applied sciences

Article

Session-Based Recommendations for e-Commerce with Graph-Based Data Modeling

Marina Delianidi, Konstantinos Diamantaras *, Dimitrios Tektonidis and Michail Salampasis

Department of Information and Electronic Engineering, International Hellenic University, 57400 Thessaloniki, Greece

* Correspondence: kdiamant@ihu.gr

Abstract: Conventional recommendation methods such as collaborative filtering cannot be applied when long-term user models are not available. In this paper, we propose two session-based recommendation methods for anonymous browsing in a generic e-commerce framework. We represent the data using a graph where items are connected to sessions and to each other based on the order of appearance or their co-occurrence. In the first approach, called Hierarchical Sequence Probability (HSP), recommendations are produced using the probabilities of items' appearances on certain structures in the graph. Specifically, given a current item during a session, to create a list of recommended next items, we first compute the probabilities of all possible sequential triplets ending in each candidate's next item, then of all candidate item pairs, and finally of the proposed item. In our second method, called Recurrent Item Co-occurrence (RIC), we generate the recommendation list based on a weighted score produced by a linear recurrent mechanism using the co-occurrence probabilities between the current item and all items. We compared our approaches with three state-of-the-art Graph Neural Network (GNN) models using four session-based datasets one of which contains data collected by us from a leather apparel e-shop. In terms of recommendation effectiveness, our methods compete favorably on a number of datasets while the time to generate the graph and produce the recommendations is significantly lower.

Keywords: recommender systems; session-based recommendations; e-commerce; data and web mining; item co-occurrence; graph data model

check for updates

Citation: Delianidi, M.; Diamantaras, K.; Tektonidis, D.; Salampasis, M. Session-Based Recommendations for e-Commerce with Graph-Based Data Modeling. *Appl. Sci.* **2023**, *13*, 394. https://doi.org/10.3390/app13010394

Academic Editor: Keun Ho Ryu

Received: 20 November 2022
Revised: 13 December 2022
Accepted: 24 December 2022
Published: 28 December 2022

1. Introduction

Intelligent recommendations and their application in e-business systems are increasingly attracting the interest of researchers and companies. Particularly, the use of the recommendations systems in e-commerce aims at increasing conversion rate, profit and customer engagement and satisfaction. Today, online sales are often made by non-registered users, and therefore there is no historical user data recorded by the e-shop platforms. In these cases, the only data that can be stored is information concerning the duration of a session, the actions performed in each session's step, the items (i.e., products) viewed, and other activities of the online customers. This information will be recorded to later generate recommendations in real time for other users visiting similar or related items. These are session data [1] and can be collected while users navigate in the e-shop platform.

There is a variety of methods used in recommendation systems, such as association rules [2], matrix factorization [3] or machine learning techniques [4], etc. Other methods recently used in recommendation systems are based on graphs [5]. Graphs can efficiently model user–item interactions within sessions enabling the easy generation of new session data in near-real-time. The most frequent items that users visit are easily found using the current complete graph where nodes represent items and edges represent the "next-item-in-session" relationship between the nodes. Additionally, the combination of consecutive item appearances during user navigation in the online store is easily identified through the graph structure. Implementing a standard co-occurrence method using graphs, a session-based recommendation method, called Pair Popularity, based on item co-appearances anywhere in the same session was presented in [6]. Having recorded the item x viewed by the user at session step t, the method recommended a list of items for step $t+1$ based on the number of times the items co-appear with x in the training sessions. Session-based recommendation with Graph Neural Networks such as SR-GNN [7], GCE-GNN [8] and IC-GAR [9] have become very popular recently because of their very good performance that often represents the state-of-the-art. However, there are also a number of drawbacks regarding GNN approaches:

- Large computational complexity and large memory requirements during training. Most GNN models require very large training times and large amounts of memory even for medium-size datasets and even with special GPU acceleration hardware;
- Difficulty with cold-start recommendations. Most GNN models base their predictions on previous items visited in the session, so it is difficult to recommend new items or make recommendations without session data;
- Relatively reduced performance when the same item never appears repeatedly in consecutive session steps. This indicates that these models have difficulty in introducing novelty and diversity in the recommendations.

The motivation behind this work is to address these problems by introducing improved graph-based recommendation models which are simple, therefore computationally efficient. Moreover, they should be able to exploit the cold-start probabilities of items when there is no available co-occurrence with other items in the session and should be able to offer novel and diverse recommendations. To that end, we propose two new session-based recommendation methods—the Hierarchical Sequence Probability (HSP) and the Recurrent Item Co-occurrence (RIC). The HSP method extends the Pair Popularity graph-based approach to improve the results by using item sequences in user sessions to produce the hierarchical recommendation list, while the Recurrent Item Co-occurrence recommendation approach focuses on the co-occurrence of the products by giving weight to the count of item appearances in the corresponding sessions.

The goal is to produce the optimal item recommendation list at time step $t+1$ according to the items observed by the user up until the current time step t. The contributions of our work are:

- The proposition of two simple yet efficient session-based recommendation methods based on the ”Next” relationship and the co-occurrence relationship between items in the data representation graph;
- The comparison of the proposed methods with state-of-the-art Graph Neural Network models using four different datasets. One of the two proposed methods outperforms the GNN models in two cases while achieving close performance in the other two;
- The study of item sequences of the recent user browsing history vs. the simple item co-occurrence. We show that the method using co-occurrence statistics can achieve considerably better results than the one using the recent item sequences data;
- Attention to the efficiency aspect, showing that the proposed methods are significantly less computationally expensive than the compared GNN approaches even though the GNN models take advantage of a special GPU accelerator hardware to be trained.

The rest of the paper is organized as follows. In Section 2, we present an overview of the existing research related to the session-based recommendation problem. The data model analysis for the proposed recommendation methods and the algorithmic details are presented in Section 3. Section 4 presents the datasets used in our experiments. In Section 5, we describe the experimental procedure and discuss the results of the proposed recommendation methods. Section 6 concludes the paper.

2. Literature Review

There is a large number of recommendation methods used for different purposes such as recommending friends, destinations, movies, products, etc [10]. These systems use, in addition to previous user transactions, features such as location, demographic profile, and user preferences to identify items that are similar to one another. The role of recommendation systems (RS) has become increasingly crucial, especially in e-commerce, due to the availability of a large group of items from which the user can choose. Users of e-commerce sites are given tailored recommendations for products that they might find interesting. After applying desired business criteria that can be applicable, RSs finally offer a list of the top *n* recommended products for each targeted user action. If they do exist, long-term user profiles are prominently used in RS techniques. Such long-term user models, however, are usually unavailable in many applications for privacy-related reasons [11].

Session-based recommendation approaches (SBR) are recommendation techniques that only consider the user's in-session behavior and other session-specific information as well as the sequential order of items in sessions [12]. They adjust their recommendations to the user's most recent actions, and their main objective is to predict and suggest the next item(s) during every active user session [1].

A general method for developing recommendation systems is matrix factorization [13,14]. A user–item rating matrix must be factorized into two low-rank matrices, each of which reflects the latent factors of users or objects. In [3], the authors propose a matrix factorization approach for session-based recommendations which is based on solving a least squares optimization problem involving item–item similarities and session–item weights. The method achieves results comparable to the state-of-the-art; however, its complexity increases quickly with the size of the itemset. The item-based neighborhood approaches [15], in which item similarities are determined by the co-occurrence within the same session, could be a rational solution by taking into account the sequential order of the objects instead of generating predictions relying on the most recent click. The sequential Markov chain approaches are suggested to be used to predict users' future actions based on their past actions [16,17]. The weakness of Markov-chain-based models is that they independently recombine the previous components. Such a significant assumption of independence affects the prediction's accuracy.

Recommendation systems using graphs have also been quite actively studied recently. In fact, graph databases (GDBs) are one of the latest approaches in data modeling [5]. In a graph model, the data entities are represented as nodes and their relationships as directed or undirected connections between the nodes; thus, any data relationship can be represented on a corresponding graph [18]. The Neo4j [19] is a popular graph database tool used for creating various recommendation systems for friends, movies and items, as well as in e-commerce and loyalty-based retail businesses [20,21]. It uses the Cypher declarative graph query language, which is similar to SQL allowing efficient creation, reading, updating and querying of the graph data [22].

A session-based recommendation solution developed using the Neo4j graph database is presented in [6]. In this paper, the authors demonstrate an efficient method for session-based next-item recommendations. This recommendation system has been developed for an e-commerce retail store. With the appropriate data modeling, by defining nodes and relationships between the nodes and executing cypher queries, the system identifies the co-occurring paired items anywhere in the same session. The frequency of co-occurring item pairs determines the degree of similarity between these items. In practice, the next-item recommendation method uses these similarities for building the model.

Deep Learning (DL) models based on Recurrent Neural Networks (RNN) have been recently proposed for session-based recommendation solutions. The work in [23] proposes the Recurrent Neural Network approach for session-based recommendations, called GRU4REC, which employs multiple layers of the GRU model and uses only item sequences. In [24], the authors propose a hierarchical Recurrent Neural Network based again on the GRU model for session-based recommendations using user information. The work presented in [25] extends the GRU4REC method by introducing data augmentation, and [26] proposes NARM which is an integration of a stacked GRU encoder attention mechanism to capture more representative item transition information of SBR. In [27], the authors mix the sequential patterns and co-occurrence signals by combining together the recurrent method and the neighborhood-based method to enhance the performance of the GRU4REC recurrent model. One more DL recommendation method is based on mixture-channel purpose routing networks (MCPRNs) [28]. To handle multi-purpose sessions, the authors suggest a mixture-channel model. To model the dependencies between items within each channel for a specified purpose, they create a purpose-specific recurrent network (PSRN), a variation of the GRU RNN model. The authors of [29] introduce an RNN model named Hierarchical Attentive Transaction Embedding (HATE), which exploits the attention mechanism to predict the next item by modeling dependencies in transactional data. The HATE model consists of two parts, the Inter-transaction Context Embedding part for the item representation, and the Intra-transaction Context Embedding part for the representation of multiple chosen items in the current transaction, integrating these embeddings using Intra-transaction attention.

Graph Neural Network (GNN) models implement recommendation systems of various scenarios such as Social Recommendation, Sequential Recommendation, Session-based Recommendation, Bundle Recommendation, Cross-Domain Recommendation or Multi-behavior Recommendation [30]. Additionally, GNN models adopt machine learning and deep learning techniques, such as Convolutional Networks, Attention Mechanism or Embeddings representation to create recommendation systems in different domains [5]. Several next-item Graph Neural recommendation models have been proposed recently for the case of e-commerce scenarios using session-based datasets. One of Graph Neural Network's next-item recommendation approaches, named the Heterogeneous Mixed Graph Learning (HMGL) framework [31], was constructed to learn the complex local and global dependencies for next-item recommendations. HMGL encodes both session information and item attribute information into one unified graph modeling both local and global dependencies to better prepare for the next-item recommendations. In SR-GNN (https://github.com/CRIPAC-DIG/SR-GNN, accessed on 15 March 2022) [7], the session sequences are modeled as graph-structured data. Each session is represented as the composition of the global preference and the current interest of the session. An attention network is used to learn item embeddings on the session graph, and then obtain a representative session embedding which is calculated according to the relevance of each item to the last one. The GCE-GNN (https://github.com/CCIIPLab/GCE-GNN, accessed on 10 May 2022) [8] extends the previous approach by employing a session-aware attention mechanism to recursively incorporate the neighbors' embeddings of each node on the global graph. First, the session sequences are converted into session graphs to construct a global graph. The GCE-GNN learns two levels of item embeddings from the session graph by modeling pairwise item-transitions within the current session and the global graph which is to learn the global-level item embedding by modeling pairwise item-transitions over all sessions. Another recent GNN model, called IC-GAR (https://github.com/Taj-Gwadabe/IC-GAR, accessed on 10 October 2022) [9], models current session representations with session co-occurrence patterns, using a modified variant of Graph Convolutional Network (GCN). The Prediction Module of the IC-GAR separates global preference, local preference, and session co-occurrence in order to estimate the probability scores of candidate items. The global and local preferences model user interest in the current session, whereas the session co-occurrence representation aggregates the higher-order transition patterns of all the items in the training sessions. IC-GAR generates a single undirected graph for every training session. The SR-GNN, CGE-GNN and IC-GAR are the most recent state-of-the-art GNN RS models for SBR where one enhances the other with additional modules in order to more accurately predict the next item. The summary of the reviewed recommendation methods is presented in Table 1.

In the present work, we focus on graph-based recommendation systems in the e-commerce domain by proposing recommendation models that compete with recent state-of-the-art GNN based recommendation models. We propose two different methods called Hierarchical Sequence Probability (HSP) and Recurrent Item Co-occurrence (RIC) which create the recommendation list using the item–item relationships: "next" and "in-same-session", respectively. Related to these two methods, the aim of this paper is to answer the following research questions:

RQ1 Can HSP and RIC models achieve a state-of-the-art performance?
RQ2 How robust are these methods, i.e., how do they perform on different datasets?
RQ3 What is the effect of the item sequences and co-occurrences on the performance of HSP and RIC?
RQ4 What computational resources are required to perform each experiment and how time-consuming is it?

Table 1. Summary of reviewed methods.

Method	**Approach**	**Citation**
Probabilistic Matrix Factorization (PMF)	Matrix Factorization	[13]
SLIS, SLIT, SLIST	Matrix Factorization, Linear item–item recommendation	[3]
Matrix Factorization	Item-based Collaborative filtering	[15]
MDP-Based Recommender Model	Markov models	[16]
FPMC	Markov chain, Matrix Factorization	[17]
Recommendation of Influenced Products Using Association Rule Mining	Neo4j, Association rules	[20]
Goods Recommendation	Neo4j, Knowledge Graph database	[21]
Pair Popularity	Neo4j, Collaborative filtering	[6]
GRU4REC	RNN	[23]
HRNN	RNN	[24]
M{1,2,3,4}(GRU Size)	RNN, Data augmentation	[25]
NARM	GRU RNN, Attention mechanism	[26]
WH (KNN, GRU)	GRU RNN, k-NN	[27]
PSRN	GRU RNN, Mixture-channel Purpose Routing Networks (MCPRNs)	[28]
HATE	Attention mechanism, Item embeddings	[29]
HMLG	Gated Graph neural network, Path-based matrix factorization model	[31]
SR-GNN	Graph neural network, Attention mechanism, Item embeddings	[7]
GCE-GNN	Graph Neural Network, Session-aware attention mechanism, Neighbors' embeddings	[8]
IC-GAR	Graph neural network, Session co-occurrence patterns	[9]

3. Recommendation Methods

In this section, we describe two session-based recommendation methods called Hierarchical Sequence Probability (HSP) and Recurrent Item Co-occurrence (RIC), respectively. For both methods, we use graphs to represent data. These graph methods can be applied in e-shop platforms that allow anonymous access from non-registered users. The difference between the two recommendation methods is that HSP exclusively uses the sequence of items appearing in the session, while in the case of RIC, the recommendation list is based primarily on the items' co-occurrences extracted from session data. In detail, the two methods are described below.

3.1. The Graph Models

In both proposed methods, the graphs consist of two types of nodes which represent sessions and items. In the HSP graph, the connections between the nodes represented relationships as follows:

- Item o appearing in session s is connected with s via the *ItemInSession* relationship;
- An item o_1 appearing in session step t and an item o_2 appearing in the same session in step $t + 1$ are connected with the relationship *Next*.

Thus, the graph data provide sequence information about items in sessions. Figure 1a shows part of the data graph including the *ItemInSession* and the *Next* relationships between the items and the sessions.

In the RIC method, similar to the HSP, the graph has two types of nodes, for the items and sessions representations and also two types of following connections:

- The *ItemInSession* relationship as in the HSP method, and
- The *InSameSession* relation connects the items' co-occurrences in the same session independently to the sequence they appeared in. This is an undirected relationship.

In this case, the graph data model does not provide the sequence information about items in sessions. Figure 1b shows part of the data graph including the *ItemInSession* and the *InSameSession* relationships between the items and the sessions.

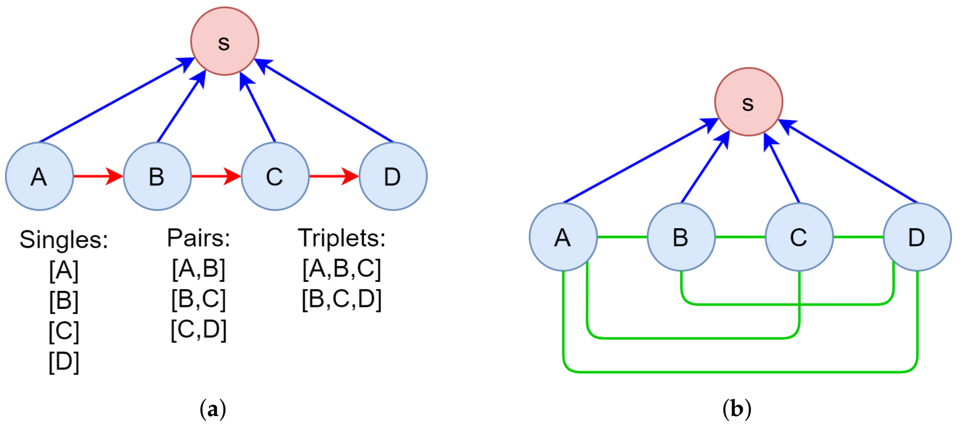

Figure 1. Representation of items (light blue nodes), sessions (pink nodes) and the relationships *ItemInSession* (blue edges), *Next* (red edges), *InSameSession* (green edges). (**a**) The HSP Graph Model, (**b**) The RIC Graph Model.

3.2. Hierarchical Sequence Probability Method—HSP

The Hierarchical Sequence Probability approach is an item–item collaborative filtering recommendation method where the list of recommended items arises from the items' sequential appearances during session navigation. We consider t the current time instance and $item_t$ the item that appears to a user in time t during the session s. To recommend the next item at time instance t during s, we introduce the concept of *"item sequence probability"*. This term derives from the visiting frequency of the item by users during the sessions on the e-shop platform. All the items have the *single item probability*, *pair sequence probability* and *triplet sequence probability* as follows:

- $P_0(A)$—*single item probability*, or *cold-start probability* is the number of appearances of item A in all sessions divided by the total number of appearances of all items:

$$P_0(A) = \frac{\text{number of appearances of } A}{\text{number of appearances of all items}} \tag{1}$$

The *single item probability* is derived from the relationship *ItemInSession* of the graph;

- $P_1(A,B)$—*pair sequence probability*, the number of appearances of the item pair (A,B) in successive instances in all sessions, i.e., $item_{t-1} = A$, $item_t = B$ divided by the number of appearances of item A

$$\begin{aligned} P_1(A,B) &= \frac{\text{number of appearances of consecutive pair } A, B}{\text{number of appearances of item } A} \\ &= P(item_t = B | item_{t-1} = A) \end{aligned} \tag{2}$$

This function is created for all pairs of items using the *Next* relationship;

- $P_2(A,B,C)$—*triplet sequence probability*, the number of appearances of the item triplet (*A*,*B*,*C*) in successive steps in all sessions, i.e., $item_{t-2} = A$, $item_{t-1} = B$, $item_t = C$ divided by the number of consecutive pairs A, B

$$\begin{aligned} P_2(A,B,C) &= \frac{\text{number of appearances of consecutive triplet } A,B,C}{\text{number of appearances of consecutive pair } A,B} \\ &= P(item_t = C | item_{t-2} = A, item_{t-1} = B) \end{aligned} \tag{3}$$

This function is created for all triplets of items connected by the *Next* relationship in a chain $A \rightarrow B \rightarrow C$.

The probabilities P_1 and P_2 are closely related to the confidences of the association rules $(A \Rightarrow B)$ and $(A, B \Rightarrow C)$ under the additional constraint that A, B, and C must be consecutive items [32].

The Algorithm

The Hierarchical Sequence Probability (HSP) recommendation method is based on item sequences observed through the users' actions in the sessions. To recommend an item, we look at its probability of appearance as well as the history of sequences of length 1 or 2 in which this item has participated during the training sessions. In the absence of a history (i.e., in the first step of a session), items are recommended based on their probability. Thus, the recommendation of the next item is based on the following cases:

1. The item recommendation list at step $t = 1$ (cold start case) contains the most frequently visited items ordered according to decreasing single item probability value P_0. We call this the *"0-history"* prediction since no previous session steps are required;
2. For step $t = 2$, the item recommendation list is compiled using navigation history of length 1. In particular, the recommendation list contains the items B that have a nonzero pair-sequence-probability value $P_1(item_1, B)$ with $item_1$ appearing at step $t = 1$. The list is ordered by decreasing the P_1 value. Since $item_1$ may be unpopular, it is likely that there are very few items B with the non-zero $P_1(item_1, B)$ value. For this reason, the recommendation list is completed with *"0-history"* predictions, i.e., appending the most popular items according to P_0, excluding those that are already included in the list;
3. For steps $t \geq 3$, the recommendation is compiled looking at the session history of length 2. In particular, the recommendation list contains the items C that have a nonzero triplet-sequence-probability value $P_2(item_{t-2}, item_{t-1}, C)$ ordered by decreasing value. Again, due to scarcity reasons, the recommendation list is complemented with predictions made using a history of length 1: we append the items B that have a nonzero pair-sequence-probability value $P_1(item_{t-1}, B)$, ordered by decreasing P_1 value, unless they are already included in the list. If that is still not sufficient, the list is finally completed with *"0-history"* predictions, i.e., with all the items A ordered by decreasing single item probability value $P_0(A)$, excluding those already in the list.

Figure 2 summarizes the flowchart of the proposed algorithm. In general, the recommended items appear only once in the final list following the hierarchy *"triplet sequence probability"* (2-length history), followed by *"pair sequence probability"* (1-length history), and followed by *"single item probability"* (0-length history).

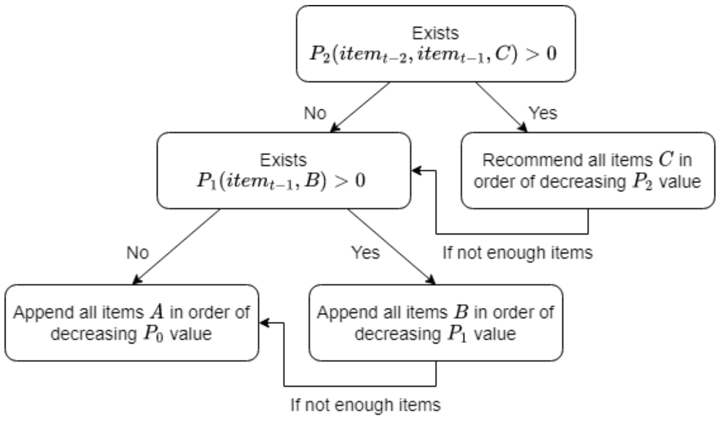
Exists
$P_2(item_{t-2}, item_{t-1}, C) > 0$
No
Yes
Exists
$P_1(item_{t-1}, B) > 0$
Recommend all items C in order of decreasing P_2 value
If not enough items
No
Yes
Append all items A in order of decreasing P_0 value
Append all items B in order of decreasing P_1 value
If not enough items

Figure 2. The flowchart of the proposed Hierarchical Sequence Probability (HSP) algorithm. Whenever appending an item in the recommendation list, we make sure it is not already included to avoid duplicate recommendations.

3.3. Recurrent Item Co-Occurrence Algorithm

Whereas HSP is based primarily on the *Next* relationship to determine the list of recommended items, our second proposed method is based primarily on the *InSameSession* relationship, which neglects the relative position of the items in the session. Let $\mathcal{I} = \{o_1, \ldots, o_N\}$ be the set of all the items. Given a current item x we define the confidence weight $\gamma(x|o_i)$ of any other item o_i as the ratio of all the *InSameSession* relations involving both x and o_i divided by the number of the *ItemInSession* relations involving x and any session s:

$$\gamma(o_i|x) = \frac{\text{count}(\mathit{InSameSession}(x, o_i))}{\text{count}(\mathit{ItemInSession}(x, s))}. \tag{4}$$

This is equivalent to the confidence value conf($x \Rightarrow o_i$) of the association rule $x \Rightarrow o_i$ [32]:

$$\begin{aligned}\text{conf}(x \Rightarrow o_i) = P(s \ni o_i | s \ni x) &= \frac{P\big((s \ni x) \cap (s \ni o_i)\big)}{P(s \ni x)} \\ &= \frac{\text{number of sessions containing both } x \text{ and } o_i}{\text{number of sessions containing } x}.\end{aligned} \tag{5}$$

A naive approach would be to build the recommendation list by simply sorting items by decreasing confidence $\gamma(o_i|x)$. This approach, however, has two major drawbacks:

(a) it poorly treats the case where there is no co-occurrence of x and o_i in any session. As it happens, this is a very common situation where, obviously, $\gamma(o_i|x) = 0$. Since all such items are put in the same ranking position, they will be randomly sorted;

(b) it is a memoryless approach since the recommendation list is built based solely on x, ignoring any other items viewed prior to x.

To alleviate these problems, we define a new confidence value c_i for item o_i which is equal to $\gamma(o_i|x)$ if x and o_i co-occur in at least one session; otherwise, c_i is equal to the cold-start probability $P_0(o_i)$ defined in Equation (1):

$$c_i = \begin{cases} \gamma(o_i|x) & \text{if } \gamma(o_i|x) > 0 \\ P_0(o_i) & \text{otherwise} \end{cases} \tag{6}$$

With this approach, items with no history of co-occurrence with x are placed in decreasing cold-start probability.

In order to introduce memory to the system, we further propose a simple, first order recurrent model that generates the item weights which will be used to build the recommendation list. Let $x = item_t$ be the item viewed at step t in some session s and $c_i(t)$ be the confidence value of any item o_i based on $item_t$ as described in Equation (6). Then, the weight $w_i(t)$ of this item at time t will be defined by the recurrent model:

$$w_i(t) = \alpha w_i(t-1) + (1-\alpha)c_i(t). \tag{7}$$

The initial condition is again the cold-start probability $w_i(0) = P_0(o_i)$. The recommendation list at any time step t is built using the top n items with the largest weights $w_i(t)$. Since the weights are computed independently for different items, the process can be easily parallelized using the vectors $\mathbf{w}_t = [w_1(t), \dots, w_N(t)]$, $\mathbf{c}_t = [c_1(t), \dots, c_N(t)]$, where now: $\mathbf{w}_t = \alpha\mathbf{w}_{t-1} + (1-\alpha)\mathbf{c}_t$.

The parameter $1-\alpha$ ($0 \le \alpha \le 1$) is the "forgetting factor," which determines the memory length of the system. For $1-\alpha = 0$ the system has infinite memory, the confidence $c_i(t)$ based on $item_t$ is ignored and $w_i(t)$ maintains a constant initial value through-out the session. If $1-\alpha = 1$ the model becomes memoryless and $w_i(t) = c_i(t)$. The parameter α is set by the user and determines the effect that the previous items $item_{t-1}, item_{t-2}, \dots$ have on the current decision. Figure 3 depicts the schematic diagram of the proposed recurrent system.

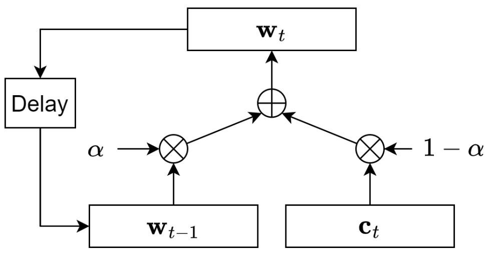

Figure 3. In the RIC method, the item weight vector $\mathbf{w}_t$ at any time step t is generated by a first order linear recurrent model. The input to the model is the current confidence vector $\mathbf{c}_t$.

Depending on how the sessions are recorded, it is possible to have repeated consecutive entries of the same item, for example, the item sequence could be A, B, B, C. In some datasets this is a frequent situation, whereas in other datasets this case never appears. We offer two-flavors of the RIC algorithm:

(a) Plain RIC where the recommendation list is provided as described above. In this case, the current item $x = item_t$ is very likely to be in the top position since $c_x = \gamma(x|x) = 1$;
(b) Current-item-Last (CiL) RIC, in which the current item is specifically moved to the last position in the list of all items, practically making it disappear from the top-n recommendation list.

4. The Datasets

We applied the experimentation on four session-based datasets: Leather (https://github.com/delmarin35/Graph-Probability-Rec-Sys/tree/main/data, accessed on 19 November 2022), Yoochoose1/64 (http://2015.recsyschallenge.com/challege.html, accessed on 3 April 2022), Diginetica (http://cikm2016.cs.iupui.edu/cikm-cup, accessed on 3 April 2022), eElectronics (https://www.kaggle.com/datasets/mkechinov/ecommerce-events-history-in-electronics-store, accessed on 18 June 2022).

Leather: The data of the Leather dataset obtained from the processing of web server log records of an e-shop with leather apparel, jackets, furs and accessories. This is real data that emerged from the log files that were recorded implicitly during the users' navigation actions in the e-shop platform for the time period of six months from March to August of 2021. The log data were preprocessed by identifying sessions, session length, user actions in each session, and the items targeted by the actions. We consider as a session step every user action during the session, for example, viewing an item or adding an item to the basket. The dataset was processed to obtain only the sessions that contain at least two behavior sequences, *"view item"* and *"add to cart"*, resulting in 102,024 records. The dataset was split into train (80%) and test (20%) sets. Thus, the number of records of the train and test sets are 81,651 and 20,373 respectively. The total number of unique sessions is 19,236 which corresponds to 15,388 unique sessions of the train set and 3848 unique sessions of the test set. In addition, the dataset contains 1448 unique items, of which 1429 appear in the train set and 1296 in the test set. The same item never appears in two consecutive steps in any session.

Yoochoose1/64: It is a public benchmark session-based dataset that has been commonly used to evaluate recommendation system performance. This dataset has 17,740 unique items, of which 17,371 appear in the train set and 6745 in the test set. In addition, 369 items of the test set do not appear in the train set, while 10,995 train set items do not exist in the test set. Moreover, 16.183% of the item pairs in the train sessions and 14.938% of the item pairs in the test sessions, respectively, contain the same item twice.

Diginetica: similarly to the Yoochoose1/64 dataset, the Diginetica dataset is often used as a benchmark for testing recommendation systems' performances. This dataset contains 43,097 items. All the items appear in the train set, but only 21,129 appear in the test set. The percentage of consecutive item pairs with a repeated item is less than in Yoochoose1/64, being approximately 9% in both train and test sets.

eElectronics: this dataset contains user behavior data recorded for a period of 5 months (October 2019–February 2020) from a large electronics online store. After removing the sessions with only one item, and splitting the total number of 68,973 sessions into train (80%) and test (20%) sets, 55,089 sessions were used for training and 13,884 sessions were used for testing. Additionally, there are 33,130 unique items, 29,917 of which appear in the train test and 14,482 appear in the test set. Furthermore, 3213 test set items are not in the train set and 1848 train test items do not appear in the test set.

The description of statistical information of the datasets is presented in Table 2. All the datasets have items that exist in the test set and do not exist in the train set or vice versa. No item was removed from either the train or the test sets.

Table 2. Datasets statistics.

Dataset	**# Sessions in**		**% Repeated Item Pairs in**		**# Items**
	Train Set	**Test Set**	**Train Set**	**Test Set**	
Leather	15,388	3848	0.000	0.000	1448
Yoochoose1/64	116,167	15,324	16.183	14.938	17,740
Diginetica	186,670	15,963	9.199	9.143	43,097
eElectronics	55,089	13,884	0.000	0.000	33,130

5. The Experimentation Procedure and Results

In this section, we first describe the evaluation metric for performance evaluation. We then intend to answer the research questions posed in Section 2.

5.1. Evaluation Metrics

The metrics that we used to evaluate the methods were the Mean Reciprocal Rank (MRR)@K and the Recall@K. The MRR is an appropriate metric for measuring the performance of recommendation algorithms on a session-based dataset as well as a good measure of the effectiveness of next-item recommendation [33]. It evaluates the accuracy of the recommended top-k list and is defined as:

$$\text{MRR@}k = \frac{1}{N}\sum_{x}\frac{1}{\text{rank}(x)},$$

where x is the next item to be predicted and $\text{rank}(x)$ is the position of x in the recommendation list, starting from position 1 for the first item. If x is not in the recommendation list, we set $\text{rank}(x) = \infty$. The value of MRR is between 0 and 1 and the higher the value, the more effective the quality of the recommendations. Assuming, as is often the case, that at least five recommended items appear on the user's screen, an MRR ≥ 0.2 indicates that the method is quite successful, since the next item chosen by the user is—on average—among the top five recommended.

The Recall@k is defined as the percentage of the target items that were actually included in the top-k recommendation list. Specifically, given a sequence of N top = k recommendation lists L_i with corresponding target items x_i, the Recall@k is defined as [29]:

$$\text{Rec@}k = \frac{1}{N} \sum_{i=1}^{N} \left| L_i \cap \{x_i\} \right|,$$

where $\left| L_i \cap \{x_i\} \right|$ denotes the cardinality of the intersection set between L_i and $\{x_i\}$ which, in this case, can either take the value 0, if the intersection is empty (i.e. $x_i \notin L_i$), or 1, if $x_i \in L_i$.

In all experiments, we used the train sets to construct the data representation graphs or to train the neural models. The evaluation of the algorithms was performed on the test sets.

5.2. The Experiments

The same session-based datasets were used for all the experimental implementations and recorded the results for MRR@k and Rec@k for the top items $k = \{10, 20, 30\}$. More specifically,

- We compared our methods against three state-of-the-art GNN recommendation models, namely, SR-GNN [7], GCE-GNN [8], and IC-GAR [9]. We trained the models using the code available from the respective GitHubs and recorded the time from the moment the training starts to receiving the results. For each model, we used the hyperparameters proposed in the corresponding Github codes. We executed the GNN models in the Google colab environment with a Tesla T4 GPU accelerator (16 GB) and Intel Xeon CPU @ 2.2 GHz;
- We also performed the same experiments using HSP and RIC and recorded the execution times from the moment we read the data until the generation of the results. The HSP method has no hyper-parameter that requires adjustment. It is worth mentioning that the HSP method extends the Pair Popularity approach and achieves better results. For the RIC method, the values of the parameter α were set to 0.1, 0.3, 0.5, 0.7 and 0.9 during the experiments. Table 3 shows the optimal parameter value per dataset. Our models were executed in Google colab in a CPU-only machine with an Intel Xeon CPU @ 2.2 GHz;
- Additionally, we run the experiments for the Pair Popularity algorithm [6] in the same scenarios of the top k recommendation items.

Table 3. Optimal values of the α parameter in the RIC method.

Dataset	$\boldsymbol{\alpha}$
Leather	0.3
Yoochoose1/64	0.1
Diginetica	0.7
eElectronics	0.7

5.3. Results and Discussion

The experimentation results show that the effectiveness of a model is affected by the dataset. Table 4 shows that the RIC-CiL recommendation method has a better MRR@k performance for any k in the case of the Leather and Electronics datasets. In these datasets, there is no session with repeated items in consecutive steps. The RIC-CiL variant achieves these results by transferring the current item to the end of the recommendation list, thus essentially excluding it from the top-k recommendations. This technique does not bring desired results in the Yoochoose1/64 and Diginetica datasets due to the existence of repeated consecutive items in the train and test sets. In these cases, the plain RIC variant works better and, especially in the Diginetica case, outperforms SG-NN and IC-GAR with respect to the MRR@*k* metric. HSP also has a very good MRR performance, outperforming the GNN models on the Leather dataset (falling only behind RIC CiL).

On the other hand, the GCE-GNN and SR-GNN models achieve a better MRR performance in the Diginetica dataset while all three GNN models have better MRR performance on the Yoochoose1/64 dataset. In both of these datasets, repeated consecutive items appear in many sessions. As shown in Table 2 the Yoochoose dataset has a very large percentage of repeated item pairs (15–16%), even higher than the Diginetica dataset (∼9%). This indicates that the GNN models have difficulty predicting the next item in a high ranking position in the recommendation list unless the next item is identical to the current one. In other words, they are not as efficient in identifying novelty. We stipulate that this phenomenon is due to overfitting, considering that these models have a lot of parameters that allow them to achieve very good fit on the training data but may not generalize as efficiently on the test data. Here, it is worth noting that online store users may not appreciate getting recommendations including the same item they are currently visiting. It seems more natural to exclude the current item from the recommendation list. However, in our experiments, we still use the Diginetica and Yoochoose1/64 datasets because they are common benchmarks studied in many papers in the field.

Table 4. Comparison results of our methods against three state-of-the-art Graph Neural Network recommendation models. The MRR@{10,20,30} and Recall@{10,20,30} are used as evaluation metrics. The best performance for each dataset and corresponding metric is marked by bold-face numbers.

Dataset	Method	Rec@10	MRR@10	Rec@20	MRR@20	Rec@30	MRR@30
Leather	SR-GNN	0.5007	0.2723	0.6028	0.2793	0.6574	0.2814
	GCE-GNN	**0.5391**	0.2686	**0.6452**	0.2760	**0.6994**	0.2781
	IC-GAR	0.4744	0.2675	0.5654	0.2734	0.6171	0.2754
	Pair Popularity	0.4776	0.2605	0.5678	0.2667	0.6206	0.2688
	HSP	0.4878	0.2837	0.5632	0.2892	0.6028	0.2905
	RIC CiL	0.5308	**0.2986**	0.6175	**0.3046**	0.6683	**0.3066**
	RIC plain	0.5083	0.1984	0.6062	0.2053	0.6594	0.2074
Yoochoose1/64	SR-GNN	0.6013	**0.2990**	0.7059	**0.3070**	0.7546	**0.3080**
	GCE-GNN	**0.6113**	0.2966	**0.7117**	0.3040	**0.7660**	0.3060
	IC-GAR	0.5776	0.2947	0.6803	0.3018	0.7310	0.3039
	Pair Popularity	0.4266	0.2108	0.5227	0.2176	0.5715	0.2196
	HSP	0.5455	0.2766	0.6409	0.2833	0.6835	0.2850
	RIC CiL	0.4365	0.2147	0.5366	0.2217	0.5865	0.2237
	RIC plain	0.5693	0.2755	0.6791	0.2832	0.7307	0.2853
Diginetica	SR-GNN	0.3877	0.1697	0.5160	0.1788	0.5945	0.1819
	GCE-GNN	**0.4104**	**0.1812**	**0.5426**	**0.1900**	**0.6193**	**0.1932**
	IC-GAR	0.3581	0.1572	0.4838	0.1659	0.5631	0.1691
	Pair Popularity	0.2664	0.1078	0.3685	0.1149	0.4271	0.1173
	HSP	0.2742	0.1225	0.3414	0.1273	0.3692	0.1283
	RIC CiL	0.3190	0.1406	0.4295	0.1483	0.4944	0.1509
	RIC plain	0.3931	0.1777	0.5134	0.1860	0.5806	0.1868
eElectronics	SR-GNN	0.4041	0.1958	0.4957	0.2023	0.5453	0.2043
	GCE-GNN	**0.4553**	0.2116	**0.5555**	0.2185	**0.6037**	0.2204
	IC-GAR	0.3868	0.2004	0.4730	0.2057	0.5188	0.2075
	Pair Popularity	0.3842	0.1872	0.4697	0.1931	0.5100	0.1948
	HSP	0.3668	0.2068	0.4243	0.2109	0.4491	0.2119
	RIC CiL	0.4525	**0.2382**	0.5421	**0.2444**	0.5892	**0.2463**
	RIC plain	0.4390	0.1685	0.5356	0.1752	0.5854	0.1772

Additionally, our experiments show that the GCE-GNN model achieves the best Recall@*k* for any *k* in all the datasets. In combination with the previous observations, we conclude that the GCE-GNN model is the most efficient one in finding the next item somewhere in the top-*k* list. However, it has still some difficulty in placing the next item in a high ranking position unless it is the same item as the current one. This is more obvious when studying the top 10 recommendations in the Leather and eElectronics datasets. In this case, we note that, although Rec@10 is almost identical for GCE-GNN and RIC-CiL, GCE-GNN has a significantly lower MRR@10 (between 2.5–3%)

Comparing HSP and RIC with each other, we find that HSP is inferior to RIC-CiL in the case of the non-repeating datasets (Leather and eElectronics) and inferior or very close to the performance of RIC-plain in the item-repeating datasets (Diginetica and Yoochoose1/64). Especially in the case of Yoochoose1/64, the HSP and RIC-plain are almost equivalent in terms of MRR performance although HSP is inferior in terms of the Rec@*k* metric. In the case of the Diginetica dataset, the performance of HSP is inferior to both versions of RIC. The difference between our two proposed methods is that, in HSP, we are basing it on the *Next* relationship, taking into account the sequence of items, while in RIC, we are basing it on the *InSameSession* taking into account the items' co-occurrence. The performance superiority of RIC against HSP indicates that focusing on item co-occurrence is more beneficial than looking strictly at the recent item sequence.

Based on these findings, we can claim that the structure of a dataset affects the performance of the recommendation models regardless of the way the recommendation model is constructed, i.e., with or without the use of neural networks. Assuming that we do not recommend the current item to the online user, the RIC-CiL variation achieves the best MRR performance compared against state-of-the-art GNN models.

Regarding the execution time, the proposed HSP and RIC methods differ significantly in the production of the recommendations list from the initial stage. In addition, simple CPU execution is sufficient for the studied datasets to quickly implement the training process or the calculations of the possible recommended items. The time it took for the entire experimental process per recommendation method and dataset, from the beginning to the appearance of the results, is shown in Table 5 and schematically illustrated in Figure 4. Despite the fact that, for the training of the state-of-the-art GNN methods, a GPU is necessary to complete the experiments in the time indicated in Table 5, for our HSP and RIC methods, significantly less time was consumed without the use of a GPU.

Table 5. The approximate experimentation execution time in minutes.

Dataset	SR-GNN 30 Epochs	GCE-GNN 20 Epochs	IC-GAR 10 Epochs	HSP	RIC
Leather	25	30	20	1	1
Yoochoose1/64	240	500	190	6	10
Diginetica	210	400	380	20	18
eElectronics	210	920	80	5	5

During the training process on the eElectronics dataset, we reduced the batch size to eight to avoid the out-of-memory problem. For the other datasets, we kept the batch size to 100 as defined in the methods' GitHub.

Based on the above findings, the proposed HSP and RIC methods are sufficiently competent against more complex, state-of-the-art methods, and can be applied in real e-commerce environments without requiring special equipment for their productive operation.

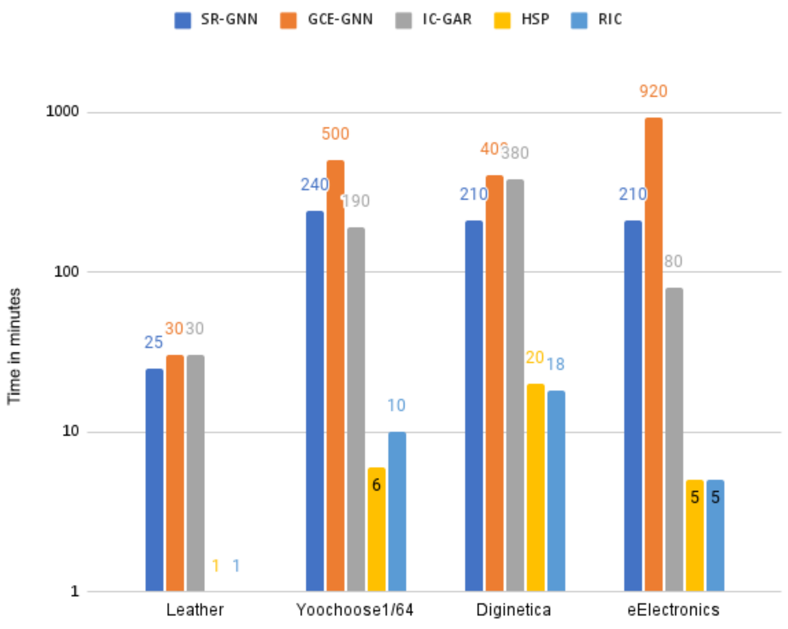

Figure 4. The execution time in minutes for all methods and each dataset. Our proposed methods are between 10× and 180× times faster than GNN models.

6. Conclusions

We have presented two graph-based methods for session-based recommendations in a generic e-commerce environment without employing user history, i.e., suitable for anonymous browsing. The methods are called Hierarchical Sequence Probability (HSP) and Recurrent Item Co-occurrence (RIC). HSP is based on the statistics of the *Next* relationship between items computing the probabilities of triplets, pairs and single items, which are then used—in that order—to determine the position of each item in the recommendation list. The RIC method is primarily based on the *InSameSession* relationship, which determines the co-occurrence of pairs of items in the same session. We introduce memory to RIC by incorporating a simple recurrent formula to determine the weight of each item which is subsequently used to place the item in its proper position in the recommendation list. Setting the value of the forgetting factor of this recurrent formula allows us to balance the effect on our current decision of previously visited items in the session.

Both proposed methods have been compared to state-of-the-art Graph Neural Network models. Our experiments, which involve four diverse datasets, show that RIC can outperform the GNN models in two cases and achieve a performance quite close to that of the winner model in the other two. The HSP approach is typically inferior to RIC, indicating that the *Next* relationship is not so important compared to the *InSameSession* relationship when building the recommendation list.

Additionally, both HSP and RIC methods are very fast compared to the GNN models. This happens despite the fact that the execution time of the GNN models is reduced thanks to the presence of a GPU accelerator, whereas the times recorded for our models are measured on a simple CPU-based machine.

In future work, we plan to investigate the improvement of the RIC method by automatically determining the optimal parameter α and also to determine whether the current item should be first or last in the recommendation list. Another important aspect of the algorithm which is worth investigating is the graph update as new data are collected in such a way that the computational cost of updating the new confidence vectors, cold-start probabilities and weight vectors is minimized.

Author Contributions: Conceptualization, M.D. and K.D.; methodology, M.D. and K.D.; software, K.D.; validation, M.D. and K.D.; formal analysis, M.D. and K.D.; investigation, M.D. and K.D.; resources, D.T. and M.S.; data curation, D.T., M.D., K.D. and M.S.; writing—original draft preparation, M.D. and K.D.; writing—review and editing, K.D., M.D. and M.S.; visualization, K.D.; supervision, K.D.; project administration, K.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by RESEARCH–CREATE–INNOVATE call (project code: T1EDK-01776).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data for this study are available on request from the corresponding authors.

Acknowledgments: This research has been co-financed by the European Regional Development Fund of the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH–CREATE–INNOVATE (project code:T1EDK-01776).

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

References

1. Wang, S.; Cao, L.; Wang, Y.; Sheng, Q.Z.; Orgun, M.A.; Lian, D. A survey on session-based recommender systems. *ACM Comput. Surv. CSUR* **2021**, *54*, 1–38. [CrossRef]
2. Han, J.; Kamber, M. *Data Mining: Concepts and Techniques*, 2nd ed.; University of Illinois at Urbana Champaign: Champaign, IL, USA; Morgan Kaufmann: Burlington, MA, USA; Elsevier: Amsterdam, The Netherlands, 2006.
3. Choi, M.; Kim, J.; Lee, J.; Shim, H.; Lee, J. Session-aware linear item–item models for session-based recommendation. In Proceedings of the Web Conference 2021, Ljubljana, Slovenia, 19–23 April 2021; pp. 2186–2197.
4. Chakraborty, S.; Hoque, M.; Rahman Jeem, N.; Biswas, M.C.; Bardhan, D.; Lobaton, E. Fashion Recommendation Systems, Models and Methods: A Review. *Informatics* **2021**, *8*, 49. [CrossRef]
5. Wang, S.; Hu, L.; Wang, Y.; He, X.; Sheng, Q.Z.; Orgun, M.A.; Cao, L.; Ricci, F.; Yu, P.S. Graph learning based recommender systems: A review. In Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, Montreal, QC, Canada, 19–27 August 2021; pp. 4644–4652.
6. Delianidi, M.; Salampasis, M.; Diamantaras, K.; Siomos, T.; Katsalis, A.; Karaveli, I. A Graph-Based Method for Session-Based Recommendations. In Proceedings of the 24th Pan-Hellenic Conference on Informatics, Athens, Greece, 20–22 November 2020; Association for Computing Machinery: New York, NY, USA, 2020; pp. 264–267. [CrossRef]
7. Wu, S.; Tang, Y.; Zhu, Y.; Wang, L.; Xie, X.; Tan, T. Session-based recommendation with graph neural networks. *AAAI Conf. Artif. Intell.* **2019**, *33*, 346–353. [CrossRef]
8. Wang, Z.; Wei, W.; Cong, G.; Li, X.L.; Mao, X.L.; Qiu, M. Global context enhanced graph neural networks for session-based recommendation. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual, 25–30 July 2020; pp. 169–178.
9. Gwadabe, T.R.; Liu, Y. IC-GAR: Item co-occurrence graph augmented session-based recommendation. *Neural Comput. Appl.* **2022**, *34*, 7581–7596. [CrossRef]
10. Roy, D.; Dutta, M. A systematic review and research perspective on recommender systems. *J. Big Data* **2022**, *9*, 1–36. [CrossRef]
11. Ludewig, M.; Jannach, D. Evaluation of session-based recommendation algorithms. *User Model. User Adapt. Interact.* **2018**, *28*, 331–390. [CrossRef]
12. Wang, S.; Hu, L.; Wang, Y.; Cao, L.; Sheng, Q.Z.; Orgun, M. Sequential recommender systems: Challenges, progress and prospects. *arXiv* **2019**, arXiv:2001.04830.
13. Mnih, A.; Salakhutdinov, R.R. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*; Curran Associates Inc.: Red Hook, NY, USA, 2007; Volume 20.
14. Koren, Y.; Rendle, S.; Bell, R. Advances in collaborative filtering. In *Recommender Systems Handbook*; Springer: Boston, MA, USA, 2022; pp. 91–142.
15. Sarwar, B.; Karypis, G.; Konstan, J.; Riedl, J. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th International Conference on World Wide Web, Hong Kong, China, 1–5 May 2001; pp. 285–295.
16. Shani, G.; Heckerman, D.; Brafman, R.I.; Boutilier, C. An MDP-based recommender system. *J. Mach. Learn. Res.* **2005**, *6*, 1265–1295.
17. Rendle, S.; Freudenthaler, C.; Schmidt-Thieme, L. Factorizing personalized markov chains for next-basket recommendation. In Proceedings of the 19th International Conference on World Wide Web, Raleigh, NC, USA, 26–30 April 2010; pp. 811–820.
18. Jyothi, D.N. Book Recommendation System using Neo4j Graph Database. *Int. J. Anal. Exp. Modal Anal.* **2020**, *12*, 498–504.

19. Neo4J. Graph Database Platform | Graph Database Management System. 2021. Available online: https://neo4j.com (accessed on 10 June 2021).
20. Sen, S.; Mehta, A.; Ganguli, R.; Sen, S. Recommendation of Influenced Products Using Association Rule Mining: Neo4j as a Case Study. *SN Comput. Sci.* **2021**, *2*, 1–17. [CrossRef]
21. Konno, T.; Huang, R.; Ban, T.; Huang, C. Goods recommendation based on retail knowledge in a Neo4j graph database combined with an inference mechanism implemented in jess. In Proceedings of the 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), San Francisco, CA, USA, 4–8 August 2017; pp. 1–8.
22. Yi, N.; Li, C.; Feng, X.; Shi, M. Design and implementation of movie recommender system based on graph database. In Proceedings of the 2017 14th Web Information Systems and Applications Conference (WISA), Liuzhou, China, 11–12 November 2017; pp. 132–135.
23. Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; Tikk, D. Session-based recommendations with recurrent neural networks. *arXiv* **2015**, arXiv:1511.06939.
24. Quadrana, M.; Karatzoglou, A.; Hidasi, B.; Cremonesi, P. Personalizing session-based recommendations with hierarchical recurrent neural networks. In Proceedings of the Eleventh ACM Conference on Recommender Systems, Como, Italy, 27–31 August 2017; pp. 130–137.
25. Tan, Y.K.; Xu, X.; Liu, Y. Improved recurrent neural networks for session-based recommendations. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, Boston, MA, USA, 15 September 2016; pp. 17–22.
26. Li, J.; Ren, P.; Chen, Z.; Ren, Z.; Lian, T.; Ma, J. Neural attentive session-based recommendation. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, Singapore, 6–10 November 2017; pp. 1419–1428.
27. Jannach, D.; Ludewig, M. When recurrent neural networks meet the neighborhood for session-based recommendation. In Proceedings of the Eleventh ACM Conference on Recommender Systems, Como, Italy, 27–31 August 2017; pp. 306–310.
28. Wang, S.; Hu, L.; Wang, Y.; Sheng, Q.Z.; Orgun, M.; Cao, L. Modeling multi-purpose sessions for next-item recommendations via mixture-channel purpose routing networks. In Proceedings of the International Joint Conference on Artificial Intelligence, Macao, China, 10–16 August 2019.
29. Wang, S.; Cao, L.; Hu, L.; Berkovsky, S.; Huang, X.; Xiao, L.; Lu, W. Hierarchical attentive transaction embedding with intra-and inter-transaction dependencies for next-item recommendation. *IEEE Intell. Syst.* **2020**, *36*, 56–64. [CrossRef]
30. Gao, C.; Zheng, Y.; Li, N.; Li, Y.; Qin, Y.; Piao, J.; Quan, Y.; Chang, J. A Survey of Graph Neural Networks for Recommender Systems: Challenges, Methods, and Directions. *ACM Trans. Rec. Sys* **2022**, *55*, 97.
31. Wang, N.; Wang, S.; Wang, Y.; Sheng, Q.Z.; Orgun, M. Modelling local and global dependencies for next-item recommendations. In *International Conference on Web Information Systems Engineering*; Springer: Cham, Switzerland, 2020; pp. 285–300.
32. Agrawal, R.; Imieliński, T.; Swami, A. Mining association rules between sets of items in large databases. *ACM SIGMOD Record* **1993**, *22*, 207–216. [CrossRef]
33. Chen, M.; Liu, P. Performance evaluation of recommender systems. *Int. J. Perform. Eng.* **2017**, *13*, 1246. [CrossRef]