S-Walk: Accurate and Scalable Session-based Recommendation with Random Walks

Minjin Choi, Jinhong Kim, Joonseok Lee, Hyunjung Shim, and Jongwuk Lee.

ACM Reference Format:
Minjin Choi, Jinhong Kim, Joonseok Lee, Hyunjung Shim, and Jongwuk Lee. 2022. S-Walk: Accurate and Scalable Session-based Recommendation with Random Walks. In Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM ’22), February 21–25, 2022, Tempe, AZ, USA. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3488560.3498464

1 INTRODUCTION

Modern recommender systems (RS) are indispensable for addressing the enormous information overload in various real-world applications, such as e-commerce platforms and online multimedia platforms, e.g., Amazon and Alibaba, YouTube, Netflix, and Spotify. Classical recommender systems [12, 15, 19, 44] usually assume that user accounts and users’ long-term interactions are available. However, this assumption rarely holds. The user may not login, or multiple users share the user account, e.g., family members. The user may also exhibit different behaviors depending on the context. Thus, it is necessary to provide personalized recommendations without explicit user information.

Recently, session-based recommendation (SR) [3, 9, 22, 42, 51] has gained considerable attention for predicting the next items from the sequential behavior consumed by an anonymous user. Unlike conventional RS, SR relies only on the users’ actions in an ongoing session. This setting of SR is well-suited for real-world scenarios but inherently results in a severe data sparsity problem. To address this issue, it is essential to understand the unique characteristics of sessions, that is, intra-session properties. First, the items within a session are coherent with the user’s hidden intent, e.g., a list of products in the same category, referred to as item consistency (or long-term dependency). Second, some items are strictly consumed in chronological order, namely sequential dependency (or short-term dependency), e.g., consecutive episodes of a TV series. Lastly, the user can repeatedly consume the same items, called repeated item consumption, e.g., user’s favorite tracks.

Most SR models utilize deep neural networks (DNNs) to learn intra-session relationships. Recurrent neural networks (RNNs) [16–18] and attention mechanisms [31, 32] have been used to model the sequential dependency of items. Recently, graph neural networks (GNNs) [1, 13, 40, 41, 50, 53, 54] have been used to effectively

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

WSDM ’22, February 21–25, 2022, Tempe, AZ, USA.
© 2022 Copyright held by the owner/authors(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9132-0/22/02...
https://doi.org/10.1145/3488560.3498464
represent both item consistency and sequential dependency. Unfortunately, they suffer from performance degradation when dealing with complex and long sessions, where it is difficult to understand user intent. For this, *inter-session* relationships among items are valuable clues. Figure 1 describes intra-session and inter-session relationships; some items do not occur within a single session but share their neighboring items for multiple sessions, implying potential item correlations. Several studies [36, 50, 52] have attempted to consider inter-session relationships using neighboring sessions or global item graphs. However, they incur substantial computational costs and are infeasible for large-scale commercial systems.

High-capacity DNN models have achieved state-of-the-art performances but usually require heavy computational overhead for runtime speed and memory consumption. Although competing for the best performance is a reasonable mission for most research problems, computational efficiency and scalability are also critical factors for dealing with practical restrictions in commercial recommender systems. [11, 33] suggested neighborhood-based models for session-based recommendations. Owing to their simplicity, they are highly scalable. Moreover, [34, 35] reported that the neighborhood-based models achieved comparable performances to DNN-based models on several benchmark datasets.

Our primary objective is to design an SR model that accounts for both accuracy and scalability. To this end, we propose a novel session-based recommendation with a random walk, namely S-Walk, (i) exploiting intra- and inter-session relationships among items to improve accuracy, and (ii) supporting cost-effective, real-time performance at scale.

(i) Whereas the basic SR learns hidden patterns between items only within a session, S-walk additionally introduces *global item graphs*, modeling item-item relationships across all sessions. By applying random walks with restart (RWR) on this graph, a random surfer can jump from one item to another adjacent item by traversing the item graph or restart from an arbitrary item in the current session. Therefore, S-Walk can capture high-order correlations among items using multi-hop connections on an item graph. S-Walk can exploit the local patterns within a session and global patterns involving the same items from other sessions.

(ii) Recently, linear item-item models [23, 45–47] have shown competitive performance in conventional RS. Motivated by their success, we devise linear item models to build two probability matrices, *item transition* and *item teleportation* matrices, to formulate the stochastic process of RWR. The item transition matrix captures the sequential dependency of items, generalizing a Markov chain model on items. The item teleportation matrix reflects the restart probability, allowing various items to participate in the model training depending on the ongoing session. Instead of having arbitrary items for restart, we utilize the co-occurrence relationship among items. The items that co-occurred with the current session items are used for restart. Notably, training our linear models is highly efficient and scalable because they have closed-form solutions whose computational complexity is determined by the number of items, independent of the number of sessions or user actions.

To summarize, the key advantages of S-Walk are as follows: (i) It can effectively capture inter- and intra-session relationships via RWR. (ii) Without complicated model tuning, it is highly efficient and scalable owing to the closed-form solution of linear models. (iii) It achieves competitive or state-of-the-art performance in various metrics (i.e., HR, MRR, recall, and MAP) on four benchmark datasets (i.e., YouChoose, Diginetica, RetailRocket, and NowPlaying). (iv) The model learned by S-Walk can be highly compressed without sacrificing accuracy, supporting fast inference time.

2 PRELIMINARIES

**Notations.** Given a set of sessions \( S = \{s^{(1)}, \ldots, s^{(m)}\} \) over a set of items \( I = \{i_1, \ldots, i_n\} \), an arbitrary session \( s \in S \) is represented by a sequence of items \( s = (s_1, s_2, \ldots, s_|s|) \) with a length \(|s|\). Here, \( s_j \in I \) is the \( j \)-th consumed item, e.g., clicked, watched, or purchased. For simplicity, \( s \in S \) is represented by a binary vector \( x \in \{0, 1\}^n \), where \( x_k = 1 \) if \( i_k \) is consumed, and \( x_k = 0 \) otherwise. By stacking \( m \) sessions, let \( X \in \mathbb{R}^{m \times n} \) denote a session-item interaction matrix, where \( m \) is the number of sessions. As a straightforward variant, the binary value in \( x_k \) can be converted to real values to quantify the importance of items within a session.

**Problem statement.** Given a sequence of items previously consumed by an anonymous user in a session, session-based recommendations predict the next items that the user is most likely to consume. Formally, a session-based recommender model takes a session \( s = (s_1, \ldots, s_j) \) as input and returns a ranked list of top-\( N \) candidate items as the recommended next items \((s_{j+1}, \ldots, s_{|s|})\). Note that this is more generalized (and challenging) than predicting the next single item \( s_{j+1} \), which has been used in existing work. (See Section 4 for the generalized evaluation metrics for this setting.)

**Random walk models.** The key concept behind random walk models is to reflect the direct and transitive relations among items. Conventional recommender models using random walks [6–8, 20, 37, 38, 55] are based on a user-item bipartite graph \( G = (\mathcal{U} \cup \mathcal{I}, \mathcal{E}) \), where \( \mathcal{U} \) and \( \mathcal{I} \) are a set of users and items. Each edge \( e \in \mathcal{E} \) represents the relationship between a user and an item. Thus, the core part of the random walk model is to determine a transition probability matrix to compute proximity scores for items.

In general, there are two possible solutions for computing the proximity scores of items. First, we can utilize the \( k \)-step landing probability distribution of a random walker. Starting from a source user \( u \), the proximity scores for all items can be computed as \( uR_{(k)} \),
where \( u \) is the user vector, and \( R_{t,k} \) is the \( k \)-step transition probability matrix. Although it is simple yet effective, existing studies [6, 7] are vulnerable to popularity bias; popular items tend to have high proximity scores as \( k \) increases (e.g., \( k \geq 3 \)), thereby degrading the recommendation quality.

As an alternative, we can compute the stationary distribution of random walks with restart (RWR), which is well-known as personalized PageRank [26]. It is effective for capturing high-order relationships among vertices. Because RWR leverages restart in addition to sequential transition, it can alleviate the problem of popularity bias, concentrating on the central node in the \( k \)-step. For this reason, we adopt RWR for session-based recommendations. (See Section 5 for empirical comparisons between the two methods, \( k \)-step and RWR.)

Adopting the random walk in session-based recommendations has the following advantages: (i) The random walk model utilizes high-order item correlations across sessions. Because sessions are usually sparse by nature, it is useful for alleviating the data sparsity problem by capturing profound relationships among items. (ii) Compared to GNN-based SR models [13, 40, 41, 53, 54], it is efficient without requiring complicated hyper-parameter tuning.

**Linear item-item models.** Given the session-item matrix \( X \), the goal of linear models [39, 46] is to estimate the item-item similarity matrix \( B \in \mathbb{R}^{n \times n} \). As a pioneering work, SLIM [39] formulated a linear model subject to the constraint that all entries in \( B \) are non-negative and zero diagonal:

\[
\begin{align*}
\text{argmin}_B & \|X - X \cdot B\|_F^2 + \lambda_1 \|B\|_1 + \lambda_2 \|B\|_F^2 \\
\text{subject to} & \quad \text{diag}(B) = 0, \quad B \geq 0,
\end{align*}
\]

where \( \| \cdot \|_1 \) and \( \| \cdot \|_F \) are the entry-wise \( \ell_1 \)-norm and the matrix Frobenius norm, respectively, \( \lambda_1 \) and \( \lambda_2 \) are the regularization coefficients, and \( \text{diag}(B) \in \mathbb{R}^n \) denotes the vector with the diagonal elements of \( B \). Although SLIM [39] shows competitive accuracy, it suffers from high computational training cost.

Recently, EASE \(R\) [46] and its variants [45, 47] only consider the zero-diagonal constraint by removing the non-negativity of \( B \) and \( \ell_1 \)-norm constraints from Eq. (1):

\[
\begin{align*}
\text{argmin}_B & \|X - X \cdot B\|_F^2 + \lambda \cdot \|B\|_F^2 \\
\text{subject to} & \quad \text{diag}(B) = 0.
\end{align*}
\]

Owing to this simpler formulation, EASE \(R\) is solved by the closed-form equation via Lagrange multipliers:

\[
\hat{B} = I - \hat{P} \cdot \text{diagMat}(1 \otimes \text{diag}(\hat{P})),
\]

where \( \hat{P} = (X^T X + I)^{-1} \). Here, \( I \in \mathbb{R}^n \) is the vector of ones, and \( \otimes \) denotes the element-wise division. Let \( \text{diagMat}(x) \) denote the diagonal matrix expanded from the vector \( x \). (See [46] for the detailed derivation of the closed-form solution.)

Although inverting the regularized Gram matrix \( X^T X \) is the computational bottleneck for large-scale datasets (i.e., the time complexity is \( \mathcal{O}(n^3) \) with the Coppersmith-Winograd algorithm), the closed-form solution is advantageous in terms of efficiency. The training complexity of EASE \(R\) [46] is proportional to the number of items, which is usually much smaller than the number of sessions \( (n \ll m) \). Besides, the linear model is beneficial to accelerate the inference because computing top-\( N \) recommended items is simply done by single matrix multiplication. Most recently, SLIST [5] reported competitive accuracy with linear models for the session-based recommendation. Although SLIST [5] tackled various characteristics of session data, it did not consider inter-session relationships. In contrast, we devise linear models with random walks, taking both intra- and inter-session correlations into account.

### 3 S-WALK: THE PROPOSED MODEL

In this section, we propose a novel session-based recommendation with a random walk, namely S-Walk. While existing random-walk-based recommender models [6–8, 20, 37, 38, 55] are based on a user-item bipartite graph, it is non-trivial to adopt them for session-based recommendations. Since user information is unavailable, we rely only on item information to learn underlying patterns. Furthermore, the session-item matrix is extremely sparse.

To address these issues, we first present the overall architecture of S-Walk using global item graphs (Section 3.1). Intuitively, walking on a global item graph can describe inter-session relationships among items because the walker can move from an item of the current session to the items of other sessions. Then, we develop two linear models to build a transition matrix and a teleportation matrix, used in S-Walk (Sections 3.2–3.3). Note that these linear models can be replaced by others as long as they are efficient and scalable. Notably, our linear models satisfy the desirable condition for efficiency and scalability. Finally, we explain model training and inference of S-Walk (Section 3.4).

#### 3.1 Model Architecture

Figure 2 overviews the S-Walk model. Given a session-item interaction matrix \( X \), we first design two models for item transition and teleportation to capture different characteristics of sessions (the blue and orange box in Figure 2, respectively). We then constitute a final global item graph using random walk with restart (RWR).

Specifically, each model produces its own relevance matrix over the transition graph \( G_R = (I, E_R) \) and the teleportation graph \( G_T = (I, E_T) \), where each node corresponds to an item and an edge indicates the relevance between a pair of items. The transition matrix \( R \) is the adjacency matrix of \( G_R \) which encodes sequential dependency and repeated item consumption in a session. On the other hand, the teleportation matrix \( T \) is the adjacency matrix of \( G_T \), which captures item consistency in the session. Introducing the two matrices captures different intra-session relationships among items, but they do not address the inter-session relationship.

By adopting the RWR using the two graphs, where a random walker jumps from one node to another or restarts on an arbitrary node regardless of her current position, we intend to consider the inter-session relationship, capturing high-order relationships among items, i.e., multi-hop connections on the item graphs. Conceptually, the RWR on the two item graphs, \( G_R \) and \( G_T \), can be thought of as tossing a biased coin that yields the head with probability \( \alpha \):

1. If the coin is head (with a probability of \( \alpha \)), the walker moves to one of the items adjacent to the current item through the transition matrix \( R \in \mathbb{R}^{n \times n} \).
2. If the coin is tail (with a probability of \( 1 - \alpha \)), the walker restarts on one of the items adjacent to the start item through the teleportation matrix \( T \in \mathbb{R}^{n \times n} \).
Through the RWR, we obtain stationary probabilities that the random walk with restart is used to build the final graph with adjacency matrix $G$.

Finally, the item transition matrix $R$ is learned with two matrices $\hat{M}$ and $\hat{E}$, capturing high-order relationships.

In this process, we devise linear models for the two models, taking the following advantages: (i) they achieve comparable performance without complicated tuning, and (ii) training and inference are much faster than DNN-based session recommender models [13, 16–18, 31, 32, 40, 41, 53, 54].

### 3.2 Item Transition Model

First, we develop a linear transition model to build the item transition matrix $R$. As a natural way of representing the transition of item sequences, we introduce *partial session representations*. A session $s$ is divided into two sub-sessions, *past* and *future*, according to each time step $t = 1, \ldots, |s|$. The past partial session consists of items consumed before the $t$-th item, i.e., $s_{t-1} = \{s_1, \ldots, s_{t-1}\}$. The future partial session consists of items consumed at or after the $t$-th item, i.e., $s_{t+1} = \{s_t, \ldots, s_{|s|}\}$. For each time $t = 2, \ldots, |s|$, we produce $|s| - 1$ past and future partial session pairs. By stacking $|s| - 1$ pairs for all $s \in S$, we build two matrices, the past session matrix $Y \in \mathbb{R}^{m \times n}$ and future session matrix $Z \in \mathbb{R}^{m' \times n}$, where $m'$ is the number of all partial sessions, i.e., $m' = \sum_{i=1}^{m} (|p(i)| - 1)$. Finally, the item transition matrix $R$ is learned with two matrices $Y$ and $Z$ according to the partial session representation.

To represent the temporal proximity of items, we adjust the weights of items in $Y$ and $Z$. As adopted in [5, 11], we consider the position gap between two items as the weight of items within a session.

$$w_{\text{pos}}(i, j, s) = \exp \left( \frac{|p(i, s) - p(j, s)|}{\delta_{\text{pos}}} \right),$$

where $\delta_{\text{pos}}$ is the hyper-parameter that controls the position decay in partial sessions, and $p(i, s)$ is the position of item $i$ in the session $s$. In this way, we decay the relevance between items $i$ and $j$ as they get farther away.

Formally, the item transition model is formulated as

$$\arg\min_{\mathbf{B}^{\text{tran}}} \| \mathbf{Z} - \mathbf{Y} \cdot \mathbf{B}^{\text{tran}} \|^2_F + \lambda \| \mathbf{B}^{\text{tran}} \|^2_F,$$

where $\mathbf{B}^{\text{tran}}$ is the item-item relevance matrix learned from the sequential dependency between $Y$ and $Z$. As $Y \neq Z$, we can naturally avoid the trivial solution $\mathbf{B}^{\text{tran}} = \mathbf{I}$. Unlike Eq. (2), we can remove the zero-diagonal constraint in $\mathbf{B}^{\text{tran}}$. The closed-form solution is given by

$$\hat{\mathbf{B}}^{\text{tran}} = \mathbf{P}'\cdot(Y^T\mathbf{Z}),$$

where $\mathbf{P}' = (Y^T\mathbf{Y} + \lambda \mathbf{I})^{-1} \in \mathbb{R}^n$. The computational complexity is independent of the number of users, as shown in Eq. (3). (See Appendix A for a detailed derivation of our solution in the supplementary material.)

To utilize the item transition matrix in random walks, each element should be the transition probability from one node to another. That is, every element is non-negative and sums to 1. However, $\mathbf{B}^{\text{tran}}$ is not normally a probability matrix. To satisfy the non-negative constraint, we first replace all negative values in $\hat{\mathbf{B}}^{\text{tran}}$ with zero, denoting $\hat{\mathbf{B}}^{\text{tran}}_{\geq 0}$. We then normalize $\hat{\mathbf{B}}^{\text{tran}}_{\geq 0}$ as follows:

$$\mathbf{R} = \text{diag(mat}(\hat{\mathbf{B}}^{\text{tran}}_{\geq 0}))^{-1} \hat{\mathbf{B}}^{\text{tran}}_{\geq 0}.$$  

Here, $\mathbf{R}$ is the item transition probability matrix, where each row is normalized, and $\mathbf{1}$ is a column vector of length $n$ filled with one.

---

1As the negative values in $\hat{\mathbf{B}}^{\text{tran}}$ mean negative correlations among items, they are less important than positive correlations. In our empirical study, it is observed that this conversion does not harm the accuracy of the linear model.
3.3 Item Teleportation Model

The item teleportation matrix is designed to capture the item consistency within a session. For this reason, we focus on modeling co-occurrence between items. A session is treated as a set of items \( s = \{s_1, s_2, \ldots, s_l\} \), ignoring the order of items. By stacking \( m \) sessions, we build a binary session-item matrix \( X \). Note that the repeated items in the session are treated as a single item.

Given the matrix \( X \), we devise a linear teleportation model. It is formulated with the same input and output matrix as used in the existing linear models [39, 46]. Meanwhile, we relax the zero-diagonal constraint for \( B \) to handle repeated item consumption, as discussed in [5]. When the diagonal element of \( B \) is loosely penalized, it allows us to predict the same items as the next item repeatedly.

\[
\arg\min_{B^{tele}} \|X - X \cdot B^{tele}\|_F^2 + \lambda \|B^{tele}\|_F^2, \quad \text{s.t. } \text{diag}(B^{tele}) \leq \xi,
\]

where \( B^{tele} \) is the item-item relevance matrix for item consistency, and \( \xi \) is the hyper-parameter to control diagonal constraints. When \( \xi = 0 \), it is equivalent to the zero-diagonal constraint for \( B^{tele} \). When \( \xi = \infty \), there is no constraint on the diagonal elements of \( B^{tele} \). Note that the objective function of EASEK [46] is a special case of Eq. (9), where \( B^{tele} \) with \( \xi = 0 \).

We can obtain the closed-form solution of \( B^{tele} \) as follows:

\[
\hat{B}^{tele} = I - \hat{P} \cdot \text{diagMat}(y), \quad \gamma_j = \begin{cases} \frac{\lambda}{\lambda P_{jj} - \xi} & \text{if } 1 - \lambda P_{jj} \leq \xi, \\ 1 & \text{otherwise}, \end{cases}
\]

where \( \hat{P} = (X^TX + \lambda I)^{-1} \), and \( y \in \mathbb{R}^n \) is the vector used to check the diagonal constraint of \( B^{tele} \). Because of the inequality condition for the diagonal elements in \( B^{tele} \), \( y_j \) is determined by \( 1 - \lambda P_{jj} \). The closed-form solution may depend on \( \xi \). If \( \xi = 0 \), \( y_j \) is equal to \( 1/P_{jj} \), corresponding to \( (1 \otimes \text{diag}(\hat{P})) \). If \( \xi = \infty \), the solution becomes \( \hat{B}^{tele} = I - \lambda \hat{P} \).

Similar to \( \hat{B}^{trap} \), the solution \( \hat{B}^{tele} \) does not satisfy the non-negativity and normalization conditions to be a probability matrix. We compute the item teleportation probability matrix \( T \) by replacing the negative values with zero and normalizing \( \hat{B}^{tele} \).

\[
T = \beta \left( \text{diagMat}(\hat{B}_{\geq 0}^{tele})^{-1} \hat{B}_{\geq 0}^{tele} \right) + (1 - \beta) I,
\]

where \( \beta \) is a hyper-parameter that controls the importance of the self-loop to guarantee the convergence of random walks. (See Appendix C for detailed analysis in the supplementary material.)

3.4 Random-walk Training and Inference

**Training.** To compute the stationary distribution of S-Walk, we utilize the power method [26]. Each proximity score corresponds to the limiting distribution using Eq. (4):

\[
x_{(\infty)} = \alpha_{\infty} x_{(0)} R_{\infty} + \sum_{k=0}^{\infty} \alpha^k (1 - \alpha) x_{(0)} TR^k
\]

\[
\approx x_{(0)} \sum_{k=0}^{\infty} \alpha^k (1 - \alpha) TR^k = x_{(0)} M.
\]

Here, \( M = \sum_{k=0}^{\infty} \alpha^k (1 - \alpha) TR^k \) is the trained item-item matrix by S-Walk. (Appendix B provides the detailed pseudo-code for computing the proximity score using the power method.)

**Inference.** Given a new session \( s_{\text{new}} \), we represent a session vector \( x_{\text{new}} \) and compute the proximity score using \( M \). The proximity score for predicting the next item is finally given by \( x_{\text{new}} M \). In this process, we decay the importance of items in \( x_{\text{new}} \) to rely more on recent consumption, similar to Eq. (5):

\[
w_{\text{inf}}(i, s_{\text{new}}) = \exp \left( \frac{|s_{\text{new}}| - p(i, s_{\text{new}})}{\delta_{\text{inf}}} \right), \quad (13)
\]

where \( \delta_{\text{inf}} \) is the hyper-parameter used to control the item weight decay, and \( |s_{\text{new}}| \) is the length of the session \( s_{\text{new}} \).

4 EXPERIMENTAL SETUP

**Benchmark datasets.** We use four public datasets collected from e-commerce and music streaming services: YooChoose [2] (YC), DigiNetica [3] (DIGI), RetailRocket [4] (RR), and NowPlaying [5] (NOWP). Following existing studies [16, 31, 32, 53], we use single training-test split (single-split) datasets (i.e., YC-1/4, DIGI1). To minimize the risk of random effects, we also split the datasets (i.e., YC5, DIGI5, RR, NOWP) into five folds (five-split) which are contiguos in time, as used in recent extensive evaluation [33–35]. Following the convention [33–35], we discard the sessions with only one interaction and items that occur less than five times. Also, we split the training and test datasets chronologically and use a portion of the training set for validation, such that the validation set size is equal to that of the test dataset. (See Appendix D for detailed statistics of all benchmark datasets.)

**Competing models.** We compare our model to nine competing models. Among the non-neural models, we compare to AR [2], SR [25], STAN [11], and SLIST [5]. AR [2] is a simple model using association rules, and SR [25] is a Markov chain model together with association rules. STAN [11] is an improved model of SKNN [21], a session-based kNN algorithm. SLIST [5] is a linear model designed for SR. Among the neural models, we compare to NARM [31], STAMP [32], SR-GNN [53], NISER+ [13], and GCE-GNN [52]. Notably, SR-GNN [53] employs GNNs to consider complex sequential dependency of items. To overcome overfitting and popularity bias, NISER+ [13] uses normalized item and session embeddings on top of SR-GNN. Lastly, GCE-GNN [52] considers inter-session relationships by modeling item transitions over all sessions. (See Appendix E for the implementation details of baselines.) All source codes are available at https://github.com/jin530/SWalk.

**Evaluation protocol and metrics.** As in the common protocol [18, 31, 53], we use the **iterative revealing scheme**, which iteratively exposes an item from a session to the model, to reflect sequential user behavior throughout the entire session. We adopt several metrics to handle two scenarios: (i) To evaluate the next single item, we use **Hit Rate (HR)** and **Mean Reciprocal Rank (MRR)**, which have been widely used in existing studies [17, 32, 53]. HR and MRR measure the next item’s existence and rank in the recommendation list, respectively.

---

2https://www.kaggle.com/chadgottop/recsys-challenge-2015  
3https://competitions.codalab.org/competitions/11161  
4https://www.kaggle.com/retailrocket/e-commerce-dataset  
5https://drive.google.com/drive/folders/1ritDnO_Zec6DFEU6UND9CSVcU70ETVp5
We evaluate the accuracy and efficiency of S-Walk by comparing it with competing models. Based on thorough and extensive evaluations, we make the following conclusions.

- S-Walk exhibits state-of-the-art performance on multiple datasets. For R@20, MAP@20, and HR@20, S-Walk consistently outperforms the other models, up to 3.16%, 4.11%, and 3.65%, respectively (Section 5.1).
- The inference of S-Walk is up to 8.9 times faster than other DNN-based models. Surprisingly, S-Walk can be compressed (sparsified by zeroing-out entries via thresholding), without sacrificing its accuracy (Section 5.2).
- For long sessions, where user intent is more difficult to capture, S-Walk significantly outperforms existing models. (Section 5.3).
- S-Walk converges within 3–5 steps and is superior to the k-step method because it utilizes the teleportation model for restarts (Section 5.4).

5.1 Evaluation of Accuracy

Table 1 reports the accuracy of S-Walk and other competitors. It is found that no single model shows the best performance over all the datasets, as reported in existing studies [33–35]. Remarkably, the existing models have different tendencies for single-split and five-split datasets; while most neural models surpass non-neural models on the single-split datasets, their performances are degraded on the five-split datasets. We conjecture that the parameters of the neural models reported in existing studies are mostly biased for optimizing the single-split datasets. For five-split datasets, it is also observed that the variance in the neural models is larger than that of non-neural models. Based on these observations, it is challenging for one model to consistently achieve outstanding performance on all the datasets.

Nevertheless, S-Walk shows competitive or state-of-the-art performances. Notably, the gain of S-Walk is up to 4.08% and 8.15% in R@20 and MAP@20 over the best competing model. For all five-split datasets, S-Walk consistently surpasses the existing models for HR@20, R@20, and MAP@20. These empirical results indicate that S-Walk captures various intra- and inter-correlation among items without being biased to specific datasets.

Compared to the other metrics, S-Walk is slightly lower on MRR@20, particularly on YC and NOWP datasets. Based on the gap between SR [25] and STAN [11], we observe that the MRR@20 scores on these datasets are mostly affected by intra-session relationships. Thus, we address this by taking fewer random walks on these datasets; e.g., S-Walk(1) considers mostly intra-session relationships, confirming superior performance in MRR@20 as well.

5.2 Evaluation of Scalability

Table 2 compares the inference time between S-Walk and the other best models on several datasets. Whereas the computational cost of S-Walk is proportional only to the number of items, that of neural models also depends on the number of layers and their dimensions. Although S-Walk runs on CPU and other models run on GPU, S-Walk shows about five times faster inference time owing to its simpler structure, even with better accuracy. This property is highly desirable for deploying S-Walk for real-world applications.

We also attempt model pruning for S-Walk. As a simple strategy, we adopt magnitude pruning [10], i.e., the model is globally sparsified by zeroing out parameter values with the smallest magnitude. For instance, 100× compression means that we retain only 1% non-zero entries in M with the highest magnitude while zeroing out the remaining 99%. Surprisingly, S-Walk preserves the accuracy with highly extreme (100×) compression ratios, as depicted in Figure 3. Random pruning is a simple baseline, i.e., the parameters are randomly removed under the given compression ratio. This result indicates that the learned relationship between items is disentangled with the locality, as observed in existing studies [4, 27–30]. Owing to this valuable property, the compressed S-Walk is memory-efficient, suitable for low-resource devices, e.g., mobile and embedded applications.

5.3 Effect of Various Session Data

To further investigate the accuracy gains of S-Walk we carefully design case studies and examine the effects of session length and retrieval size. (See Appendix F for the effect of the data size on S-Walk and competing models.)

To observe how the session length affects the performance, we compare it with other best models (i.e., STAN [11], SR-GNN [53] and NISER+ [13]) by categorizing the entire sessions into two groups: short (≤ 5 items) and long sessions (> 5 items). Note that the ratio of short sessions is 77.2% (DIGI5) and 79.2% (RR), respectively. Figure 4 indicates that long sessions are much more challenging to predict users’ hidden intents effectively. Nonetheless, S-Walk significantly outperforms NISER+ [13] in long sessions by 6.2%–20.2% and 5.8%–21.0% on R@20 and MAP@20, respectively. Based on this result, we confirm that S-Walk effectively captures complicated item patterns, further improving the accuracy of longer sessions.
Table 1: Accuracy comparison of S-Walk and competing models, following experimental setup in [34, 35]. Gains indicate how much better the best proposed model is than the best baseline model. S-Walk(1) is a variant of S-Walk trained only up to the first step M(1). The best model is marked in **bold** and the second best model is *underlined*.

| Dataset | Metric | Non-Neural Models | Neural Models | Ours | Gain(%) |
|---------|--------|-------------------|--------------|------|---------|
|         | AR     | SR    | STAN  | SLIST | NARM | STAMP | SR-GNN | NISER+ | GCE-GNN | S-Walk(1) | S-Walk |        |
| DIGIS   |        |       |       |       |      |       |       |       |         |           |        |        |
| RR      |        |       |       |       |      |       |       |       |         |           |        |        |
| YC5     |        |       |       |       |      |       |       |       |         |           |        |        |
| NOWP    |        |       |       |       |      |       |       |       |         |           |        |        |
| DIGI1   |        |       |       |       |      |       |       |       |         |           |        |        |
| YC-1/4  |        |       |       |       |      |       |       |       |         |           |        |        |

Table 2: The number of floating point operations and run-time (in seconds) for inference of S-Walk and DNN-based models. Gain indicate how fast S-Walk is compared to GCE-GNN [52]. Runtime for DNN models was measured on GPU, while that for S-Walk on CPU.

| Models | YC1/4 | DIGIS | RR |
|--------|-------|-------|----|
|        | GFLOPs | Time  | GFLOPs | Time  | GFLOPs | Time  |
| SR-GNN | 1282.8 | 70.8  | 1247.2 | 12.7  |
| NISER+ | 2605.8 | 87.1  | 1551.0 | 59.7  | 501.8  | 15.7  |
| GCE-GNN| 51094.8| 108.8 | 10445.9| 74.0  | 9446.0 | 19.8  |
| S-Walk | 11.0   | 20.5  | 4.9    | 8.3   | 2.3    | 5.2   |

Gain | 4632.3x | 5.3x | 2131.3x | 8.9x | 4133.2x | 3.8x |

Figure 5 depicts the comparison results between S-Walk and the competitive models for various numbers of recommended items. For all cut-off sizes, we observe that S-Walk consistently surpasses NISER+ by 5.2%–16.9% and 5.4%–17.2% gains on Recall and MAP, respectively. In this sense, S-Walk can expand item coverage by increasing the number of steps through the random walk process in the item graph.

5.4 Ablation Study

We analyze the effect of each component, i.e., the transition model and the teleportation model, in S-Walk. As shown in Table 3, the complete S-Walk shows the best performance, compared to using SR [25] for the transition model. For the teleportation model, AR [2] shows worse accuracy than the identity matrix in the YC-1/4 dataset, where AR [2] shows the worst performance. This implies that incorrect teleportation may hinder random walks.

Figure 6 depicts the effect of the damping factor $\alpha$ in S-Walk, controlling the ratio of walks and restarts. With $\alpha = 0.7$, S-Walk shows the best performance, implying that the transition model is more dominant than the teleportation model. Finally, with various
numbers of random walk steps, we compare S-walk with the k-step method, which uses the k-step landing probability on the transition graph. (i) Without restart using the teleportation graph, performance significantly degrades, and (ii) when k \geq 2 in k-step, the accuracy continues to decrease. (iii) S-Walk usually converges within 3–5 steps, achieving the best performance on both datasets.

6 RELATED WORK

Random walk-based models. With the success of PageRank [26], the idea of the random walk has been widely adopted to address the data sparsity problem in recommender systems [55]. TrustWalker [20] addressed cold-start user/item problems using trust

Figure 7: R@20 of S-Walk using random walks with restart (RWR) and a k-step landing probability.

information between users. Starting from a target user vertex, [6, 7] estimated the proximity score using the transition probabilities after short random walks on the user-item bipartite network. Besides, random-walk-based models have been successfully deployed in the large-scale industrial systems [8]. Recently, RecWalk [37, 38] leveraged spectral properties of nearly decoupled Markov chains and combined the item model with random walks.

Session-based recommendation (SR). SR models can be categorized into three groups, i.e., Markov chain models, neighborhood-based models, and DNN-based models. For more details, please refer to recent survey papers [3, 22]. Firstly, Markov chains (MC) are useful for modeling consecutive item dependency. FPMC [43] proposed the tensor factorization using MF, and FOSSIL [14] combined FISM [24] with factorized MC. SR [25] proposed an improved MC model by combining association rules. Although they are effective for addressing the short-term item dependency, it does not utilize various patterns among items. Secondly, SKNN [21] and STAN [11] adopted the K-nearest neighbor (KNN) for SR. Although the KNN-based models have shown competitive performance on various datasets [33–35], they are generally limited in representing high-order dependency among items. Lastly, various DNN-based models have been used for SR. GRU4Rec [16, 17] employed gated recurrent units (GRU) for SR. Later, NARM [31] and STAMP [32] utilized attention mechanisms to distinguish short- and long-term item dependency. Recently, SR-GNN [53], NISER+ [13], and GCE-GNN [52] recently exploited graph neural networks (GNNs) to further analyze complex item transitions.

7 CONCLUSION

In this work, we propose a session-based recommendation using item random walks, S-Walk. To complement the drawbacks of existing models, we utilize the random walks with restart to fully capture intra- and inter-correlations of sessions. We incorporate efficient linear item models, i.e., the transition model and the teleportation model, into the item random walks process. Our extensive evaluation shows that S-Walk achieves comparable or state-of-the-art accuracy, high scalability, and fast inference speed over various benchmark datasets.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea (NRF) (NRF-2018R1A5A1060031, NRF-2021R1F1A1063843, and NRF-2021H1D3A2A03038607). Also, this work was supported by Institute of Information & communications Technology Planning & evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2019-0-00421, AI Graduate School Support Program).
REFERENCES

[1] Sami Abu-El-Haija, Amol Kapoor, Bryan Perozzi, and Jouneosek Lee. 2020. NGCN. Multi-scale graph convolution for semi-supervised node classification. In Uncertainty in Artificial Intelligence (UAI).

[2] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. 1993. Mining Association Rules between Sets of Items in Large Databases. In SIGMOD 207–216.

[3] Geoffrroy Bonnin and Dietmar Jannach. 2014. Automated Generation of Music Playlists: Survey and Experiments. ACM Comput. Surv. 47, 2 (2014), 26:1–26:35.

[4] Balázs Hidasi, Alexandros Karatzoglou, and Domonkos Tikk. 2018. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. In CIKM.

[5] Minjin Choi, Jinhong Kim, Jouneosek Lee, Hyunjung Shim, and Jongwuk Lee. 2021. Session-aware Linear Item-Item Models for Session-based Recommendation. In WWW.

[6] Fabian Christoffel, Bibek Paudel, Chris Newell, and Abraham Bernstein. 2015. Blockbusters and Wallflowers: Accurate, Diverse, and Scalable Recommendations with Random Walks. In RecSys.

[7] Colin Cooper, Sang-Hyuk Lee, Tomasz Radzik, and Yiannis Siantos. 2014. Random walks in recommender systems: exact computation and simulations. In WWW ( Companion Volume).

[8] Chantat Eskomboatchai, Pranav Jindal, Jerry Zitau Liu, Yuchen Liu, Rahul Sharma, Charles Sagnet, Mark Ulrich, and Jure Leskovec. 2018. Pixie: A System for Recommending 3+ Billion Items to 200+ Million Users in Real-Time. In WWW.

[9] Hui Fang, Guibing Guo, Danning Zhang, and Yiheng Shu. 2019. Deep Learning-based Sequential Recommender Systems: Concepts, Algorithms, and Evaluations. In IJCAI.

[10] Jonathan Franke and Michael Carbin. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In ICLR.

[11] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Web Search and Data Mining. 734–742.

[12] Harald Steck. 2019. Collaborative Filtering via Nearly Uncoupled Random Walks. In RecSys.

[13] Maksims Volkovs and Guang Wei Yu. 2015. Effective Latent Models for Binary Classification. In NeurIPS.

[14] Athanasios Nikolopoulos and George Karypis. 2019. RecWalk: Nearly Uncoupled Random Walks for Top-N Recommendation. In WSDM. 150–158.

[15] Iman Kamehkhosh, Dietmar Jannach, and Malte Ludewig. 2017. A Comparison of Frequent Pattern Techniques and a Deep Learning Method for Session-based Recommendation. In RecSys.

[16] Ziyang Wang, Wei Wei, Gao Cong, Xiao-Li Li, Xianling Mao, and Minghui Qiu. 2019. GCN: Multi-scale graph convolution for semi-supervised node classification. In CoRR.

[17] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2020. Global Context Enhanced Graph Neural Networks for Session-based Recommendation. In ICLR.

[18] Hui Fang, Guibing Guo, Danning Zhang, and Yiheng Shu. 2019. Deep Learning-based Sequential Recommender Systems: Concepts, Algorithms, and Evaluations. In IJCAI.

[19] Jonathan Franke and Michael Carbin. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In ICLR.

[20] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Web Search and Data Mining. 734–742.

[21] Shoujin Wang, Longbing Cao, and Yan Wang. 2019. A Survey on Session-based Recommender Systems. In RecSys.

[22] Meirui Wang, Pengjie Ren, Lei Mei, Zhumin Chen, Jun Ma, and Maarten de Rijke. 2019. A Collaborative Session-based Recommendation Approach with Parallel Memory Modules. In SIGIR.

[23] Iman Kamehkhosh, Dietmar Jannach, and Malte Ludewig. 2017. When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation. In RecSys.

[24] Santosh Kabbur, Xia Ning, and George Karypis. 2013. FISM: factored item similarity models for top-N recommender systems. In KDD.

[25] Balázs Hidasi, Massimo Quadrana, and Dominos Tikk. 2016. Parallel Recurrent Neural Network Architectures for Feature-rich Session-based Recommendations. In RecSys.

[26] Mohsen Jamali and Martin Ester. 2009. Automatic Generation of Music Playlists: Survey and Experiments. In ACM Comput. Surv. 47, 2 (2014), 26:1–26:35.

[27] Marie-Christine Garg, Priyanka Gupta, Pankaj Malhotra, Lovekesh Vig, and Gautam M. 2010. Collaborative Filtering with Top-k Gains for Session-based Recommendations. In CIKM.

[28] Joonseok Lee, Samy Bengio, Seungyeon Kim, Guy Lebanon, and Yoram Singer. 2014. Local collaborative ranking. In WWW.

[29] Meirui Wang, Pengjie Ren, Lei Mei, Zhumin Chen, Jun Ma, and Maarten de Rijke. 2019. A Collaborative Session-based Recommendation Approach with Parallel Memory Modules. In SIGIR.

[30] Ziyang Wang, Wei Wei, Gao Cong, Xiao-Li Li, Xianling Mao, and Minghui Qiu. 2019. GCN: Multi-scale graph convolution for semi-supervised node classification. In CoRR.

[31] Shoujin Wang, Longbing Cao, and Yan Wang. 2019. A Survey on Session-based Recommender Systems. In RecSys.

[32] Zhiqiang Pan, Fei Cai, Wanyu Chen, Honghui Chen, and Maarten de Rijke. 2020. Star Graph Neural Networks for Session-based Recommendation. In CIKM.

[33] Zhiqiang Pan, Fei Cai, Yanxiang Ling, and Maarten de Rijke. 2020. Rethinking Item Importance in Session-based Recommendation. In SIGIR.

[34] Zhiqiang Pan, Fei Cai, Yanxiang Ling, and Maarten de Rijke. 2020. Rethinking Item Importance in Session-based Recommendation. In SIGIR.

[35] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. In ICLR.

[36] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Dominos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In ICLR.

[37] Malte Ludewig and Dietmar Jannach. 2018. Evaluation of session-based recommendation algorithms. User Model. User Adapt. Interact. 28, 4–5 (2018), 331–390.

[38] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, and Jiajie Xu. 2019. Neural Attentive Session-based Recommendation. In CIKM.

[39] Iman Kamehkhosh, Dietmar Jannach, and Malte Ludewig. 2017. When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation. In RecSys.

[40] Shoujin Wang, Longbing Cao, and Yan Wang. 2019. A Survey on Session-based Recommender Systems. In RecSys.

[41] Zhiqiang Pan, Fei Cai, Yanxiang Ling, and Maarten de Rijke. 2020. Rethinking Item Importance in Session-based Recommendation. In SIGIR.

[42] Maksims Volkovs and Guang Wei Yu. 2015. Effective Latent Models for Binary Classification. In NeurIPS.

[43] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In WWW.

[44] Iman Kamehkhosh, Dietmar Jannach, and Malte Ludewig. 2017. When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation. In RecSys.

[45] Hui Fang, Guibing Guo, Danning Zhang, and Yiheng Shu. 2019. Deep Learning-based Sequential Recommender Systems: Concepts, Algorithms, and Evaluations. In IJCAI.

[46] Jonathan Franke and Michael Carbin. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In ICLR.

[47] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In ICDM.

[48] Mohsen Jamali and Martin Ester. 2009. TrustWalker: a random walk model for combining trust-based and item-based recommendation. In KDD.

[49] Dietmar Jannach and Malte Ludewig. 2017. When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation. In RecSys.

[50] Dietmar Jannach, Malte Ludewig, and Lukas Lerche. 2017. Session-based item recommendation in e-commerce: on short-term intents, reminders, trends and discounts. User Model. User Adapt. Interact. 27, 3–5 (2017), 351–392.

[51] Olivier Jeunen, Jan Van Balen, and Bart Goethals. 2020. Closed-Form Models for Recommender Systems with Next-Basket Recommendations. In IJCAI.

[52] Zhiqiang Pan, Fei Cai, Yanxiang Ling, and Maarten de Rijke. 2020. Rethinking Item Importance in Session-based Recommendation. In SIGIR.

[53] Shoujin Wang, Longbing Cao, and Yan Wang. 2019. A Survey on Session-based Recommender Systems. In RecSys.

[54] Ziyang Wang, Wei Wei, Gao Cong, Xiao-Li Li, Xianling Mao, and Minghui Qiu. 2020. Global Context Enhanced Graph Neural Networks for Session-based Recommendation. In SIGIR.

[55] Shoujin Wang, Longbing Cao, and Yan Wang. 2019. A Survey on Session-based Recommender Systems. In RecSys.

[56] Zhiqiang Pan, Fei Cai, Yanxiang Ling, and Maarten de Rijke. 2020. Rethinking Item Importance in Session-based Recommendation. In SIGIR.

[57] Shoujin Wang, Longbing Cao, and Yan Wang. 2019. A Survey on Session-based Recommender Systems. In RecSys.
A CLOSED-FORM SOLUTIONS

We provide a detailed derivation of the closed-form solutions for the item transition model and the item teleportation model in Section 3.2 and 3.3. Based on the closed-form solutions, we can further apply for row-level weights to reflect the timeliness of sessions, i.e., the more important, the more recent sessions.

A.1 Linear Item Transition Model

Given the input matrix $Y \in \mathbb{R}^{m \times n}$ and the output matrix $Z \in \mathbb{R}^{m \times n}$, the objective function is expressed as

$$
\arg\min_{\tilde{B}^{\text{tran}}} \mathcal{L}(\tilde{B}^{\text{tran}}) = \| (Z - Y \cdot \tilde{B}^{\text{tran}}) \|^2_F + \lambda \|B^{\text{tran}}\|^2_F.
$$

(14)

The first-order derivative of Eq. (14) over $B^{\text{tran}}$ is then given by

$$
\frac{1}{2} \cdot \frac{\partial \mathcal{L}}{\partial B^{\text{tran}}} = (-Z^T)(Z - Y \cdot B^{\text{tran}}) + \lambda B^{\text{tran}},
$$

(15)

Solving for $B^{\text{tran}}$ so that Eq. (15) becomes 0 gives the closed-form solution of Eq. (14):

$$
\tilde{B}^{\text{tran}} = \hat{P}^T \cdot (Y^T Z),
$$

(16)

where $\hat{P} = (Y^T Y + \lambda I)^{-1}$.

A.2 Linear Item Teleportation Model

To solve the constrained optimization problem, we define a new objective function $\mathcal{L}(B^{\text{tele}}, \mu)$ by applying a Lagrangian multiplier and a KKT condition:

$$
\mathcal{L}(B^{\text{tele}}, \mu) = \| (X - X \cdot B^{\text{tele}}) \|^2_F + \|B^{\text{tele}}\|^2_F + \|\mu\|^2_F
$$

(17)

s.t.

$$
\text{diag}(B^{\text{tele}}) \leq \xi
$$

(18)

where $\mu \in \mathbb{R}^n$ is the KKT multiplier that satisfies $\forall i, \mu_i \geq 0$. Then, we differentiate $\mathcal{L}(B^{\text{tele}}, \mu)$ with respect to $B^{\text{tele}}$ to minimize Eq. (18):

$$
\frac{1}{2} \frac{\partial \mathcal{L}(B^{\text{tele}}, \mu)}{\partial B^{\text{tele}}} = (-X^T)(X - X \cdot B^{\text{tele}}) + \lambda B^{\text{tele}} + \lambda \text{diag}(\mu)
$$

(19)

Setting this to 0 and solving by $B^{\text{tele}}$ gives the optimal $\tilde{B}^{\text{tele}}$ as

$$
\tilde{B}^{\text{tele}} = (X^T X + \lambda I)^{-1} \cdot (X^T X + \lambda I - \mu \cdot \text{diag}(\mu))
$$

(20)

Finally, $\mu$ can be expressed as follows:

$$
\mu_j = \frac{(1 - \lambda P_{jj} - \xi)}{P_{jj}} = \frac{(1 - \xi)}{P_{jj}} - \lambda.
$$

(22)

Substituting $\mu$ in Eq. (20) and enforcing non-negative elements in $\tilde{B}^{\text{tele}}$ give $\tilde{B}^{\text{tele}}$:

$$
\tilde{B}^{\text{tele}} = I - \hat{P} \cdot \text{diag}(\gamma),
$$

(23)

$$
\gamma_j = \begin{cases} 
\frac{\lambda}{1 - \lambda P_{jj}} & \text{if } 1 - \lambda P_{jj} \leq \xi \\
\frac{\xi}{P_{jj}} & \text{otherwise}.
\end{cases}
$$

(24)

Here, $\gamma$ is a vector defined by $\gamma = \mu + \lambda \cdot 1$.

B TRAINING OF S-WALK

Algorithm 1 describes how the final item-item matrix $M$ is trained using the transition matrix $R$ and the teleportation matrix $T$. First, we define $M_{(0)}$ as the identity matrix. Then, we repeatedly update $M_{(k)}$ using $R$ and $T$. For example, $M_{(1)}$ can be thought as $R$ with a probability of $\alpha$ and $T$ with a probability of $\lambda (1 - \alpha)$. After $M_{(k)}$ converges, we use $M_{(k)}$ as the item-item matrix for inference.

Algorithm 1: Training procedure for S-Walk

Input: Transition matrix $R$, teleportation matrix $T$.

Output: S-Walk model $M$.

1. $M_{(0)} \leftarrow I$
2. $k = 0$
3. repeat
4. $k \leftarrow k + 1$
5. $M_{(k)} \leftarrow \alpha M_{(k-1)} R + (1 - \alpha) T$
6. until $\|M_{(k)} - M_{(k-1)}\|_1 \leq \epsilon$
7. $M \leftarrow M_{(k)}$

C THEORETICAL ANALYSIS

Property of the item transition matrix. The item transition matrix $R$ can be viewed as a generalized Markov model. When representing partial representations, the importance of items can be different. As $\delta_{\text{pos}}$ in Eq. (5) is small, it can be close to a simple Markov model [25], capturing consecutive transitions between items. We can also consider a broader range to represent the transition between items. While the simple Markov model is mostly based on the frequency of consecutive items, our linear model can estimate the sequential correlation between items. As a result, it can outperform existing Markov models [14, 24, 25, 43].

Property of the item teleportation matrix. As discussed in [46], assume that a training matrix $X$ is $|S|$ samples with $|I|$ random variables, i.e., $x \sim N(0, \Sigma)$ following a Gaussian distribution with zero mean and the covariance matrix $\Sigma \in \mathbb{R}^{n \times n}$. Here, the estimate of the covariance matrix is $\hat{\Sigma} = X^T X / |S|$, and $\hat{P} = \hat{\Sigma}^{-1}$ is the estimate of the precision (or concentration) matrix. By solving the closed-form equation in Eq. (10), the item teleportation matrix $T$ can be interpreted as the precision matrix by estimating the co-occurrence matrix $X^T X / |S|$. In other words, the precision matrix can be viewed as the similarity matrix between items, which has been used in the recent neighborhood-based approach [48, 49].
Table 4: Statistics of the benchmark datasets. #Actions indicates the number of entire user-item interactions.

| Split  | Dataset | Actions / Sess. | Sessions / Sess. | Items / Sess. | Actions | Items |
|--------|---------|-----------------|------------------|--------------|---------|-------|
| 1-split| YC-1/4  | 7,909,307       | 1,939,891        | 30,638       | 4.08    | 3.28  |
|        | DIGI1   | 916,370         | 188,807          | 43,105       | 4.85    | 4.08  |
| 5-split| YC5     | 5,426,961       | 1,375,128        | 32,137       | 4.08    | 3.28  |
|        | DIGI5   | 203,488         | 41,755           | 31,968       | 4.86    | 4.08  |

Model convergence. The graph $\mathcal{G}_M$ is defined by a combination of $R$ and $T$. The non-negative values in both matrices imply the positive correlations in covariance matrix $R^T$. The edges based on the correlations make graph $\mathcal{G}_M$. Besides, $T$ has a self-loop connection as in Eq. (11). Based on these structures, the transition matrix $R$ is an ergodic Markov model, i.e., irreducible and aperiodic. That is, the landing probabilities of S-Walk converge to a limiting distribution. It is found that S-Walk converges by 3–5 steps, as observed in 5.4.

D DATASET DESCRIPTION

Table 4 summarizes the detailed statistics of all the benchmark datasets. These datasets are collected from e-commerce and music streaming services: YooChoose$^6$ (YC), Diginetica$^7$ (DIGI), Retail-Rocket$^8$ (RR), and NowPlaying$^9$ (NOWP). For YC and DIGI, we use single-split datasets (i.e., YC-1/4 and DIGI1), following existing studies [16, 31, 32, 53]. To evaluate on large-scale datasets, we further experiment on five-split experiments (i.e., YC, DIGI5, RR, NOWP), used in the recent empirical analysis [33–35]. For single-split datasets, we use the last day as the test set for YC-1/4 and the sessions of the last seven days as the test set for DIGI1, as done in [33–35]. For five-split datasets, we divide the datasets into five disjoint successive splits. For each split, the last $N$-days sessions were used for the test set. We choose $N$ for the test set as follows: one for YC, two for RR, five for NOWP, and seven for DIGI.

E REPRODUCIBILITY

Implementation details. For S-Walk, we tuned $\alpha$ and $\beta$ among $[0.1, 0.3, 0.5, 0.7, 0.9]$, $\delta_{pos}$ and $\delta_{inf}$ among $[0.125, 0.25, 0.5, 1, 2, 4, 8]$ using the validation set selected from the training set for the same period as the testing set. For the baseline models, we used the best hyper-parameters reported in [33, 34]. We implemented the proposed model and STAN [11] using NumPy. We used public source code for SR-GNN$^{10}$ [53] and implemented NISER+ [13] using SR-GNN code. We used all source code for the other baseline models$^{11}$ in [35]. For reproducibility, we conducted the reported results for all the baseline models, and actually verified whether the performance of baseline models could be reproduced with an error of 1–2% or less in our implementation environments. We conducted

\begin{table}[h]
\centering
\begin{tabular}{ |c|c|c|c|c|c|c|c| } \hline
Dataset & $\alpha$ & $\beta$ & $\lambda$ & $\delta_{pos}$ & $\delta_{inf}$ \\
\hline
YC-1/4  & 0.5 & 0.7 & 10 & 1 & 1 \\
DIGI1   & 0.5 & 0.9 & 10 & 0.5 & 2 \\
\hline
YC5     & 0.5 & 0.7 & 10 & 1 & 1 \\
DIGI5   & 0.7 & 0.7 & 10 & 0.5 & 4 \\
RR      & 0.5 & 0.9 & 10 & 0.25 & 4 \\
NOWP    & 0.5 & 0.9 & 10 & 1 & 1 \\
\hline
\end{tabular}
\caption{Hyper-parameter settings of S-Walk. $\lambda$ is a L2-weight decay, $\alpha$ is a damping factor and $\beta$ is the self-loop factor. $\delta_{pos}$ is a weight decay by item position and $\delta_{inf}$ is a weight decay by inference item position.}
\end{table}

Figure 8: R@20 and MAP@20 of S-Walk and competing models over various size of training session matrices.

Hyper-parameter setting. Table 5 reports all hyper-parameters for S-Walk. Firstly, the hyper-parameters of the left column are the factor that controls the involvement of random walk models. $\alpha$ is a damping factor that determines the proportion of the transition matrix $S$ and teleportation matrix $T$. $\beta$ controls the probability that the surfer jumps to the item rather than to itself. Lastly, the hyper-parameters of the right column are model hyper-parameters used for the transition model and the teleportation model.

F ADDITIONAL EXPERIMENTAL RESULTS

We also evaluate the accuracy of S-Walk and the competing models by varying the size of the entire session from 10–100%. As shown in Figure 8, S-Walk consistently achieves better accuracy than other baselines. Even though all models degrade performance with a smaller training set, S-Walk suffers the least from sparser data. Notably, DNN-based models show significant degradation in accuracy with smaller training sets owing to their complex structures and enormous parameters. (For the other datasets, we still observe similar tendencies.) Based on this observation, S-Walk is still more effective than the baseline models, even on small-scale datasets.