Human-Centric Intelligent Systems (2024) 4:344–362
https://doi.org/10.1007/s44230-024-00073-3

RESEARCH ARTICLE

X-Model4Rec: An Extensible Recommender Model Based on the User's Dynamic Taste Profile

Rogério Xavier de Azambuja[1,2,3] · **A. Jorge Morais**[2,4] · **Vítor Filipe**[3,4]

Received: 28 December 2023 / Accepted: 16 April 2024 / Published online: 27 June 2024

Abstract
Several approaches have been proposed to obtain successful models to solve complex next-item recommendation problem in non-prohibitive computational time, such as by using heuristics, designing architectures, and applying information filtering techniques. In the current technological scenario of artificial intelligence, sequential recommender systems have been gaining attention and they are a highly demanding research area, especially using deep learning in their development. Our research focuses on an efficient and practical model for managing sequential session-based recommendations of specific products for users using the wine and movie domains as case studies. Through an innovative recommender model called X-Model4Rec – eXtensible Model for Recommendation, we explore the user's dynamic taste profile using architectures with transformer and multi-head attention mechanisms to solve the next-item recommendation problem. The performance of the proposed model is compared to that of classical and baseline recommender models on two real-world datasets of wines and movies, and the results are better for most of the evaluation metrics.

Keywords Recommender Systems · Sequential Recommendation · Session-based Recommendation · Deep Neural Network · Transformer · Attention Model

1 Introduction

The recommendation problem can be formally defined as finding a utility function to recommend one or more highest-rated items for each user, ranked by a single output score [1]. Recommender Systems (RSs) assist users in discovering relevant items, creating an attractive user experience. Mathematics and statistical methods are used to enhance human perception, detecting similarities or variations in recommended items. Recently, machine learning methods involving Deep Neural Networks (DNNs) have been used to capture users' interest to generate personalized recommendations [2–4]. Due to their utility, RSs have gained attention in several fields, such as e-commerce, e-business, e-learning, e-tourism, etc., attracting the interest of companies such as Netflix, Amazon, Google, Meta, and Alibaba, among others.

Usually, a RS receives as an input a set of users $U = \{u_1, \ldots, u_{|U|}\}$, a set of items $I = \{i_1, \ldots, i_{|I|}\}$, and user-item preference ratings $R = \{r_{u,i}\}$ available at the contextual moment (C), which can be created by gathering various sets obtained through explicit and implicit user feedback. User preferences can be sparse, scarce (cold start problem), undergo temporal changes, or be estimated through prediction mechanisms. Different approaches to information filtering based on: collaboration, content, knowledge, demographics, context, or even hybrid [1–3] usually provide as an output a top set of items for each user containing relevant recommendations $top@k_u = \{i_1, \ldots, i_k\}$, where $k \in \mathbb{N}^*$. When evaluating the performance of RSs, specific metrics are employed. These metrics can ignore ranks, such as precision, recall, F1-Score, etc., or explore rank-aware positions, such as the Mean Reciprocal Rank (MRR), Mean Average Precision (MAP), and Normalized Discounted Cumulative Gain (NDCG), among others. Evaluation metrics check whether

✉ Rogério Xavier de Azambuja
rogerio.xavier@farroupilha.ifrs.edu.br

1 Instituto Federal do Rio Grande do Sul (IFRS), Farroupilha, RS 95174-274, Brazil

2 Department of Science and Technology, Universidade Aberta (UAb), 1269-001 Lisbon, Portugal

3 School of Science and Technology, Universidade de Trás-os-Montes e Alto Douro (UTAD), 5000-801 Vila Real, Portugal

4 INESC TEC–INESC Tecnologia e Ciência, 4200-465 Porto, Portugal

the user liked or interacted with a particular item among the recommended *top@k* items [5–7], which marks the next-item recommendation problem when only one target item is desired by each user.

In addition to RSs features previously presented, Sequential Recommender Systems (SRSs) are being increasingly used [8]. SRS is a type of RS that process serialized data as input to generate recommendations. Sequential recommendation categorizes user interactions by date and time stamp to help in capturing users' historical behavioral patterns to anticipate items that may be of interest to them. SRS in the online environment requires continuous adaptation mechanisms because new and obsolete users, items, and feedback need to be considered in recurring iterations over time intervals.

In this study, the wine domain was chosen for most experiments due to its significance in Portugal and Brazil, particularly in the Serra Gaúcha region, and its global recognition and appreciation. Wine has a production chain that requires knowledge, talent, and creativity and presents a very small data volume for exploratory works on recommender systems. In addition, for scientific experimentation purposes in more than one dataset, the movie domain was chosen for comparison. Movies also have their peculiar characteristics that could stimulate users' future interest. The research challenge is to provide adaptive recommendations to mitigate information overload, especially in online environments, due to their dynamic nature. We want to contribute to this field of research by recognizing the potential of next-item recommendation to improve the user experience.

This paper proposes a less complex model for sequential recommendation called X-Model4Rec – eXtensible Model for Recommendation that explores the user's dynamic taste profile to recommend relevant items to users. Related works on the next-item recommendation problem and sequential recommender systems, as well as algorithms and a simple demonstration of the proposed model using a new wine dataset [9] and a traditional movie dataset [10], are presented as partial results of our research. The results obtained with the proposed model are compared to that of classical and some baseline recommender systems. The main contributions are highlighted hereafter:

- The proposed X-Model4Rec is an extensible sequential and single-session-based recommender for effective next-item recommendation. This model proposes a data modeling to obtain sessions with a defined length and it was designed to allow the exchange of internal components, such as the transformer block, which specifically focuses on prediction and ranking to explore some identified gaps.
- The prediction model focuses on capturing intricate data on the user-item relationship by modeling built to explore the user's dynamic taste profile, which is sensitive to variations in the item categories with which they have interacted in the past. By analyzing variations or continuities in item categorization, the model recommends the next item for each user based on their sequential historical data. This is achieved using advanced DNN mechanisms such as transformers and multi-head attention.

Table 1 Notations used in the next-item recommendation problem

Notation	Description
$\hat{\imath}$	predicted next item
τ	verified next item
C	set of user feedback at the contextual moment, $C = \{(U \times I)_1, \ldots, (U \times I)_{\lvert C \rvert}\}$
emb	embeddings representation
i	item entity with its attributes
I	set of items i, $I = \{i_1, \ldots, i_{\lvert I \rvert}\}$
j	index
k	number of items, $k \in \mathbb{N}^*$
p	probability
r	preference rating value
R	set of preference ratings r, $R = \{r_{u,i}\}$, where $u \in U$ e $i \in I$
$rank_j$	the j-th item of $top@k$, where $1 \leq j \leq k$
rel_j	the j-th item relevance score of $top@k$, where $1 \leq j \leq k$
s	session, $s = \{i_1, \ldots, i_{\lvert s \rvert}\}$
S	set of sessions s, $S = \{s_1, \ldots, s_{\lvert S \rvert}\}$
t	time interval, $t =$ |minutes, hours, days, weeks, months, all range, …|
u	user entity with its attributes
U	set of users u, $U = \{u_1, \ldots, u_{\lvert U \rvert}\}$
$top@k$	set of k relevant items recommended for each user u, $top@k = \{i_1, \ldots, i_k\}$

- Extensive experiments were conducted on two real-world transaction datasets on specific products (wines and movies), and promising results were obtained, outperforming other classical and baseline models.

2 Background

Identifying items that will be of interest to users in the future is a complex and challenging task. The main symbol notations used in this context are listed in Table 1.

2.1 Next-Item Recommendation Problem

The next item recommendation is studied in the scenario where only one target item is desired by each user. Based on the information presented in [11–13], the next-item recommendation problem can be formally defined as accurately classifying a specific next item (target) for each user among all items (classes) established from user and item features and contextual information with a time stamp. This classification problem can be expressed by the following general formula:

$$\hat{i}_{t+1}^{u} = \text{argmax}\ p(i \in I | U, I, C(t)) \quad (1)$$

where $U = \{u_1, \ldots, u_{|U|}\}$ denotes a set of users, $I = \{i_1, \ldots, i_{|I|}\}$ denotes a set of items, and $C = \{(U \times I)_1, \ldots, (U \times I)_{|C|}\}$ denotes one or more sets of user feedback at contextual moment $C = f(t)$, which vary as a function of the time interval (t).

Normally, several factors need to be considered to generate a feasible solution to the next-item recommendation problem. Each user can exhibit timeless personalized behavior and make the process of finding, scaling, and generalizing a solution difficult. This is a problem in which others are concatenated, such as data retrieval, classification, prediction, and ranking. In this sense, the recommender model aims to classify a list of items and identify only one target item among the k recommended items for each user [14–16]. The problem increases with scalability since anyone among all items is a potential candidate to be the next item, with a probability considered $p = \frac{1}{|I|}$.

Different applications use rich metadata comprising structured and unstructured data to represent various characteristics, which can be explored to generate recommendations. The sets of users (U) and items (I) are typically created by continuous structured data, such as numeric values, and categorical structured data, such as wine type, movie genre, and gender. These sets may also include unstructured data, such as text, images, audio, and videos. Preference ratings and additional user feedback are available resources at contextual moment (C) which can be composed of explicit and implicit user feedback.

To make experimentation possible by recommending the next item from a defined data sample, the item that each user last interacted with is typically removed beforehand. It is subsequently compared with the recommendation generated afterward. All the data in the sample, except for the last interaction taken for each user, can be used to generate the *top@k* recommendation. This process makes it challenging for the recommender system to classify items for each user among all available items, "without knowing" the next targeted item given by Eq. (1). The generated recommendation is evaluated using specific metrics, such as those presented in Section 5.3. The next item previously removed from the training set is considered a constant (τ) in the measurements, where $|\tau| = 1$ and is the target compared to the generated *top@k* recommendation. Here, k is the number of recommended items, i.e., 1, 10, 20, 30, 40, 50, …

2.2 Sequential and Session-based Recommendation

Recent studies have focused on exploring sequential recommendations (SRec) supported by session-based recommendation (SBRec) to solve recommendation problem in online environments [3, 17, 18]. As illustrated in Fig. 1, when using historical information, the time interval (t) can be uniform or varied and the treatment of sequential events (with a time stamp) will model the user's behavior in short- and long-term sequential dynamics.

Considering the user's interactions containing time stamps in the interval $[-\infty, t]$, it is possible to form temporal user sessions $S_u = \{s_1, \ldots, s_t\}$ that can aid in the development of sequential recommendation heuristics, where each session s denotes its order in S_u w.r.t. its occurring time $s_t^u = \{i_1, i_2, \ldots, i, \ldots\}$. A monolithic session records user interaction in a system execution and normally has a time interval $t =$ minutes, hours, days, etc. On the Web, a session starts when a user accesses the website and continues on it, and it finishes when the user leaves [19]. Sessions can also be simulated in databases by dividing user interactions into a defined time interval or even a defined length of iterations for the considered long-term. It is important to consider the use of single session that makes next-item recommendations based on the current session only and multi-session that manipulates other sessions to complement the information contained in the current session for next-item recommendations [15].

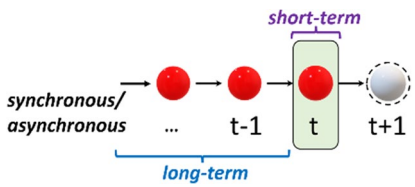

Fig. 1 Mapping user behavior over time intervals

In theory, SRec considers the order of events arising from user behavior (historical sequential data) to obtain user preferences, which are modeled by recurrence to predict the next item of interest to the same user [20]. Sessions can be created by recording the system execution during a time interval, and sessions can also be simulated from databases. Personalization can use event data from the current session and stored data from previous sessions to provide SBRec in short- and long-term sequential dynamics. In practice, the SRec can be constructed either within the session or outside of it. On the other hand, SBRec enables the use of this information without any concern for its sequential arrangement. In this way, SRec and SBRec complement each other in constructing Sequential Recommender Systems (SRSs).

3 Related Work

Prominent research can be found in the literature references. Deep learning has become a state-of-the-art method in various fields, including speech and visual object recognition, as well as Natural Language Processing (NLP). It is constantly evolving in the SRSs [3, 21]. The use of deep learning for recommendations presents a challenge and an object of study for research. In recent years, the increase in attention mechanisms [22] and more recently, the use of transformers [23] have also been considered.

The recommendation problem is summarized as a matrix filling task in [24]. When there is data scarcity for the recommendation (sparse matrices), several estimates need to be made [25], and some authors have proposed works based on collective recommendations for user groups to mitigate the cold start problem [26, 27]. Neighborhood-based models use user-based and item-based methods to find similar users with related interests. Despite their widespread use, neighborhood-based models seek to overcome limitations using specific heuristics, as presented in [28, 29].

SRSs are characterized as special forms of context-based RSs in [8], since the short-term dynamics involve the user behavior context. The characterization of a specific utility function is also described in [30], where SRSs are classified into three distinct types: traditional sequence, latent representation, and DNN. Remarkable results in SBRec with the architecture Recurrent Neural Network (RNN) were obtained for the first time in [16]. The authors proposed a model in which the first item clicked by a user is used as the initial input of the RNN that generates the recommendation via recursion. Each new click of the user is used in the network, considering all previous clicks in the output layer. Despite good results, the item set to be considered may be on the order of thousands, and the performance of the computational model may be compromised.

To capture sequential dynamics, models have used Markov chains to model the probability of choosing one item from the ordered list of the items already selected. A broad study is presented in [6], where methods such as sequential rule mining and session-based neighborhoods were compared with more complex methods, such as those using DNNs. The less complex methods resulted in better performance in accuracy measurements. SBRec was generated in [31] by means of Graph Neural Network (GNN) that combines a neural network with graph theory. The authors modeled session-based sequences as structured data in graphs to represent the current interest and user's global preference in the active session.

Self-Attention based Sequential Recommendation (SASRec) model was proposed in [32], unlike models based on Markov chains, RNNs, or Convolutional Neural Networks (CNNs), SASRec uses the economical Multilayer Perceptron (MLP) architecture with optimal computational performance. It can model semantic and syntactic patterns between recurrent iterations over time intervals. The self-attention model was evaluated with long-term scenarios represented by dense datasets and simultaneously with sparse datasets generated by more recent user interactions.

Adaptively Distilled Exemplar Replay (ADER) model [18], which specializes in SASRec for building robust and scalable SBRec in online environments, was presented at ACM/RecSys'2020 [33]. The ADER model proved to be an innovative proposal for mitigating a classic problem in SBRec: the catastrophic forgetting problem from past data. Repeated data examples in previous cycles chosen dynamically by the frequency of the items (herding method) are proposed, considering a loss of adaptive distillation (knowledge distillation). The portion of the selected data is interpolated at each time interval (t) with new data before a new training cycle and the next model update for the future cycle ($t + 1$). By considering a part of the data previously labeled in the training, ADER achieved excellent results, even when compared to the models that use all the historical data again for training at each time interval. However, practical tests revealed that the model needs computational power because the training is still slow.

Other SRS models studied are widely used in practice with good results evaluated offline and mainly present variations between their architectures. The Gated Recurrent Unit for Recommender model (GRU4Rec) [16] makes use of RNN, the Convolutional Sequence Embedding Recommendation model (Caser) [20] uses CNN, and Bidirectional Encoder Representations from Transformer for sequential Recommendation model (BERT4Rec) [23] presents a bidirectional model attention-based to the recommendation using transformer mechanisms.

Finally, some DNN models utilize methods that allow switching of the calculation formula between more than one similarity and diversity metric in information filtering [19, 34, 35]. According to the extensive reviews conducted in [2], the use of DNNs has been growing in recent years in the field of RSs, mainly using reinforcement learning.

4 Research Overview

4.1 Key Challenges

Heuristics play a crucial role in enabling software to understand users' personalized behaviors and predict their preferences individually to meet their needs. Some heuristics recommend popular items highly rated by other users, or naively assume that past interactions with items make them more relevant, which may not be true, as the user may simply not be aware of all the available items and the next item may have received a low rating from the user.

Some explicit gaps related to SRSs are identified as challenges from our research. Most of the authors surveyed agree that users normally interact with only some items from the total available items and that there is no linear correlation between the historical items that the user has interacted with in the past and the predicted next item. The lack of a unified systematic categorization is presented in [36], where to address this, a framework has been proposed to reduce some theoretical and practical SRSs confusions and inconsistencies. Regarding using DNNs, the task is learning from embeddings given U, I, C with time stamp sets, the goal is to create a function of the user's history and context that is useful for classifying the target next item among all items. However, there is considerable difficulty in creating models that can quickly converge on a good solution. Most of the models are evaluated offline and aim to find realistic practices for evaluating a recommender when meeting user needs, especially considering short- and long-term sequential dynamics. Most of the existing studies on SRSs are based on long-term sessions, as short-term sessions involve limited contextual information and "making the next-item recommendation very challenging" [15]. Personalization of the time, user experience [19], and explanation can be researched to improve the recommendations to the users, seeking to increase their trust in the SRSs. In addition, the prediction can be used to promote diversity and uncover new, popular, and unexpected items. However, there are still unanswered questions regarding the practical computational performance of DNNs [6].

4.2 Methodology

Our motivation is to find emerging solutions that can be replicated with grounded insights in the next-item recommendation problem. To achieve this goal, we organized our methodology using design science research as follows:

I. The first stage contemplates a review of the literature reference and laboratory reproduction of existing and selected models, such as: SASRec [32], ADER [18], GRU4Rec [16], Caser [20], BERT4Rec [23], Long Short-Term Memory bias optimized for recommender model named here Bias-LSTM [37], and Long- and Short-Term Preference Model (LSPM) [14], among others with theoretical and practical potential for collaborative expertise.
II. In the second stage, we aim to develop models that can adapt to the realities of Web applications. To achieve this goal, we evaluate the performance of different models through controlled experiments with varying parameters. The input datasets obtained by implicit and explicit feedback were alternated to explore the complexity of the architectures versus their computational performance. Our primary goal is to specialize in our models to construct an efficient model that produces concrete results.
III. Based on the results, in the third stage, we aim to make substantiated contributions on the specific theme using a recommender model that considers the best answers found, justified, and applicable for the next-item recommendation problem. The parameters used and analysis of the concrete results obtained in the comparisons between the methods used will be highlighted in corrective and refutative ways.

Our research focused on improving the quality of recommendations in an online environment using X-Model4Rec built through SRec- and SBRec-based methods. To implement this model, we use single session processing via a DNN with multi-head attention and transformer mechanisms.

5 The eXtensible Model for Recommendation (X-Model4Rec)

The practical use of SRec and SBRec is related to next-item prediction for users who have previously been interested in or purchased other items. When recommending products, it is essential to consider their specific characteristics. Wines, for example, have attributes such as type, grape variety, acidity, body, etc., while movies are cultural products of the film industry with other specific features such as artists, genre, duration, scenario, etc. The recommender model must be adaptable to the peculiarities of the input data. A wine that is appreciated by the user may always be of interest to them, but a movie that the user has already watched, or another product that the user has previously interacted with, may no longer be of interest to them.

The proposed X-Model4Rec – eXtensible Model for Recommendation is a recommender model adapted from the input data that aims to explore the alternations and the continuities identified in the historical interactions conducted by the users. Unlike other models that disregard more than one user interaction on the same item, often considering only the unique or most recent reviews, in this

case, multiple ratings on the same item are very welcome for dynamic taste profile (DTP) processing.

The X-Model4Rec illustrated in Fig. 2 includes the data preprocessing to transform raw data into a set of sessions S, in which each session $s \in S$ is defined between the minimum and maximum length that respects the temporal sequence; the training and prediction from the DNN process contain a simplified architecture to analyze the set of input sessions; this way the data represented in embeddings are explored by a few architectural blocks to form an intermediate spectrum, a matrix $p_{|U|\times|I|}$ of correlated values that will be ranked to form the recommendation *top@k* items for each user. The DNN process looks at intricate data about the alternation or continuity of behavior based on the user's historical preferences. In the wine and movie examples, the users were asked whether they liked the same types of items over time and how they evaluated them positively, neutrally, or negatively. This continuity or alternation in the taste profile is assigned dynamically calculated weights in the DNN layers to find the mathematical formulation that predicts positive or negative scores for each pair of item-user in the output layer that performs the categorical classification task.

The sequential multi-head attention network, powered by transformer mechanisms, considers both short-term and long-term user behavior records in nonlinear correlation networks. This approach helps determine user preferences and assigns weights to positive and negative variations to suggest the most suitable next item for the target user based on their past interactions.

Three algorithms were created for experimentation with the X-Model4Rec using both the new wine dataset and the traditional movie dataset. Algorithm 1 preprocesses the datasets to form the training and test sets using SRec and SBRec. Algorithm 2 is responsible for training the model, making predictions, and ranking the results for the *top@k* next-item recommendation study cases. This algorithm uses the dynamic taste profile, which is monitored for loss rate and Area Under the Receiver Operating Characteristic (ROC) Curve (AUC) rate. Finally, Algorithm 3 evaluates the model performance using precision, recall, F1-Score, MRR, MAP, and NDCG metrics. In the following sections, we will provide details about all three stages, including the design decisions and algorithms used.

5.1 Preprocessing

Data preprocessing is a crucial and often expensive task. Its goal is to convert raw data into validated and organized data for appropriate use through the sessions in the second stage, as described in Section 5.2. Regardless of the dataset used, we seek to work with some valuable features. In addition to sets of identification codes such as user IDs, item IDs and ratings values, the categorization of items and the date and time stamps in the evaluation conducted by the user are used. Many other features are important and can also be explored by computational recommender models, such as implicit information obtained mainly in online environments, e.g., clicks, view timers, spatial location in the interface, etc., which we have not yet used in our experiments. Algorithm 1 performs the explicit data preprocessing, which involves the formation of training and testing sets, here instances named *trainFlow* and *testFlow*, respectively.

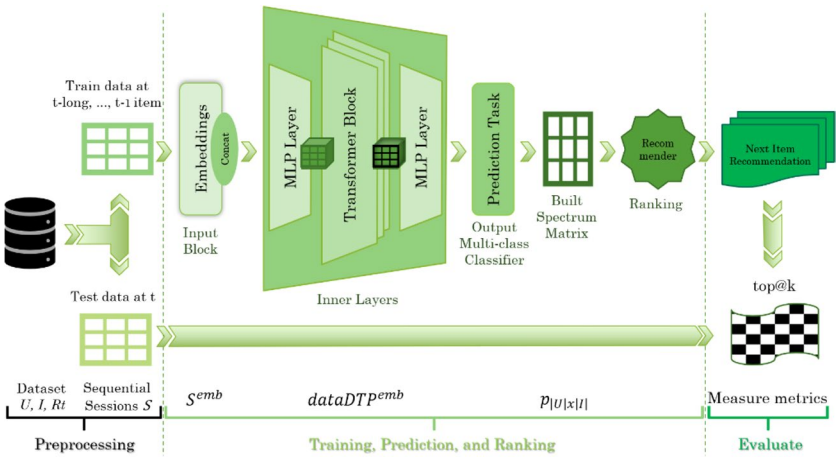

Fig. 2 The X-Model4Rec design is composed of three stages

Algorithm 1 Preprocessing

Input: Dataset
Output:Training (*trainFlow*) and testing (*testFlow*) sets containing user-item sessions
1. Define the global time range to filter all user-item interactions
2. Define the time interval t that makes up a session s, where $t = |$minutes, hours, days, weeks, months, all range, …$|$ and define the minimum (*min*) and maximum (*max*) session lengths, where *min*, *max* $\in \mathbb{N}^*$
3. Process the categorization of the sets: Users U, Items I, and Ratings $R \in C$ with threshold positive/negative
4. For each user u from U do: *# find single sessions with a defined length*
5. Concatenate user-item interactions to form sessions $S_u = \{s_1, \ldots, s_t\}$, where $s_t^u = \{i_1, \ldots, i_{max}\}$ by a countdown time counter $[t - max, t]$
6. Process the DTP user sequence by $DTP_u = \bigcup_{1}^{max} \begin{cases} 1, & \text{changed the last category} \\ 0, & \text{kept the last category} \end{cases}$
7. Find the most recent short-term user session $S_t^u = \{s_t^u\}$
8. Find the set of long-term sessions formed by the remainder of the user sessions slot $S_{t-1}^u = \{s_1^u, \ldots, s_{t-1}^u\}$
9. Pick the last item i from $S_t^u = \{s_t^u - i\}$ to form the testing set (*testFlow*) truncated by the *max* session length
10. The remainder of the user short-term session slot S_t^u and long-term sessions S_{t-1}^u form the training set (*trainFlow*) truncated by the *max* session length

Figure 3 presents a data flow diagram that explains Algorithm 1 graphically. The diagram was created using the NVIDIA NV Tabular framework [38]. It illustrates the dataflows of the schema created, starting from the raw data inputs and continuing through the preprocessing stage, which involves operations to form the training and test datasets. This modeling enables the input data to be transformed by adding reliable labels that can be used for supervised learning and other machine learning methods to train the X-Model4Rec computational model.

The diagram in orange shows the data flow process using five inputs: *UserID*, *Date*, *ItemID*, *Category*, and *Rating*. The data are categorized and transformed by building single sessions for each user with a defined length (lines 1–3 in Algorithm 1). The main operations performed on these input data are illustrated in green: *Groupby*, *ListSlice*, and *SessionFilter* (lines 5, 7–10 in Algorithm 1). In the center, in cyan, the user's dynamic taste profile is identified through a binary classification based on the category of items with which the user has interacted in the past. This taste profile can be updated adaptively with each new preprocessing run based on new input data obtained at different contextual times. Finally, the categorized and labeled data are divided into two distinct sets for training (*trainFlow*) and testing (*testFlow)*. The desired result is illustrated in yellow at the last vertex of the diagram, as shown in Fig. 4. This is an extract of illustrative data obtained from an execution of Algorithm 1.

For each user presented in column *UserID*, a row of data is created in a table with several lists of correlated information between the elements of the lists. These lists include columns such as *ItemID_list*, *Category_list*, *binaryRating_list*, and *DTPSequence_list*. The last value in each list of the *testFlow* set is not present in the *trainFlow* set, as it represents the next item previously removed from the whole, i.e., the target for which our recommender model will strive. The *trainFlow* set consists of defined length lists truncated between the parameters $(1, max)$ up to the penultimate sequential item found in the input data. It is also worth noting that numerical values are categorized starting from zero or true and false Booleans such as 1 and 0. However, in this case, values 3 and 4 are used because of the categorization performed by the NVIDIA NV Tabular framework, which models the raw data inputs. The software reserves indices 0, 1, and 2 for its internal control and produces categorical values from index 3. The *ItemCountSession* column represents a normalized value that expresses the total number of items evaluated by the

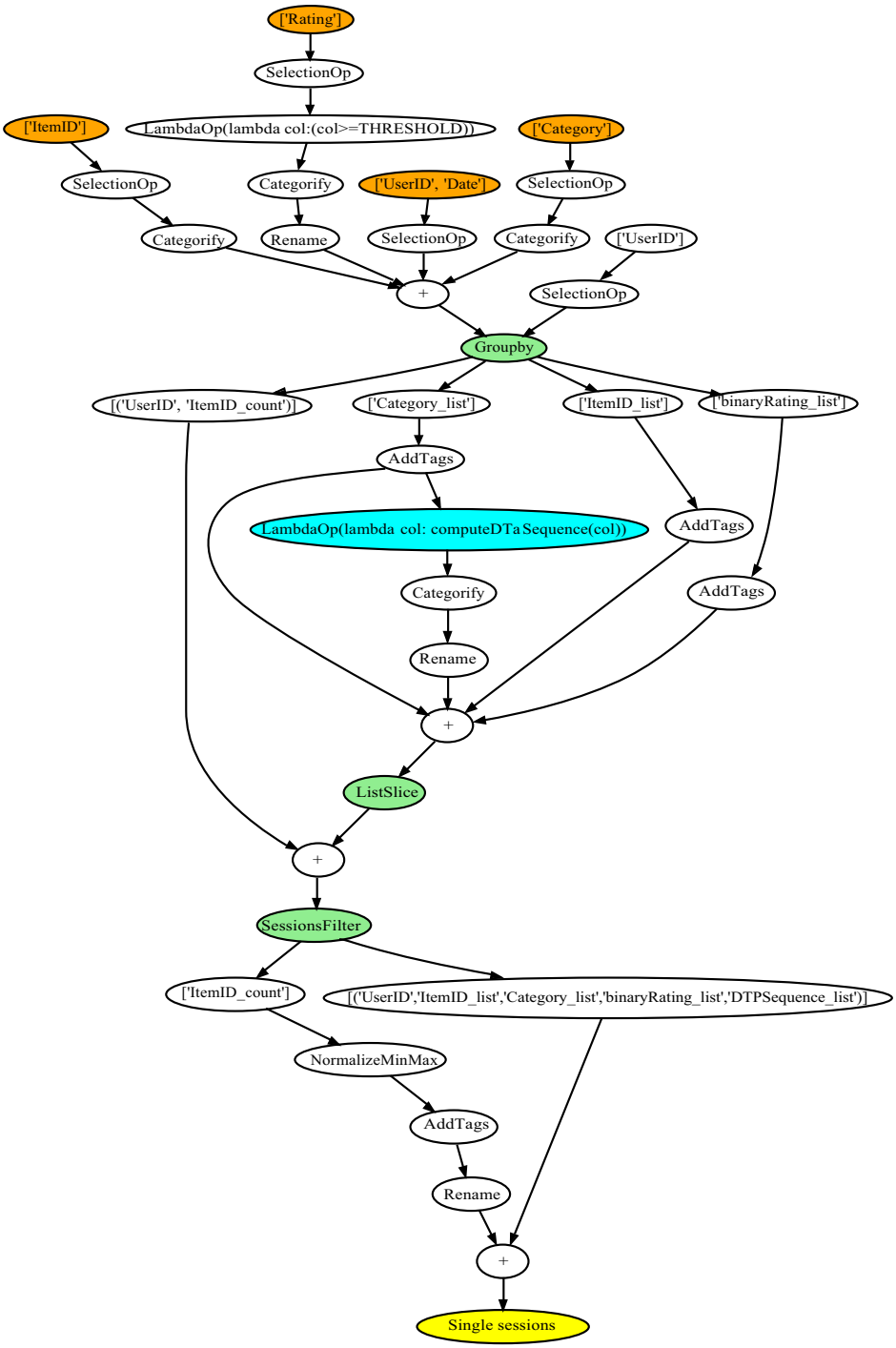

Fig. 3 Data entry modeling for training and testing

respective user. With this number, it is possible to determine whether the user has greater or lesser participation in the preference evaluations, given that the length of the sessions has been truncated to maximum defined values.

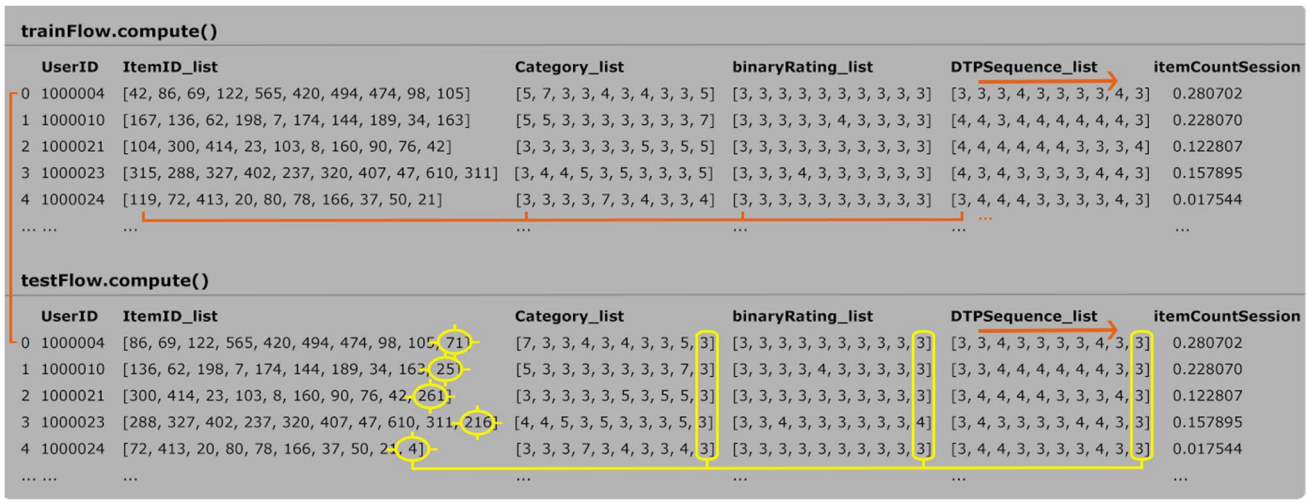

Fig. 4 Data entry sequence for training and testing

5.2 Training, Prediction and Ranking

The visualization of X-Model4Rec is shown in Fig. 2. offers a high-level overview of all the layers in the training model for a specific data input, an instance of the *trainFlow* set containing the sessions modeled using binary classification DTP. To start processing in this second stage, multiple resources are concatenated from the input block to form a unique set of data represented in the embeddings. The use of embeddings has proved to be extremely useful in the developed model by adding resources in multi-dimensions. The data flow diagram followed by the recommender algorithm is shown in Fig. 5.

The transformation of embedding data is modeled and implemented by separate blocks in the X-Model4Rec architecture, represented by the inner layers, maintaining the same dimensionality in the dataset processed between the model layers. This simplified architectural configuration allows the transformer block in the center to be changed. Thus, this approach allows us to explore different pre-trained transformer mechanisms with various architectures through an extensible model.

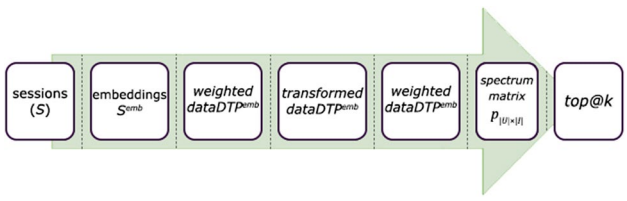

Fig. 5 Data flow for recommendation generation

The transformer [22] is an autoregressive mechanism and its architecture is proposed without recurrence or convolutions that can improve computational performance but may require a large amount of memory for processing. In the transformer block, parallelism is exploited to modify weighted data sequences represented in embeddings. It consists of two internal building blocks, an encoder block and a decoder block. Both internal blocks are formed by several pre-tested and interconnected layers to calculate linear transformations in the data through processes that can be parallelized using a feed-forward network (FFN) fully connected in each internal module. Furthermore, position encodes are injected into the data to mark the absolute and relative positions in the input sequence of these data, producing tokens used in the internal multi-head attention mechanisms to direct attention to some positional token. We used attention to the next token during model training and attention to the last sequential token in the prediction step, as faster convergence was identified during model development.

Although the use of different attention heads in processing data does not guarantee a significant improvement in the performance of the model, it does increase interpretability, making it easier to find the best formulation for a dynamic objective function given by Eq. (2). It is achieved using the initial data modeling (S) for the model to capture intricate data on the relationship between the user and the item ($dataDTP^{emb}$). These data are subsequently used during the execution of the model to direct attention toward the next and last item in the input sequence to form a probability spectrum matrix $p_{|U|x|I|}$ that will lead to the recommendation.

$$\hat{i}_{t+1}^{u} = argmax\ p(i \in I | multi\_class\_classifier(MLP(Transformer(MLP(S))))^{attention}) \quad (2)$$

The built modeling in Section 5.1 is used to explore the user's dynamic taste profile, which is sensitive to variations in the item categories with which they have interacted in the past. The steps used are described in Algorithm 2.

Algorithm 2 X-Model4Rec Recommender

Input: Training set (*trainFlow*) and hyperparameters (*EPOCHS, EMBEDDING_DIM, NEURONS, HEADS, DROPOUT, BATCH_SIZE_TRAINING, BATCH_SIZE_TEST, OPTIMIZER, LOSS, TEMPERATURE_SCALING*)

Output: Set of $top@k$ recommendations, where $k = \{1, 10, 20, 30, 40, 50, \ldots\}$

 # step 1: training

1.For each of the *EPOCHS* using *BATCH_SIZE_TRAINING*, do:
2. Process sessions S to embeddings using the *EMBEDDING_DIM* dimension and concatenate all features to form a unique set of multi-dimensional data S^{emb}
3. Process combinations in S^{emb} to form weighted user preferences $dataDTP^{emb}$ using Multilayer Perceptron (MLP) with linear activation, *NEURONS,* and *DROPOUT* values
4. For each of the *HEADS*, do: *# find weights using transformers and multi-attention*
5. Transform the weighted $dataDTP^{emb}$ by each head separately into attention to the next position from the shifted sequence using *NEURONS* values
6. Process combinations in transformed $dataDTP^{emb}$ to form weighted user preferences using MLP with linear activation, *NEURONS,* and *DROPOUT* values
7. Compute the multi-class classification task using linear activation and the *TEMPERATURE_SCALING* value
8. Calculate partial scores to track training metrics: true positive rate (TPR) and false positive rate (FPR) to form the AUC rate, Precision, Recall, F1-Score, MRR, MAP, and NDCG (equations 3-9 presented in Section 5.3)
9. Calculate the loss rate using the *LOSS* function
10. Update all computed scores using the *OPTIMIZER* function by learning rate

 # step 2: prediction

11. Predict the next-item probability from sessions S in the training set (*trainFlow*) into attention to the last position of the sequence to form a spectrum matrix $p_{|U|\times|I|}$ using *BATCH_SIZE_TEST*

 # step 3: ranking

13. For each user u from U, do:
14. Compute the ranking from the probability spectrum matrix $p_{|U|\times|I|}[u]$
15. Compute $top@k_u$ given the best scores

In Algorithm 2 of our coding process, we used the NVIDIA Merlin-Models framework [38], which provides an effective way to construct models, prepare sequential inputs, and code support for next-item prediction. A limitation of using black-box tools is that performing a global computational asymptotic analysis of the model can be challenging mainly because it is an extensible model. Conducting X-Model4Rec implementation in layers to form structural blocks and increase sequential interactions during training (line 5 in Algorithm 2), we used the *SequencePredictNext* class, which extracts the target from the shifted sequence of positions and truncates the sequential input features in the last position. The sequential input column *ItemID_list* in *trainFlow* was used to extract the target. For prediction in step 2 (line 11 in Algorithm 2), we used the S*equencePredictLast* class, which reserves the last item in each *ItemID_list* column as a target and does not use it as an input sequence during training. Thus, the last item as a target is selected (masked) to be predicted following the attention concept [22]. To combat the problem of overfitting, we utilized

weight tying as a form of regularization by sharing the same weight matrix between the input block and the output multi-class classifier (with linear activation) in our classification task (line 7 in Algorithm 2). We conducted experiments using the temperature parameter [39] in the classifier to vary the probability distribution. From the spectrum scores generated by the classifier, we obtained the candidate items to compose the *top@k* items recommendation for each user.

Although Merlin-Models do not provide access to the newest proprietary technologies such as transformer mechanisms, they do offer free versions of these software tools for scientific experimentation. In this way, four pre-trained transformer models, which were originally developed for natural language processing and content generation (generative artificial intelligence), are now being applied to a classification problem, are they: Bidirectional Encoder Representations from Transformers (BERT) [40], Robustly Optimized BERT Approach (RoBERTa) [41], generalized autoregressive pretraining method XLNet [42], and Generative Pre-trained Transformer 2 (GPT-2) [43]. BERT works bidirectionally for pre-training over an unlabeled dataset to learn a language representation in natural language processing that can be used to fine-tune specific machine learning tasks, such as next structure prediction. RoBERTa is a variant of BERT that uses a dynamic masking technique to improve training performance and was trained on a much larger dataset than BERT. XLNet is also a variant of BERT that enables the learning of bidirectional contexts by an autoregressive formulation that maximizes the expected likelihood over all permutations of the factorization order, and GPT-2, achieves better performance in our experiments. It is a generational transformer (GPT-1, 2, 3, and 4) that enables massive parallelization to perform various tasks because of its general ability to accurately predict the next item in a sequence.

5.3 Model Evaluation

After training X-Model4Rec, the next item for each user is predicted and the evaluation of the recommendation is computed. The following information retrieval metrics are used to calculate the *top@k* accuracy of recommendation lists containing k items [5–7]. The results produced as outputs in Algorithm 2 were measured considering the average score among all users, by the following metrics:

Precision-at-k (*P@k*) It is the most intuitive metric. It calculates the proportion of the *top@k* recommendation that includes items $rank_j$ in the test set τ (expected next item, $|\tau| = 1$) for each user.

$$P@k = \frac{1}{k}\sum_{j=1}^{k}\begin{cases} 1, & rank_j \in \tau \\ 0, & otherwise \end{cases} \tag{3}$$

Recall-at-k (*R@k*) Measures what proportion of the test set τ was recovered from the recommended items for each user:

$$R@k = \frac{1}{|\tau|}\sum_{j=1}^{k}\begin{cases} 1, & rank_j \in \tau \\ 0, & otherwise \end{cases} \tag{4}$$

F1-Score-at-k (*F1-Score@k* or *F1@k*) Measures a balance between precision and recall:

$$F1 - Score@k = 2\frac{P@k \times R@k}{P@k + R@k} \tag{5}$$

Mean Reciprocal Rank-at-k (*MRR@k*) Measures the rank of the first relevant item (rank) found in a *top@k* ranked list:

$$MRR@k = \frac{1}{|\mathrm{U}|}\sum_{j=1}^{|\mathrm{U}|} RR@k_j, \quad where \quad RR@k = \begin{cases} \frac{1}{\min j}, & \mathrm{rank}_{j=1..k} \in \tau \\ 0, & otherwise \end{cases} \tag{6}$$

Mean Average Precision-at-k (*MAP@k*) Calculates the average accuracy at each position in the *top@k* sorted list. The average precision is defined as the number of relevant items in the list up to a certain position divided by the total number of items in the list up to that position:

$$MAP@k = \frac{1}{|\mathrm{U}|}\sum_{j=1}^{|\mathrm{U}|} AP@k_j, where\ AP@k = \frac{1}{|\tau|}\sum_{j=1}^{k}\begin{cases} P@j, & \mathrm{rank}_j \in \tau \\ 0, & otherwise \end{cases} \tag{7}$$

Normalized Discounted Cumulative Gain-at-k (*NDCG@k*) Measures the quality of a *top@k* list returned by the model for the items that each user indicated as relevant. The positions in the *top@k* are used to estimate the relevance in which the first items in the ranking receive the most credit, decreasing until the final position:

$$NDCG@k = \frac{1}{|U|}\sum_{j=1}^{|U|}\left(\frac{DCG@k}{IDCG@k}\right)_j, \ where \ DCG@k = \sum_{j=1}^{k\ (actual\ order)}\frac{2^{rel_j}-1}{\log_2(j+1)} \ and \ IDCG@k = \sum_{j=1}^{k\ (ideal\ order)}\frac{2^{rel_j}-1}{\log_2(j+1)} \tag{8}$$

Algorithm 3 processes the model evaluation using the above metrics as described.

Algorithm 3 Testing

Input: Set of recommendations *top@k*, where $k = \{1, 10, 20, 30, 40, 50, ...\}$ and testing set (*testFlow*)
Output: Metric scores

1. For each user u from U, do:
2. Check if the recommended items in *top@k* match the test set τ (target)
3. Compute the *hit* and position at k
4. Calculate Precision, Recall, and F1-Score as rank-less metrics via equations (3-5)
5. Calculate MRR, MAP, and NDCG as rank-award metrics via equations (6-8)

The Area Under the Receiver Operating Characteristic (ROC) Curve (AUC) metric was used during training and in model validation. It offers an aggregate measure of performance across all possible classification limits, calculating the performance between the true positive rate (TPR) and the false positive rate (FPR):

$$AUC = \int_0^1 TPR(FPR)\, dFPR, \ where \ TPR = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \ and \ FPR = \frac{\text{false positives}}{\text{false positives} + \text{true negatives}} \tag{9}$$

Finally, as illustrated in Figs. 2 and 4, it is important to highlight that the *trainFlow* and *testFlow* sets are produced as the output of Algorithm 1. Each one is used separately, with the *trainFlow* set only used in Algorithm 2 and the *testFlow* set only used in Algorithm 3.

6 Experiments and Results

6.1 Datasets

Currently, there is a large reference movie dataset from GroupLens [10], electronic items on e-commerce from Taobao [44], fashion products from Amazon [45], and books and ratings from Book-Crossing [46]. These and other datasets may be found in scientific publications, but few references have been found to the wine domain for recommender systems, which we are interested in investigating. Some wine datasets may be found in repositories such as Kaggle Datasets (https://www.kaggle.com/datasets, last accessed 2023/10/15) and GitHub Data Packaged Core Datasets (https://github.com/datasets, last accessed 2023/10/15); however, they present a scarcity of relevant data or are organized without the necessary rigor for wider use.

Table 2 shows the details of the original datasets chosen for experimentation. The first dataset was produced as part of our research called X-Wines [9]. It is a large and consistent wine dataset consisting of 100,646 wine labels and 21 million 5-star ratings assigned by 1 million users. The public dataset can be accessed in the official repository at https://github.com/rogerioxavier/X-Wines (last accessed 2023/12/13).

The traditional movie dataset MovieLens [10] from the GroupLens Research Laboratory at the University of Minnesota (https://grouplens.org/datasets/movielens, last accessed 2023/12/13) was also used for X-Model4Rec assessment. Their repository has millions of movie reviews by real users and offers these datasets systematically in a range of sizes. It is one of the well-known datasets used in scientific experimentation.

Table 2 Datasets used in the experiments

Dataset	# Items	# Users	# Ratings	# Categories	Density
X-Wines version Slim	1,007	10,561	150,000	6 types	14.2
MovieLens version 1 M	3,900	6,040	1,000,209	18 genres	165.6

6.2 Experiments

Various experiments were conducted to test and evaluate the performance of the X-Model4Rec recommender. The algorithms were implemented with Python 3.10.10, TensorFlow 2.10.0, Pandas 2.0.1, Numpy 1.24.3, Transformers 4.31.0, NVIDIA Merlin-Models 23.6.0 and NVTabular 23.6.0 on Debian GNU/Linux version 11 run on a computer i7-10750H 6-core CPU @ 2.60 GHz with an integrated 10th Gen Intel UHD GPU and NVIDIA GeForce GTX 1650 GPU, NVMe M.2 SSD and 16 GB of RAM.

The three experiments presented here aimed to keep the parameter values as close as possible to those used in the literature reference to ensure fair comparisons. Therefore, the default hyperparameters of each tested model were maintained. Algorithm 1 was built to be as similar as possible to the case of single sessions with a length of 2–10 interactions. This approach allowed greater confidence in the short- and long-term session samples. To ensure a positive rating, the threshold was set to 3 in the 5-stars classification, meaning that ratings equal to or above 3 were considered positive, while anything below was considered negative. Two main challenges were faced during our experiments – a lack of information and the risk of overfitting that can occur between training and testing in DNNs. To address these concerns, only users who had performed at least 10 evaluations and items with 8 or more evaluations in the datasets were considered. Wine type and movie genres were also used as item categories in the models that required them. A detailed breakdown of the filtered datasets used in our experiments is provided in Table 3.

The hyperparameters used in the three experiments described below were as follows:

$EMBEDDING\_DIM = 64$, $NEURONS = [128, 64]$,
$HEADS = 8$, $DROPOUT = 0.1$,
$BATCH\_SIZE\_TRAINING = 256$,
$BATCH\_SIZE\_TEST = 1024$,
$OPTIMIZER = Adam\ (learning\_rate = 0.005)$,
$LOSS = CategoricalCrossEntropy\ (from\_logits = True, label\_smoothing = 0.2)$, and
$TEMPERATURE\_SCALING = 1.0$.

Table 3 Filtered datasets used in the experiments

Filtered Dataset	$\|I\|$	$\|U\|$	$\|R\| \in C$	Categories	Density
X-Wines_Slim	894	8,369	133,278	6 types	15.93
MovieLens_1M	3,319	6,040	992,997	18 genres	164.40

6.2.1 X-Model4Rec's Performance Assessment with Various Transformer Mechanisms

In the first experiment, the main goal was to evaluate the scalability and extensibility of the proposed X-Model4Rec. The model was designed in blocks to make it more flexible, allowing for architectural variations while maintaining the dimensionality of the data embeddings. Both filtered datasets were trained for 50 epochs using the same hyperparameters presented above. However, each run was executed with a different pre-trained transformer block, such as BERT, RoBERTa, XLNet, and GPT-2. Figure 6 shows the partial monitoring during the training process by running Algorithm 2.

The metrics presented in Section 5.3 are automatically calculated by functions available in the NVIDIA Merlin-Models framework. This allows us to interpret situations that led us to some parameters of continuity in the development of the X-Model4Rec model, among which the verification that the extensible architecture was successful in allowing the transformer block to be changed, an internal block that performs most of the data processing. The area under the ROC curve (AUC) metric represents an important indicator for assessing the performance of classification. It helps evaluate how well a monitored model can classify between desired positives and negatives by measuring the performance of correct positive predictions versus the false positive rate given by Eq. (9). Due to the stability presented by the AUC metric, illustrated in Fig. 6a and b, transformer GPT-2 was chosen as the best mechanism, and we will continue all the other experiments described here using only GPT-2 as a resource applied in the transformer block. The model execution responded positively on two different data volumes tested, allowing greater scalability in future experiments. During the training of the X-Model4Rec using the GPT-2 transformer, Fig. 6c and d were generated to demonstrate the obtained results for the upcoming experiments. The performance after seven epochs is proportional to the density from two samples used.

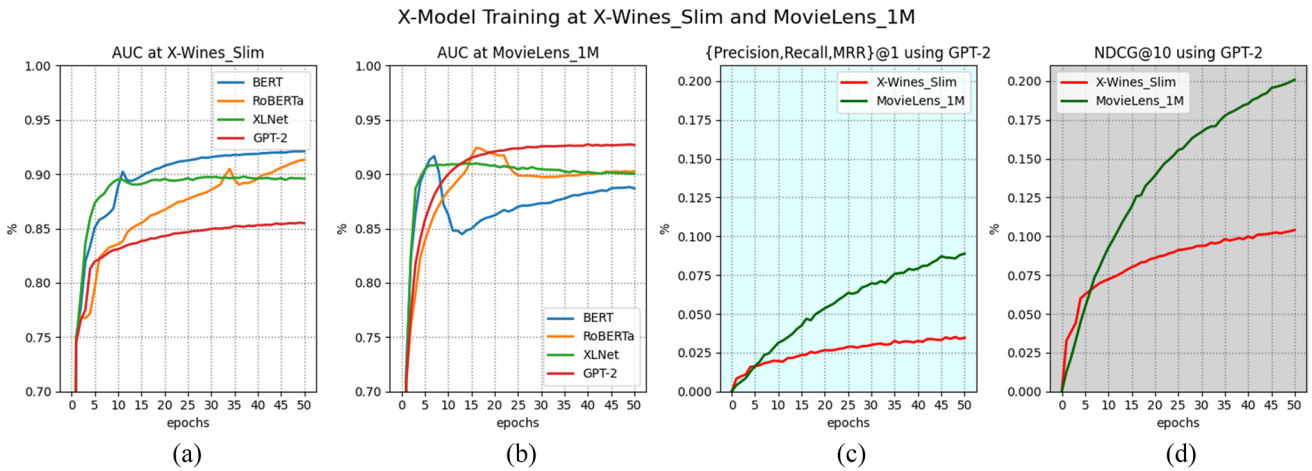

Fig. 6 Metrics AUC on different transformers, Precision, Recall, MRR, and NDCG obtained when training the X-Model4Rec

6.2.2 X-Model4Rec Compared to Classical Methodologies

In the second experiment, we aimed to calculate the metrics that are commonly used to evaluate the *top@k* recommendations generated by the X-Model4Rec. Our objective was to compare the performance of X-Model4Rec with that of other classic recommender models cited in the literature reference.

In the study on recommender systems, an extensive research about various approaches was performed to identify the most effective models. Although all the approaches could not be covered, several classic models were tested; these include Collaborative Filtering, Content-Based Filtering, and Hybrid Filtering through different implementations, as listed in the legend of Fig. 7. The models selected for comparison in the second experiment are well-known and widely used worldwide.

Each model adheres to specific principles and offers some structural variations, such as singular value decomposition (SVD) algorithms and their variants SVD++or SVDpp, similarity-based algorithms that use different distance measures, such as Cosine, Euclidean, Haversine, Manhattan, and short or longer neighborhoods. These models do not rely on random draws but use advanced information filtering techniques. For comparison purposes, three models that use random draws were included and are identified by the prefix 'Simple' in the legend of Fig. 7. These models generate *top@k* recommendations based on a few elements, such as the items that the user has not interacted with in the past or the most popular items among other users. We used the same hyperparameters as in the previous experiment, but in this run, the model was trained for only 20 epochs.

The results obtained after executing Algorithms 1, 2, and 3 on two filtered datasets from different domains described in Table 3, have shown that the X-Model4Rec produced in our research, outperformed all the other models evaluated here, as shown by the green line on the graphs. In some cases, it produced much better results than did the recommendations generated by other models for the wine and movie samples evaluated. There is efficiency and consistency in the results produced by our model, even when low scores on average across all users are observed that negatively affect the F1-Score due to $P@k \ll R@k$ and make MAP performance similar to MRR. These situations are typically encountered in the next-item recommendation problem where the test set size is $|\tau| = 1$, because only one target item is desired by each user when compared to the *top@k* recommendation generated. At most, one relevant item can be found and not several items, as in other problems. It was also observed that classical approaches are more effective than random approaches, which was expected.

6.2.3 X-Model4Rec Compared to Baselines Models

In the third experiment, we compared the results measured by the AUC, precision and recall evaluation metrics from X-Model4Rec with four baseline models using the X-Wines_Slim and MovieLens_1M samples described in Table 3 as input. The models were chosen due to similarities, between more traditional models, and models that consider both short- and long-term sequential in their implementation. The four baseline models are as follows:

- ATRank [47] considers the user's heterogeneous behaviors using only the attention model (attention multi-head). It makes use of the DNN in the self-attention layer and vanilla attention layer to obtain users' preferences;
- The named here Bias-LSTM [37] is a bidirectional short- and long-term memory network for capturing implicit sequential user data through multiple LSTM gates;
- BPR-MF [48] is a traditional model that uses a Bayesian Personalized Ranking with matrix factorization (MF)

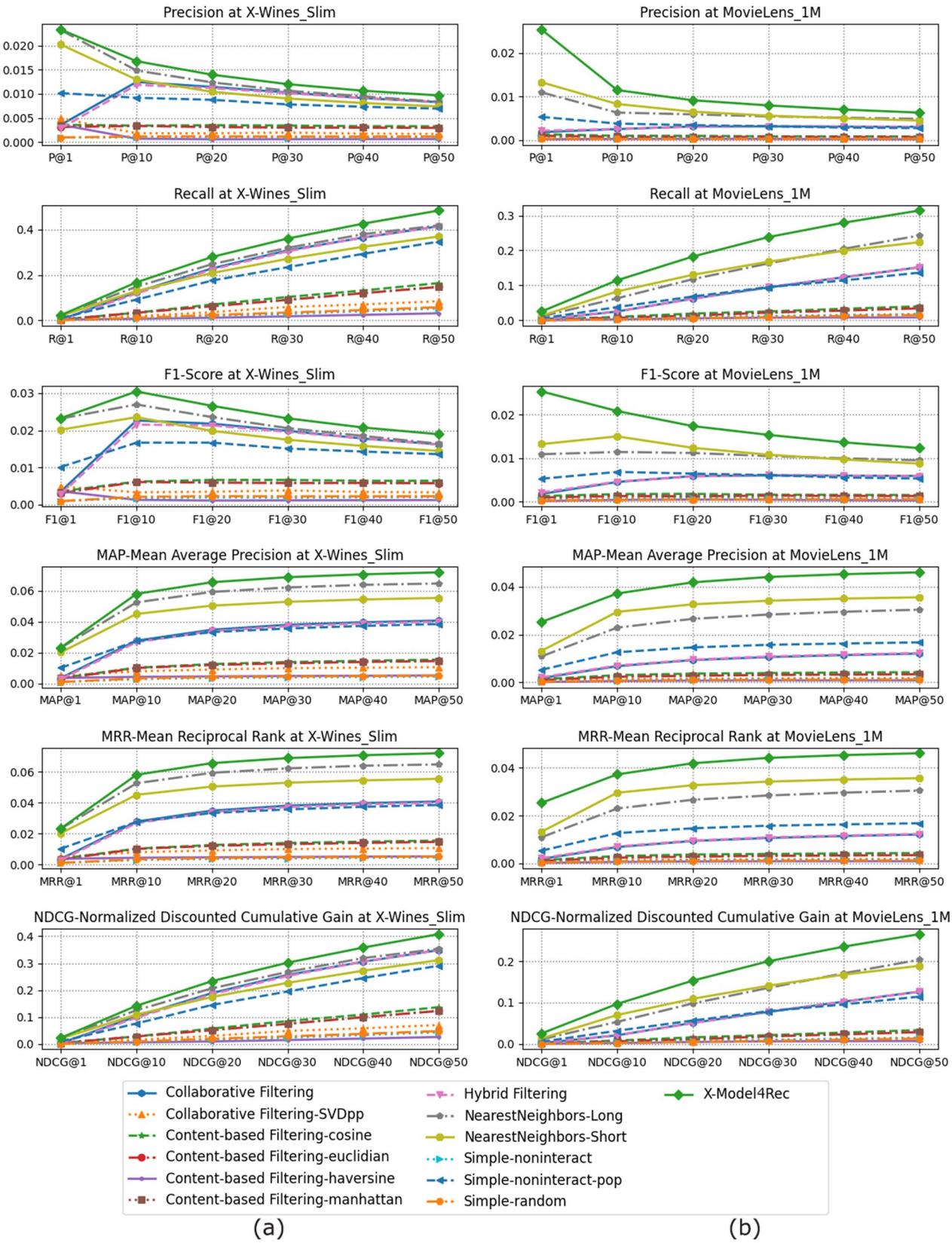

Fig. 7 Metrics Precision, Recall, F1-Score, MAP, MRR, and NDCG obtained when evaluating the X-Model4Rec and classic models

Table 4 AUC, Precision, and Recall on the X-Wines_Slim filtered dataset

Model	AUC	Precision@						Recall@					
		1	10	20	30	40	50	1	10	20	30	40	50
ATRank	**0.8859**	0.0180	0.0143	0.0123	0.0108	0.0098	0.0089	0.0180	0.1430	0.2452	0.3253	0.3906	0.4453
Bias-LSTM	0.8830	0.0175	0.0143	0.0122	0.0108	0.0098	0.0089	0.0175	0.1427	0.2443	0.3244	0.3905	0.4463
BPR-MF	0.8161	0.0081	0.0075	0.0070	0.0066	0.0063	0.0061	0.0081	0.0751	0.1391	0.1974	0.2534	0.3063
LSPM	0.8045	0.0086	0.0073	0.0067	0.0063	0.0060	0.0057	0.0086	0.0731	0.1339	0.1898	0.2398	0.2856
X-Model4Rec	0.8444	**0.0260**	**0.0177**	**0.0144**	**0.0122**	**0.0107**	**0.0097**	**0.0260**	**0.1773**	**0.2889**	**0.3669**	**0.4299**	**0.4842**
%	-4.68	+44.44	+23.78	+17.07	+12.96	+9.18	+8.99	+44.44	+23.99	+17.82	+12.79	+10.06	+8.74

Table 5 AUC, Precision, and Recall on the MovieLens_1M filtered dataset

Model	AUC	Precision@						Recall@					
		1	10	20	30	40	50	1	10	20	30	40	50
ATRank	0.9118	0.0127	0.0094	0.0080	0.0071	0.0065	0.0059	0.0127	0.0942	0.1601	0.2131	0.2581	0.2965
Bias-LSTM	**0.9240**	0.0091	0.0097	0.0089	0.0081	0.0075	0.0069	0.0091	0.0966	0.1776	0.2439	0.2988	0.3452
BPR-MF	0.8225	0.0032	0.0024	0.0023	0.0022	0.0021	0.0022	0.0032	0.0237	0.0464	0.0673	0.0859	0.1096
LSPM	0.8881	0.0065	0.0050	0.0045	0.0042	0.0039	0.0037	0.0065	0.0505	0.0905	0.1253	0.1562	0.1840
X-Model4Rec	0.8914	**0.0253**	**0.0150**	**0.0110**	**0.0091**	**0.0078**	**0.0070**	**0.0253**	**0.1505**	**0.2215**	**0.2715**	**0.3124**	**0.3478**
%	-3.53	+99.22	+54.64	+23.60	+12.35	+4.00	+1.45	+99.22	+55.80	+24.72	+11.32	+4.55	+0.75

method to train positive and negative user ratings and embedding concatenations of item IDs + categories;
- The Long- and Short-Term Preference Model (LSPM) [14] uses a trainable matrix to capture long-term user preferences and combines each short-term user preferences to obtain users' profiles to generate personalized online recommendations.

The same hyperparameters as in the previous experiment were used, all the models were trained for 20 epochs, and the pre-trained transformer block GPT-2 was used in X-Model4Rec. Tables 4 and 5 present the results obtained after executing each compared model. The bold font and underlined values indicate better results in each column.

The experiment produced impressive results for the next-item recommendation problem using wine and movie samples. The results surpassed even those of known models. Tables 4 and 5 show a percentage comparison of the X-Model4Rec results to the best results obtained by the other methods presented in each column, and the last line demonstrates the significant improvement. The multi-head attention and GPT-2 transformer mechanisms used focused on the next element of the shifted sequential input list, which resulted in good outcomes. However, to further enhance the proposed X-Model4Rec model, further experiments will be conducted, including bias adjustment, a revised setup, and an internal check architecture. The runtimes of each model were excluded from this report, as some implementations in the literature require different versions of Python and libraries, running in different virtual environments. It can be stated that under the same conditions, X-Model4Rec is trained faster than the other verified models, obtaining the following times in each input sample:

- X-Wines_Slim – preprocessing in 12.7 s and training and testing in 301.5 s.
- MovieLens_1M – preprocessing in 28.5 s and training and testing in 574.1 s.

With these runtimes, this approach ensures a non-prohibitive computational time for generating recommendations. Finally, the main scores obtained by the evaluation metrics to provide a basis for further implementation were retained. These data, along with the source codes used in the experiments, are available at https://github.com/rogerioxavier/X-Model4Rec (last accessed 2024/01/26).

7 Conclusion

Sequential Recommendation Systems (SRSs) are receiving increased research attention for improve the relationship between users' short- and long-term interests, contextual information, and dynamic preferences via Deep Neural Network (DNN) architectures.

In this paper, the X-Model4Rec – eXtensible Model for Recommendation was presented, proposing a modeling to explore the user's dynamic taste profile in sequential single sessions with a defined length, as a contribution to the next-item recommendation problem. The proposed model demonstrated the use of a generative technology designed to create content that was successfully applied in this research to a classification problem making predictions in classes (items).

Our results show that a set of bidirectional transformers using improved training methodology can achieve better prediction metrics than previous models. X-Model4Rec is designed to be an extensible recommender model that allows the exchange of architectural blocks, if necessary, to direct the model's attention to the next item from the data input sequence. This approach proved to be a highly effective mechanism for the proposed model to adapt to the input data, improving the results.

X-Model4Rec has shown promising results, outperforming classic and baseline methods in terms of prediction scores. The best value obtained indicates the good performance of X-Model4Rec for the next-item recommendation problem on two real-world transaction datasets about specific products (wines and movies) across different domains, sizes, and data densities. Nonetheless, we will continue to improve the model by experimenting with new setups and architectural modeling to obtain even better results. In our research, we utilized proprietary transformer mechanisms such as BERT and GPT-2, which are available in free versions; however, new experiments can be conducted with improved versions. We encountered limitations due to insufficient user sequence data for SRS training. The X-Wines dataset and the X-Model4Rec recommender are being used in practice on our research project's collaborative Web platform to recommend wines to actual users. In the future, it is planned to release new findings, including evaluations from real users of this platform, should be released, as well as new evaluations based on standard metrics used in recommender systems.

Finally, this paper contributes to the study of adaptive recommendations to mitigate information overload in online environments under development by the authors [49]. Some challenges that need to be addressed in SRSs research have been presented, but we believe it is possible to specialize in existing models and create entirely innovative models that can further explore feedback to obtain better results.

Abbreviations ADER: Adaptively Distilled Exemplar Replay recommender model; AUC: Area under the Receiver Operating Characteristic (ROC) Curve; BERT: Bidirectional Encoder Representations from Transformers; BERT4Rec: Bidirectional Encoder Representations from Transformer for sequential Recommendation; Bias-LSTM: Long Short-Term Memory bias optimized model; BPR-MF: Bayesian Personalized Ranking—Matrix Factorization model; Caser: Convolutional Sequence Embedding Recommendation model; CNN: Convolutional Neural Network; DNN: Deep Neural Network; DTP: Dynamic Taste Profile; FFN: Feed-Forward Network; FPR: False Positive Rate; GNN: Graph Neural Network; GPT: Generative Pre-trained Transformer; GRU4Rec: Gated Recurrent Unit for Recommendation; LSPM: Long- and Short-Term Preference Model; LSTM: Long Short-Term Memory; MAP: Mean Average Precision; MLP: Multilayer Perceptron; MRR: Mean Reciprocal Rank; NDCG: Normalized Discounted Cumulative Gain; NLP: Natural Language Processing; RNN: Recurrent Neural Network; RoBERTa: Robustly Optimized BERT Approach; RS: Recommender System; SASRec: Self-Attention based Sequential Recommendation model; SBRec: Session-based recommendation; SRec: Sequential recommendation; SRS: Sequential Recommender System; SVD: Singular Value Decomposition; TPR: True Positive Rate; X-Model4Rec: EXtensible Model for Recommendation

Acknowledgements This research has been supported by the Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS) – Brazil.

Authors' contributions Conceptualization and analysis, R.X.d.A., A.J.M. and V.F.; methodology, software engineering, and programming, R.X.d.A.; writing—original draft preparation, R.X.d.A.; writing—review and editing, R.X.d.A., A.J.M. and V.F.; supervision, A.J.M. and V.F. All authors have read and agreed to the published version of the manuscript.

Funding This research received no external funding.

Data Availability The experimental data and the simulation results that support the findings of this paper are available online at https://github.com/rogerioxavier/X-Model4Rec (last accessed 2024/01/26).

Declarations

Competing interests Not applicable.

References

1. Adomavicius G, Tuzhilin A. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-art and Possible Extensions. IEEE Trans Knowl Data Eng. 2005;17:734–49. https://doi.org/10.1109/TKDE.2005.99.
2. Shao B, Li X, Bian G. A Survey of Research Hotspots and Frontier Trends of Recommendation Systems from the Perspective of Knowledge Graph. Expert Syst Appl. 2021;165: 113764. https://doi.org/10.1016/j.eswa.2020.113764.
3. Zhang Q, Lu J, Jin Y. Artificial Intelligence in Recommender Systems. Complex & Intelligent Systems. 2021;7:439–57. https://doi.org/10.1007/s40747-020-00212-w.

4. Zhang S, Yao L, Sun A, Tay Y. Deep Learning Based Recommender System: A Survey and New Perspectives. ACM Comput Surv. 2019;52:1–38. https://doi.org/10.1145/3285029.
5. Shani, G., Gunawardana, A. Evaluating Recommendation Systems. In: Ricci, F., Rokach, L., Shapira, B., and Kantor, P.B. (eds.) Recommender Systems Handbook. pp. 257–297. Springer US, Boston, MA 2010. https://doi.org/10.1007/978-1-4899-7637-6_8.
6. Ludewig M, Jannach D. Evaluation of Session-Based Recommendation Algorithms UMUAI. 2018;28:331–90. https://doi.org/10.1007/s11257-018-9209-6.
7. Fang, H., Zhang, D., Shu, Y., Guo, G. Deep Learning for Sequential Recommendation: Algorithms, Influential Factors, and Evaluations. ACM Trans Inf Syst. 39, 2020. https://doi.org/10.1145/3426723.
8. Quadrana, M., Cremonesi, P., Jannach, D. Sequence-Aware Recommender Systems. ACM Comput Surv. 51, (2018). https://doi.org/10.1145/3190616.
9. de Azambuja, R.X., Morais, A.J., Filipe, V. X-Wines: A Wine Dataset for Recommender Systems and Machine Learning. Big Data and Cognitive Computing. 7, 2023. https://doi.org/10.3390/bdcc7010020.
10. Harper FM, Konstan JA. The MovieLens Datasets: History and Context. ACM Trans Interact Intell Syst. 2016;5:1–19. https://doi.org/10.1145/2827872.
11. Covington, P., Adams, J., Sargin, E. Deep Neural Networks for YouTube Recommendations. In: Proceedings of the 10th ACM Conference on Recommender Systems. pp. 191–198. Association for Computing Machinery, New York, NY, USA 2016. https://doi.org/10.1145/2959100.2959190.
12. Singh PK, Pramanik PKD, Dey AK, Choudhury P. Recommender Systems an overview, research trends, and future directions. Int J Business Syst Res (IJBSR). 2021;15(1):14–52. https://doi.org/10.1504/IJBSR.2021.111753.
13. Zheng Y, Wang D. (Xuejun): A survey of recommender systems with multi-objective optimization. Neurocomputing. 2022;474:141–53. https://doi.org/10.1016/j.neucom.2021.11.041.
14. Du, Y., Liu, H., Qu, Y., Wu, Z. Online Personalized Next-Item Recommendation via Long Short Term Preference Learning. In: Geng, X. and Kang, B.-H. (eds.) PRICAI 2018: Trends in Artificial Intelligence. pp. 915–927. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-97304-3_70.
15. Song, W., Wang, S., Wang, Y., Wang, S. Next-Item Recommendations in Short Sessions. In: Proceedings of the 15th ACM Conference on Recommender Systems. pp. 282–291. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3460231.3474238.
16. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D. Session-based Recommendations with Recurrent Neural Networks. In: ICLR 2016 (Poster). San Juan, Puerto Rico, arXiv preprint arXiv: 1511.06939. (2016). https://arxiv.org/abs/1511.06939.
17. Kim J, Wi J, Kim Y. Sequential Recommendations on GitHub Repository. Journal Applied Sciences. 2021;11:14. https://doi.org/10.3390/app11041585.
18. Mi, F., Lin, X., Faltings, B. ADER: Adaptively Distilled Exemplar Replay Towards Continual Learning for Session-based Recommendation. In: ACM RecSys'20. Virtual Event, Brazil. pp. 408–413 (2020). https://doi.org/10.1145/3383313.3412218.
19. Gharahighehi A, Vens C. Personalizing Diversity Versus Accuracy in Session-Based Recommender Systems. SN Comput Sci. 2021;2:39. https://doi.org/10.1007/s42979-020-00399-2.
20. Tang, J., Wang, K. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In: ACM WSDM'18. LA. USA. pp. 565–573 (2018). https://doi.org/10.1145/3159652.3159656.
21. Martinez AD, Del Ser J, Villar-Rodriguez E, Osaba E, Poyatos J, Tabik S, Molina D, Herrera F. Lights and shadows in Evolutionary Deep Learning. Information Fusion. 2021;67:161–94. https://doi.org/10.1016/j.inffus.2020.10.014.
22. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I. Attention Is All You Need, Attention is All You Need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. p. 6000–6010. Curran Associates Inc., Red Hook, NY, USA, 2017, arXiv preprint arXiv: 1706.03762. 2023. https://doi.org/10.48550/arXiv.1706.03762.
23. Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., Jiang, P. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In: ACM CIKM '19. Beijing, China. pp. 1441–1450 2019. https://doi.org/10.1145/3357384.3357895.
24. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In: Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work. pp. 175–186. Association for Computing Machinery, New York, NY, USA 1994. https://doi.org/10.1145/192844.192905.
25. Dokoohaki, N. ed Fashion Recommender Systems. In: Fashion Recommender Systems. pp. 3–21. Springer International Publishing, Cham 2020. https://doi.org/10.1007/978-3-030-55218-3_1.
26. Dara S, Chowdary CR, Kumar C. A Survey on Group Recommender Systems. J Intell Inf Syst. 2020;54:271–95. https://doi.org/10.1007/s10844-018-0542-3.
27. Felfernig, A., Boratto, L., Stettinger, M., Tkalčič, M. Group Recommender Systems : An Introduction. Springer International Publishing 2018https://doi.org/10.1007/978-3-319-75067-5.
28. Nguyen, L.V., Vo, Q.-T., Nguyen, T.-H. Adaptive KNN-Based Extended Collaborative Filtering Recommendation Services. Big Data and Cognitive Computing. 7, 2023. https://doi.org/10.3390/bdcc7020106.
29. Nguyen, L.V., Nguyen, T.-H., Pham, H.-T.-N., Vo, Q.-T., Duong, H.-T., Nguyen-Thi, T.-A. Bio-Inspired Clustering: An Ensemble Method for User-Based Collaborative Filtering. In: Dao, N.-N., Thinh, T.N., and Nguyen, N.T. (eds.) Intelligence of Things: Technologies and Applications. pp. 26–35. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-46573-4_3.
30. Wang, S., Hu, L., Wang, Y., Cao, L., Sheng, Q.Z., Orgun, M. Sequential Recommender Systems: Challenges, Progress and Prospects. In: International Joint Conference on Artificial Intelligence, IJCAI-19. pp. 6332–6338 2019. https://doi.org/10.24963/ijcai.2019/883.
31. Wu S, Tang Y, Zhu Y, Wang L, Xie X, Tan T. Session-Based Recommendation with Graph Neural Networks. AAAI. 2019;33:346–53. https://doi.org/10.1609/aaai.v33i01.3301346.
32. Kang, W.-C., McAuley, J. Self-Attentive Sequential Recommendation. In: IEEE ICDM'18. pp. 197–206. Singapore 2018. https://doi.org/10.1109/ICDM.2018.00035.
33. ACM RecSys Conferences Series on Recommender Systems. https://recsys.acm.org. Accessed 2023/12/15.
34. Eberle, O., Büttner, J., Kräutli, F., Müller, K.-R., Valleriani, M., Montavon, G. Building and Interpreting Deep Similarity Models. IEEE Trans Pattern Anal Mach Intell. 1–1 2020. https://doi.org/10.1109/TPAMI.2020.3020738.
35. Hiriyannaiah, S., Siddesh, G.M., Srinivasa, K.G. Deep visual ensemble similarity (DVESM) approach for visually aware recommendation and search in smart communityJ King Saud

Univ Comput Inf Sci.https://doi.org/10.1016/j.jksuci.2020.03.009.
36. Wang S, Cao L, Wang Y, Sheng QZ, Orgun M, Lian D. A Survey on Session-based Recommender Systems. ACM Comput Surv. 2021;54(7):1–38. https://doi.org/10.1145/3465401.
37. Jozefowicz, R., Zaremba, W., Sutskever, I. An Empirical Exploration of Recurrent Network Architectures. In: Bach, F. and Blei, D. (eds.) Proceedings of the 32nd International Conference on Machine Learning. pp. 2342–2350. PMLR, Lille, France 2015. https://proceedings.mlr.press/v37/jozefowicz15.html, last accessed 2023/12/15.
38. NVIDIA AI Merlin Project, https://developer.nvidia.com/merlin, last accessed 2023/12/15.
39. Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q. On Calibration of Modern Neural Networks, In: Proceedings of the 34th International Conference on Machine Learning - Volume 70. p. 1321–1330. JMLR.org, Sydney, NSW, Australia 2017. https://doi.org/10.5555/3305381.3305518.
40. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Presented at the Proceedings of NAACL-HLT 2019, Minneapolis, Minnesota May 24 2019. https://doi.org/10.18653/v1/N19-1423.
41. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach, arXiv preprint arXiv:1907.11692 2019. https://doi.org/10.48550/arXiv.1907.11692.
42. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., Le, Q.V. XLNet: Generalized Autoregressive Pretraining for Language Understanding, arXiv preprint arXiv:1906.08237 2020. https://doi.org/10.48550/arXiv.1906.08237.
43. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I. Language Models are Unsupervised Multitask Learners. OpenAI. San Francisco, California, United States, 2019. https://api.semanticscholar.org/CorpusID:160025533, last accessed 2023/12/15.
44. Tianchi Taobao Dataset 淘宝网(淘寶網), https://tianchi.aliyun.com/datalab/dataSet.html?dataId=649, last accessed 2022/12/28.
45. He, R., McAuley, J. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In: Proceedings of the 25th International Conference on World Wide Web. pp. 507–517. International World Wide Web Conferences Steering Committee, Montréal Québec Canada 2016. https://doi.org/10.1145/2872427.2883037.
46. Ziegler, C.-N., McNee, S.M., Konstan, J.A., Lausen, G. Improving recommendation lists through topic diversification. In: Proceedings of the 14th international conference on World Wide Web - WWW '05. p. 22. ACM Press, Chiba, Japan 2005. https://doi.org/10.1145/1060745.1060754.
47. Zhou, C., Bai, J., Song, J., Liu, X., Zhao, Z., Chen, X., Gao, J. ATRank: An Attention-Based User Behavior Modeling Framework for Recommendation. AAAI. 32, 2018. https://doi.org/10.1609/aaai.v32i1.11618.
48. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L. BPR: Bayesian Personalized Ranking from Implicit Feedback. Presented at the Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Arlington, Virginia, USA 2009. https://doi.org/10.5555/1795114.1795167.
49. de Azambuja, R.X., Morais, A.J., Filipe, V. Adaptive Recommendation in Online Environments. In: González, S.R., Machado, J.M., González-Briones, A., Wikarek, J., Loukanova, R., Katranas, G., and Casado-Vara, R. (eds.) Distributed Computing and Artificial Intelligence, Volume 2: Special Sessions 18th International Conference. pp. 185–189. Springer International Publishing, Cham 2022. https://doi.org/10.1007/978-3-030-86887-1_17.