex user–item interactions and integrating diverse data types, such as textual and visual information, into the recommendation process.

Zhang, Shuai et al. [29] surveyed and offered new perspectives on deep-learning-based recommender systems. Neural collaborative filtering (NCF) demonstrated how deep neural networks can learn user–item interaction patterns more efficiently than traditional matrix factorization methods. This approach leverages a multi-layer perception to learn the non-linear interactions between user and item, significantly enhancing the recommendation accuracy. While neural-network-based models and the application of Restricted Boltzmann Machines have offered a nuanced understanding of user–item interactions, deep learning also has enhanced content analysis capabilities, with convolutional neural networks (CNNs) being applied for feature extraction from non-textual content, such as images and videos, significantly improving content-based recommendations. Auto Rec and CDAE utilize an autoencoder framework to extract latent vectors for user–item dynamics, predicting user ratings and generating high-quality movie recommendations. In the realm of recommendation systems, Xinchang, Khamphaphone, Phonexay Vilakone, and Doo-Soon Park [30] proposed using social network analysis and collaborative filtering to form the recommendation system, and the authors’ method solved the cold-start problem in the traditional approach. Moreover, they applied the community detection method to cluster user similarities and recommend a movie list to a target user based on similar preferences. Aiming to develop an improved collaborative movie recommendation system which combines K-means clustering with neural networks, Jing and Hui [31] introduced the hybrid approach, which applies K-means to cluster movies into groups and then trains

a neural network model to learn users' preferences based on the clusters to provide more accurate and personalized recommendations, especially for new users with sparse data. Wang, Kai et al. [32] presented a novel model that combines k-means clustering with a deep neural network to generate personalized recommendations for users in e-commerce applications, which can effectively solve problems of sparse data and information overload. Chen, Jianguo et al. [33] applied K-means clustering in healthcare recommendation systems to group patients with similar medical histories. Clustering patients based on demographic and medical data facilitated personalized treatment recommendations, improving healthcare outcomes.

The traditional domain of collaborative filtering has undergone substantial evolution with the interpolation of deep learning models, particularly matrix factorization techniques combined with neural networks that can capture temporal dynamics in user-item interactions to predict the next items. Early work on sequential recommendation systems was generally based on Markov Chains (MC). For instance, for Rendle, Steffen, Christoph Freudenthaler, and Lars Schmidt-Thieme [34], FPMC was the most commonly used technique when combining power matrix factorization and MC to make next-basket recommendations, encoding users' short-term interests, which demonstrated a good performance using sparse datasets. Tang, Jiayi, and Ke Wang [35] also proposed a convolutional sequence embedding (caser), using the Top-N model of the embedded sequential pattern as the local features of the image. Both horizontal and vertical convolutional filters from the embedding matrix of high-order Markov chains were used as the image in order to capture high-level sequential patterns. Hidasi, Balázs et al. [36] developed a novel CF model that incorporates recurrent neural networks (RNN) to track changes in user preferences over time (as in the case of Netflix). This approach addresses the traditional limitations of CF, such as cold start and data sparsity problems, by providing a dynamic representation of user interests, leading to more accurate and timely recommendations. The recurrent neural networks (RNNs) can capture the sequence dependence among user-item interactions in a behavior sequence to predict the possible interactions of the next item. Basically, when dealing with long-term sequences using RNN, backpropagation will face a few acute problems, such as gradient disappearance. In addition, it only handles point-wise dependency. Choe, Byeongjin, Taegwan Kang, and Kyomin Jung [37] used gated recurrent units (GRU), and Duan, Jiasheng et al. [38] used long short-term memory (LSTM), popular models that encode items into a dense vector to reflect users' interest in various recurrent architectures to improve their version of the model and further improve the Top-N recommendation performance. Those complex models require large amounts of data to capture long-term patterns, i.e., overfitting easily occurs in high-sparsity settings. They cannot be run in parallel, which is time-consuming.

Even if the current trend of using other state-of-the-art recommendation technology could provide satisfactory results, studies of recommendation models are still a hot research topic, improving their performance in an unstoppable way. The integration of deep learning into RS using a transformer model has been a significant area of innovation. The transformer architecture, originally used for natural language processing, has led to a significant leap forward in creating adaptive systems that can capture long-range dependencies in sequences of user-item interactions and improving recommendation performance and interpretability. Kang, Wang-Cheng, and Julian McAuley [39] introduced self-attentive sequential recommendations (SASRec), while Yu, Saisai et al. [40] proposed personalized movie recommendation algorithms that fuse visual and textual features using multi-head attention with neural networks that can address data sparsity and cold start problems. Chen, Qiwei et al. [41] presented a Behavior Sequence Transformer for E-commerce Recommendation in Alibaba, a model designed to enhance recommendation systems within the e-commerce domain. This is specifically built upon the transformer architecture, a popular deep-learning architecture that is effective in sequence-to-sequence tasks. Wang, Dongjing et al. [42], revolutionized a transformer-based RS that adapts to both sequential and contextual information in user interactions. By employing self-attention mechanisms,

the system can dynamically weigh the importance of different items in a user’s history, leading to highly personalized and context-aware recommendations. In addition, existing works focus on understanding users’ playlists or listening sequences, which have inspired many sequential recommendation models. Chen, Quanzhen et al. [43] introduced a hybrid model incorporating GNNs to leverage both user–item interactions and content features. The approach allows for a more comprehensive understanding of user preferences and item characteristics, leading to improved recommendation diversity and accuracy. Additionally, hybrid systems employing the transformer architecture have been developed to better capture sequential user behavior and item attributes to make dynamic recommendations.

3. Methodology of Movie Recommendation Systems

In this section, we provide an overview of the recommendation architecture and the important requirements for predictive tasks. Additionally, we introduce the process, from data processing to the point where the model can generate a list of movie recommendations for the target users.

3.1. Data Processing and Sequence Creation

Firstly, we explain the preprocessing steps required in our implementation. In our work, we utilize users’ ratings from the MovieLens dataset to construct a recommendation system. Initially, we merge all the rating information for an individual user into the required input format for our transformer model [44]. We then construct a vocabulary for movie IDs and user IDs and create sequences of user interactions in chronological order. All user interactions are first sorted by their interaction timestamp and then divided into subsequences for the training model. To facilitate subsequent calculations, we convert the list of movie IDs and the movie ratings to a fixed length and retain a position set of subsequence information. These sequences are further divided into subsequences to create a structured input format that encapsulates a user’s interaction history up to a fixed length, thereby preparing the data for the transformer model. After processing, the input is generated for each individual user in a sequential manner, including user ID, movie ID, sequence rating, the target (label) that the model attempts to predict—which would be the target movie ID—and the rating of the last item in the sequences, as shown in Table 1.

Table 1. Sequence interaction of users and movies.

User ID	Movie ID in Sequence				Sequence Rating				Target Movie ID	Target Rating
User_1	2926	2915	2344	2968	2.0	3.0	3.0	4.0	2968	4.0
User_2	1307	2791	73	3269	3.0	5.0	1.0	5.0	3269	5.0
User_3	420	3688	838	2174	2.0	1.0	4.0	3.0	2174	3.0
User_4	1704	541	1219	3521	4.0	2.0	3.0	2.0	3521	2.0
User_5	44	1608	2064	3835	5.0	1.0	1.0	4.0	2064	4.0

To enhance the input data, we integrate additional attributes, such as movie genres in the sequences and the demographic information (age, gender, occupation) of the users, into our final dataset. The final dataset was then subjected to a series of cleaning procedures, including removing duplicates, handling missing values, and addressing outliers, before being finalized for analysis. Furthermore, some data had to be converted from categorical to numerical formats to be compatible with the transformer model and k-means clustering in the embedding layer. Variables like gender and occupation were encoded from categorical to numerical formats through a process of factorization. The age feature underwent normalization using the MinMaxScaler, which scales the data to fall within a specified range. This enhances model performance by ensuring numerical features are on a similar scale. Movie genres were expanded into binary features, each indicating the presence of a specific genre, thus enriching the dataset with explicit genre information, as shown in Figure 1 on step 1 and in Figure 2.

3.2. Model Architecture

This section introduces the model architecture of the entire recommendation system as detailed in Figure 1. There are two main processes in our system. The first process involves applying deep learning and key transformer features to predict the next movie based on the user's sequence and information. The transformer model was implemented with a multi-head attention layer and positional embedding, which are adept at understanding the complex viewing patterns of users. The multi-head attention mechanism allowed the model to focus on different parts of the user's movie history to provide a comprehensive understanding of their preferences. The positional embeddings provide the model with an understanding of the order in which movies were watched, which was crucial for predicting future interests. Subsequently, we fed an embedding layer of the user behavior sequence, which was implemented using a transformer architecture. This was then combined with user demographics following multi-layer perceptron (MLP) to predict the rating. In the second step, after training the transformer model, we integrated the output predicted rating into K-mean clustering to generate a Top-N recommendation system for the target user. K-means clustering is a popular method in cluster analysis, which is designed to partition a set of objects into K clusters in such a way that the sum of the squared distances between the objects and their assigned cluster mean is minimized. Our objective was not only to develop recommendation algorithms that personalized the results based on historical preferences but also to dynamically adjust diversity in the recommendations list according to the user's interest. To achieve this, we utilized the K-means approach to personalized diversity, which automatically segments movies into distinct groups based on user preferences (target next movie) according to certain predefined categories (movie embeddings associated with the movie's genre). When creating the recommended list, we investigated these clusters to ensure a diverse suggestion of Top-N movies when a user interacts with our system. Similarly, for new movies with few or no ratings, clustering helps us identify and recommend unseen movies to target users based on their similarity score with other movies. The K-means clusters aid in diversifying the selected top N by ensuring that movies from different clusters are included in the recommendations. Therefore, our approach helps in providing a variety of recommendations to the user, ensuring that these recommendations are not limited to a specific genre or type of movie [45,46].

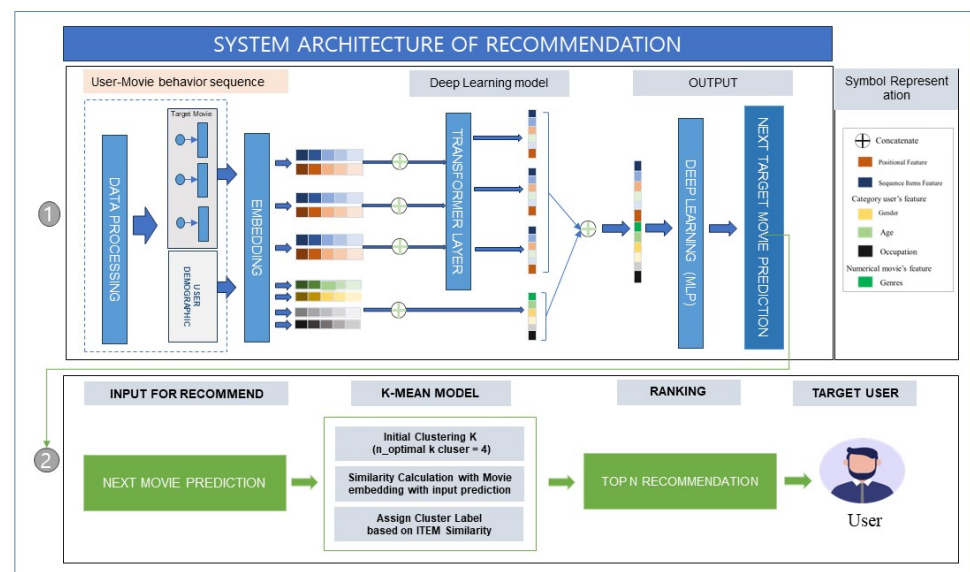


Figure 1. Movie recommendation system architecture.

3.2.1. Process ONE—Predicting User Ratings for Movies

This section introduces the first process used for model rating prediction. The primary purpose of our work was to predict how a particular user might rate a given movie.

The process includes various necessary elements suitable for our predictive model. The model took in features such as user ID, movie ID, user's demographic data, and historical movie ratings (sequence experiences of the target user), and output predicted ratings.

(a) User-Movie Embedding Layer

We divided learning data into two parts: user demographic and movie sequence embedding. User demographic data consisted of data such as user ID, gender, age, and occupation. Each category of input features first passed through the embedding layer to become a dense feature vector. An embedding matrix $u_j \in \mathbb{R}^{u \times d}$ was created to transform the integer index into a dense vector of fixed size, where u represented the vocabulary size (the total number of unique categorical elements), and d was the embedding dimension. These dense vectors could typically reduce the dimensionality of the input features, capturing the underlying relationships and characteristics of each categorical feature to feed them directly into the multi-layer perceptron (MLP).

Movie features contained the movie IDs in sequence, and movie genre. Firstly, the movie IDs, in sequence, were transformed into dense feature vectors through the embedding layer. The input movies' genre features were transformed into feature vectors through multi-hot encoding. We also obtained embeddings for each movie ID in the behavior sequence, including the target movie, which was essential for the transformer model to understand the sequence's temporal dynamics. Furthermore, we incorporated a learnable position encoding matrix to enhance the input order sequence to address the lack of inherent positional information in the transformer architecture. For each movie, we then concatenated the movie ID sequence with the positional encoding matrix and created an embedding matrix $\hat{E}_i \in \mathbb{R}^{n \times d}$, which represents the embedding vector of the i -th movie in each behavior sequence of user u after adding the position vector. In our process, we fed our sequence of movie embeddings into a single transformer layer, before concatenating the output with the user features.

(b) Model Training

The transformer model was a type of deep learning model that was primarily used in the processing of sequential data such as natural language. It could capture the user dynamic nature influences on users' recent activities. In our study, we analyzed the structure of the transformer layer, which enhanced the model's ability to capture the long-range dependencies and relationships between movies in a sequence, allowing it to make more accurate and personalized recommendations. The transformer layer comprised a multi-head attention layer and a Position-Wise Feed-Forward Neural Network (FFN).

The multi-head attention mechanism allowed for the model to focus on different sections of the input sequence in different ways (e.g., longer-term dependencies versus shorter-term dependencies) when making prediction ratings for each target movie. The multi-head attention module can be used in algorithms. The module is expected to be process information multiple times in parallel via an attention mechanism. This mechanism allows the model to jointly attend to information from different representation subspaces at different positions, providing a more complex understanding of the input sequence. After that, the output of the distinct attention was connected and linearly transformed into the aspect that was predicted. In our case, the multi-attention operation takes the embedding \hat{E}_i as input. The input is passed through three separate linear layers to produce the Q, K, and V matrices, as presented below:

$$A_i = \text{MultiHead}(\hat{E}) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W_0 \quad (1)$$

$$\text{head}_i = \text{Attention}(\hat{E}W^Q, \hat{E}W^K, \hat{E}W^V) \quad (2)$$

where W^Q , W^K , and $W^V \in \mathbb{R}^{d \times d}$ are the projection matrices that makes the model more flexible, \hat{E} is the embedding matrix of all movies, W_0 is a learnable weight matrix that can form the final representation of the movie at i , and h is the number of heads.

In our transformer layer, a point-wise feed-forward network (FFN) enabled the model to adapt, providing the flexibility needed to understand complex movie interactions within the sequence. These networks had hidden layers and non-linear activation functions, which allowed the model to learn complex patterns in the movie ID sequence. FFN consisted of two fully connected layers. At the same time, we utilized dropout layer normalization to optimize the model, while avoiding overfitting and speeding convergence with LeakyReLU in FFN. These were applied to each position, separately and identically, during training. The overall output of the transformer layer between the multi-head attention and stacking FFN layers, which represents the candidate movie that contains characteristics of the users' behavior sequence, can be defined as follows:

$$N_i = \text{LayerNormalization}((\hat{E}) + \text{Dropout}(A_i)) \quad (3)$$

$$F_i = \text{FFN}(N_i) \quad (4)$$

$$T_i = \text{LayerNormalization}(N_i + \text{Dropout}(\text{LeakyReLU}(F_i W_1 + b_1) W_2 + b_2)) \quad (5)$$

where $W_1, W_2 \in \mathbb{R}^{d \times d}$, b_1 , and $b_2 \in \mathbb{R}^d$ refer to the learnable weight's matrix and bias parameters.

Next, we concatenated the user demographic embedding layer u_j and the output of the transformer, with target item as T_i , for the representation operation of the two vectors and passed them together through a multilayer perceptron (MLP) network. The output layer computed the predicted rating of the next target movie using a dense layer. The dense layer allowed for a linear transformation of all flattened and concatenated features, as expressed below:

$$x_{i,j} = \text{Concat}(T_i, u_j) \quad (6)$$

$$\hat{y}_{i,j} = \sigma(W \cdot x_{i,j} + B) \quad (7)$$

where $\hat{y}_{i,j}$ is the predicted rating for user j on movie i -th, W is the weight matrix for the output layer, (\cdot) is the dot product between the weight matrix W and the input vector $x_{i,j}$, B is the bias vector for the output layer, and σ is the sigmoid activation function.

3.2.2. Process TWO: Leveraging Predictions for Recommendations

In this second process, we aimed to leverage the predictive tasks resulting from Process One's output to model a recommendation for a Top-N recommendation list. After training, we applied KMeans clustering to movie embeddings to group movies into clusters based on the learned features. This clustering could assist in the recommendation process by identifying movies that are similar to each other.

To recommend movies, we first predicted ratings for a user's movies and then selected the top-N movies based on these predicted ratings. Clustering information was used to refine recommendations, ensuring diversity and focusing on a specific genre or group of similar movies. The linkage of KMeans clustering to our model could be explained as follows:

After training the recommendation model, we extracted embeddings for each movie. These embeddings were the output of the movie embedding layer in our neural network model. This step involved KMeans clustering, with the following processes: (1) Choose the number of clusters, K , based on domain knowledge, heuristic methods (like the elbow method), or experimentation. (2) Initialize the KMeans algorithm with K clusters and fit this on movie embeddings. (3) Each movie should be assigned to the nearest cluster centroid. To generate Top-N recommendations, the system predicted ratings for movies that had not yet been rated by a user, leveraged clustering information to ensure diversity or focus on specific interests, and then selected the top-rated movies as recommendations.

4. Experimental Study

To verify the effectiveness of our proposed model, we applied two well-established versions, 100 K and 1 M [47], to the experiment, as presented in Table 2. MovieLens is a popular benchmark dataset for R.S. evaluation, which consists of many user attributes. The 100 K datasets consisted of interactions between 944 users and 1682 items (movies)

that were used for the experiment. Meanwhile, 1 M datasets were larger, with interactions between 6040 users and 3883 items. The dataset was structured into three distinct CSV files, which provided a brief description of important attributes such as user ID, movie ID, movie title, rating, timestamp, gender, age, occupation, zip code, title, and genres. Each dataset consists of various ratings from anonymous users, from 1 to 5. We gathered relevant data on user–item interactions and user demographics such as age, gender, movie genres, and occupation as input features to speed up our system, to obtain a better result in terms of recommendations.

Table 2. Statistics of the Movielens dataset.

Dataset	#User	#Item	#Rating	Sparsity
100 K	943	1682	100,000	93.69%
1 M	6040	3883	1,000,209	95.73%

After the data were preprocessed, our dataset was divided into training and testing datasets, respectively. The training dataset contained 80 percent of the Movielens dataset, and the remaining 20 percent belonged to testing data. In the model, ratings were considered target values in a sequence, which needed to be fit. Furthermore, we saved the training and testing datasets to CSV files for implementation in our training and evaluation model. The sample data in Figure 2 refer to the data obtained from the data processing of Movielens.

user_id	movie_id	rating	gender	age	occupation	genres
438424 user_5749	movie_30,movie_3265,movie_2692,movie_198	3.0,5.0,5.0,1.0	1	0.43636363636363634	16	[Drama, Action Crime, Action Crime]Romance, Ac...
375249 user_5046	movie_1278,movie_1276,movie_2396,movie_2384	4.0,5.0,4.0,2.0	1	0.43636363636363634	1	[Comedy Horror, Comedy Drama, Comedy Romance, ...
455396 user_5936	movie_69,movie_1136,movie_1036,movie_2791	5.0,5.0,4.0,5.0	1	0.3090909090909091	12	[Comedy, Comedy, Action Thriller, Comedy]
459062 user_5976	movie_2791,movie_1220,movie_2455,movie_1090	4.0,3.0,2.0,2.0	0	0.43636363636363634	6	[Comedy, Action Comedy Musical, Horror Sci-Fi,...
30810 user_1297	movie_2912,movie_159,movie_337,movie_1784	3.0,3.0,4.0,4.0	1	0.43636363636363634	3	[Action Crime Drama, Drama, Drama, Comedy Drama]
...
447756 user_5842	movie_2194,movie_1374,movie_2949,movie_2948	4.0,3.0,3.0,3.0	1	0.43636363636363634	12	[Action Crime Drama, Action Adventure Sci-Fi, ...
80033 user_1793	movie_3256,movie_349,movie_112,movie_2278	4.0,4.0,4.0,2.0	1	0.43636363636363634	12	[Action Thriller, Action Adventure Thriller, A...
384736 user_5150	movie_1210,movie_318,movie_1,movie_2779	4.0,5.0,4.0,3.0	1	0.43636363636363634	10	[Action Adventure Romance Sci-Fi War, Drama, A...
254226 user_3735	movie_3765,movie_3182,movie_2726,movie_858	2.0,5.0,5.0,5.0	1	0.43636363636363634	9	[Drama Romance, Documentary, Crime Film-Noir, ...
1170 user_1010	movie_3248,movie_1410,movie_1468,movie_1772	1.0,1.0,1.0,1.0	1	0.43636363636363634	9	[Comedy, Comedy Drama, Comedy Romance, Action ...

279228 rows × 7 columns

Figure 2. Illustrated input features of Movielens datasets.

4.1. Implementation Detail

4.1.1. Environment Set-up of Transformer Model

In our architecture model, we performed parameter optimization with Adam, a learning rate of 0.001, and a default batch size of 128. We set hyper-parameters of the transformer layer as $L = 2$, attention head as $h = 8$, sequence length as $N = 4$, inner size as 256 (e.g., FNN layers), and hidden size of MLP = [128, 128]. After each layer, we used a dropout layer with a dropout chance of 0.2 to reduce overfitting. We set a feed-forward layer followed by a Softmax layer to predict the probability of a movie. The training process was terminated after a maximum of 100 epochs. These procedures were implemented using TensorFlow 2.13.0 and Python 3.8.10. The hardware and software environment used for implementing the task included Windows 11 Enterprise 64-bit, a 12th Gen Intel(R) Core(TM) i7-12700 CPU @ 4.02 GHz, and 32.0 GB RAM.

4.1.2. Environment Set up of K-Means Algorithm

In this section, we determined the number of clusters based on the k-value setting using the Elbow method. This method involves iteratively fitting K-Means models with varying values of k to evaluate their performance. We applied the MovieLens dataset to generate the top 10 recommendations by predicting the next target movie, specifically movie_9 in the 100 K dataset and movie_2926 in the 1 M dataset. These movies were

identified among the distinct clusters of other movies after model training, and classified into separate groups using the k-means algorithm, along with genre information. To identify the optimal number of clusters, we applied the Elbow method by running k-means clustering on the dataset for a range of k-values (e.g., from 1 to 50), and then computed the sum of squared distances from each point to its assigned center (inertia) for each k. This process requires running the algorithm multiple times in a loop, with an increasing number of cluster choices, and then calculating a clustering score as a function of the number of clusters. The optimal cluster values for different pairs of genres are 4, 5, 7, and 10. In our experiments, the elbow point was observed at $k = 4$, indicating this to be the most optimal number of clusters for our K-Means clustering to provide a suitable balance between capturing the underlying data structure and avoiding excessive model complexity.

4.2. Evaluation Metric

To check the performance of our sequential recommendation system, we adopted the leave-one-out evaluation for the next-item recommendation task. For each user, we used the last movie of the behavior sequence as the test set and utilized the remaining movie for the training set. We followed the common convention of pairing each ground truth in these test sets with 100 random negative items that the user had not interacted with, giving them all the same context as their corresponding positive test item, and ranking the positive test item among them. Lastly, we truncated the rank list at the threshold value for each user. We measured the overall quality using the hit rate ($HitRate@k$) and normalized discounted cumulative gain ($NDCG@k$). These metrics can be calculated as follows:

$$HitRate@k = \frac{Number\ of\ HitRate@k}{k} \quad (8)$$

where $HitRate@k$ is the number of correct predictions in the Top-N recommendations, and k is the number of recommendations or the cutoff value for which we want to calculate the HitRate. In our experiments, all datasets had a rating range of 1–5, and we set the threshold at 3.5. If the threshold for an item was not met, we omitted the item.

The second metric that was adopted was the normalized discounted cumulative gain ($NDCG@k$), which measured the Top-N recommendation system list quality, expressed as follows:

$$DCG_k = \sum_{i=1}^k \frac{2^{rel_i} - 1}{\log_2 i + 1}, NDCG_k = \frac{DCG_K}{IDCG_K} \quad (9)$$

where rel_i was the actual rating score of an item at the rank position i . For our experiment, we gave results for HitRate and NDCG with $k = [5, 10]$. For all metrics, a high value corresponded to a better performance.

Moreover, we used root mean squared error (RMSE) to measure the proposed model performance. RMSE was the approach we used to measure the error rate a user gave to the system and the error predicted by the model. For example, the RMSE is shown as follows:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_{i,j} - \hat{y}_{i,j})^2} \quad (10)$$

where y_i is the actual value, \hat{y}_i refers to the predicted value of the observation, and n refers to the total number of data points.

To assess the diversity of the recommended movies, we utilized a metric based on the pairwise distances between the movie's embedding. These embeddings are high-dimensional vectors that represent movies in a latent space, capturing various aspects of the movie's genre. Then, the K-means algorithm was used to cluster these embeddings, grouping similar movies together based on their features to make our recommendations. We calculated the Euclidean distance between every pair of movie embeddings in the

recommended list. Finally, the diversity score was defined by taking the average of all these pairwise distances, expressed as follows:

$$ILD = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^n \|E_i - E_j\| \quad (11)$$

where $\|E_i - E_j\|$ represents the Euclidean distance between the embeddings of movies i and j , and n is the total number of recommended movies.

4.3. Benchmark Models

In this section, we selected benchmarks including BPR, NeuMF, Caser, GRU4Rec, and SASRec to evaluate the performance. We implemented them using TensorFlow and the RecBole library and considered latent dimensions {10, 20, 30, 40, 50} as the common hyperparameter in all models. The maximum number of training epochs is 100. In addition, we adopted the NeuMF model with the hidden size $h = [64, 32, 16]$. We tuned the hyperparameter using the validation set and terminated training if the validation performance did not improve at 40 epochs. The benchmark model is detailed as follows:

- Bayesian Personalized Ranking (BPR): The model is an optimization technique applied to matrix factorization, explicitly tailored to the handling of implicit feedback to enhance the factorization process by incorporating a pairwise ranking loss function.
- NeuMF: This is a collaborative filtering model that uses user-item interactions with an MLP instead of the inner product in matrix factorization when calculating the relationship between the user and item.
- Caser: This adopts convolutional neural networks (CNNs) in horizontal and vertical dimensions to effectively model high-order Markov Chains (M.C.s) for sequential recommendations.
- GRU4Rec: This is a unidirectional recurrent neural network (RNN)-based framework that is used to capture sequential dependencies and make predictions.
- SASRec: This is a state-of-the-art sequential recommendation model leveraging self-attention blocks to predict the next item for recommendation. Additionally, it employs the dot product computation between sequential latent features of the latest item and embeddings of the target item to establish the scoring mechanism.

4.4. Results Analysis

This subsection explains the result of our proposed system during the experiment using the performance metrics across two Movielens datasets, 100 K and 1 M. We evaluated the performance of our algorithm using common Top-N evaluation metrics, HitRat@K and NDCG@K, highlighting its effectiveness in providing accurate and relevant recommendations. Firstly, we compared our transformer model training with a non-sequence model such as BPR, NeuMF, and sequence models including GRU4Rec, Caser, and SASRec. The non-sequence model only suggests movies to the user, without explicitly considering the sequential order of their preferences or viewing history, and ignores temporal information. The sequence model considers the temporal order or sequence of a user's interactions with the recommender. The user's behavior sequence information can effectively characterize the user's changing preferences to a certain extent, improve the recommendation performance on sparse datasets, and learn the long-term dependencies.

In Table 3, our transformer model is shown to achieve the best HitRat@5, HitRat@10, NDCG@5, and NDCG@10 results on all datasets compared to the BPR, NeuMF, GRU4Rec, Caser, and SASRec models. The experimental results were HitRat@5, HitRat@10, NDCG@5, and NDCG@10 at (0.5676, 0.6633, 0.3714, 0.3783) on 100 K and (0.7034, 0.7309, 0.5869, 0.6238) on 1 M. This indicated that the sequential movie recommendation system of our proposed transformer architect could effectively extract useful information from movies to help us automatically select information with a long-term dependency and achieve a more accurate predicted rating of next target movies than other models. Furthermore, our

approach conveys the effectiveness of utilizing the multi-head attention mechanism to capture relevant information in different representation subspaces for different tasks and is time-consuming. The mechanism only pays attention to important movies. It drowns out irrelevant movies for recommendation.

Table 3. Overall performance comparison with state-of-art efficient movie model at baseline.

Dataset	Metric	BPR	NeuFM	GRU4Rec	Caser	SASRec	Our Model	Improvement
Movielens 100 K	HitRat@5	0.4507	0.5610	0.3701	0.3595	0.3637	0.5676	1.17%
	HitRat@10	0.5801	0.6925	0.5260	0.5111	0.5419	0.6633	−4.21%
	NDCG@5	0.1656	0.2325	0.2325	0.2371	0.2390	0.3714	55.39%
	NDCG@10	0.1588	0.2314	0.2826	0.2861	0.2965	0.3783	27.58%
MovieLeng 1 M	HitRat@5	0.4780	0.6752	0.5823	0.5889	0.5743	0.7034	4.17%
	HitRat@10	0.5927	0.5808	0.6866	0.7003	0.6861	0.7309	4.36%
	NDCG@5	0.1927	0.2323	0.4487	0.4463	0.4428	0.5869	30.80%
	NDCG@10	0.1770	0.2442	0.4826	0.4825	0.4791	0.6238	29.25%

Figures 3 and 4 show our analysis of the latent dimensionality hyperparameter, highlighting the significance of higher latent dimensions d in achieving optimal results, ranging from 5 to 50, across metrics with k values 5 and 10 of HitRate and NDCG to understand the impact on algorithm performance. In this illustration, we study models such as BPR, NeuFM, GRU4Rec, Caser, SasRec, and our model, with various d values, which increased its value in both datasets. Moreover, our proposed model's performance steadily improved the embedding dimensions increased from 20 to 50. This indicated the best performance, with the model learning more efficiently when a larger d was used to solve overfitting.

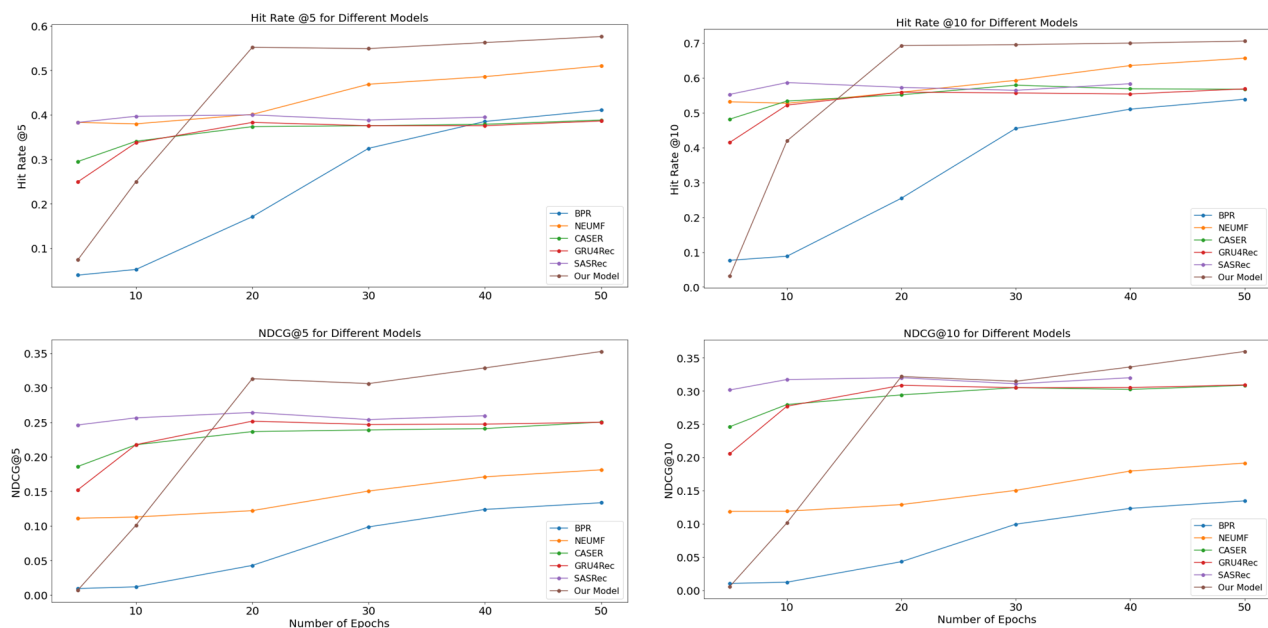


Figure 3. Performance evaluation in terms of NDCG and with a hit rate of 100 K from positions 5 to 50 (i.e., $k = [5, 10]$).

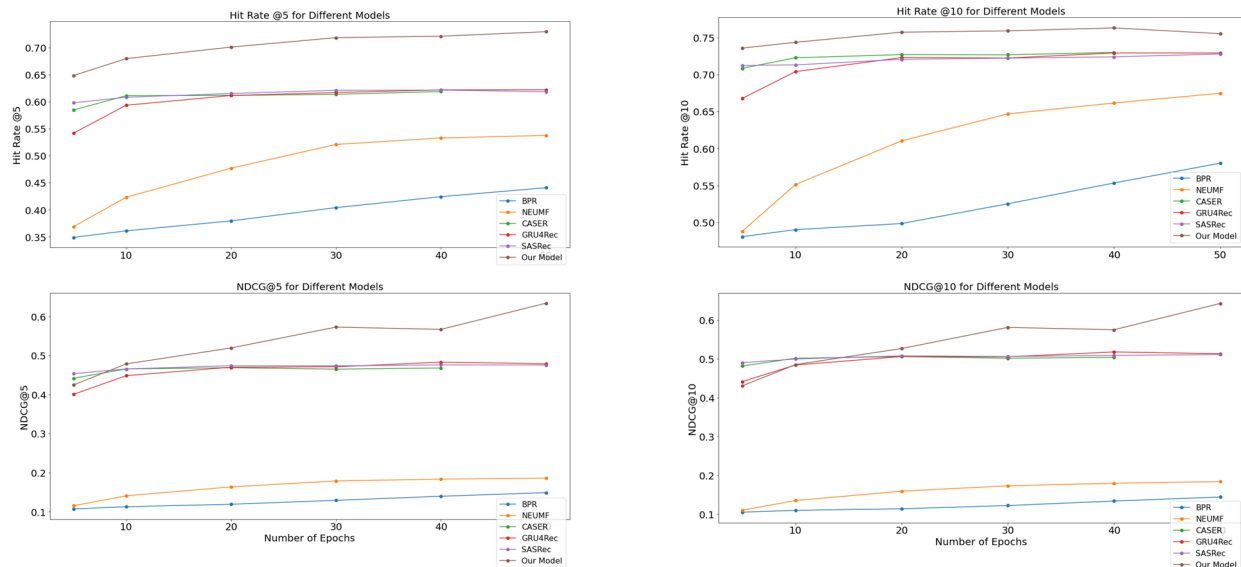


Figure 4. Performance evaluation in terms of NDCG and with a hit rate of 1 M from positions 5 to 50 (i.e., $k = [5, 10]$).

Overall, for sparse datasets, the larger the embedding dimension, the better the performance that can be achieved. This variation has a relatively stronger effect on 1 M than 100 K. A higher HitRate@10 value indicates a greater number of relevant results, making the model more effective in suggesting the next_target movie.

The performance evaluation of our movie recommendation system was carried out after utilizing the transformer model integrated K-means. Table 4 demonstrates effectiveness in enhancing recommendation quality and diversity. On the MovieLens 100 K dataset, our model achieved an RMSE of 1.0756, MAE of 0.8741, precision of 0.5516, recall of 0.3260, F1-score of 0.4098, and item coverage of 0.1165. The intra-list diversity, when applied with K-means, was 0.2447, which is an improvement over the 0.2173 intra-list diversity in the non-K-means scenario. In contrast, in the larger 1 M dataset, performance was enhanced, as evidenced by the improved metrics: an RMSE of 0.9927, MAE of 0.8007, precision of 0.5838, recall of 0.4723, F1-score of 0.5222, and item coverage of 0.3216. The intra-list diversity for this dataset improved to 0.3007 with K-means, compared to 0.2878 without K-means. These outcomes illustrated the model's scalability and its enhanced ability to make more accurate predictions with larger datasets. The higher precision, recall, and F1-score on the 1 M dataset indicated a stronger ability to identify and recommend relevant movies to users. The significant increase in item coverage for the 1 M dataset highlights that our method is effectiveness in recommending a broader spectrum of movies, thereby potentially enriching the user experience by exposing them to a wider variety of content. Although intra-list diversity slightly decreases with a larger dataset, the model maintains a balance when recommending popular movies, ensuring a diverse set of recommendations. This balance demonstrated the system's capacity to address critical challenges in recommendation systems, such as accuracy, diversity, and coverage, making it a valuable tool for personalized movie recommendations in an era of information overload.

Table 4. Overall performance of our model on training and testing datasets.

Dataset	RMSE	MAE	Precision	Recall	F1-Score	Item Coverage	Intra-List Diversity	
							K-Means	Non K-Means
100 K	1.0756	0.8741	0.5516	0.3260	0.4098	0.1165	0.2447	0.2173
1 M	0.9927	0.8007	0.5838	0.4723	0.5222	0.3216	0.3007	0.2878

Figure 5 illustrates the training error with the initial setting. The model showed a consistent decrease in both training and validation losses as the number of epochs increased. This suggests that the model is improving its predictions over time. The model trained on the larger dataset 1 M demonstrated a lower initial loss and achieved a smaller overall final loss compared to the model that was trained on the smaller dataset “100 K”. This suggests that the larger dataset might provide more information, allowing the model to learn a better representation of the underlying data distribution and become capable of making accurate predictions on unseen data.

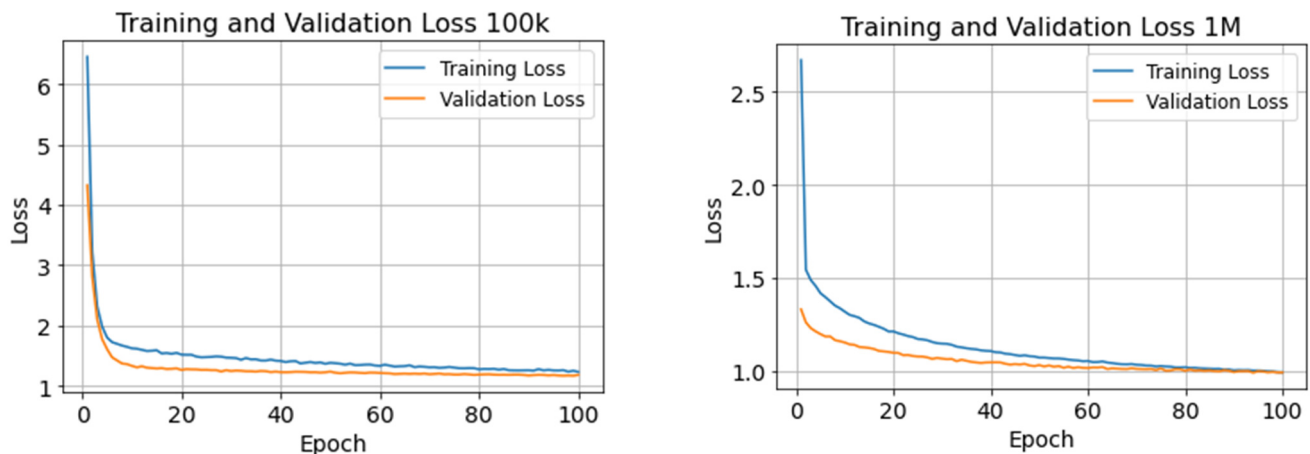


Figure 5. Training and validation loss performances of our proposed model on both datasets.

5. Conclusions

This work offers a unique sequence movie recommendation system that employs deep learning techniques to enhance user experiences by providing tailored movie suggestions based on the captured sequential relations. Our study mainly focuses on user demographics and user-item interactions in a sequence to help overcome the issues in standard algorithms' recommendations caused by cold-start and data sparsity or scalability. We used user demographic and movie sequence embedding as input and added this to the transformer architecture to handle sequential data to generate the next target movie.

The transformer model in our system exploits movie sequence embeddings, translating these dense vectors into a multi-head attention mechanism. This comprehensive technique helps increase the understanding of user behavior dynamics. It interacts with a multilayer perceptron (MLP) fed by user demographic data to discover intricate relationships, hence refining the personalized recommendations for the next target movie. Additionally, we applied KMeans clustering to detect underlying patterns in movie qualities that may not be visible from the anticipated ratings post-model training. This concept enables our system to widen the range of choices by suggesting movies from various clusters with comparable genre features. This technique considerably enhances the personalized Top-N movie recommendations, matching them with the user's essential preferences, including those that are not immediately apparent.

In our studies, we conducted detailed assessments utilizing the MovieLens datasets (100 K and 1 M), which include essential variables such as movie ID, user ID, genres, gender, age, and employment. The findings demonstrate that our recommendation system (RS) model outperforms state-of-the-art baseline methods in terms of RMSE, MAE, precision, recall, and F1 scores, recording values of 1.0756, 0.8741, 0.5516, 0.3260, 0.4098 for the 100 K dataset and 0.9927, 0.8007, 0.5838, 0.4723, and 0.5222 for the 1 M dataset. Additionally, we examined the model, utilizing hit rate and normalized discounted cumulative gain (NDCG) measures, such as Hit Rate@5, Hit Rate@10, NDCG@5, and NDCG@10, with scores of 0.734, 0.7309, 0.5869, and 0.6238, respectively. These values represent improvements of 4.17%, 4.36%, 30.80%, and 29.25% compared to the 1 M dataset. Similarly, for the MovieLens 100 K dataset, the model obtains Hit Rate@5, Hit Rate@10, NDCG@5, and NDCG@10

with scores of 0.5676, 0.6633, 0.6925, 0.3714, and 0.3783, showing gains of 1.17%, −4.21%, 55.39%, and 27.58% over the baseline model. These results illustrate the higher performance and accuracy of our method when compared to other techniques, such as BPR, NeuMF, GRU4Rec, Caser, and SaSRec, especially with extensive datasets.

In future studies, we seek to expand our model by including more large-scale datasets to further evaluate its efficacy across multiple domains. Our key goal is to develop the system's capacity to provide more accurate and relevant suggestions to the target audience. This expansion will allow us to examine the model's adaptability and effectiveness in multiple circumstances, ensuring that our recommendations remain topical and beneficial to users with different tastes and needs.

Author Contributions: Conceptualization, S.S., S.P., M.K. and D.-S.P.; methodology, S.S. and D.-S.P.; software, S.S. and S.P.; writing—original draft preparation, S.S. and S.P.; writing—review and editing, D.-S.P.; visualization, S.S. and S.I.; supervision, D.-S.P. and M.K.; funding acquisition, D.-S.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Research Foundation of Korea, grant number, NRF-2022R1A2C1005921) and BK21 FOUR (Fostering Outstanding Universities for Research, grant number, 5199990914048).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: <https://grouplens.org/datasets/movielens/> (30 January 2024).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Djedouboum, A.C.; Abba Ari, A.A.; Gueroui, A.M.; Mohamadou, A.; Aliouat, Z. Big data collection in large-scale wireless sensor networks. *Sensors* **2018**, *18*, 4474. [\[CrossRef\]](#)
2. Qolomany, B.; Al-Fuqaha, A.; Gupta, A.; Benhaddou, D.; Alwajidi, S.; Qadir, J.; Fong, A.C. Leveraging machine learning and big data for smart buildings: A comprehensive survey. *IEEE Access* **2019**, *7*, 90316–90356. [\[CrossRef\]](#)
3. Guk, K.; Han, G.; Lim, J.; Jeong, K.; Kang, T.; Lim, E.K.; Jung, J. Evolution of wearable devices with real-time disease monitoring for personalized healthcare. *Nanomaterials* **2019**, *9*, 813. [\[CrossRef\]](#)
4. Lemonde, C.; Arsenio, E.; Henriques, R. Integrative analysis of multimodal traffic data: Addressing open challenges using big data analytics in the city of Lisbon. *Eur. Transp. Res. Rev.* **2021**, *13*, 64. [\[CrossRef\]](#)
5. Kirmani, S.; Mazid, A.; Khan, I.A.; Abid, M.A. Survey on IoT-Enabled Smart Grids: Technologies, Architectures, Applications, and Challenges. *Sustainability* **2022**, *15*, 717. [\[CrossRef\]](#)
6. Fayyaz, Z.; Ebrahimian, M.; Nawara, D.; Ibrahim, A.; Kashef, R. Recommendation systems: Algorithms, challenges, metrics, and business opportunities. *Appl. Sci.* **2020**, *10*, 7748. [\[CrossRef\]](#)
7. Qin, L.; Xu, X.; Li, J. A real-time professional content recommendation system for healthcare providers' knowledge acquisition. In *Big Data–BigData 2018: 7th International Congress, Held as Part of the Services Conference Federation, SCF 2018, Seattle, WA, USA, 25–30 June 2018, Proceedings*; Springer International Publishing: Berlin/Heidelberg, Germany, 2018; pp. 367–371.
8. Rabiou, I.; Salim, N.; Da'ou, A.; Osman, A. Recommender system based on temporal models: A systematic review. *Appl. Sci.* **2020**, *10*, 2204. [\[CrossRef\]](#)
9. Bennett, J.; Lanning, S. The netflix prize. In *Proceedings of the KDD Cup and Workshop, San Jose, CA, USA, 12 August 2007*; Volume 2007, p. 35.
10. Covington, P.; Adams, J.; Sargin, E. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, 15–19 September 2016*; pp. 191–198.
11. Smith, B.; Linden, G. Two decades of recommender systems at Amazon.com. *IEEE Internet Comput.* **2017**, *21*, 12–18. [\[CrossRef\]](#)
12. Cheng, H.T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, Boston, MA, USA, 15 September 2016*; pp. 7–10.
13. Roy, D.; Dutta, M. A systematic review and research perspective on recommender systems. *J. Big Data* **2022**, *9*, 59. [\[CrossRef\]](#)
14. Vilakone, P.; Xinchang, K.; Park, D.S. Movie recommendation system based on users' personal information and movies rated using the method of k-clique and normalized discounted cumulative gain. *J. Inf. Process. Syst.* **2020**, *16*, 494–507.
15. Selimi, D.; Nuci, K.P. The use of Recommender Systems in web technology and an in-depth analysis of Cold State problem. *arXiv* **2020**, arXiv:2009.04780.

16. Shambour, Q.Y.; Hussein, A.H.; Kharm, Q.M.; Abualhaj, M.M. Effective Hybrid Content-Based Collaborative Filtering Approach for Requirements Engineering. *Comput. Syst. Sci. Eng.* **2022**, *40*, 113–125. [\[CrossRef\]](#)
17. Khanal, S.S.; Prasad PW, C.; Alsadoon, A.; Maag, A. A systematic review: Machine learning based recommendation systems for e-learning. *Educ. Inf. Technol.* **2020**, *25*, 2635–2664. [\[CrossRef\]](#)
18. Park, S.T.; Chu, W. Pairwise preference regression for cold-start recommendation. In Proceedings of the 3rd ACM Conference on Recommender Systems, New York, NY, USA, 23–25 October 2009; pp. 21–28.
19. Martins, G.B.; Papa, J.P.; Adeli, H. Deep learning techniques for recommender systems based on collaborative filtering. *Expert Syst.* **2020**, *37*, e12647. [\[CrossRef\]](#)
20. Peng, S.; Park, D.S.; Kim, D.Y.; Yang, Y.; Siet, S.; Ugli SI, R.; Lee, H. A Modern Recommendation System Survey in the Big Data Era. In Proceedings of the International Conference on Computer Science and Its Applications and the International Conference on Ubiquitous Information Technologies and Applications, Vientiane, Laos, 19–21 December 2022; Springer Nature: Singapore, 2023; pp. 577–582.
21. Pazzani, M.J.; Billsus, D. Content-based recommendation systems. In *The Adaptive Web: Methods and Strategies of Web Personalization*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 325–341.
22. Koren, Y.; Rendle, S.; Bell, R. Advances in collaborative filtering. In *Recommender Systems Handbook*; Springer: New York, NY, USA, 2021; pp. 91–142.
23. Sarwar, B.; Karypis, G.; Konstan, J.; Riedl, J. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th International Conference on World Wide Web, Hong Kong, China, 1–5 May 2001; pp. 285–295.
24. Takács, G.; Pilászy, I.; Németh, B.; Tikk, D. Scalable collaborative filtering approaches for large recommender systems. *J. Mach. Learn. Res.* **2009**, *10*, 623–656.
25. Barathy, R.; Chitra, P. Applying matrix factorization in collaborative filtering recommender systems. In Proceedings of the 6th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 6–7 March 2020; pp. 635–639.
26. Rendle, S.; Freudenthaler, C.; Gantner, Z.; Schmidt-Thieme, L. BPR: Bayesian personalized ranking from implicit feedback. *arXiv* **2012**, arXiv:1205.2618.
27. Sun, C.; Sun, G.; Ding, Z.; Liu, Q.; Ma, Z. A News Recommendation Algorithm Based on SVD and Improved K-means. In Proceedings of the 2021 International Conference on Networking, Communications and Information Technology (NetCIT), Manchester, UK, 26–27 December 2021; IEEE: New York, NY, USA, 2021; pp. 130–134.
28. Patoulia, A.A.; Kiourtis, A.; Mavrogiorgou, A.; Kyriazis, D. A comparative study of collaborative filtering in product recommendation. *Emerg. Sci. J.* **2023**, *7*, 1–15. [\[CrossRef\]](#)
29. Zhang, S.; Yao, L.; Sun, A.; Tay, Y. Deep learning based recommender system: A survey and new perspectives. *ACM Comput. Surv. (CSUR)* **2019**, *52*, 1–38. [\[CrossRef\]](#)
30. Xinchang, K.; Vilakone, P.; Park, D.S. Movie recommendation algorithm using social network analysis to alleviate cold-start problem. *J. Inf. Process. Syst.* **2019**, *15*, 616–631.
31. Jing, H. Application of Improved K-Means Algorithm in Collaborative Recommendation System. *J. Appl. Math.* **2022**, *2022*, 2213173. [\[CrossRef\]](#)
32. Wang, K.; Zhang, T.; Xue, T.; Lu, Y.; Na, S.G. E-commerce personalized recommendation analysis by deeply-learned clustering. *J. Vis. Commun. Image Represent.* **2020**, *71*, 102735. [\[CrossRef\]](#)
33. Chen, J.; Li, K.; Rong, H.; Bilal, K.; Yang, N.; Li, K. A disease diagnosis and treatment recommendation system based on big data mining and cloud computing. *Inf. Sci.* **2018**, *435*, 124–149. [\[CrossRef\]](#)
34. Rendle, S.; Freudenthaler, C.; Schmidt-Thieme, L. Factorizing personalized markov chains for next-basket recommendation. In Proceedings of the 19th International Conference on World Wide Web, Raleigh, NC, USA, 26–30 April 2010; pp. 811–820.
35. Tang, J.; Wang, K. Personalized top-n sequential recommendation via convolutional sequence embedding. In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, Marina del Rey, CA, USA, 5–9 February 2018; pp. 565–573.
36. Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; Tikk, D. Session-based recommendations with recurrent neural networks. *arXiv* **2015**, arXiv:1511.06939.
37. Choe, B.; Kang, T.; Jung, K. Recommendation system with hierarchical recurrent neural network for long-term time series. *IEEE Access* **2021**, *9*, 72033–72039. [\[CrossRef\]](#)
38. Duan, J.; Zhang, P.F.; Qiu, R.; Huang, Z. Long short-term enhanced memory for sequential recommendation. *World Wide Web* **2023**, *26*, 561–583. [\[CrossRef\]](#)
39. Kang, W.C.; McAuley, J. Self-attentive sequential recommendation. In Proceedings of the IEEE International Conference on Data Mining (ICDM), Singapore, 17–20 November 2018; pp. 197–206.
40. Yu, S.; Guo, M.; Chen, X.; Qiu, J.; Sun, J. Personalized Movie Recommendations Based on a Multi-Feature Attention Mechanism with Neural Networks. *Mathematics* **2023**, *11*, 1355. [\[CrossRef\]](#)
41. Chen, Q.; Zhao, H.; Li, W.; Huang, P.; Ou, W. Behavior sequence transformer for e-commerce recommendation in alibaba. In Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data, Anchorage, AK, USA, 5 August 2019; pp. 1–4.

42. Wang, D.; Zhang, X.; Yu, D.; Xu, G.; Deng, S. Came: Content-and context-aware music embedding for recommendation. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 1375–1388. [[CrossRef](#)] [[PubMed](#)]
43. Chen, Q.; Jiang, F.; Guo, X.; Chen, J.; Sha, K.; Wang, Y. Combine temporal information in session-based recommendation with graph neural networks. *Expert Syst. Appl.* **2024**, *238*, 121969. [[CrossRef](#)]
44. Mavrogiorgos, K.; Kiourtis, A.; Mavrogiorgou, A.; Kleftakis, S.; Kyriazis, D. A multi-layer approach for data cleaning in the healthcare domain. In Proceedings of the 2022 8th International Conference on Computing and Data Engineering, Bangkok, Thailand, 11–13 January 2022; pp. 22–28.
45. Eskandarian, F.; Mobasher, B.; Burke, R. A clustering approach for personalizing diversity in collaborative recommender systems. In Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization, Bratislava, Slovakia, 9–12 July 2017; pp. 280–284.
46. Saeed, M.; Mehrdad, M.; Farahnaz, H. Optimal Diversity of Recommendation List for Recommender Systems based on the Users' Desire Diversity. *J. Inf. Sci. Theory Pract.* **2019**, *7*, 31–39.
47. Movielens, GroupLens. Retrieved 31 January 2023. Available online: <https://grouplens.org/datasets/movielens/> (accessed on 8 December 2021).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.