11-1-2023

A Survey and Taxonomy of Sequential Recommender Systems for E-commerce Product Recommendation

Mahreen Nasir
University of Windsor

C. I. Ezeife
University of Windsor

SN Computer Science (2023) 4:708
https://doi.org/10.1007/s42979-023-02166-5

SURVEY ARTICLE

A Survey and Taxonomy of Sequential Recommender Systems for E-commerce Product Recommendation

Mahreen Nasir[1] · **C. I. Ezeife**[1]

Received: 18 November 2021 / Accepted: 19 July 2023

Abstract
E-commerce recommendation systems facilitate customers' purchase decision by recommending products or services of interest (e.g., Amazon). Designing a recommender system tailored toward an individual customer's need is crucial for retailers to increase revenue and retain customers' loyalty. As users' interests and preferences change with time, the time stamp of a user interaction (click, view or purchase event) is an important characteristic to learn sequential patterns from these user interactions and, hence, understand users' long- and short-term preferences to predict the next item(s) for recommendation. This paper presents a taxonomy of sequential recommendation systems (SRecSys) with a focus on e-commerce product recommendation as an application and classifies SRecSys under three main categories as: (i) traditional approaches (sequence similarity, frequent pattern mining and sequential pattern mining), (ii) factorization and latent representation (matrix factorization and Markov models) and (iii) neural network-based approaches (deep neural networks, advanced models). This classification contributes towards enhancing the understanding of existing SRecSys in the literature with the application domain of e-commerce product recommendation and provides current status of the solutions available alongwith future research directions. Furthermore, a classification of surveyed systems according to eight important key features supported by the techniques along with their limitations is also presented. A comparative performance analysis of the presented SRecSys based on experiments performed on e-commerce data sets (Amazon and Online Retail) showed that integrating sequential purchase patterns into the recommendation process and modeling users' sequential behavior improves the quality of recommendations.

Keywords Recommendation systems · Sequential pattern mining · Clickstream data · Historical purchases · Recsys · Collaborative filtering · E-commerce

Introduction

E-commerce Recommendation Systems

A recommendation system (RS) provides suggestions to users according to their interests [1, 2]. For example, many companies and service providers such as Amazon, [1] Facebook, [2] Netflix, [3] and career builders[4] provide recommendations to users for products, friends, movies, and jobs, respectively [3]. E-commerce recommendation systems assist customers in decision making by recommending items that are likely to be in their interest [4]. They facilitate the customers' search process and narrow down the choices out of thousands of available products to save customers' time. Top tech e-commerce companies (e.g., Amazon, Alibaba) deploy customized recommendation systems to cater to their growing customers' needs and maintain their market share.

One of the most widely used recommendation techniques is *collaborative filtering (CF)* which recommend users' products that they might be interested in based on the taste of other similar users [5]. In CF, customer's interest in a product is saved in a user–item rating matrix. For example, we have a list of m users $U = \{u_1, u_2, ..., u_m\}$ and n

✉ Mahreen Nasir
nasir11d@uwindsor.ca

C. I. Ezeife
cezeife@uwindsor.ca

[1] School of Computer Science, University of Windsor, 401 Sunset Avenue, Windsor, ON N9B 3A6, Canada

[1] https://www.amazon.com/.
[2] https://www.facebook.com/.
[3] https://www.netflix.com/ca/.
[4] https://www.careerbuilder.ca/.

Published online: 15 September 2023

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	1	1	1	1	1
User 2	1	1	0	1	0
User 3	0	1	1	1	1
User 4	1	1	1	1	0
User 5	1	0	1	1	1

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	7	6	7	4	5
User 2	6	7	?	4	3
User 3	?	3	3	1	1
User 4	1	2	2	3	3
User 5	1	?	1	2	3

Fig. 1 Implicit and Explicit User–Item Rating in CF-Based Recommendation Systems

items $P = \{p_1, p_2, ..., p_n\}$. The system constructs an $m \times n$ user–item matrix, where each entry $r_{i,j}$ denotes user interactions (clicks, purchases, views) u_i for an item p_j, and the CF algorithm then either predicts a rating for an item or recommends a list of most likable Top-K items for user u_i. The rating matrix can be (a) *implicit,* in which ratings are collected based on clickstream or historical purchase data such as tracking users' navigational behavior on a site, or (b) *explicit*, where customers provide ratings, for example, users rating movies on a scale of 1–5. Figure 1 shows an illustration of an implicit and explicit rating matrix created from user's browsing and reviews' history from an e-commerce site. The entry "1" or "0" in the implicit matrix show that the user has interacted with those items and similarly an explicit rating provided by the customer for an item is recorded in the explicit matrix. A "0" or a "?" indicates the absence of users' interaction with that particular item.

Content based, on the other hand, generates recommendations based on the content (features) of the item [6]. The system creates user profiles based on item features (key words) extracted from items, which the user has consumed in the past. Different models (e.g., TF-IDF) are then used to compute similarity (relationship) between already consumed items and the items available in the catalog. Items with higher similarity score are then recommended to the customer.

While collaborative filtering-based recommendation does not consider the items' properties and only uses user interactions with items, it suffers from some limitations such as (1) sparse rating matrix, as users only rate a small number of items out of tens of thousands of available products and (2) cold start, where a new item cannot be recommended initially when it is introduced to a collaborative filtering system with no ratings. Alternately, content-based approaches suffer from (1) content overspecialization (lack of diversity in recommended products) due to the use of specific item features only [6].

Hybrid recommendation systems were introduced to address the limitations of both CF and content-based systems. The idea is to generate recommendations by considering user preferences (implicit or explicit) on items along with the content (product features) [7].

Challenges in E-commerce Recommendation Systems

E-commerce recommendation systems face certain challenges including:

1. *Cold Start problem*: In cold start problem, items cannot be recommended due to the lack of information about the user and/or the item. Cold start problem presents a collective issue of new item and new user to RSs. There are two types of cold start. (i) New item: a new item cannot be recommended initially when it is introduced to a collaborative filtering system with no ratings. For instance, MovieLens (movielens.org) cannot recommend new movies until these have got some initial ratings. (ii) New user: when a new user enters the system, it is a bit hard to find similar users or to create a content-based profile, as previous preferences (such as browsing history, likes/dislikes) of user are not available.

2. *Sparsity and Scalability*: Sparsity occurs when the majority of the users do not rate most of the items and consequently the ratings matrix becomes very sparse, which limits the chances of finding a set of users with similar ratings. In other words, this problem arises when there are many items to be recommended, but only few recommendations are provided (because of fewer available ratings on items). Also, it is becoming challenging to deal with huge and dynamic data sets produced by item–users interactions such as preferences, ratings, and reviews. It is possible that when some recommendation algorithms are applied on relatively small data sets, they provide the best results, but may reflect inefficient or worst behavior on very large data sets. So, algorithms need to be scalable to perform well on large data sets.
3. *Content Overspecialization and Diversity*: This means that the system cannot provide something unexpected. Such systems would not find anything novel since they only recommend items similar to the user profile. This problem affects the user and the system. The user may get bored easily and not return. The system may not be able to introduce new items. Differently speaking, generally, a user can opt for an item of his interest from a recommendation list if the list reflects some diversity in the recommended items to some extent. Seamless recommendations for a restricted type of product have no value until or unless it is desired or explicitly described by the user with a narrow set of preferences. In the initial stage, when the RS is used as a knowledge discovery tool, the users may wish to explore new and different options available. So, it is required to have such solution that could achieve the goal of diversity of items together with the accuracy of recommendation.
4. *Lack of Sequential Information in Recommendation*: These traditional recommender systems model users' interactions with items (e.g., clicks, add to cart, views, or purchases) in a static way and can only capture users' general preferences without considering the sequential behavior of user–item interactions. As users' interests and preferences change with time, learning the sequential behaviors in users' interactions based on the time stamps are useful to (i) understand their long- and short-term preferences and (ii) predict the next items for purchase based on finding the sequential dependency of an event (e.g., click or purchase) on its preceding event, as the time interval between any such interactions provide useful insights about users' behavior. So, a sequential recommendation system views the interactions of a user as a sequence and aims to predict which item the user will interact with next. Hence, the problem of sequential recommendation system can be stated as: given past item interactions (e.g., clicks, views, purchases) of a user, the goal of a sequential recommendation system is to predict the future item(s) that are of interest to the user.

How E-commerce Recommendation is Different from Recommendation in Other Domains

1. *Increased Number of Products*: Retailors and vendors keep on adding new products to their catalog to attract customers. Although this addition of products is good to provide more choices to customers, at the same time it brings the challenge of sparsity (as customers do not interact with all products out of millions of available products).
2. *Long- and Short-Term User Preferences*: Users' browsing and purchasing preferences change with time. For example, one of the scenarios can be where the user has a long-term preference toward a particular dairy product, but recently due to certain dietary restrictions, they shift towards other brands/dairy products. Another scenario could be change in user's browsing and purchase preferences due to the COVID-19 pandemic. For example, we can see a high demand for sanitizing products, masks and other such products. So, monitoring these static (long-term) preferences (e.g., a particular clothing brand/style) and dynamic short-term intent (e.g., purchase of sanitizing products) is important to generate tailored recommendations to customers.
3. *Regular Buyers and Temporary Buyers*: Is is both important and challenging to differentiate between regular and temporary customers. The target should be to retain regular customers while at the same time convert the temporary customers to regular customers (e.g., by offering them incentives, cash back offers, etc). For example, we have seen in recent times during the COVID-19 pandemic that due to forced lockdowns, people turned to online shopping even if they did not prefer it. However, after the ease in lockdown restrictions, these customers might prefer in-store shopping from their regular store of choice.
4. *Huge Diversity Across Various Product Categories*: Big e-commerce platforms like Amazon and Alibaba have a diverse collection of product categories ranging from digital devices, books, video games, electronics, home décor, furniture, office supplies, grocery, health and beauty, and clothing sports to name a few. These product categories are then further subclassified into several categories which creates huge diversity among all available products.
5. *Variations in Product Features Within the Same Product Category*: Lots of variation exists among product features in the e-commerce domain. For example, consider the "Clothing" category with subcategories of "Men", "Women" and "Baby" clothing for simplicity (as there

are many other subclassifications that exist as well). For these three subcategories, a single product feature "size" will have different specifications. For example, for "Men", it can be "Small", "Medium", "Large" and "Extra Large", however, for women the same size range could be interpreted differently (e.g., in case if someone is petite or looking for maternity wear). Similarly, for babies, the size will be according to their age groups (e.g., 0–6 months, 1–2 years, infants etc.). Hence, generating recommendations relevant to customer interest and requirements is vital.

6. *Availability of Substitute Products*: Due to the ever increasing number of products, a lot of substitutes exist for products. For example, in case of household (e.g., kitchen items), if the user is looking for a non-stick cookware set, there will be lots of subsitutes available (e.g., similar products from other brands). So, if the customer is not looking for a product from a specific brand, she may like to choose from the shown substitute products. Additionally, cookwares with other specifications (e.g., ceramic) can also be available as a substitute.
7. *Complementary Products*: As substitute products can be used in place of the other products, complementary products refers to items which are consumed together to satisfy a particular want. For example, a user looking for a party wear, can be recommended complementary items (e.g., accessories including shoes, purse) that complement her party look along with the dress. Similarly, for the kitchen scenario presented above, along with the cooking ware, complementary products such as cutlery set can be recommended. However, finding suitable set of substitute or complementary products is also important to satisfy customers' needs.

Related Surveys

There have been some surveys on sequential recommendation. Smirnova et al. [8] proposed a categorization of the recommendation tasks and goals, and summarized existing solutions. A comprehensive discussion and analysis on sequential recommendations systems which utilize sequential pattern mining techniques for recommendation is provided in [9]. Specifically, they presented a taxonomy on sequential pattern mining algorithms (one of the traditional methods for sequential recommendation) and provided an in-depth analysis of the presented techniques. An extension to the previous taxonomies and surveys by providing a more complete review including further analysis of the research was presented by Mooney et al. [10]. Another survey [11] on recent advances and research opportunities in sequential pattern mining provided a comprehensive overview of the main approaches and strategies to solve sequential pattern mining problems along with their limitations. The survey also presented the open-source implementations of sequential pattern mining algorithms. All these surveys provide a comprehensive overview; however, they are only targeted toward one key technique (sequential pattern mining) for generating sequential recommendation(s) by sequential recommendation systems and do not discuss other techniques and methods utilized by sequential recommendation systems. Other recent surveys on sequential recommendation systems [12–15] focus only on deep learning-based sequential recommendation and are not domain specific (e.g., sequential recommendation in the e-commerce domain).

Different from the above-mentioned surveys, this survey presents a taxonomy of sequential recommendation systems in the literature with e-commerce product recommendation as an application. Furthermore, existing surveys were not domain specific and did not specifically focus on the e-commerce domain. Therefore, this paper aims to investigate sequential recommendation systems in the e-commerce domain by (a) introducing a taxonomy for classifying SRecSys under three main categories as: (i) traditional approaches (sequence similarity, frequent pattern mining and sequential pattern mining), (ii) factorization and latent representation approaches (matrix factorization and Markov models), and (iii) neural network-based approaches (deep neural network approaches and advanced models). This classification enhances the understanding of sequential recommendation systems, provides current status of available solutions, and explores future research directions in this area. The survey also presents a classification of presented systems according to eight important key features supported by the techniques and discusses their limitations as well. Additionally, it provides a comparative performance analysis of various key approaches based on experiments performed on publicly available benchmark data sets (Amazon and Online Retail) along with a discussion about the theoretical aspects of the presented categories in the taxonomy.

Contributions and Survey Structure

The contributions made by this survey are as follows:

1. We provide a comprehensive overview of the sequential recommendation systems in the e-commerce domain in terms of the techniques used to build such systems.
2. We present a classification (i.e., a taxonomy, which refers to the classification of sequential recommendation system techniques in the literature) corresponding to key features supported by the techniques along with their limitations.
3. We provide a discussion of eight key features to further classify e-commerce SRecSys surveyed in this paper.
4. We also provide a comparative performance analysis of various SRecSys based on experiments performed on

publicly available benchmark data sets (Amazon and Online Retail).

The presented taxonomy contributes to research because:

1. It helps to gain an in-depth understanding of different techniques along with their features, and also methods used in the research.
2. It discusses the advantages and shortcomings of the various approaches.
3. This taxonomy for Sequential Recommendation Systems presents a hierarchical and a tabular ordering of the key approaches along with their features.
4. It also presents experimental performance including comparative analysis of the presented techniques using e-commerce data sets and evaluated on the basis of various evaluation metrics including precision, recall and mean reciprocal rank.

The rest of this paper is organized as follows. Section "Overview of sequential recommendation" presents the formal definition of the problem of sequential recommendation. Section "Taxonomy of sequential recommendation" presents the taxonomy and surveys of sequential recommendation systems. Techniques presented in the taxonomy belong to three main classification categories: (i) traditional approaches (sequence similarity, frequent pattern mining and sequential pattern mining), (ii) factorization and latent representation approaches (matrix factorization and Markov models) and (iii) neural network-based approaches (deep neural networks, advanced models). Section "Features to classify sequential recommendation systems" provides a tabular classification of sequential recommendation systems under the three presented categories based on eight key features. Section "Experimental results and analysis" provides comparative experimental analysis of the reviewed systems according to various evaluation metrics (precision, recall and mean reciprocal rank) using publicly available data sets (Amazon and Online Retail). The survey is concluded in Section "Conclusions and future research directions" along with providing future research directions.

Overview of Sequential Recommendation

Why Sequential Recommendation Systems?

Sequential recommendation systems suggest items which may be of interest to a user by modeling the sequential dependencies over the user–item interactions (e.g., clicking, viewing or purchasing items on an online shopping platform) in a sequence [12]. Learning sequential dependencies between items for the next item recommendation assists in modeling user preferences by reflecting the sequential dependency of an event (e.g., click or purchase) on its preceding event. A sequential recommender views user interactions as a sequence and predicts the next items with which the user will interact. Items with which a user interacts (e.g., clicked, rated, or purchased) can provide a strong indication of her interests and facilitate in learning a good user profile leading to recommendations that match her interests. However, users' interests and preferences change with time. The time stamp of a user interaction (click or purchase event) is an important attribute and learning the sequential patterns of user interactions based on the time stamps are useful to (i) understand the long- and short-term preferences of the user and (ii) predict the next items for purchase by users, as the time interval between any such interactions provides useful insights about users' behavior.

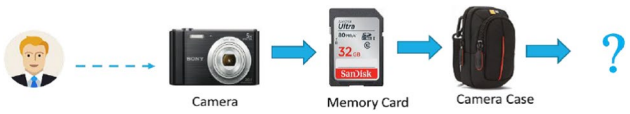

Fig. 2 Example of a sequential recommender system: after Smith has purchased a camera, memory card and a camera case, what item will he purchase next?

In the real world, user–item interactions (e.g., shopping behaviors) are mostly sequentially dependent. For example, in Fig. 2, we can see the sequential dependencies for a user Smith. After Smith has purchased a camera, memory card, and a camera case, what item will he purchase next? Such kind of sequential dependencies commonly exist in transaction data, but traditional collaborative filtering and content-based recommender systems are unable to capture these sequential dependencies which necessitates the need for developing sequential recommendation systems.

Problem Formulation

Consider a set of users $U = \{u_1, u_2, ..., u_n\}$ and a set of products $P = \{p_1, p_2, ..., p_n\}$. Each user u is associated with a sequence of interactions with some items from P such that $S = \{S_1, S_2, ..., S_n\}$. Given a sequence of user–item interactions (click, view, add to cart, purchase), the task of a sequential recommendation system is to generate a personalized Top-K ranked candidate item list. The system aims to maximize users' future needs, by considering their long- and short-term preferences in terms of finding sequential relationships between various user–item interactions (e.g., clicks, views and purchases). In other words, the sequential recommendation task is to learn a complex function f for accurately predicting the probability that user u will choose each product p at time $t + 1$ based on the input behavior sequence and the user profile. Mathematically,

$$O = argmax\, f(I),$$

where f represents a utility function which outputs a ranking score for the candidate items and can be in the form of conditional probability or an interaction score. I is the input sequence such that $I = \{a_1, a_2, ..., a_{|S|}\}$ and represents a sequence of user–item interactions, where each interaction $a_i = \langle u, i, p \rangle$ is a triplet comprising a user u, the user's interaction i, (click, view or purchase) and the corresponding item p. Sometimes, there is some metadata associated with the users (e.g., user demographics or features), items (e.g., product descriptions, title, brand) and user–item interactions (e.g., click, add to the cart, purchase) which can occur under different scenarios (contexts) such as a particular time, location and weather. The output O (i.e., recommendation) is a list of items ordered by the ranking score.

Taxonomy of Sequential Recommendation

A taxonomy of existing sequential recommendation systems is provided in Fig. 3, which provides a classification of the sequential recommendation systems according to three main categories: (i) traditional approaches (sequence similarity, frequent pattern mining and sequential pattern mining), (ii) factorization and latent representation approaches (matrix factorization and Markov models), and (iii) neural network-based approaches (deep neural network, advanced models) along with providing details on their theoretical aspects, potential for solving problems and their limitations.

The subsequent sections will review related approaches used in sequential recommendation systems for learning sequential relationships between items for next item recommendation.

Traditional Approaches for Sequential Recommendation Systems

To learn sequential associations between items, these approaches work either by computing similarities between sequences of items (e.g., click or purchase sequences) according to a similarity measure (similarity based) or by mining sequential patterns that occur frequently in the data. Section "Sequence similarity based" discusses about sequence similarity-based approaches, while Sect. "Frequent pattern mining" and Sect. "Frequent sequential pattern mining" will review pattern mining-based methods including frequent pattern mining and frequent sequential pattern mining, respectively.

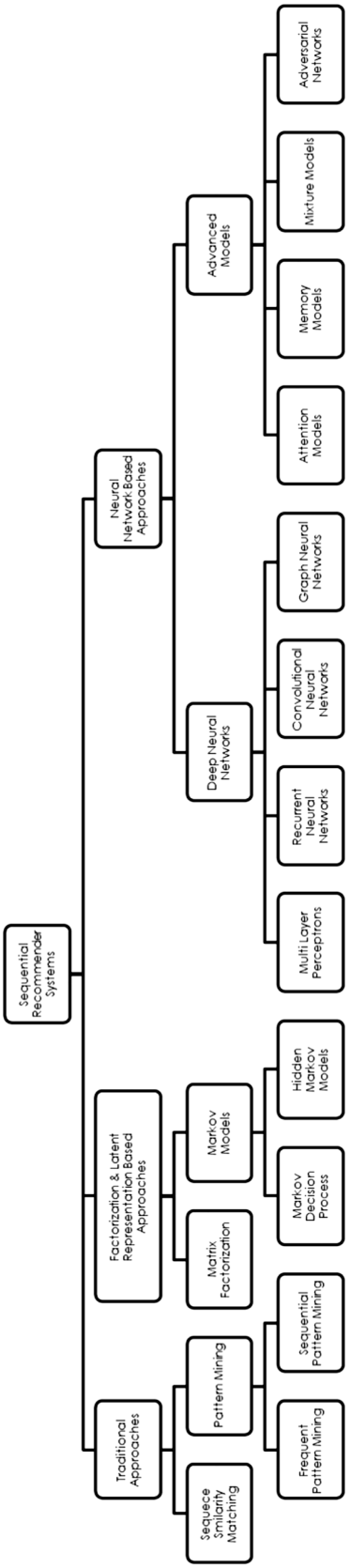

Fig. 3 Taxonomy of SecRecSys

Sequence Similarity Based

The objective of these approaches is to find relationships between items or sequences by computing similarity based on a similarity measure and then utilizing sequences similar to the target user sequence for the next item recommendation. These relationships reflect the sequential dependency of an event (e.g., click or purchase) on its preceding event. The granularity of sequential dependency can be at item level (single item) or sequence level (comprising multiple itemsets, where each itemset can be a single item or collection of items).

Problem Find relationships between item sequences by computing similarity using a similarity measure and then utilizing sequences similar to the target user sequence for the next item recommendation. For example, consider a purchase sequence of a customer as $< (a,b),(b,c,a,d),(a),(c,d,d) >$ which shows items purchased by the customer on different time stamps. The sequence consists of four itemsets which are (a, b), (b, c, a, d), (a) and (c, d, d). The itemset (a, b) consists of two items a and b which were purchased together and the itemset a just represents a single item which was purchased individually in another transaction. Sequence similarity can be computed by finding the longest common subsequence (LCS) rate between the two sequences and then utilizing the obtained rate to recommend items from the sequence with the highest rate to the target sequence. Consider purchase sequences of two customers as $X =< (a,b),(b,c,a,d),(a),(c,d,d) >$ and $Y =< (a,a,b),(b,c,c),(a,b) >$, then the LCS rate [16] between sequence X and Y can be computed using Eq. (1) as:

$$\begin{aligned} LCSR(X,Y) &= \frac{\mathrm{LCS}(X,Y)}{\max(X,Y)} \\ &= \frac{< (a,b),(b,c),(a) >}{< (a,b),(b,c,a,d),(a),(c,d,d) >} \\ &= \frac{5}{10} = 0.5, \end{aligned} \tag{1}$$

where LCS [17] can be computed using Eq. (2) as:

$$LCS(X,Y) = \begin{cases} \phi & if \quad i=0 \quad or \quad j=0 \\ \mathrm{LCS}\left(X_{i-1},Y_{j-1}\right) \cap X_i & if \quad x_i = y_i \\ longest \quad LCS\left(X_i,Y_{j-1}\right), & \\ \qquad\qquad LCS\left(X_{i-1},Y_j\right) if x_i \neq y_i. & \end{cases} \tag{2}$$

Another method of computing the similarity between sequences is by representing each sequence as a vector of events (e.g., items purchased). Considering the same purchase sequences X and Y, their corresponding vectors in terms of the frequency of items purchased will be $\vec{X}$= [a b c d] = [3 2 1 3] and $\vec{Y}$ = [a b c d] = [3 3 2 0]. The similarity between $\vec{X}, \vec{Y}$is calculated using cosine similarity as:

$$\begin{aligned} \mathrm{sim}(X,Y) = \cos(\vec{X},\vec{Y}) &= \frac{\sum_{i=1}^{n} X_i Y_i}{\sqrt{\sum_{i=1}^{n} X_i^2}\sqrt{\sum_{i=1}^{n} Y_i^2}} \\ &= \frac{(3*3)+(2*3)+(1*2)+(3*0)}{\sqrt{3^2+2^2+1^2+3^2} * \sqrt{3^2+3^2+2^2+0^2}} \\ &= 0.87. \end{aligned} \tag{3}$$

A sequential recommendation system, ChenRec15 [18], finds the similarity between two users based on their clickstream sequences. They measure indicators like category visiting path, category browsing frequency and category access time, then cluster users into different clusters, by selecting top-N similar users, and then provide better top-N recommendations. However, they only focus on the category level visits, and their technique for mining the whole data set is not very efficient. To improve the effectiveness of recommendation systems by incorporating the sequence information about user's item ratings, a system is proposed [19]. To depict users' dynamic interest evolution patterns, [19] uses the concept of "interest sequences" (IS) which takes the sequences of users' behaviors based on the time of their interactions (ratings). Assume two users u and v. User u rated items C, E, and D at time stamps t_1, t_2 and t_3 with ratings 4.0, 3.5, and 2.0, respectively. User v rated the same items, but in different order as items E, D, C at time stamp t_1, t_2 and t_3 with ratings 2.0, 4.5, 2.5. So, IS of user u (i_{su}) according to time stamp t_1, t_2 and $t_3 = (C, 4.0), (E, 3.5), (D, 2.0)$. Similarly, IS of user v at these time stamps will be $i_{sv} = (E, 2.0), (D, 4.5), (C, 2.5)$. Next, an interest sequence match (ISM) between users is calculated if (a) the rated items between them are same and (b) the deviation between the ratings given by the user u and user v for the same item is less than a specified threshold θ. Using these interest sequences, they computed the length of longest common sub-IS (LCSIS) (which refers to an interest sequence match (ISM) between two interest sequences if and only if there is no other longer ISM detected between them) and the count of all common sub-IS (ACSIS) which is used to count all the ISM between two interest sequences, including empty ISM. Considering the length of LCSIS and the count of ACSIS, users' similarities are computed based on IS, which are then used to find the Top-K nearest neighbors for the target user.

HPCRec18 [20] used normalized purchased frequency matrix to improve the quality of ratings by mining the consequential bond (similarity between click and purchase sequences) in each session to predict the ratings for next possible purchase. The proposed system [20] enriched the quantity (finding the value for unknown ratings) and quality (finding precise value for already rated items) of user–item

Table 1 Implicit user–item rating matrix

User\Items	Apple	Orange	Banana	Bread	Butter	Milk
User1	0	0	0	1	1	1
User2	0	1	0	0	1	1
User3	0	0	0	1	1	0
User4	1	1	1	1	1	1
User5	1	0	1	0	0	0
User6	1	1	1	0	0	0
User7	1	1	0	0	1	0

rating matrix. However, the main limitation was that customers' historical sequential behavior patterns were not integrated into the item rating matrix and during the mining process of consequential bond. This limitation was addressed by HSPRec19 [16] which incorporated sequential information while mining the consequential bond and enriching the user–item matrix.

Frequent Pattern Mining

The objective of frequent pattern mining is to find relationships (patterns) between items that occur frequently in the transaction database and then utilizing the mined patterns to guide the subsequent next item recommendations. Association rules [21] were first introduced to discover relationship between items in supermarket data. Consider a transaction database of customer purchases, $T = \{T_1, T_2, ..., T_m\}$, containing m transactions, which are defined on items I. Each transaction T_i is a subset of the items in I. To find these relationships between items, two measures, support and confidence are used. The support of an item is the measure of the frequency (occurrence) of the item in the transaction database. A minimum support threshold s is used to determine if the item is frequent or not. An item is said to be a frequent item if its support is greater than or equal to the minimum support s. These frequent items or frequent patterns can be useful in providing valuable insights about customers' buying behavior. For example, consider the data from customers' purchase in Table 1. The rows correspond to customers and columns correspond to items. The 1s refer to the situation in which a particular customer has bought an item.

These are partitioned into two sets of closely related items. One of these sets is $\{Apple, Orange, Banana\}$ and the other is $\{Bread, Butter, Milk\}$. These are the only itemsets with at least three items, which also have a support of at least 0.2. Therefore, both of these itemsets are frequent itemsets or frequent patterns. Finding such patterns with high support is useful to the merchant, because he can use them to make recommendations and other targeted marketing decisions. For example, it can be concluded that User2 is likely to eventually buy *Bread*, because he has already bought $\{Butter, Milk\}$. Similarly, User5 is likely to buy *Oranges* because he has also bought $\{Apple, Banana\}$. Such inferences are very useful from the point of view of a sequential recommendation system. Another insight can be obtained in terms of the directions of these correlations using the concept of association rules and confidence. An association rule is denoted in the form $X \rightarrow Y$, which shows the correlation between the set of items X and Y. For example, a rule such as $\{Butter, Milk\} \rightarrow \{Bread\}$ would be very useful to recommend *Bread* to User2, because it is already known that he has bought *Milk* and *Butter*. The strength of such a rule is measured by its confidence. The confidence of the rule $X \rightarrow Y$ is the "conditional probability that a transaction in T contains Y, given that it also contains X". Higher values of the confidence are always indicative of greater strength of the rule. An association rule is defined on the basis of a minimum support s and minimum confidence c. One of the earliest works in frequent pattern mining and association rule mining is the Apriori algorithm [21]. Given a transaction database and a minimum support threshold s, the algorithm mines the transaction database to find frequent itemsets for association rules. The algorithm is based on two important steps which are (a) join and (b) prune. Below are the steps for generating candidate items using the Apriori algorithm [21].

1. Generate all singleton (one-item) candidate items (C_k) from the transaction database.
2. Prune any items that do not meet the minimum support threshold s. This gives the frequent L_k items.
3. Generate two-itemset candidate items C_{k+1} using the apriori property of L_k join L_k.
4. Repeat step 2 to generate L_{k+1} frequent itemsets.
5. The process is repeated until there are no further candidate itemsets as the apriori algorithm is based on the apriori property which means that, any $(k-1)$- non frequent itemset cannot be a subset of a frequent k-itemset.
6. The set of frequent itemsets L will be the union of all $L_k, L_{k+1}, ...L_n$. The above steps can be explained with the help of a running example (Example 3.1).

Example 3.1 Association rule mining using the Apriori algorithm

Consider Table 2, which shows four transactions by customers. Each transaction is represented by a transaction id (TID) and the set of items purchased in that transaction. For example, transaction 001 shows that *Bread*, *Butter* and *Jam* were purchased. Given these transactions and a minimum support $s = 2$, which means an item should appear at least twice to be considered as frequent, the algorithm works by finding all singleton candidate items. The one-itemset candidate items C_k with their support count are $\{Bread : 4, Butter : 2, Eggs : 2, Tea : 1, Milk : 1, Jam : 1, Cheese : 1\}$. After the pruning step, that is, elements not meeting the minimum support which are $\{Tea, Milk, JamandCheese\}$, we have the frequent L_k items as $\{Bread : 3, Butter : 3, Eggs : 3\}$. Next to create two-itemset candidate items C_{k+1}, we will join each L_k with itself which gives the items $\{Bread, Butter : 3\}, \{Bread, Eggs : 3\}, \{Butter, Eggs : 2\}$. Repeating the steps, we will get the final set of three-item frequent items as $\{Bread, Butter, Eggs : 2\}$. Since we do not have any more candidate items to join, the algorithm will terminate. The set of frequent itemsets is $L = \{\{Bread\}, \{Butter\}, \{Eggs\}, \{Bread, Butter\}, \{Bread, Eggs\}, \{Butter, Eggs\}, \{Bread, Butter, Eggs\}\}$.

Generating Rules from Frequent Items The rules are generated by following these steps:

1. For each frequent itemset l, generate all nonempty subsets of l.
2. For every nonempty subset s of l, output the rule $s \rightarrow (l - s)$ if support_count (l) / support_count(s) >= *min_conf*, where *min_conf* is the minimum confidence threshold. From our example, if we take the itemset $I = \{Bread, Butter, Eggs\}$, its all non-empty subsets are $\{Bread, Butter\}, \{Bread, Eggs\}, \{ButterEggs\}, \{Bread\}, \{Butter\}, \{Eggs\}$. Given a minimum confidence of 0.5, some of the rules can be: (a) $\{Bread, Butter\} \rightarrow Eggs$ and (b) $\{Bread, Eggs\} \rightarrow Butter$ with a confidence of 0.6.

Kim11Rec [22], a recommendation system used association rule mining and calculated confidence between products at different levels such as clicks, basket placement and purchases. The main idea is to find the most relevant clicked item. This is repeated for all the other stages, i.e., basket places and purchases. A final score is calculated by assigning weights to the scores of the above-mentioned three stages. The disadvantage of association rule methods is that it does not include connections between different users who share special interests [22]. These systems simply measure the interest by browsing paths from the clickstream data.

Table 2 Customer's transaction database

TID	Items
001	{Bread, Butter, Jam}
002	{Bread, Eggs, Butter}
003	{Bread, Eggs}
004	{Milk, Tea}
005	{Bread, Butter, Eggs, Cheese}

Frequent Sequential Pattern Mining

These techniques mine patterns which occur frequently in a sequential database and then use the mined sequential patterns for generating subsequent items for recommendation. A common application is mining customers' purchase sequence database for generating frequent sequential purchase patterns which can predict the next item recommendations. A sequence database consists of ordered elements or events [9, 16]. Formally stated:

Given (i) a set of sequential records (called sequences) representing a sequential database SDB= $\{s_1, s_2, s_3, ..., s_n\}$ with sequence identifiers 1,2,3,....n; (ii) a minimum support threshold called min sup ξ and (iii) a set of k unique items or events $I = \{i_1, i_2, ..., i_k\}$. The goal of mining sequential patterns is to find the set of all frequent subsequences S in the given sequence database SDB of items I at the given min sup ξ that are interesting to the user. Consider the example in Table 3, showing a sequence database containing sequences representing four transactions made by four customers at a retail store in a week. Items between curly brackets represent an itemset purchased in one market visit. For example, the sequence with SID 3 indicates that a customer first purchased *Bread* in one transaction, then *milk*, followed by the purchase of *Egg* and *Butter* in the same transaction and lastly *cream* was purchased in another transaction.

Problem: Given an input of a sequential database (SDB) and a minimum support, the goal of sequential pattern mining algorithms such as GSP [21] is to mine all frequent sequential patterns from a sequence database having the support count greater than or equal to the user-defined threshold of minimum support.

A sequence s is said to be a frequent sequential pattern if and only if the support (number of occurrence of s) is greater than or equal to a given minimum support. One of the earliest sequential pattern mining algorithms GSP [21], which works on the apriori principle while maintaining the order of items, will generate some of the frequent sequences as $< \{Bread\} >, < \{Salt\}, < \{Bread\}, \{Butter\} >, < \{Bread, Cream\} >, < \{Bread, Milk\} >$, if the given minimum support is 2. So, the sequence $< \{Bread, Milk\} >$ is frequent, as it occurs in two sequences (SID 1 and 2)

Table 3 A sequence database (SDB)

SID	Sequence
1	{Bread, Milk},{Salt},{Egg, Butter},{Butter}
2	{Bread,Cheese},{Salt},{Milk},{Bread,Milk,Cream}
3	{Bread},{Milk},{Egg, Butter},{Cream}
4	{Milk},{Egg, Butter}

in the sequence database. In this case, the sequence $< \{Bread, Milk\} >$ and $< \{Milk, Bread\} >$ will be considered different, as their order is different in the sequences.

Many sequential pattern mining algorithms were introduced in the literature including GSP [21], SPAM [23], Prefix Span [24] and PLWAP [25]. An extensive survey on various sequential pattern mining algorithms for next item prediction is presented in [9–11].

Other research works [26, 27] also use sequential pattern mining to explore the sequential relationship between purchase sequences for item recommendations. These models are hybrid as they combine collaborative filtering with sequential pattern mining for providing recommendations. The idea is to find neighbor users not just on the basis of ratings on similar items, but also considering the sequential relationship between purchase sequences which can lead towards more accurate recommendations. Some of the recent works [16, 17, 26–28] have utilized sequential pattern mining techniques in e-commerce recommendation systems for mining customers' purchase patterns and recommending next items for purchase. A hybrid online product recommendation system (HOPE) [26] was proposed, which integrates collaborative filtering (CF)-based recommendations using implicit rating and sequential pattern mining-based recommendations. The system uses implicit rating information instead of explicit rating information by computing implicit rating information from the transaction data set. The recommendation quality was improved by integrating collaborative filtering and sequential pattern analysis of customer purchases each of which considers the rating information of users on items and the associations among items [26].

In another study, Yap et al. [17] stated that most of the existing sequential pattern mining methods are not user specific and proposed a personalized sequential pattern mining-based approach using a novel competence score to overcome such disadvantages. More weight was assigned to sequences that had similar items/products as the target user (e.g., user's purchasing/viewing the same books as the target user). To mine sequences in the purchase patterns of customer transactions, a framework is proposed [27] which aims to find the time gap between products purchased to help improve the recommendation of next products in sequence. SPADE [29] algorithm was used to mine sequences of all products which are (1) bought regularly, e.g., every month and (2) bought one after another in a sequence, e.g., most users find mobile phones and mobile covers as common purchase sequence.

HSPRec19 [16] mined frequent sequential purchase patterns and augmented the item rating matrix with sequential purchase patterns of customers for the next item recommendation. It mined frequent sequential click and purchase behavior patterns using the consequential bond between click and purchase sequences and then used this quantitatively and qualitatively rich matrix for Collaborative Filtering to provide better recommendations. The results [16] showed significant improvement over the systems without using sequential pattern mining methods such as those by [18, 20, 22, 30]. Therefore, mining sequential patterns from customers' historical transactions can be very beneficial for retailers and can increase revenue and customer satisfaction by recommending tailored products to customers.

In summary, rule-based approaches suffer from some limitations in terms of generating patterns including deciding an optimal value for the minimum support threshold ξ; setting it to a value which is too low will result in unnecessary patterns and setting it too high may lead to very few patterns which may not be very useful. Furthermore, the patterns are not personalized, that is, they do not take into account an individual's preferences while they are generated as they are based on the historical data of transactions from all customers. This also limits the capability of these systems as they can only recommend items which exist in the historical data (for example, items purchased by the customers) and meet the minimum support threshold ξ, thereby not exploring thousands of products available in the catalog for recommendation.

Factorization and Latent Representation-Based Methods

These techniques create a model from the training data (observed user ratings) and then predict the ratings for users on unseen items based on the observations learned from the trained model. Model-based methods apply machine learning techniques during the model training process. Model-based methods are assumed to capture underlying relations from the data. In the next subsections, a review of model-based techniques for sequential recommendation such as (i) matrix factorization and (ii) Markov models will be presented.

Matrix Factorization

Matrix factorization [31] is used to discover latent features (concepts) underlying the interactions between two different kinds of entities such as users and items. The idea is

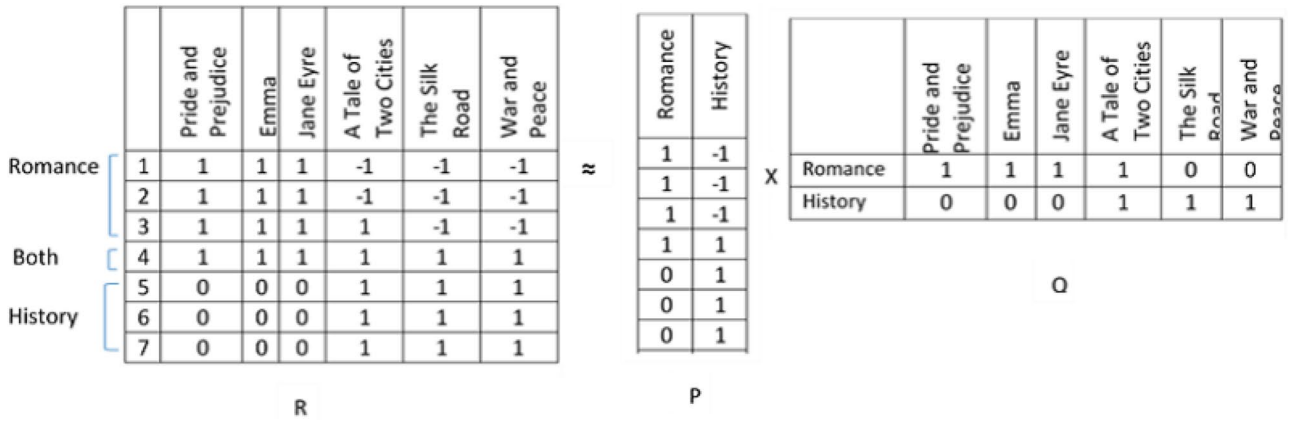

Fig. 4 An example of rank-2 (k=2) matrix factorization [2]

to decompose the rating matrix into two smaller matrices in such a way that when multiplied together (by taking dot product), they approximate the original matrix and predict the unknown ratings. In the basic matrix factorization model, the $m \times n$ ratings matrix R is approximately factorized into an $m \times k$ matrix P and an $n \times k$ matrix Q as follows:

$$R \approx P \times Q^T = \hat{R}, \tag{4}$$

where m and n represent the number of users and the number of items, respectively, and k represents the number of latent features. Each column of P (or Q) is referred to as a latent vector or latent component (concepts), whereas each row of P (or Q) is referred to as a latent factor. The ith row p_i of P is referred to as a user factor, and it contains k entries corresponding to the association of user i toward the k concepts in the ratings matrix. Similarly, the ith row q_i of Q is referred to as item factor, and it contains k entries corresponding to the association of item i toward the k concepts in the ratings matrix. The concept behind matrix factorization is that there should be some latent features which determine the preference of a user toward an item. For example, two users may give high ratings to a certain book if they both like the writer of the book, or if the book is based on a historical event. In this case, the latent feature can be the "genre" which both users like. So, the latent feature "historic genre" can be used to describe both the user and the item. Thus, the discovery of such latent features can help to predict the rating of a particular user on a particular item because the features associated with both the user and the item are same. While determining those different features, an assumption is also made that the number of features (k) would be less than the number of users and the number of items. To get the prediction of a rating of an item q_j by u_i, we can calculate the dot product of the two vectors using Eq. (5)

$$\hat{r_{lj}} = p_i^T q_j = \sum_{k=1}^{k} p_{ik} q_{kj}. \tag{5}$$

Figure 4 shows a conceptual view of matrix factorization of the rating matrix R into two matrices U and V. In the matrix U, u_i is a two-dimensional vector containing the association of user i toward the history and romance genres in the ratings matrix. Similarly, each row v_i of V is referred to as an item factor and represents the association of the ith item toward these k concepts. In the matrix Q, the item factor contains the association of the item toward the two categories of books.

Markov Models

The Markov chain-based models [32–35] are an early approach to Top-N sequential recommendation. Markov chain (MC) model-based RS utilize sequential data by predicting the users' next action based on the last actions. Therefore, a transition matrix is estimated that gives the probability of transitioning to the next state based on the previous state (for example, buying an item based on the last purchase of the user). The transition matrix of the MC models is assumed to be the same over all users. An $L-order$ Markov chain makes recommendations based on L previous actions. The first-order Markov chain is an item-to-item transition matrix learned using maximum likelihood estimation (MLE). Figure 5 shows a state diagram and computation process to compute probability for transitioning to next state from the current state (e.g., buying the next item after the purchase of a recent item) from time t to $t+1$ using first-order Markov model. Consider we have three states representing three products A, B and C and the transition probabilities of moving from one state to the next (probabilities of purchasing these products). Assume an initial transition

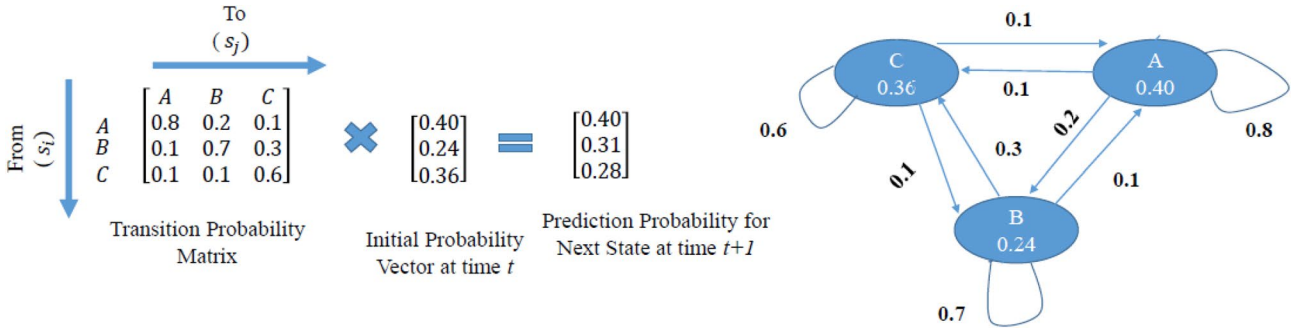

Fig. 5 Example to compute the probability for transitioning to the next state from the current state (e.g., buying an item from time t to $t+1$ using first-order Markov Model

matrix $\mathbf{P^t}$ with transition probabilities defining the probability of going from one state to another (based on analysis of customers' purchase data), in this case the probability of purchasing the next product. The column indicates the start state (s_i) and the rows indicate the end state (s_j). For example, at current time t, the probability of going from state A to B is 0.1 (probability of purchasing product B after product A). An initial probability vector $\mathbf{p_s}$ for each state to be the start state (chosen randomly) is also provided. For example, for the given vector $\mathbf{p_s}$ with values [0.40 0.24 0.36], the probability that the customer will start its purchasing journey from product A is 40% (0.40) (based on analysis from historic purchases on the company's site), whereas the probability that product B will be purchased first is 24% (0.24). To predict the transition probabilities from each of these states to next states at time $t+1$, we compute product of the transition matrix $\mathbf{P^t}$ with the initial probabilities vector $\mathbf{p_s}$. For example, as can be seen from the new probabilities $\mathbf{P^{t+1}}$ (at time $t+1$), the probability of purchasing product B after product A is 0.31 (second row in the probability transition matrix $\mathbf{P_{t+1}}$).

Markov chains-based recommender systems have been studied by several researchers [32–36]. A sequential recommender based on Markov chains [36] investigates the extraction of sequential patterns to learn the next state with a standard predictor—e.g., a decision tree. Another recommender based on Markov decision processes (MDP) and also an MC-based recommender [35] enhanced the maximum likelihood estimates (MLE) of the MC transition graphs along with the use of several heuristic approaches like clustering and skipping.

In summary, Markov chain-based RSs have some limitations: they can only capture the short-term dependencies while ignoring long-term ones due to the Markov property which assumes that the current interaction depends on one or several most recent interactions only; on the other hand, they can only capture the pointwise dependencies while ignoring the collective dependencies over user–item interactions.

Neural Network-Based Methods

Neural networks [37] have natural strength to model and capture the comprehensive relations over different entities (e.g., users, items, interactions) in a sequence. Many deep neural networks for modeling sequential behavior have been studied including recurrent neural networks (RNN), convolutional neural networks, and graph neural networks. In the next subsections, we provide a review of neural network-based sequential recommendation methods subclassified as deep neural networks (multilayer perceptrons, recurrent neural networks, convolutional neural networks, graph neural networks) and advanced models (attention models, memory models, mixture models, adversarial networks).

Deep Neural Network Methods

Deep learning has its origins from the field of artificial neural networks [38]. It deals with training multilayer artificial neural networks. Deep learning is an evolving field in machine learning research [39] and it aims to interpret data (e.g., sounds, images and texts) [40] by mimicking the function of the human brain. Hinton et al. [41] proposed the idea of deep learning in 2006. In deep learning, low-level features are combined to generate a high-level abstract representation of features (e.g., attribute categories). This high-level feature representation facilitates the discovery of distributed representation of features in the data.

Multilayer Perceptrons A fully connected multilayer neural network is called a multilayer perceptron (MLP). It has three layers, input layer, output layer, and one or more hidden layer(s). The inclusion of more than one hidden layer in the architecture makes it a deep neural network. MLPs can also be interpreted as multiple layers stacked together performing nonlinear transformations to learn hierarchical feature representations. A multilayer perceptron begins with

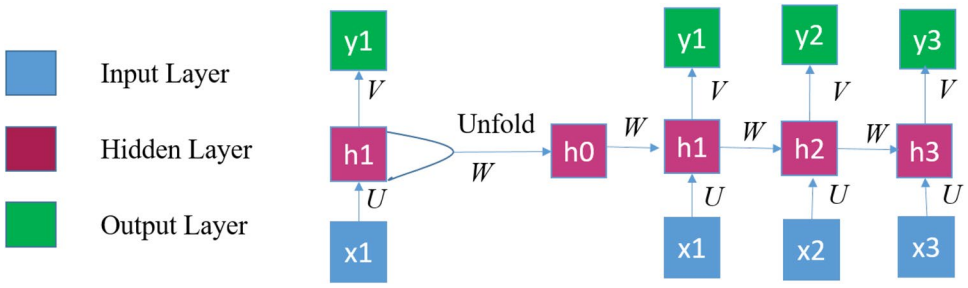

Fig. 6 Architecture of an RNN model

the input layer and forwards the data to the output layer after going through processing at the hidden layer(s). Depending on the output, the error is computed and, to minimize the error, it is back propagated to the model. The weights at each layer are adjusted after computing the derivate and the model is updated. This process is repeated until the ideal weights are learned and error is minimized (i.e., models' prediction accuracy improves). The number of layers and the number of neurons are the hyper-parameters of the model and can be tuned (to find ideal values) for optimized performance.

For next basket recommendation, NNrec [42] utilized neural network. It predicts a user's next basket by capturing the local context in user's last *k* baskets. The proposed model has four layers as: (1) input layer, which inputs a one-hot vector encoding of user id and item ids present in user's recent k baskets, (2) embedding layer, which performs an average pooling operation (down-sampling the input by extracting important features) to product user's basket vector representations, (3) hidden layer, which inputs the concatenated *k* basket vectors and user vector from the embedding layer, and finally the (4) output layer, which predicts the probabilities (using a Softmax function) of each candidate item to be in the user's next basket.

HRM [43] is another MLP-based model for next basket recommendation. It aims to find correlations between users' long-term and short-term preferences. The model has two aggregation layers where each layer performs any of the max-pooling or average pooling operation to extract important features from the input. The first layer finds item correlations in the user's recent basket (short-term preferences) by forming transaction-level item vector representations and the second layer models users' general interests (long-term preferences) by learning user representation among all users. The next items in the basket are then predicted on the basis of this learned hybrid representation.

Recurrent Neural Networks These neural networks are suitable for sequence problems. Given a sequence of historical user–item interactions, an RNN-based RS tries to predict the next possible interaction by modeling the sequential dependencies over the given interactions. A basic RNN is a standard feed-forward multilayer perceptron network (MLP) architecture (a network with single input, multiple hidden and single output layer) with addition of loops to the architecture. RNNs have internal memory which helps the model to store important things about the input and enables it to predict precisely about what is coming next. For this reason, RNNs are preferred for processing data which is sequential such as time series, speech, text, financial data, audio, video, and weather. They are designed to handle variable-length sequence data. Figure 6 shows architecture of an RNN model.

A walk-through example for next item prediction using RNN is provided in Example 3.2.

Problem Given a sequence of historical user–item interactions, predict the next possible interaction by modeling the sequential dependencies over the given interactions.

Example 3.2 Next item prediction using RNN

A sample example is shown to explain the working of RNN based on [44]. A gated recurrent unit (GRU)-based RNN for session-based recommendation (GRU4Rec) was proposed. The main contribution was to propose modification to GRU by introducing session parallel mini-batches (to handle variable-length sequences) and mini-batch-based output sampling (to provide ranking on a subset of items). The input is the actual state of session with 1-of-N encoding, where *N* is the number of items. The coordinate will be 1 if the corresponding item is active in this session, otherwise 0. The output is the likelihood of the item being next in the session for each item. It starts by taking the input of user click sessions and then creating an order for the sessions. The first event of each of the first *X* sessions forms the input of the first mini-batch. The second mini-batch is formed from the second events and so on. If any of the sessions end, the next available session is put in its place. The desired output is the second events of active sessions. Sessions are considered to be independent, and thus the appropriate hidden state (GRU) is reset to zero when this switch occurs. A user click session consists of session id, item id, and time stamps. The actual

vocabulary size is in hundreds and millions, but for the example, here, a small vocabulary size of three items is used.

Input User click sessions, hidden unit dimension=3, input layers=3, output layer=3, mini-batch size=3, weight vectors (U,V,W) as $U = \begin{bmatrix} -0.75 & 0.50 \\ -0.25 & 0.00 \\ 0.25 & 0.25 \end{bmatrix}$, $W = \begin{bmatrix} 0.10 & -0.25 \\ 0.50 & 0.50 \end{bmatrix}$, $V = \begin{bmatrix} -0.90 & 0.45 & 0.00 \\ 0.15 & 0.25 & 0.10 \end{bmatrix}$, item vocabulary = purse, wallet, mobile, laptop

Output Likelihood of item being next in the sequence

Step 1: Parallel Mini-Batch Creation

The mini-batch will be created by taking the first event of each of the first X sessions to form the input of the first mini-batch (the desired output is the second events of active sessions). Table 4 shows sample click sessions. Here, we use batch size = 3. So, mbtch1 = (purse, wallet, mobile), mbtch2 = (wallet, laptop, purse), mbtch3 =(mobile, mobile, wallet).

Then, create encodings as shown in Table 5, where the presence of 1 indicates the item position in the vector.

Step 2: Hidden Layer Processing (Score Calculation on Items)

Pass the input word vector X_0 (input) at each timestep into the RNN cell to compute the next hidden state s_t using Eq. (6):

$$s_t = \sigma\left(x_t U + s_{t-1} W\right). \tag{6}$$

After substituting the values,we get the states as shown in Fig. 7.

Step 3: Prediction at Timestep t

Calculating the output at each timestep using $o_t = softmax(V_{st})$, where V are the weights at the output layer and s_t is the previously calculated hidden state. The steps and computations are shown in Fig. 8.

Table 4 User click sessions

Session no.	Click sequences
S1	Purse, wallet, laptop, mobile
S2	Wallet, laptop, mobile
S3	Mobile, purse, wallet, wallet, laptop, laptop
S4	Mobile, purse
S5	Wallet, laptop, mobile

Table 5 One-N hot vector encoding

Timestep	Input	(Vector encoding)
1	(Purse, wallet, mobile)	(1, 0, 0, 0, 1, 0, 0, 0, 1)
2	(Wallet, laptop, purse)	(0, 1, 0, 0, 0, 1, 1, 0, 0)
3	(Mobile, mobile, wallet)	(0, 0, 1, 0, 0, 1, 0, 1, 0)

Step 4: Item Ranking

From output o_4, o_5 and o_6, the results show likelihood for items to be next in the sequence. For example, we have the output from the layers as:

$$o_4 = \begin{bmatrix} 0.07 \\ 0.18 \\ 0.12 \end{bmatrix}, o_5 = \begin{bmatrix} 0.20 \\ 0.46 \\ 0.33 \end{bmatrix}, o_6 = \begin{bmatrix} 0.20 \\ 0.46 \\ 0.33 \end{bmatrix},$$

which means in o_4, out of three items "purse, wallet and mobile", it ranked "wallet" as expected to be the next item (0.18) with high probability. In the next timestep t_4 and t_5, the output ranked wallet with the high score, i.e., 0.46, whereas we expected it for items "laptop" and "mobile".

Fig. 7 Hidden state computations in RNN

$$s_1 = \sigma\left[1\ 0\ 0\right] * \begin{bmatrix} -0.75 & 0.50 \\ -0.25 & 0.00 \\ 0.25 & 0.25 \end{bmatrix} + \left[0\ 0\right] = \sigma[-0.75 \quad 0.50] = \left[0.32\ 0.62\right]$$

$$s_2 = \sigma\left[0\ 1\ 0\right] * \begin{bmatrix} -0.75 & 0.50 \\ -0.25 & 0.00 \\ 0.25 & 0.25 \end{bmatrix} + \left[0.32\ 0.62\right] * \begin{bmatrix} 0.10 & -0.25 \\ 0.50 & 0.50 \end{bmatrix} = \sigma\left[0.9\ 0.23\right] = \left[0.71\ 0.55\right]$$

$$s_3 = \sigma\left[0\ 0\ 1\right] * \begin{bmatrix} -0.75 & 0.50 \\ -0.25 & 0.00 \\ 0.25 & 0.25 \end{bmatrix} + \left[0.71\ 0.55\right] * \begin{bmatrix} 0.10 & -0.25 \\ 0.50 & 0.50 \end{bmatrix} = \sigma\left[0.59\ 0.7\right] = \left[0.64\ 0.66\right]$$

$$s_4 = \sigma\left[0.64\ 0.66\right] * \begin{bmatrix} 0.10 & -0.25 \\ 0.50 & 0.50 \end{bmatrix} = \sigma\left[0.39\ 0.17\right] = \left[0.59\ 0.54\right],$$

similarly,

$$s_5 = \left[0.57\ 0.52\right],$$

$$s_6 = \left[0.57\ 0.52\right]$$

$$o_4 = \text{softmax}\begin{bmatrix}0.59 & 0.54\end{bmatrix} * \begin{bmatrix}-0.90 & 0.45 & 0.00\\ 0.15 & 0.25 & 0.10\end{bmatrix} = \text{softmax}\begin{bmatrix}-0.45 & 0.40 & 0.05\end{bmatrix}$$

Now using formula for calculating Softmax which is:

$S(y_i) = \frac{e^{yi}}{\sum e^{yj}}$, hence,

$S(-0.45) = \frac{e^{-0.45}}{e^{-0.45}+e^{0.40}+e^{0.05}}$,

$o_4 = \begin{bmatrix}0.07 & 0.18 & 0.12\end{bmatrix}, o_5 = \begin{bmatrix}0.20 & 0.46 & 0.33\end{bmatrix}, o_6 = \begin{bmatrix}0.20 & 0.46 & 0.33\end{bmatrix}$

Fig. 8 Output layer computation in RNN

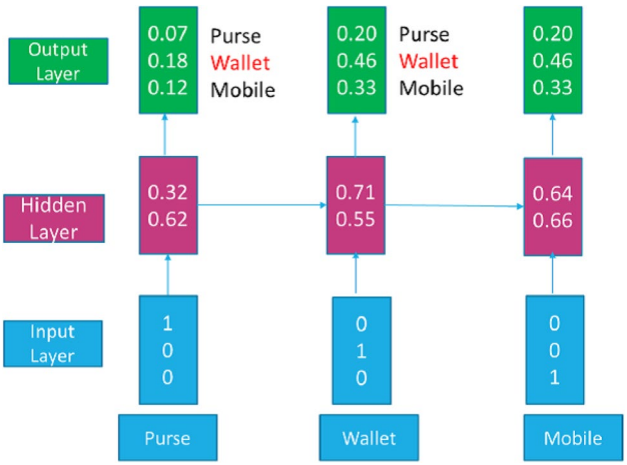

Fig. 9 Recommended items (output) through the RNN model for example 3.2

However, the results may vary depending on the selection of weights. So, from this we can see that the model has ranked "wallet" as the next item in sequence for the next mini-batch which is the same as the actual item in the next mini-batch (next in sequence). The items are ordered according to their probability (descending order) and their position is their rank as shown in Table 6. If the recommended (predicted) item is not the actual item, loss is computed and weights are updated by backpropagation.

Step 5: Loss Computation:

Expected loss is:

$$Ls = \frac{1}{Ns} \cdot \sum_{j=1}^{Ns} \sigma\left(r_{s,j-}r_{s,i}\right) + \sigma\left(r_{s,j}^2\right), \tag{7}$$

where N_s is the number of samples in the session and $r_{s,i}$ is the score on item i (next desired item), and $r_{s,j}$ are scores on negative samples which are taken from other sessions not present in the current session showing customer dislike. Hence,

$$\begin{aligned} L_s &= 1/3 * \sigma(0.46 - 0.18) + \sigma(0.46)^2 \\ &= 0.37. \end{aligned}$$

Similarly, loss for all sessions will be calculated for each epoch. After each epoch, the weight matrices U, W and V will be updated using backpropagation through time (BPTT). Update to each weight matrix is proportional to the gradient of the error with respect to that matrix and BPTT computes the gradients using the chain rule. Figure 9 shows the computation steps by RNN architecture to generate the final recommendation for Example 3.2.

Some other variants of RNNs are proposed to capture more complex dependencies in a sequence, like hierarchical RNN [45]. An extension to GRU4Rec [44] was proposed [46] by providing four main features which are (1) click sequence augmentation, (2) temporal change adaptation, (3) privilege information usage and (4) using item embedding. Smirnova et al. [8] proposed the use of contextual information besides considering the user's past sequence. They suggest using context such as the time gaps between events and the time of day for each interaction and the associated types of user–item interactions. Additionally, a new class of contextual recurrent neural networks for recommendation (CRNNs) was proposed which incorporates the contextual information both in the input and output layers and modifies the behavior of the RNN by combining the context embedding with the item embedding.

One of the limitations of the previous session-based systems was that they were unable to capture evolution in user preferences across sessions, the reason being that those approaches handled sessions independently and there is no cross transfer of information across sessions. This issue was addressed through a hierarchical RNN [45]. The model consists of a hierarchy of two GRUs, the session-level GRU (GRUses) and the user-level GRU (GRUusr). The session-level GRU generates recommendations by modeling the user activity within sessions. The user-level GRU(GRUusr) models user's evolution across various sessions and coordinates with session-level GRU by providing personalization capabilities through initializing its hidden state or, optionally, by propagating the user representation in input. Another study [44] introduced a parallel architecture consisting of multiple

Table 6 Scores and items' rank

Time step	Item scores	Scores (descending)	Item order	Expected order
1	(0.07, 0.18, 0.12)	(0.18, 0.12, 0.07)	Wallet, mobile, purse	Wallet, mobile...
2	(0.20, 0.46, 0.33)	(0.46, 0.33, 0.20)	Wallet, mobile, purse	Laptop, purse...

RNN's (p-RNN) for session-based recommendation. Three GRUs are used for learning item representations from text feature vectors, image features and one-hot vectors. To predict the next item in the session, the output of the three GRUs are weightedly concatenated and fed into a nonlinear activation. They categorize the architecture as baseline RNN and parallel RNNs. The idea is to first create multiple baseline models, each with a different item representation such as item ID, image and text. The proposed parallel architectures use a subset of the above three item representations (ID, image, or text) as input. The output is a score for every item indicating the likelihood of being the next item in the session. The hidden states of these networks are merged to produce the score for all items. As a baseline, the standard best RNN setting from GRU4Rec15 was chosen. The input of the networks is the item ID of a transaction.

In a different study, a two-stage GRU-based RNN interactive recommendation model was introduced [47], which first asks users about topics of interest and then recommends items such as videos. This first stage is question ranking. It involves capturing user interest by asking questions about the topic of their interest. The second stage is video response model (ranking stage model), which will predict Top-K videos based on the feedback (selected topic) from stage 1.

In summary, RNNs turn out to be promising models for modeling sequential dependencies between items; however, they suffer from some limitations: (1) it is easy to generate fake dependencies due to the overly strong assumption that any adjacent interactions in a sequence must be dependent, which may not be the case in the real world because there are usually irrelevant or noisy interactions inside a sequence; and (2) it is likely to capture the pointwise dependencies only while ignoring the collective dependencies (e.g., several interactions collaboratively affect the next one), and (3) difficulty to train the model and optimize the hyperparameters.

Convolutional Neural Networks A convolutional neural Network (CNN) is a feed-forward neural network and consists of convolution layers, pooling layers, and feed-forward full-connected layers. It can efficiently capture the local and global features in the input. For example, common applications include processing time series and image data such as finding correlations between pixels in a specific part of an image or finding dependencies between various adjacent words in a sentence [48, 49].

For sequential recommendation, given a sequence of user–item interactions, a CNN first generates embeddings of these user–item interactions and creates a matrix of these embeddings. This matrix is then treated as an "image" in time and space. The CNN then uses convolutional filters to learn sequential patterns considering them as an image's local features and generates next recommendations. CNNs, however, are not able to effectively capture long-term dependencies due to limitation in the size of filters used. Some research works such as 3D-CNN [50] concatenated the embeddings of item id, name, and category to create the input embedding matrix for the CNN. CASER [48], another convolutional neural network (CNN)-based method, provides sequential recommendation by applying convolution operations on the embedding matrix of L most recent items. It views the embedding matrix of L previous items as an 'image', and then captures the sequential patterns using horizontal and vertical convolutional layers. Long-term preferences are also captured through user embedding. NextitNet [49], a CNN model proposes to capture short- and long-term item dependencies through the use of residual block structure.

Graph Neural Networks Recently, graph neural networks (GNN)-based systems have been adapted to sequential recommendation. They can capture complex relations and dependencies between user–item interactions in a sequence. These systems work by constructing a directed graph on the sequence data. Each interaction is considered as a node in the graph and each sequence gets mapped to a path in the graph. In the next step, user and item embeddings are learned on the graph and complex relationships are embedded into the graph [51]. A graph neural network-based sequential recommender system (SDE-GNN) [52] captures sequential dependencies and item transition relations within sessions for generating accurate next item recommendations. Another recommender system [53] generates unified friend and item recommendation by incorporating a mutualistic mechanism which models mutual relationships between consumption and social behavior of users. A hierarchical and interactive gate network (HIGnet) model for items' rating prediction is proposed [54] which explores users' and items' textual features to capture their correlations by modeling informativeness of local words and captures global semantics from customer reviews in a hierarchical way. GNN [55] is proposed to collectively aggregate information from graph structure. Wu et al. [56] first used GNN for session-based recommendation by capturing more complex relationships between items in a sequence, and each session is represented as the composition of the long-term preference and short-term interests within a session using an attention network. Wu et al. [56] fused GNN for session-based recommendation and proposed modeling each session as a directed graph. Session representations include local and global session embeddings and the probabilities of next items are computed based on representations from sessions and item embeddings. A deep listNet framework was proposed by Wu et al. [57] with two phases as : (1) SIE, which performs pooling operation (average or max) to create click and view embedding vectors from user's interaction sequences. These are then concatenated with user and

target item embedding. A feed-forward network then learns a sessions' hybrid representation according to the concatenation. (2) Listwise ranking, which computes relevance score between candidate items and session representations. These techniques capture high-order sequential interaction between items and users; however, they usually require a high computational time and are complex.

Advanced Deep Neural Networks

Attention Models Attention mechanisms are also used for sequential recommendation. Given a user's historical behavior (sequence), these models aim to identify items more relevant to the user. They focus more on interactions that are important in a sequence and ignore the ones that are less relevant. An encoder–decoder-based attention framework NARM7 [58] is proposed for transaction-based sequential recommendation. The encoder is a hybrid of global and local encoder where the local encoder uses RNN with vanilla attention (which uses a linear combination of previous hidden vectors in the input sequence) to capture user's interest in a sequence and the global encoder uses RNN for modeling sequential dependencies between items in the current sequence. A unified representation for the sequence is then obtained from this hybrid encoder. Wang et al. [59] also utilized vanilla mechanism and reduced the impact of irrelevant interactions (e.g., accidental clicks, views) by assigning weights to each item. A short-term attention model proposed by Li et al. [58] computes (a) items' attention score in a sequence and (b) attention correlation between the most recent and previous items in a sequence. A hierarchical attentive architecture [60] deploys attention mechanism at item feature level and identifies users' preferences toward items SASRec. [61] uses attention mechanism to capture long-term user preferences and uses user's historical sequence to identify preferences for next items. The model comprises an embedding and self-attention layer along with a feed-forward network. The model can learn complex transitions between items by including additional attention layer and feed-forward network (self-attention blocks) in the architecture. The probability of a candidate item to be the next item (recommended next item in a sequence) is computed as a result of the output of the self-attention block. BERT4Rec [62], a sequential recommendation model with bidirectional encoder representations from transformer utilizes deep bidirectional self-attention mechanism for modeling user behavioral sequences and learns a bidirectional representation model which makes recommendations by allowing each item in users' historical behavior to integrate information from both left and right sides in a given sequence. Zhang et al. [63] infer relationships between items by accepting an input of an item embedding vector and predicts the next item by combining user's short-term and long-term preferences by estimating item weights in user's sequence using self-attention and a metric learning framework, respectively.

Memory Models Memory model's memory networks utilize an external memory matrix to store and then later update the historical user–item interactions in a given sequence. This facilitates finding dependencies between historical and future user–item interactions dynamically and explicitly, hence improving model effectiveness and reducing interactions that are irrelevant [64]. Some other works [65] proposed using a key-value memory network to store and then later update the historical user–item interactions to better learn user preferences for recommendation.

Mixture Models A mixture model [66, 67] generates sequential recommendation by integrating different submodels where each submodel captures different dependencies and contributes to the models' performance in generating better sequential recommendation. To learn short-term and long-term dependencies, [66] combines encoders of different kinds where each encoder learns a representation of a sequence to generate the subsequent next recommendations.

Adversarial Networks The adversarial network (AN) [68] is a generative neural network (GNN) which consists of a discriminator and a generator. The two neural networks (discriminator and generator) are simultaneously trained by competing with each other in a minimax game framework. Recently, generative adversarial networks (GANs) have been used in deep learning fields for image and audio generation [69]. GAN works on the principle of playing an adversarial minimax game between a generator and a discriminator. The task of the generator is to (i) capture the distribution of real observed data, (ii) generate samples which are adversarial, and (iii) fool the discriminator. The discriminator, on the other hand, focuses on distinguishing whether the input sample is generated by the generator. Recommendation systems DAN [70] utilizes an adversarial approach and transfers the representation of items and users to the target domain from different domains. Another GAN-based model, AB-GAN [69], is a virtual try-on clothing model which generates images for 2D images modeling. Using four features including user-image feature, desired-posture feature, body shape mask and new clothing feature, it generates an image of the user wearing new clothes. The generator creates the person's original image to ensure that data distribution is similar to the real image.

Some other sequential recommendation systems [71, 72] explored to integrate items' semantic knowledge obtained by utilizing item (products') metadata (e.g., title, description and brand) according to their semantic context (co-purchased and co-reviewed products) into the sequential recommendation process. Distributional hypothesis (which learns an item's representation by analyzing the context in which it is used) captures the semantics of these user–item interactions and then incorporate items semantic knowledge into different recommendation phases such as pre-processing, candidate generation and the output (recommendation).

A summary of sequential recommendation systems as classified in the taxonomy based on the techniques used to build those systems along with their limitations and notable research works is presented in Tables 7, 8 and 9.

In the next section, we will present a classification of the sequential recommendation systems based on eight key features and a classification of sequential recommendation systems according to these eight features will be presented in Tables 14, 15 and 16.

Features to Classify Sequential Recommendation Systems

Types of Sequences

Sequences in the database may represent information regarding single dimension or multiple dimensions. Below, we will discuss both of these sequence types.

Single Dimensional Sequences

These are the sequences which represent only one aspect (dimension) of customers' transactions (for example, sequences representing products purchased by customers). Consider Table 10 which shows a single dimension sequence database that shows purchase sequences of four customers where SID represents the sequence ID for a unique customer and the sequence represents the items bought by the customer on different time stamps. For example, the customer with SID 100 has the purchase sequence $< (be)(ce) >$,

Table 7 Summary of sequential recommendation system approaches

Category	Subcategory	Limitations	Research work
Traditional approaches	Sequence similarity matching	Static—cannot learn user's evolving preferences	Xiao et al. [20], Zhang et al. [73], Gurbanov et al. [74], Song et al. [75], Salehi [28]
		Item similarity computed using exact product match (e.g., name), no other content attribute information included	
		Sequence can only handle one event type (e.g., click/purchase events)	
		No use of side information (users/items)	
		Do not capture temporal information	
	Sequential pattern mining	Depends on choice of suitable threshold for min support	Bhatta et al. [16], Saini et al. [27], Choi et al. [26], Yap et al. [17], Rudin et al. [76], Parameswaran et al. [77], Liu et al. [78], Mobasher et al. [79] Agrawal et al. [21]
		Unable to generate patterns that are less frequent (may be of interest to the target user)	
		Cannot model individual user's preferences (lack of personalization)	
		Cannot infer semantic relationships between items	
		Only recommending frequent (popular) products	
Factorization and latent representation	Factorization techniques	Latent representations are not interpretable	Zeng et al. [80], Pasricha et al. [81],Wan et al. [82], He et al. [83], Lian et al. [84], Zhao et al. [85] Rendle et al. [34]
		Scalability issues	
		Good for only particular type of feedback (e.g., implicit user behavior)	

Table 8 Summary of sequential recommendation system Approaches (continued from Table 7)

Category	Subcategory	Limitations	Research Work
Factorization and latent representation	Markov Models	Can only model short-term dependencies (upto a limited no. of states) Transition probability matrix suffers from ambigiuos prediction problem (same probability for transitioning to next state) and sparsity (lack of user–item interactions)	Bernhard et al. [32], Tavakol et al. [86], Sahoo et al. [87], Shani et al. [35], Brafman et al. [33], Dasphande et al. [88], Zimdars et al. [36]
Deep Neural Network Based	Multilayer Perceptrons	Sensitive to feature scaling Hyperparameter tuning (e.g., no. of layers, iterations)	Xiao et al. [53], Dziugaite et al. [89], Lian et al. [90], Wang et al. [91], Xue et al. [92], Zhang et al. [93]
	Recurrent neural networks	Assumes that adjacent interactions in a sequencemust be dependent (which may not be true) Difficult to train model and find optimal hyper-parameters Interpretability is low	Hidasi et al. [44, 94, 95], Villatel et al. [96], Donkers et al. [97] Wu et al. [98], Yu et al. [99] Christakopoulou et al. [47], Qyadrana et al. [45], Tan et al. [46]
	Convolutional neural networks	Can be slower due to max-pooling operation Difficulty in image classification if the image is tilted or rotated A possiblesolution is to use data augmentation during classification Requires lots of training data	Tang et al. [48],Tuan et al. [50], Yuan et al. [49], Hsu et al. [100]

representing two events, each comprising two items. This shows that the customer first purchased items *b* and *e* together and then later on a different time stamp purchased items *c* and *e* together. Since, no other information besides the items purchased is represented by the sequence, these sequences are single dimensional. To mine frequent sequential purchase patterns according to a minimum support threshold from this single dimensional sequence database, any of the sequential pattern mining algorithms such as GSP [21], SPAM [23], Prefix Span [24] and PLWAP [25] can be used, as given the same sequence database and the minimum support threshold, any sequential pattern algorithm will yield the same set of frequent sequential patterns and will only differ in the process of generating sequential patterns (i.e., the structure for extracting sequential patterns).

Multidimensional Sequences

In real world, most sequence patterns are associated with different circumstances and such circumstances form a multiple dimensional space. For example, customer purchase sequences are associated with location, time, customer group, occupation, etc. Consider Table 11, which shows a multidimensional sequence database of customer purchases from Table 10, where it contains the columns (cust_type, city, age_range) representing three dimensions (aspects) of customers' information along with their unique SIDs and purchase sequences. For example, for customer with SID 100, we can see that he is a student from the Windsor region and of young age. In other words, a multidimensional sequence will take the form $(a_i,a_2,\ldots..,a_n,S)$, where a_i represents the dimension and S represents the sequence. In this case, for customer with SID 100, the multidimensional sequence will be represented as (student, Windsor, young, $<(be)(ce)>$). A multidimensional sequence is said to match a tuple in a multidimensional sequence database if any of the dimensions in the multidimensional sequence matches with the corresponding dimension in the multidimensional sequence database and the sequence present in the multidimensional sequence is the subset of the sequence in the database. For example, a multidimensional sequence $M = (student, *, *, <(b)>)$ matches tuples 100 and 103, where $*$ in the multidimensional sequence represents any domain not present in the sequence database and the support (no. of matching tuples) of M in the sequence data base is 2.

Mining sequential patterns associated with multidimensional information are useful. Given a multidimensional sequence database and a minimum support threshold, a multidimensional frequent sequential pattern will be the one whose support is greater than the minimum support threshold and can be mined through embedding multidimensional information into sequences and then mining the whole set using a sequential pattern mining method.

For example, consider Table 12 created as an extension from Table 11 by embedding the multidimensional information in the sequence as a special element. For instance, for tuple 100 in Table 11 (100, *student*, *Windsor*, *young*, $<(be, ce)>$), the sequence $<(be)(ce)>$ in this tuple can be extended as $<(student\ Windsor\ young)\,(be)\,(ce)>$ as shown in Table 12

Table 9 Summary of sequential recommendation system Approaches (Continued from Table 8)

Category	Subcategory	Limitations	Research work
Deep neural network based	Graph neural networks	Use of same parameters across different layers	Guo et al. [52], Xiao et al. [53], Wu et al. [51], Wu et al. [57], Sachdeva et al. [101], Zhong et al. [54], Zhou et al. [55]
		Difficulty in effective modeling of informative features on graph edges	
Advanced deep learning models	Attention models	Addition of more weight parameters, and hence increased model training time	Wang et al. [59], Ying et al. [102], Li et al. [58], Liu et al. [103], Ren et al. [104], Sachdeva et al. [105], Bai et al. [60], Kang et al. [61],Sun et al. [62], Zhang et al. [63]
	Memory Networks	An increase in each supporting memory level k leads to an increase in the embedding matrix size according to models' vocabulary size which becomes impractical for huge vocabularies (e.g., with thousands of words)	Chen et al. [64], Huang et al. [106]
	Mixture models	Identifiability (existence of a unique characterization for any one) of the models which requires minor constraints for estimation (e.g., in case of finite mixture models)	Tang et al. [66], Wang et al. [107], Kang et al. [67]
	Adversarial networks	Mode collapse (occurs when the generator only produces a single output type or a small set of outputs)	Ian et al. [68], Unger et al. [108], Wang et al. [56]
		Non-convergence	

Table 10 Single dimensional sequence database

SID	Sequences
100	(be) (ce)
101	(ah)abf
102	(bf) (ce) (fg)
103	(bd)cba

Table 11 Multidimensional sequence Database

SID	Cust_ type	City	Age_range	Sequence
100	Student	Windsor	Young	(be) (ce)
101	Manager	Toronto	Middle	(ah)abf
102	Skilled worker	Toronto	Retired	(bf) (ce) (fg)
103	Student	Waterloo	Young	(bd)cba

(tuple 100). The other sequence can be extended in a similar fashion. Next, multidimensional sequential patterns can be mined from this extended sequence database using any traditional sequential pattern mining algorithm such as prefix span [24].

Use of Side Information (Customers' and Items' Metadata)

Traditional information sources consist of gathering users' interest on items by monitoring their implicit or explicit behaviors.

Implicit

Implicit actions are captured by inferring how a user responds to an item, for example, on an e-commerce site, which items a user has browsed, clicked, added to his favorite items' list, added to cart or purchased. These implicit actions indicate a user's interest on an item and are helpful in determining their future preferences. One way of expressing implicit behaviors is by storing binary values in the user item rating matrix where "1" indicates behavior on an item such as clicked or purchased and "0" otherwise. A disadvantage is that it cannot fully reflect users' preference on the item as a value of "0" does not necessarily mean that a user is not interested in the item. Since there are millions of products, it is very likely possible that the user is unaware of the existence of such items.

Explicit

Explicit behaviors are recorded when users' explicitly provide feedback on the item. A common example is rating an item on a five-star scale where "1" is the lowest rating and "5" is the highest. Another way of extracting users' explicit

Table 12 Multidimensional extension sequence database created from Table 11

SID	Sequences
100	(Student Windsor young) (be) (ce)
101	(Manager Toronto middle) (ah)abf
102	(Skilled worker Toronto retired) (bf) (ce) (fg)
103	(Manager Waterloo middle) (bd)cba

feedback is from text reviews. In this case, if the review is positive, a value of "1" is stored in the user–item matrix, "0" for a neutral review and "−1" for a negative feedback. A common example is extracting user's preferences from the reviews they provide after consuming a product.

Auxiliary Input Sources

These information sources complement the traditional information sources and can better reflect user's interest by capturing more insights about users' behavior. These information can be classified as (a) user and item information (e.g., meta data), (b) information contributed by users (e.g., tags, geotags, multimedia content, free comments and reviews) and (c) information associated with user–item interaction also known as context. An example of such information can be when a user is interacting with an item, such as purchasing an item, listening to a song or watching a movie [6].

Consequential Bond Between Click Stream and Purchase Data

User interactions (clicks, purchases) in e-commerce data contain information which can be used to derive consequential bond (relationships) between these user interactions and thus facilitate understanding user preferences [20]. More specifically, consequential bond refers to the concept that when a customer clicks on some items, it is most likely that some of those clicked items will be purchased. For example, consider Table 13, presenting historical purchase data of various customers.

We can see that customer with user id 4 had clicked items 2 and 7 in a single transaction and notice that item 2 had been purchased in that transaction. Similar patterns can be seen in the behavior of other customers which shows that there is a relationship between clicks and purchases and it can be used to derive the consequential bond between clicks and purchases.

Use of Contextual Information (Location, Occasion, Season)

Unlike the traditional recommendation systems which collects users' preferences as ratings, context-aware systems also include the "contextual information" to understand users' preferences. Context represents a set of factors or situations under which a user interacted with an item: for example, time, location, surroundings, purpose of purchase, device and occasion while interacting with an item [6]. Each context factor can be characterized by a structure such as the time factor and can be described as seconds, minutes, hours, days, months and years. Such contextual information [109] can be gathered from users' implicit or explicit feedback: for example, a user's rating for a hotel during her summer vacation stay.

Structure for Extracting and Modeling Sequential Behavior

Different structures are used by sequential recommendation systems for extracting and modeling sequential behaviors. In traditional sequential recommendation systems utilizing sequence similarity matching (Sect. "Sequence Similarity Based"), the technique of longest common subsequence is used to find the matching sequences for recommendation to the target user, and for sequential pattern mining-based approaches (Sect. "Frequent Sequential Pattern Mining"), several algorithms are designed to mine sequential patterns from sequence databases such as GSP [21], SPADE [29], SPAM [23], Prefix Span [24] and PLWAP [25]. The input to these sequential pattern mining algorithms is a sequence database and a minimum support threshold (set by the user) and the output is the set of frequent sequential patterns. Given the same sequence database and the minimum support threshold, all sequential pattern mining algorithms will output the same set of sequential patterns, as these algorithms do not differ in the output but differ from each other according to the (1) type of search technique used, such as breadth first, e.g., GSP [21], depth first, e.g., SPADE [29] and SPAM [23], (2) the type of database representation, such as, horizontal (e.g., GSP [21] or vertical (e.g., SPADE [29] and SPAM [23] and (3) how the next patterns are determined and generated from the search space, such as, candidate generate and test, e.g., Apriori [110] and GSP [21], pattern growth using projected databases, e.g., Prefix Span [24]. Use of efficient strategies and data structures results in some algorithms being more efficient than others [9, 11]. For more details on structures for extracting sequential patterns using sequential pattern mining algorithms, we encourage the reader to follow the survey on sequential pattern mining by [9–11].

Factorization and latent representation-based approaches model user's sequential behavior by creating a model from the training data (observed user ratings) and then predicting the ratings for users on unseen items based on the observations learned from the trained model. Model-based methods apply machine learning techniques during the model training

Table 13 Historical purchase records of customers

UID	Clicks sequence	Purchases sequence
1	(1, 2, 3), (7, 5, 3), (1, 6), (6), (1, 5)	(1, 2), (3), (6), (7), (5)
2	(1, 4), (6, 3), (1, 2), (1, 2, 5, 6)	(1, 4), (3), (2), (1, 2, 5, 6)
3	(1, 5), (6, 5, 2), (6), (5)	(1), (2), (6), (5)
4	(2, 7), (6, 6, 7)	(2), (6, 7)

process. Model-based methods are assumed to capture the underlying relations from the data. Common approaches are matrix factorization [31] and Markov models [32–35]. The concept behind matrix factorization is that there should be some latent (hidden) features which determine the preference of a user toward an item. For example, two users may give high ratings to a certain book if they both like the writer of the book, or if the book is based on an historical event. In this case, the latent feature can be the "genre" which both users like. So, the latent feature "historic genre" can be used to describe both the user and the item. Thus, the discovery of such latent features can help to predict the rating of a particular user on a particular item because the features associated with both the user and the item are same. While determining those different features, an assumption is also made that the number of features k would be less than the number of users and the number of items. The Markov chain-based models [32–35] are successful probabilistic models that model sequential patterns by predicting user's future actions based on the past actions. To achieve this goal, an item-to-item transition probability matrix is estimated where items which are nearest (with high probability of transitioning) after the recently interacted item by the user are recommended: for example, recommending the item for purchase that has the highest transitioning probability after the last purchased item by the user as explained in Sect. "Matrix Factorization".

Deep neural network-based approaches [37] model user's sequential behavior capturing the comprehensive relations over different entities (e.g., users, items, interactions) in a sequence. The most commonly used neural networks for modeling sequential behavior are recurrent neural networks (RNN) [45–47, 94, 94] due to their natural strength in sequence modeling.

Given a sequence of historical user–item interactions, an RNN-based RS tries to predict the next possible interaction by modeling the sequential dependencies over the given interactions. A basic RNN is a standard feed-forward multilayer perceptron network (MLP) architecture (a network with single input, multiple hidden and single output layer) with addition of loops to the architecture. RNNs have internal memory which helps the model to store important things about the input and enables it to predict precisely about what is coming next. For this reason, RNNs are preferred for sequential data like time series, speech, text, financial data, audio, video and weather. They are designed to handle variable length sequence data. Several sequential recommender systems based on RNNs had been proposed to model users' sequential behavior [8, 44–47, 94, 96].

Sequences Having Long User–Item Interactions

A long user–item interaction sequence has a large number of user–item interactions which makes it challenging to model comprehensive and complex dependencies between multiple interactions that form the sequence. Two key features in long user–item interaction sequences are (i) learning higher-order sequential dependencies and (ii) learning long-term sequential dependencies.

Higher-Order Sequential Dependencies

Compared to the lower-order sequential dependencies, which are moderately simple and can be easily modeled by Markov chain models [32–35] or factorization machines [31], higher-order sequential dependencies are more complex and difficult to capture due to their complicated multi-level cascading dependencies crossing multiple user–item interactions. High-order sequential dependencies can be addressed through the use of higher-order Markov chain models [83] and recurrent neural networks [8, 44–47, 94]. However, each approach has its own limitations, for example, an increase in the order of Markov model also increases the number of states and the model complexity. Alternately, reducing the number of states leads to inaccurate transition probability matrix and limits the predictive power, whereas in case of RNN models, some sequential dependencies may not be modeled efficiently due to the assumption that any adjacent items in a sequence are highly dependent, which might not be true in the real world as a sequence may contain irrelevant or noisy interactions.

Long-Term Sequential Dependencies

In a sequence, long-term dependencies involve dependencies between interactions which are far from each other: for example, in a given shopping sequence $S = \{towel, eggs, bread, butter, soap\}$, which consists of items purchased successively by a user Smith. Obviously, the towel and the soap are highly dependent (in the context of being used for bath) besides even they are far from each other. These situations are quite common in real world, as users' behaviors tend to change with time and is highly uncertain, leading to placement of any items in any order in the cart. To address this, long short-term memory-based

Table 14 Classification of sequential recommender systems according to features

System	Sequences type	Side information (metadata) (users, items)	Consequential bond	Information	Structure to extract sequential patterns	Handling long user sequences	Flexible order sequences	Temporal characteristics (sequence granularity)
ChoiRec12 (Choi, K., Yoo, D., Kim, G., & Suh, Y. , 2012)	Single level	x	x	x	Sequential pattern analysis	x	x	x
(Yap, G. E., Li, X. L., Philip, S. Y., 2012)	Multi-level	x	x	x	Pattern growth (Prefix Span)	x	x	Yes(weighted backward and forward compatibility of sequences with the target sequence)
Hybrid (Salehi, M. , 2013)	Single level	Yes (Product Attributes)	x	x	Apriori(introduced weighted association rules)	x	x	Yes (more weight assigned to products purchased recently)
BSSM (Zhang, Y., & Cao, J., 2013)	Single level	x	x	x	Sequence similarity based on behavior sequence similarity	x	x	x

x means the presence of a particular feature in the system under discussion

[98] and gated recurrent unit-based RNN [44] have been used in sequential recommendation systems to capture the long-term dependencies among the user–item interactions in a sequence. However, it is easy for RNN models to generate false dependencies by overly assuming any adjacent items in a sequence are highly dependent. In the above example of Smith's shopping sequence, an RNN usually models S by assuming the butter and the soap are dependent due to the close distance between them, but actually they are not. Overall, the works that are able to tackle this are quite limited and need further investigation.

Sequences with Flexible Order

In real world, it is not necessary that all adjacent user interactions in a sequence are dependent sequentially. For instance, in a shopping sequence $S = \{ butter, eggs, cheese, bread \}$, the order of purchase of bread, eggs and cheese is not important; however, the collective purchase of *butter*, *eggs* and *cheese* leads to an increased probability to purchase *bread*. This shows that while there is no strict order among *butter*, *eggs* and *cheese*, the purchase of *bread* will depend sequentially on their union. Hence, in a flexible order sequence, it is important to capture collective sequential dependencies instead of pointwise dependencies (which do not consider a strict order) between user–item interactions. Existing sequential recommendation systems designed based on Markov chains, factorization machines or RNNs are good at handling the pointwise dependencies, but are not very good at modeling and capturing collective dependencies.

Temporal Patterns

Temporal databases integrate the concept of time to capture past, present and future data. Various forms of temporal data can include (i) start time (when the event actually began) and end time (when the event ended): for example, when a customer started his session by navigating an on line e-commerce site and ended the session after purchase or without a purchase by leaving the site, (ii) granularity (for example, events occurring on the same day or happening with N weeks from a specific day or in a specific month, etc.) and (iii) periodicity, which refers to regular repetition of a certain event within a specific time interval. For instance, the event "fall" is repeated once within the time interval of a year. Such events are called periodic events and such intervals are called periodic intervals [111].

Data are collected in the form of event time sequences where each event lasts for a certain time interval. Each

Table 15 Classification of sequential recommender systems according to features (continued from Table 14)

System	Sequences type	Side information (metadata) (users, items)	Consequential bond	Contextual information	Structure to extract sequential patterns	Handling long user sequences	Flexible order sequences	Temporal characteristics (sequence granularity)
Wei Song Kai Yang, 2014)	Single level	x	x	x	Weighted sequence matching	x	x	x
SainiRec17 (Saini, S., Saumya, S., and Singh, J. P., 2017)	Single level	x	x	x	Vertical Id list based (SPADE)	x	x	Yes (monthly basis)
Hybrid (Gurbanov, T., and Ricci, F. (2017)	Single-level collection of various actions on the event (click, bookmark)	x	x	x	Probabilistic (Bayes and naïve Bayes)	x	x	Yes (timeless and time-aware sessions)represents time gap between action a_i and action a_{i+1} on the same item
HPCRec18 (Xiao, Y., and Ezeife, C. I. 2018)	Single level	x	Yes (without considering sequentiality)	x	Consequential bond similarity computation between clickstream and purchase sequences	x	x	x

x means the presence of a particular feature in the system under discussion

Table 16 Classification of sequential recommender systems according to features (continued from Table 15)

System	Sequences type	Side information (metadata) (users, items)	Consequential bond	Contextual information	Structure to extract sequential patterns	Handling long user sequences	Flexible order sequences	Temporal characteristics (sequence granularity)
HSPRec19 (Bhatta et al. 2019)	Single level	x	Yes	x	GSP (generalized sequential patterns)	x	x	Yes (Creating sequences with granularity of daily, weekly and monthly purchases)
SEMSRec20 (Nasir, M., Ezeife, C. I., 2020)	Single-level	Yes (customers' reviews)	Yes	x	Pattern growth (PrefixSpan), use of semantics during mining	x	x	Yes (creating sequences with granularity of daily, weekly and monthly purchases)
SEMSRec21 (Nasir, M., Ezeife, C. I., Gidado, Abdulrauf,2021)	Single level	Yes (customers' reviews & product metadata)	Yes	x	Pattern growth (PrefixSpan) Use of semantics during mining	Yes	x	Yes (Creating sequences with granularity of daily, weekly and monthly purchases)

x means the presence of a particular feature in the system under discussion

Table 17 Data set statistics

Data set/statistics		Amazon			Online retail
		Fashion	Movies and TV	Beauty	
Reviews (Amazon) and purchases (Online Retail) Data	Total no. of transactions	883636	8290109	353956	240007
	No. of unique users	743216	3755907	317982	2974
	No. of unique items	186054	181996	32586	3282
	Avg No. of reviews (purchases per item)	4.66	45.55	10.86	58.10
	Max. sequence length	40	4036	23	294
Product metadata	No. of unique items	186637	203970	32992	4497

record in the database stores the start time and end time during which the tuple is valid. For instance, for an e-commerce business, customers' purchase records are stored to determine customers' purchase behavior over a certain period of time: for example, records like "Customer *A* had purchased items *X* and *Y* on 20 August" or Customer *A* visited the site from 13:00 to 14:00 on May 11 and purchased items *X*, *Y* and *Z* are stored. The temporal nature of data provides valuable information about varying trends or patterns. For example, we can find patterns like "80% of the customers who bought item *X* and then bought item *Y* within an hour are likely to visit the site the following day to view new promotions". Such frequent temporal patterns from customers are useful to identify correlations between items for further marketing and promotion strategies.

Incorporating temporal aspect is an important extension as it offers the capability to infer causal and temporal proximity relationships. The time component helps in analyzing the changes in data overtime. The time component may facilitate in identifying the validity of rules like $hiking\ gear \rightarrow hiking\ boots$, $years : months(5 : 3)$ during $\{ years(2015), years(2020) \}$ which reveals that every spring time from 2015 to 2020, customers who bought hiking gear also purchased hiking boots. Such a rule may not be valid before 2015 or after 2020. Therefore, by adding the temporal information to the rule set, more accurate and clear information is obtained. Furthermore, it is possible to predict how quickly a domain changes by discovering the change in knowledge obtained from the underlying data, thus leading to better marketing strategies.

A classification of sequential recommender systems is presented in Tables 14, 15 and 16 according to the categories presented in the taxonomy and based on the eight features explained above.

Experimental Results and Analysis

In this section, we present our experimental setup, followed by results and analysis.

Data Sets and Implementation Details

1. Online Retail[5]: This data set represents purchases from a UK-based company, which sells unique all-occasion gifts. The data include purchases between an 8-month time period, i.e., between 01/12/2010 and 09/12/2011.
2. Amazon[6]: Amazon data set includes data for various product categories where each category includes data about (i) customer reviews such as ratings, text and helpfulness votes), (ii) product metadata including product description, category information, price, brand, image, and (iii) links such as products also viewed/also bought graphs.

To test various surveyed sequential recommender systems, we selected the categories fashion, beauty, movies and TV and for each category, the data set was K-core, i.e., all items must have at least K reviews (where $K = 5$). Table 17 presents statistics about the data set. For experimental purposes, we utilized the implementation provided from the respective authors and kept all parameters at their optimal settings as done by their authors to have a fair comparison.

Preprocessing and Hyper-parameter Tuning

To partition the data set, we used methods of (i) leave one out and (ii) temporal user splitting, where the former keeps the most recent sequence of each user for testing and the remaining sequences form the training data set, and the latter approach considers a percentage of user's last interactions for testing purpose. A user–item interaction is indicated by the presence of a purchase record in the Online Retail data and the availability of a review or a rating in the Amazon data set. To determine the sequential order of actions, time stamps were used and purchases of customers grouped to create purchase sequences based on these time stamps.

[5] https://archive.ics.uci.edu/ml/datasets/online+retail.

[6] http://jmcauley.ucsd.edu/data/amazon/.

Table 18 Comparative performance analysis of different non-sequential and sequential recommender systems

Data set	K	Metric	POP[1]	BPR[1]	HSPRec[2]	FPMC[3]	(GRU4Rec)[4]	SASRec[4]	Caser[4]	SEMSRec[4]
Online retail	1	Prec@1	0.0084	0.0324	0.0656	0.0054	0.0961	0.1599	0.1524	0.1617
		Recall@1	0.0084	0.0324	0.0214	0.0054	0.0961	0.0281	0.0286	0.0214
		MRR@1	0.0084	0.0324	0.0656	0.0054	0.0961	0.1432	0.1317	0.1617
	5	Prec@5	0.0051	0.0214	0.0403	0.0025	0.0425	0.1139	0.1189	0.1231
		Recall@5	0.0025	0.0675	0.0561	0.0128	0.2126	0.0461	0.0321	0.0561
		MRR@5	0.0149	0.0423	0.0974	0.0079	0.1380	0.2011	0.1924	0.2196
	10	Prec@10	0.0043	0.0117	0.0334	0.0018	0.0279	0.0942	0.0899	0.1059
		Recall@10	0.0430	0.1223	0.0750	0.0189	0.2796	0.0699	0.0601	0.0750
		MRR@10	0.0171	0.0912	0.1026	0.0087	0.1471	0.2154	0.2032	0.2275

[1] Non-sequential RS

[2] Sequential RS

[3] Factorization-based RS

[4] Deep neural network-based RS

These purchase sequences constitute the training data set for the leave-one-out method and the test data set comprised the most recent purchase sequence of each customer to evaluate the model's performance. For temporal user splitting, a standard train and test splits of (i)70%, 30% and (ii) 80%, 20% were used and users having at least five interactions (e.g., purchase records, ratings/reviews) were selected.

Evaluation Metrics

We evaluated the discussed models on standard metrices including Precision@K, Recall@K and mean reciprocal rank (MRR). Table 18 shows the comparative performance analysis of different sequential recommendation systems. The different evaluation metrices used for performance evaluation are defined as:

1. *Precision@K:* represents the proportion of items recommended in the Top-K set that are also relevant.
2. *Recall@K:* represents the proportion of relevant items in the Top-K recommendations.
3. *Mean Reciprocal Rank (MRR):* for a sample of ground truths R, it is the average of the reciprocal ranks of results and $rank_i$ refers to the rank position of the first relevant item for the $i-th$ ground truth.

Sequential Recommendation Systems' Performance Comparison

To show a comparative analysis between some sequential recommenders from the presented surveyed systems, we selected some baseline systems from each surveyed category.

The first group represents general recommendation (non-sequential) methods based on user feedback.

1. *Popularity Based (POP)*: The items' popularity is used for item ranking the items where popularity is determined by the number of user interactions.
2. *Baysian Personalized Ranking (BPR)* [112]: A method based on matrix factorization technique for non-sequential item recommendation on implicit feedback.

The second group includes sequential recommenders from (i) traditional approaches (e.g., sequential pattern mining based) and (ii) factorization and latent representation based (e.g., first-order Markov chains, which consider the last visited item).

3. *Historical Purchase Click (HPCRec)* [20] *and Historical Sequential Purchase (HSPRec)* [16]: It mines frequent sequential click and purchase patterns by utilizing the consequential bond between click and purchase sequences and then using this quantitatively and qualitatively rich matrix for collaborative filtering to provide better recommendations.
4. *Factorized Personalized Markov Chain (FPMC)* [34]: A hybrid method that is designed using matrix factorization and Markov chains where matrix factorization learn users' general taste by factorizing the matrix on user–item preferences and Markov models are used to predict users' sequential behavior by building an item transition graph.

The third group involves sequential recommender systems from deep neural network-based approaches, which include various or all previously visited items.

5. *GRU4Rec* [94]: They proposed gated recurrent unit (GRU)-based recurrent neural network (RNN) and modeled sequential dependencies for session-based recommendation (GRU4Rec).

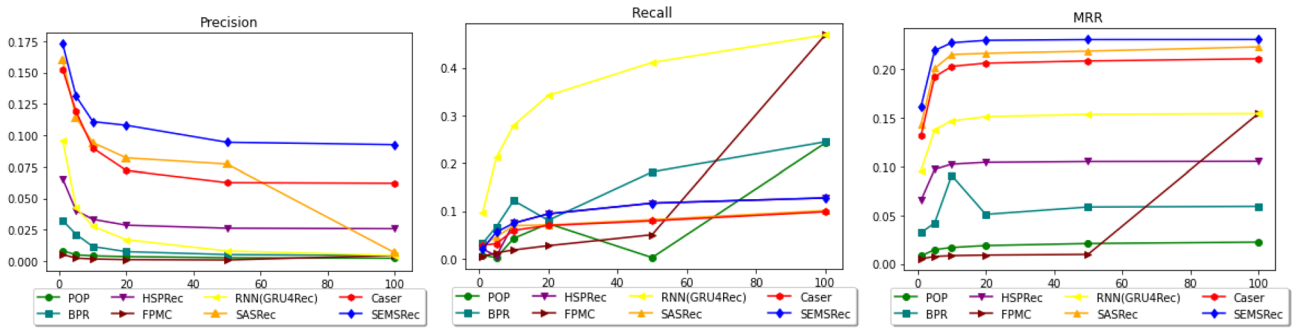

Fig. 10 Performance comparison of different sequential recommender systems on the basis of precision, recall and MRR

6. *Convolutional Sequence Embeddings Caser* [48]: A convolutional neural network (CNN)-based method provides sequential recommendation by applying convolution operations on the embedding matrix of L most recent items.
7. *Self-Attentive Sequential (SASRec)* [113]: It uses the attention mechanism to capture long-term user preferences.

Results and Analysis

Sequential recommenders such as SASRec [113], Caser [48]), HSPRec19 [16] and RNN [94] exhibited high precision and recall measures which indicate the importance of learning users' sequential behavior to capture variations in user's interests and improving the product recommendation quality.

Another important observation leads to the result that amplifying the sequential purchase patterns through integration of products' metadata to learn semantic knowledge about products and then utilizing this knowledge to compute similarities among products [71, 72] significantly improve model performance and provide recommendations that are of interest to the user. On the other hand, least performance (lowest precision and recall score) was shown from factorized personalized Markov chain (FPMC) [34] model, which indicated that it is unable to effectively capture the sequential purchase behaviors of customers. A slight increase in the performance was shown by SEMSRec [71], which indicated that by incorporating products' semantic knowledge for computing product similarities, the quality of recommendations can be improved as it helps to capture similarities between products' purchased in the past and other available products (e.g., items in the catalog that are relevant to the user but not purchased before). A comparative performance of non-sequential and sequential recommenders under various categories as presented in this survey is shown in Fig. 10. The performance is evaluated using precision, recall and mean reciprocal rank metrices.

Conclusions and Future Research Directions

We presented a taxonomy of sequential recommendation systems in the literature with e-commerce product recommendation as an application. This paper presented a taxonomy for classifying sequential recommender systems under three main categories as: (i) traditional approaches (sequence similarity, frequent pattern mining and sequential pattern mining), (ii) factorization and latent representation methods (matrix factorization and Markov models) and (iii) neural network-based methods (deep neural network, advanced models). The classification tends to enhance the understanding of sequential recommendation systems and provides solutions and future research directions in this area. The paper also provides a classification of the presented systems according to eight important key features supported by the techniques and discusses their limitations as well. A comparative performance analysis of many of the sequential recommendation systems on publicly available e-commerce data sets (Amazon and Online Retail) using several evaluation metrics (precision, recall and mean reciprocal rank) is also presented.

Research on sequential recommender systems has gained attention in the past several years. While summarizing and categorizing the various research directions followed, we ascertain further open research directions including:

1. *Context Adaptation in Sequential Recommender Systems*: Estimating the current context for a user to understand its preferences can greatly impact the quality of recommendation and therefore increase user satisfaction. For example, a context may represent a set of factors or situations under which a user interacted with an item such as time, location, surroundings, purpose of purchase, device and occasion while interacting with an item. Knowing the context in sequential recommender systems is more essential as user's intent are short term and may evolve quickly with time. For

example, in an e-commerce platform, it is important to determine if the user is going over the catalog to find different range of options or she is interested in just reviewing shortlisted candidate items for purchase. However, most existing sequential recommendation systems ignore this significant aspect. Hence, further work in this direction can be promising.

2. *Social Influence in Sequential Recommender Systems*: Users tend to trust more on the recommendations from their friends (social network) than recommendations by unknown people. Collecting social information of users such as facebook friends, followers on twitter, gathering preferences of their friends from social networks and then utilize their ratings on items to generate recommendations for the target user needs to be taken into account in Sequential Recommendation Systems. A common example could be recommending movie to watch based on the movies recently watched by target user's friends or recommending products by collecting information on products bought by target users' friends and are similar to her interest.
3. *Cross-Domain Sequential Recommender Systems*: Cross-domain recommendation aims to leverage data in different domains by transferring knowledge from the source domain to the target domain (for example, recommending books on the basis of movies watched by the user), hence amplifying the target domains' recommendation performance. Future research can investigate deep learning approaches to determine the characteristics of sequential data in the source and target domains and design sequential recommendation systems generating cross-domain recommendations.
4. *Explainability in sequential recommender systems*: Recently, most advanced neural network and deep learning-based models are deemed as a black box and lack explainability for users and model practitioners. Therefore, designing sequential recommendation systems which are explainable is significant, as without knowing the reasons behind recommendations, users' may not trust these and hence do not take an action accordingly (e.g., purchasing an item). Similarly, it is vital for the practitioners to understand the influence of various factors such as data, features and other model hyperparameters on the model output (i.e., recommendations).
5. *Incorporating General Trends and Additional Data*: Usually, few specific types of user interactions (e.g., items views, clicks and purchases) are considered while creating users' profile (recording its preferences). However, in real world, other rich information sources are available relevant to items (e.g., add to favorites, add to wishlist, add to cart, trending items) and users (e.g., navigation across different categories, purchase behaviors during several occasions). Focus on these additional information sources can also impact the quality of recommendations.
6. *Comprehensive and Standardized Evaluations Across Different Models*: There has been a debate that only complex and advanced deep learning models cannot always guarantee better and more robust recommender systems. Additionally, one critical issue for evaluation in sequential recommendation systems is the lack of effective standardized benchmarks. Therefore, it is imperative to lay emphasis on benchmarking study for standardized evaluations.
7. *Domain-Specific Sequential Recommendation*: Most of the research conducted for sequential recommendation is considered to be applicable to all domains (e.g., music, news, movies, product recommendation). However, to address the needs for real world applications, more focus can be given to design recommendation algorithms for a particular domain based on domain-specific features and hence have common data sets and standard baselines for comparison. For example, in case of movie recommendation, common features include "genre", "artists", "director", "writer","release year", "awards", etc.; however, in case of e-commerce domain for product recommendation, product features vary not just in comparison to other domains, but also between various product categories, as most of the product information (features) areembedded in the text descriptions. For example, consider the description of a baby girl clothing with description as " Cute Blue Red Cranberry Taped Girl Ruffle long Sleeve Sunflower Dress", so here "Cranberry" and "Sunflower" may refer to a "flavor" , an "ingredient' or simply a "color". Accurate identification of these features is important to generate accurate and relevant recommendations.
8. *Inclusion of Products' Semantic Knowledge*: Extracting products' semantic knowledge (e.g., using textual features and context) and then including it in the sequential recommendation process can improve the recommendation process. Nasir et al. [71, 72] proposed using customers' purchase history and products' metadata (e.g., title, description and brand) and then extract products' sequential and semantic knowledge according to their (a) usage (e.g., products co-purchased or co-reviewed) and (b) textual features by finding similarity between products based on their characteristics considering the context of items' usage.
9. *Exploring Consequential Bond Between Customers' Different Sequential Interactions*: As stated in Sect. "Features to classify sequential recommendation systems", consequential bond is useful to find similarities between customers' click and purchase sequences.

This relationship (consequential bond) can be further extended to other user–item interactions including items viewed, added to cart or added to wishlist to further explore sequential dependencies between users' various interactions and hence recommend items of interest.

10. *Mining Historical (Long-Term) and Short-Term User Preferences*: Considering the shift in users' interest and preferences, it will be good to consider users' historical (long-term) records (e.g., past purchases) along with their current interests (e.g., short-term user–item interactions). This can facilitate in learning their static (long-term) preferences (e.g., a particular clothing brand/style) and dynamic short-term intent (e.g., a particular color). Models integrating these can lead to improved recommendations tailored to customers' needs.

Funding This research was supported by the Natural Science and Engineering Research Council (NSERC) of Canada under an operating grant (OGP-0194134) and a University of Windsor grant received by Dr. C. I. Ezeife.

Data Availability The dataset that is generated or analysed during this study are included in the published article.

Declarations

Conflict of Interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Aggarwal CC. Recommender systems: the textbook. In: Aggarwal CC, editor. Cham: Springer International Publishing; 2016. p. 1–28. https://doi.org/10.1007/978-3-319-29659-3_1.
2. Ricci F, Rokach L, Shapira B. Recommender systems handbook. In: Ricci F, Rokach L, Shapira B, editors. US: Springer; 2015. p. 1–34. https://doi.org/10.1007/978-1-4899-7637-6_1.
3. Schafer JB, Konstan JA, Riedl J. E-commerce recommendation applications. Data Min Knowl Discov. 2001;5(1):115–53. https://doi.org/10.1023/A:1009804230409.
4. Jannach D, Zanker M, Felfernig A, Friedrich G. Recommender systems: an introduction. Cambridge: Cambridge University Press; 2010.
5. Schafer JB, Frankowski D, Herlocker J, Sen S. The adaptive web: methods and strategies of web personalization. In: Brusilovsky P, Kobsa A, Nejdl W, editors. Lecture notes in computer science. Berlin: Springer; 2007. p. 291–324. https://doi.org/10.1007/978-3-540-72079-9_9.
6. Adomavicius G, Tuzhilin A. Recommender systems handbook. In: Ricci F, Rokach L, Shapira B, Kantor PB, editors. Boston: Springer; 2011. p. 217–53. https://doi.org/10.1007/978-0-387-85820-3_7.
7. Sinha BB, Dhanalakshmi R. Evolution of recommender system over the time. Soft Comput. 2019;23(23):12169–88.
8. Smirnova E, Vasile F. Proceedings of the 2nd workshop on deep learning for recommender systems. DLRS. 2017;2017:2–9. https://doi.org/10.1145/3125486.3125488.
9. Mabroukeh NR, Ezeife CI. A taxonomy of sequential pattern mining algorithms. ACM Comput Surv. 2010;43(1):1–41. https://doi.org/10.1145/1824795.1824798.
10. Mooney CH, Roddick JF. Sequential pattern mining—approaches and algorithms. ACM Comput Surv. 2013;45(2):1–39. https://doi.org/10.1145/2431211.2431218.
11. Fournier-Viger P, Lin JCW. A Survey of Sequential Pattern Mining p. 24.
12. Wang S, Cao L, Wang Y, Sheng QZ, Orgun M, Lian D. A Survey on Session-based Recommender Systems. arXiv:1902.04864 [cs] (2021).
13. Fang H, Guo G, Zhang D, Shu Y. International conference on web engineering. Cham: Springer; 2019. p. 574–7.
14. Zhang S, Yao L, Sun A, Tay Y. Deep learning based recommender system: a survey and new perspectives. ACM Comput Surv. 2019;52(1):1–38.
15. Mu R. A survey of recommender systems based on deep learning. Ieee Access. 2018;6:69009–22.
16. Bhatta R, Ezeife CI, Butt MN. Big data analytics and knowledge discovery. In: Ordonez C, Song IY, Anderst-Kotsis G, Tjoa AM, Khalil I, editors. Lecture notes in computer science. Cham: Springer International Publishing; 2019. p. 57–72. https://doi.org/10.1007/978-3-030-27520-4_5.
17. Yap GE, Li XL, Yu PS. Database systems for advanced applications. In: Lee SG, Peng Z, Zhou X, Moon YS, Unland R, Yoo J, editors. Lecture notes in computer science. Berlin: Springer; 2012. p. 48–64. https://doi.org/10.1007/978-3-642-29035-0_4.
18. Su Q, Chen L. A method for discovering clusters of e-commerce interest patterns using click-stream data. Electron Commer Res Appl. 2015;14(1):1–13. https://doi.org/10.1016/j.elerap.2014.10.002.
19. Cheng W, Yin G, Dong Y, Dong H, Zhang W. Collaborative filtering recommendation on users' interest sequences. PLOS ONE. 2016;11(5):e0155739. https://doi.org/10.1371/journal.pone.0155739.
20. Xiao Y, Ezeife CI. Big data analytics and knowledge discovery. In: Ordonez C, Bellatreche L, editors. Lecture notes in computer science. Cham: Springer International Publishing; 2018. p. 70–82. https://doi.org/10.1007/978-3-319-98539-8_6.
21. Agrawal R, Srikant R. In: Proceedings of the Eleventh International Conference on Data Engineering, 1995. p. 3–14. https://doi.org/10.1109/ICDE.1995.380415.
22. Kim YS, Yum BJ. Recommender system based on click stream data using association rule mining. Expert Syst Appl. 2011;38(10):13320–7. https://doi.org/10.1016/j.eswa.2011.04.154.
23. Ayres J, Flannick J, Gehrke J, Yiu T. Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. New York: Association for Computing Machinery; 2002, KDD' 02. p. 429–35. https://doi.org/10.1145/775047.775109.
24. Pei Jian, Han Jiawei, Mortazavi-Asl B, Pinto H, Chen Qiming, Dayal U, Hsu Mei-Chun. In: Proceedings 17th International Conference on Data Engineering (IEEE Comput. Soc, Heidelberg, Germany), 2001. p. 215–224. https://doi.org/10.1109/ICDE.2001.914830.http://ieeexplore.ieee.org/document/914830/.
25. Ezeife CI, Lu Y, Liu Y. In: Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations. New York: Association for Computing

Machinery; 2005, OSDM' 05. p. 26–35. https://doi.org/10.1145/1133905.1133910.
26. Choi K, Yoo D, Kim G, Suh Y. A hybrid online-product recommendation system: combining implicit rating-based collaborative filtering and sequential pattern analysis. Electron Comm Res Appl. 2012;11(4):309–17. https://doi.org/10.1016/j.elerap.2012.02.004.
27. Saini S, Saumya S, Singh JP. Computer information systems and industrial management. In: Saeed K, Homenda W, Chaki R, editors. Lecture notes in computer science. Cham: Springer International Publishing; 2017. p. 366–75. https://doi.org/10.1007/978-3-319-59105-6_31.
28. Salehi M, Nakhai Kamalabadi I. Hybrid recommendation approach for learning material based on sequential pattern of the accessed material and the learner's preference tree. Knowl-Based Syst. 2013;48:57–69. https://doi.org/10.1016/j.knosys.2013.04.012.
29. Zaki MJ. SPADE: an efficient algorithm for mining frequent sequences. Mach Learn. 2001;42(1):31–60. https://doi.org/10.1023/A:1007652502315.
30. Kim YS, Yum BJ, Song J, Kim SM. Development of a recommender system based on navigational and behavioral patterns of customers in e-commerce sites. Expert Syst Appl. 2005;28(2):381–93. https://doi.org/10.1016/j.eswa.2004.10.017.
31. Koren Y. Predings of the 15th ACM SIGKDD international cference on Knowledge discovery and data mining. New York: Association for Computing Machinery; 2009. p. 447–56.
32. Bernhard SD, Leung CK, Reimer VJ, Westlake J. Proceedings of the 20th international database engineering and applications symposium. New York: Association for Computing Machinery; 2016. p. 24–33. https://doi.org/10.1145/2938503.2938535.
33. Brafman RI, Heckerman D, Shani G. Recommendation as a Stochastic Sequential Decision Problem p. 10.
34. Rendle S, Freudenthaler C, Schmidt-Thieme L. Procdings of the 19th international confece on World wide web. New York: Association for Computing Machinery; 2010. p. 811–20.
35. Shani G, Heckerman D, Brafman RI. An MDP-based recommender system. J Mach Learn Res. 2005;6:1265–95.
36. Zimdars A, Chickering DM, Meek C. Using Temporal Data for Making Recommendations. 2013. arXiv:1301.2320 [cs].
37. Nielsen M. Neural Networks and Deep Learning p. 224.
38. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Van Den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M, et al. Mastering the game of go with deep neural networks and tree search. Nature. 2016;529(7587):484–9.
39. Zhu J, Shan Y, Mao J, Yu D, Rahmanian H, Zhang Y. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, 2017. p. 1703–1711.
40. Guo H, Tang R, Ye Y, Li Z, He X. Deepfm: a factorization-machine based neural network for ctr prediction. 2017. arXiv preprint arXiv:1703.04247.
41. Hinton GE, Osindero S, Teh YW. A fast learning algorithm for deep belief nets. Neural Comput. 2006;18(7):1527–54.
42. Wan S, Lan Y, Wang P, Guo J, Xu J, Cheng X. In: RecSys Posters, 2015.
43. Wang P, Guo J, Lan Y, Xu J, Wan S, Cheng X. In: Proceedings of the 38th International ACM SIGIR conference on Research and Development in Information Retrieval, 2015. p. 403–412.
44. Hidasi B, Quadrana M, Karatzoglou A, Tikk D. In: Proceedings of the 10th ACM Conference on Recommender Systems (Association for Computing Machinery, New York, NY, USA), RecSys '16, 2016. p. 241–248. https://doi.org/10.1145/2959100.2959167.
45. Quadrana M, Karatzoglou A, Hidasi B, Cremonesi P. In: Proceedings of the Eleventh ACM Conference on Recommender Systems (Association for Computing Machinery, New York, NY, USA), RecSys '17, 2017. p. 130–137. https://doi.org/10.1145/3109859.3109896.
46. Tan YK, Xu X, Liu Y. In: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (Association for Computing Machinery, New York, NY, USA), DLRS 2016, 2016. p. 17–22. https://doi.org/10.1145/2988450.2988452.
47. Christakopoulou K, Beutel A, Li R, Jain S, Chi EH. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (Association for Computing Machinery, New York, NY, USA), KDD '18, 2018. p. 139–148. https://doi.org/10.1145/3219819.3219894.
48. Tang J, Wang K. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. 2018. arXiv:1809.07426 [cs].
49. Yuan F, Karatzoglou A, Arapakis I, Jose JM, He X. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, 2019. p. 582–590.
50. Tuan TX, Phuong TM. In: Proceedings of the eleventh ACM conference on recommender systems, 2017. p. 138–146.
51. Wu S, Tang Y, Zhu Y, Wang L, Xie X, Tan T. Proceedings of the AAAI conference on artificial intelligence. 2019;33:346–53.
52. Guo W, Wang S, Lu W, Wu H, Zhang Q, Shao Z. In: 2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA), IEEE, 2021. p. 1–10.
53. Xiao Y, Yao L, Pei Q, Wang X, Yang J, Sheng QZ. Mgnn: mutualistic graph neural network for joint friend and item recommendation. IEEE Intell Syst. 2020;35(5):7–17.
54. Zhong M, Li C, Wen J, Liu L, Ma J, Zhang G, Yang Y. Hignet: hierarchical and interactive gate networks for item recommendation. IEEE Intell Syst. 2020;35(5):50–61.
55. Zhou J, Cui G, Hu S, Zhang Z, Yang C, Liu Z, Wang L, Li C, Sun M. Graph neural networks: a review of methods and applications. AI Open. 2020;1:57–81.
56. Wang Q, Yin H, Hu Z, Lian D, Wang H, Huang Z. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018. p. 2467–2475.
57. Wu C, Yan M. In: Proceedings of the 2017 ACM on conference on information and knowledge management, 2017. p. 2379–2382.
58. Li J, Ren P, Chen Z, Ren Z, Lian T, Ma J. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017. p. 1419–1428.
59. Wang S, Hu L, Cao L, Huang X, Lian D, Liu W. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2018;32.
60. Bai T, Nie JY, Zhao WX, Zhu Y, Du P, Wen JR. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 2018. p. 1201–1204.
61. Kang WC, McAuley J. In: 2018 IEEE International Conference on Data Mining (ICDM) (IEEE), 2018. p. 197–206.
62. Sun F, Liu J, Wu J, Pei C, Lin X, Ou W, Jiang P. In: Proceedings of the 28th ACM international conference on information and knowledge management, 2019. p. 1441–1450.
63. Zhang S, Tay Y, Yao L, Sun A. Next item recommendation with self-attention. 2018. arXiv preprint arXiv:1808.06414.
64. Chen X, Xu H, Zhang Y, Tang J, Cao Y, Qin Z, Zha H. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (ACM, Marina Del Rey CA USA), 2018. p. 108–116. https://doi.org/10.1145/3159652.3159668.
65. Huang J, Zhao WX, Dou H, Wen JR, Chang EY. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 2018. p. 505–514.
66. Tang J, Belletti F, Jain S, Chen M, Beutel A, Xu C, Chi EH. In: The World Wide Web Conference, 2019. p. 1782–1793.

67. Kang WC, Wan M, McAuley J. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, 2018. p. 1143–1152.
68. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial nets. Adv Neural Inform Process Syst. 2014;27.
69. Zheng N, Song X, Chen Z, Hu L, Cao D, Nie L. In: Proceedings of the 27th ACM International Conference on Multimedia, 2019. p. 266–274.
70. Wang C, Niepert M, Li H. Recsys-dan: discriminative adversarial networks for cross-domain recommender systems. IEEE transactions on neural networks and learning systems. 2019;31(8):2731–40.
71. Nasir M, Ezeife CI. In: 2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), 2020. p. 270–274. https://doi.org/10.1109/ASONAM49781.2020.9381352. ISSN: 2473-991X.
72. Nasir M, Ezeife CI, Gidado A. Improving e-commerce product recommendation using semantic context and sequential historical purchases. Soc Netw Anal Min. 2021;11(1):82. https://doi.org/10.1007/s13278-021-00784-6.
73. Zhang Y, Cao J. Behavior and Social Computing. Springer; 2013. p. 165–77.
74. Gurbanov T, Ricci F. In: Proceedings of the Symposium on Applied Computing, 2017. p. 1655–1661.
75. Song W, Yang K. Practical Applications of Intelligent Systems. Springer; 2014. p. 657–66.
76. Rudin C, Letham B, Salleb-Aouissi A, Kogan E, Madigan D. In: Proceedings of the 24th annual conference on learning theory (JMLR Workshop and Conference Proceedings), 2011. p. 615–634.
77. Parameswaran AG, Koutrika G, Bercovitz B, Garcia-Molina H. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, 2010. p. 87–98.
78. Liu DR, Lai CH, Lee WJ. A hybrid of sequential rules and collaborative filtering for product recommendation. Inf Sci. 2009;179(20):3505–19.
79. Mobasher B, Dai H, Luo T, Nakagawa M. Discovery and evaluation of aggregate usage profiles for web personalization. Data Min Knowl Disc. 2002;6(1):61–82.
80. Zeng Z, Lin J, Li L, Pan W, Ming Z. Next-item recommendation via collaborative filtering with bidirectional item similarity. ACM Trans Inform Syst. 2019;38(1):1–22.
81. Pasricha R, McAuley J. In: Proceedings of the 12th ACM Conference on Recommender Systems, 2018. p. 63–71.
82. Wan M, McAuley J. In: Proceedings of the 12th ACM conference on recommender systems, 2018. p. 86–94.
83. He R, McAuley J. In: 2016 IEEE 16th International Conference on Data Mining (ICDM) (IEEE, Barcelona, Spain), 2016. p. 191–200. https://doi.org/10.1109/ICDM.2016.0030. http://ieeexplore.ieee.org/document/7837843/.
84. Lian D, Xie X, Chen E. Discrete matrix factorization and extension for fast item recommendation. IEEE Trans Knowl Data Eng. 2019;33(5):1919–33.
85. Zhao G, Lee ML, Hsu W, Chen W. In: Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval, 2012, p. 165–174.
86. Tavakol M, Brefeld U. In: Proceedings of the 8th ACM Conference on Recommender Systems, 2014. p. 33–40.
87. Sahoo N, Singh PV, Mukhopadhyay T. A hidden markov model for collaborative filtering. MIS quarterly, 2012;1329–1356.
88. Deshpande M, Karypis G. Selective Markov models for predicting web page accesses. ACM Trans Internet Technol. 2004;4(2):22.
89. Dziugaite GK, Roy DM. Neural network matrix factorization. 2015. arXiv preprint arXiv:1511.06443.
90. Lian J, Zhang F, Xie X, Sun G. In: Proceedings of the 26th international conference on World Wide Web companion, 2017. p. 817–818.
91. Wang X, He X, Nie L, Chua TS. In: Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval, 2017. p. 185–194.
92. Xue HJ, Dai X, Zhang J, Huang S, Chen J. in IJCAI, vol. 17. Melbourne: Australia; 2017. p. 3203–9.
93. Zhang S, Yao L, Sun A, Wang S, Long G, Dong M. Neurec: On nonlinear transformation for personalized ranking. 2018. arXiv preprint arXiv:1805.03002.
94. Hidasi B, Karatzoglou A, Baltrunas L, Tikk D. Session-based Recommendations with Recurrent Neural Networks. 2016. arXiv:1511.06939 [cs].
95. Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. (2015). Session-based recommendations with recurrent neural networks. arXiv preprint arXiv:1511.06939.
96. Villatel K, Smirnova E, Mary J, Preux P. Recurrent Neural Networks for Long and Short-Term Sequential Recommendation. 2018. arXiv:1807.09142 [cs, stat].
97. Donkers T, Loepp B, Ziegler J. In: Proceedings of the eleventh ACM conference on recommender systems, 2017. p. 152–160.
98. Wu CY, Ahmed A, Beutel A, Smola AJ, Jing H. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (Association for Computing Machinery, New York, NY, USA), WSDM '17, 2017. p. 495–503. https://doi.org/10.1145/3018661.3018689.
99. Yu F, Liu Q, Wu S, Wang L, Tan T. In: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, 2016. p. 729–732.
100. Hsu KC, Chou SY, Yang YH, Chi TS. Neural network based next-song recommendation. 2016. arXiv preprint arXiv:1606.07722.
101. Sachdeva N, Manco G, Ritacco E, Pudi V. In: Proceedings of the twelfth ACM international conference on web search and data mining, 2019. p. 600–608.
102. Ying H, Zhuang F, Zhang F, Liu Y, Xu G, Xie X, Xiong H, Wu J. In: IJCAI International Joint Conference on Artificial Intelligence, 2018.
103. Q. Liu, Y. Zeng, R. Mokhosi, H. Zhang, in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2018), pp. 1831–1839
104. Ren P, Chen Z, Li J, Ren Z, Ma J, De Rijke M. Proceedings of the AAAI conference on. Artif Intell. 2019;33:4806–13.
105. Sachdeva N, Gupta K, Pudi V. In: Proceedings of the 12th ACM Conference on Recommender Systems, 2018. p. 417–421.
106. Huang J, Zhao WX, Dou H, Wen JR, Chang EY. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (ACM, Ann Arbor MI USA), 2018. p. 505–514. https://doi.org/10.1145/3209978.3210017.
107. Wang S, Hu L, Wang Y, Sheng QZ, Orgun M, Cao L. International Joint Conference on Artificial Intelligence. International Joint Conferences on Artificial Intelligence; 2019.
108. Unger M, Bar A, Shapira B, Rokach L. Towards latent context-aware recommendation systems. Knowl-Based Syst. 2016;104:165–78.

109. Mukherjee D, Banerjee S, Bhattacharya S, Misra P. In: 2011 7th International Conference on Information Technology in Asia, IEEE, 2011. p. 1–7.
110. Agrawal R, Srikant R, Road H, Jose S. Fast Algorithms for Mining Association Rules p 32
111. Alkilany AAA. An overview: temporal-side of sequential patterns discovery. Int J Data Min Knowl Manag Process. 2013;3(1):1 (**Publisher: Academy & Industry Research Collaboration Center (AIRCC)**).
112. Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L. BPR: Bayesian Personalized Ranking from Implicit Feedback, 2009. p. 10.
113. Kang WC, McAuley J. Self-Attentive Sequential Recommendation. 2018. https://doi.org/10.48550/arXiv.1511.06939 [cs].