PARSRec: Explainable Personalized Attention-fused Recurrent Sequential Recommendation Using Session Partial Actions

Ehsan Gholami
egholami@ucdavis.com
University of California, Davis
Davis, California, USA

Mohammad Motamedi
mmotamedi@ucdavis.edu
University of California, Davis
Davis, California, USA

Ashwin Aravindakshan
aaravind@ucdavis.com
University of California, Davis
Davis, California, USA

ABSTRACT

The emerging meta- and multi-verse landscape is yet another step towards the more prevalent use of already ubiquitous online markets. In such markets, recommender systems play critical roles by offering items of interest to the users, thereby narrowing down a vast search space that comprises hundreds of thousands of products. Recommender systems are usually designed to learn common user behaviors and rely on them for inference. This approach, while effective, is oblivious to subtle idiosyncrasies that differentiate humans from each other. Focusing on this observation, we propose an architecture that relies on common patterns as well as individual behaviors to tailor its recommendations for each person. Simulations under a controlled environment show that our proposed model learns interpretable personalized user behaviors. Our empirical results on Nielsen Consumer Panel dataset indicate that the proposed approach achieves up to 27.9% performance improvement compared to the state-of-the-art.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; *Personalization*; *Personalization*; • **Human-centered computing** → Information visualization; • **Computing methodologies** → Machine learning.

KEYWORDS

Sequential Recommendation; Next Item Recommendation; Personalized User Attention; Assortment; Attention; Embedding

ACM Reference Format:
Ehsan Gholami, Mohammad Motamedi, and Ashwin Aravindakshan. 2022. PARSRec: Explainable Personalized Attention-fused Recurrent Sequential Recommendation Using Session Partial Actions. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534678.3539432

1 INTRODUCTION

The task of recommender systems is to delineate users' interests accurately. Recommender systems help providers offer viable alternatives to users as they navigate amongst a vast number of available choices. They achieve this by leveraging users' historical behavior to extract meaningful patterns that help predict users' future interests. These patterns often change over time and are heterogeneous across users. Therefore, deriving functional patterns becomes increasingly challenging with growing numbers of users, items, and user-item actions. Recommender systems focus on capturing these evolving, diverse, and high-dimensional behaviors.

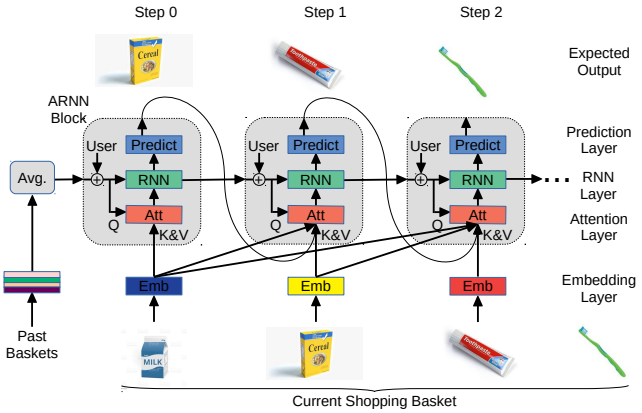


Figure 1: Coarse Architecture of PARSRec: In every step, first, the Attention module investigates the items that currently exist in the shopping basket to identify those items that will considerably impact the selection of the next item. Then, it represents such items in the latent space as a hint to the recurrent network. Furthermore, the user-aware nature of the proposed architecture makes it possible to leverage users' idiosyncrasies for predicting and suggesting the next item.

Two types of recommender systems have gained popularity in recent research, i) sequential and ii) session-based recommenders. Sequential recommenders often consider all historical user actions as a single ordered sequence and try to successively infer each user action based on the user's prior actions in the sequence. Session-based recommenders only leverage the user's most recent actions called session (e.g., anonymous online shopper without an existing historical behavior). The state-of-the-art approaches benefit from deep neural nets to enhance the performance of recommendation tasks. Recurrent Neural Networks (RNN) [6, 32] and their improved variants such as Gated Recurrent Units (GRU) [17, 18] and Long Short-Term Memory (LSTM) [7] often capture all the previous actions of the user in the past via hidden states. This strategy allows them to understand complex user behaviors. However, RNNs suffer from long-range dependencies because long-range back-propagated gradients can vanish or explode. LSTM and GRU prove effective in some fields by resolving this issue but have their limitations in the

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08.
https://doi.org/10.1145/3534678.3539432

field of sequential recommendation. They tend to summarize the session information into a single representation. For example, in machine translation, words within a sentence are related to each other (in various degrees). However, not all items within a shopping basket are necessarily related. For example, in the shopping basket *(milk, cereal, laundry detergent)*, the choice of laundry detergent could be completely independent of milk and cereal. Encoding the entire session into one (or a few) representation(s) will entangle irrelevant information together and would possibly make it harder for the decoder to detangle them. These methods assume a natural order to historical user actions, which does not always hold in real-world applications. Moreover, they tend to become slow when processing long sequences due to their sequential nature.

More recent recommender systems use attention mechanisms to overcome the issues mentioned above by identifying a smaller number of actions most relevant to the next item recommendation. They are effective in many applications and can provide helpful interpretable visuals for item-item relationships. The self-attention mechanism in Transformers learns long-range global item-item relationships and utilizes that along with the items in the user's historical actions to learn heterogeneous user behaviors implicitly. They require less dense data and can run faster in parallel. However, the state-of-the-art transformer recommender systems usually can handle limited input length, limiting the number of user actions[3, 21, 36]. This is due to the extra positional embedding required to capture the relative or absolute order of the items in a sequence. On top of that, item relations generally differ from one user to another. For instance, user A might purchase milk for cookies, while user B might purchase milk for cereal and independently buy cookies simultaneously. Most attention-based models capture only the universal item-item relationships based only on item sequence and lack personalized interpretable user behavior representations.

Inspired by these methods, we propose PARSRec, a **P**ersonalized **A**ttention-based **R**ecurrent **S**equential **Rec**ommender that fuses the attention mechanism with RNNs, illustrated in Figure 1, to address the limitations mentioned above. Our framework partitions a user's actions into two groups: *i)* information from past sessions and *ii)* items interacted with so far during the current session. There are multiple reasons for this approach. In a wide range of applications, the partial knowledge of the current session provides much richer information than the previous sessions [13]. For example, one may purchase cereal and milk on a trip to a grocery store. Knowing the partial information of the cereal is more likely to help predict the next item, milk, accurately compared to the shopping baskets in the past. In another example, the list of the music tracks that a user listens to in a session is more likely to be related to other songs in the same session than tracks in other sessions [12]. Second, in many real-world cases, there is no order to the items in a session (e.g., items within a basket), whereas the sessions themselves may follow a chronological order. Lastly, our model eliminates the need for positional embeddings by such partitioning of the historical interactions. This, in turn, reduces the network's memory footprint.

The hidden states of the RNN network in our model carry the user information and the user's historical behavior. The attention layer, which is agnostic to orders, uses the hidden state to determine which items within the current session are more relevant in predicting the next item. PARSRec outperforms the state-of-the-art methods on Nielsen's real-world Consumer Panel dataset, as detailed in Section 5. The model extracts interpretable personalized user behavior by using explicit user representations in the attention layer's queries. We show that PARSRec can accurately explain personalized user behavior in a controlled environment on a synthetic dataset. This powerful explanation allows the provider to fully understand the underlying user behavior beyond a simple next item recommendation to make informed decisions on many tasks. Examples of tasks that can benefit from this knowledge are assortment optimization, assortment allocation (what items go together on the same shelf or a webpage design), and personalized coupons, discounts, and displays. Our recurrent model capacity is independent of the sequence length. Its complexity depends only on the number of items within a session which is usually small as detailed in Section 5. The network can utilize any length of user history without increasing the capacity or complexity of the model. The key contributions of our work are:

- we propose a model that uses attention layers combined with RNNs for the task of sequential recommendation. We show that our model outperforms various state-of-the-art methods on synthetic and real-world data under different evaluation metrics.
- we show that our model learns personalized user behaviors and offers interpretable results through visualizing item relationships.
- we conduct an ablation study on variations of the proposed model to evaluate the contribution of components of the model and report the most effective architecture.

2 RELATED WORK

We review the works on closely related recommender systems to our framework.

General Recommendation: Collaborative Filtering (CF) is one of the classic approaches in the field of recommender systems [16, 22, 35]. CF infers user preferences from their historical interactions. Matrix Factorization (MF) is a successful CF method that uses a shared space to represent both users and items [23, 30]. More recent approaches use deep learning to improve the effectiveness of models. However, the state-of-the-art works rely on deep learning to provide recommendations [11, 12, 31, 37, 42].

Sequential Recommendation: Studies of sequential recommendation aim to extract item transitions in a sequence of items a user interacts with. Markov Chain (MC) models capture such transitions. Factorizing Personalized Markov Chain (FPMC) [34] and its extension Hierarchical Representation Model (HRM) [39] combine MF and MC to extract a personalized item transition. Recurrent models have also shown promising results in the field of sequential recommendation. RNNs and their variants (e.g., GRU and LSTM) have been used for modeling user interaction sequences. Most RNN models encode a user's historical behavior into a representation vector and use that along with the current interaction as the input to the model to predict the next action. DREAM [41] adopted MCs in a recurrent setting to create dynamic representations of users. GRU4Rec [18] and GRU4Rec$^+$ [17] use Gated Recurrent Units for session-based recommendations. Some works utilize memory networks to store users' actions in RNNs for recommendation [4, 20].

BINN [27] uses two components to capture long and short-term preferences in the RNN setting.

Attention-based Recommendation: Attention mechanism has gained popularity in many fields (e.g., machine translation) due to its promising performance and interpretability. Recent state-of-the-art recommender systems use the attention mechanism. NARM uses attention and encoder layers to model the user's sequential behavior and the user's session purpose [25]. ACA-GRU leverages an attention mechanism to build a context-aware recommender system [43]. STAMP uses attention to capture users' general interests from the long-term memory of a session context and users' current interests from the short-term memory [28]. KGAT [40] uses the attention mechanism on knowledge graphs for the recommendation. SASRec [21] and BERT4Rec [36] use uni-directional and bi-directional self-attention mechanisms (i.e., Transformers) to capture item-item relationships and have achieved state-of-the-art performances. TiSASRec [26] incorporates time intervals in attention mechanism for a time-aware recommender system.

Existing methods that use attention mechanisms usually learn an implicit representation of users using the global item-item relationships. These methods often limit the length of the input sequence (number of user's historical actions) and require cropping the input to a pre-set max-length. Some methods also assume a rigid order to the sequence of user actions. We seek to design a model that addresses these limitations. Our model outperforms the state-of-the-art methods and learns explainable personalized item-item relationships that provide insights into user choices.

3 PROBLEM STATEMENT

In recommender systems, a set of users, $U = \{u_1, u_2, ..., u_{|U|}\}$, interact with a set of items, $V = \{v_1, v_2, ..., v_{|V|}\}$. User-item interactions may constitute implicit feedback from the user. Examples of user-item interactions are purchasing an item, listening to a track, watching a movie, or clicking on a link. A sequence of interactions by user $u \in U$, denoted by S_u, is partitioned into sessions $(S^u_{t_1}, S^u_{t_2}, ..., S^u_{t_{|S_u|}})$. Each session $S^u_{t_i} \subseteq V$ is the interaction of user u with a set of items $\{v^{(S)}_j | v^{(S)}_j \in V, 1 < j < n^u_i\}$ at time t_i. The total number of items in the session is denoted by $n^u_i = |S^u_{t_i}|$, and the sessions are in chronological order $(t_1 < t_2 < ... < t_{|S_u|})$. An example of $S^u_{t_i}$ would be a shopping basket or a session of listening to music tracks. In this paper, we use the terms session and basket interchangeably. We assume there is no specific chronological order to items within a session. For instance, a shopping basket at a brick-and-mortar store does not provide a meaningful order. However, if there is a meaningful order to items within a session, an ideal solution would be able to capture that relationship as well. The task of sequential recommendation is to predict the next item in the session $v^{(S)}_{j+1}$ given the history of user behavior $(S^u_{t_1}, ..., S^u_{t_{i-1}})$, and the subset of items $[v^{(S)}_1, ..., v^{(S)}_j]$ that user has interacted with so far during the current session $S^u_{t_i}$.

4 MODEL ARCHITECTURE

In this section, we introduce a new sequential recommendation model called **PARSRec**. A **P**ersonalized **A**ttention-based **R**ecurrent **S**equential **Rec**ommender that combines the power of recurrent neural networks (RNN) and scaled dot-product attention into a modified **A**ttention-fused **RNN**: **ARNN** (Figure 2). The outer structure of the model is similar to vanilla RNN, i.e., each ARNN block receives a hidden state and an input and produces an output and the next hidden state. However, the inner architecture of the ARNN block is different than that of vanilla RNN. All ARNN blocks in Figure 2 use the same parameters. Next we detail the architecture. A formal and detailed explanation of each layer is provided in Appendix B.

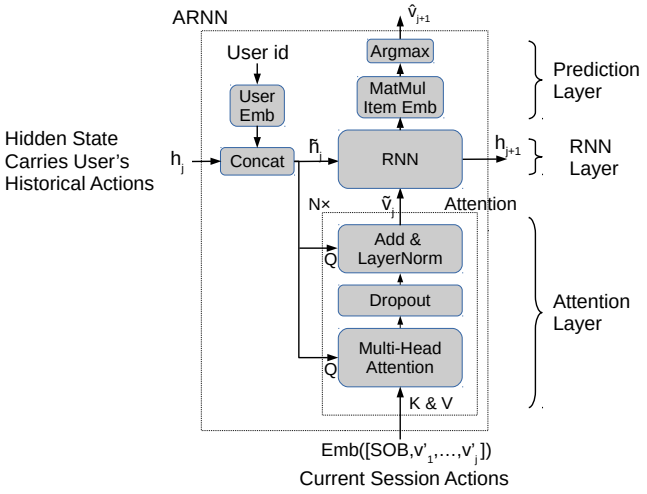


Figure 2: ARNN: Attention-fused RNN Block

4.1 Embedding Layer

We build two embedding tables, $\mathbf{E}^U \in \mathbb{R}^{|U| \times d_u}$ and $\mathbf{E}^V \in \mathbb{R}^{|V| \times d_v}$, to represent users and items in the latent space, respectively. We denote the latent dimensions of users by d_u and items by d_v.

4.2 Attention Layer

The attention layer can learn dependencies between two representations regardless of their position in a sequence. We use the attention layer to identify the existing items in the current session that will considerably impact the selection of the next merchandise. The attention module in our model consists of three sub-layers:

Multi-Head Attention Block: We adopt the multi-head attention module described in [38]. It is shown that learning from multiple sub-spaces of representations is more flexible than a single representation. Multi-head attention inherently breaks down the representations into smaller sub-spaces, applies attention on each sub-space, and then concatenates the outputs back into a single representation. The attention block is a scaled dot-product of three vectors, key **K**, value **V**, and query **Q**. Intuitively, attention is a weighted sum of rows in **V** and weights are defined by similarity of rows in matrices **Q** and **K**. In our model, **K** and **V** are the same and their rows are embeddings of items $\{v^{(S)}_1, ..., v^{(S)}_{n^u_i}\}$. The matrix **Q**, however, is different and is the concatenation of user embedding and the previous hidden state. Similar to [38], to alleviate overfitting, help with stability, and speed up the training process, we add

Layer Normalization (LN), and Dropout. We refer to the described architecture as attention layer in the rest of the paper.

Stacking Attention Layers: Utilizing more numbers ($N > 1$) of attention layers hierarchically allows the model to learn richer and more complex relationships. We investigate this structure by stacking multiple attention layers using the same query $\mathbf{Q}$ and feeding the output of the lower layer as the next layer's key $\mathbf{K}$ and value $\mathbf{V}$. It may be beneficial to use multiple attention layers depending on the application and complexity of the relationships.

4.3 Recurrent Architecture

We combine the attention layer with a vanilla RNN by using the attention layer output as the input $\tilde{v}_j$ to the RNN block and attention query $\mathbf{Q}$ as the input hidden state $\tilde{h}_j$ to the RNN block.

The hidden state $\tilde{h}_j$ carries the user information and the current state of the session. We concatenate user embedding to the hidden state at every step to ensure it does not vanish as we progress. The initial hidden state encodes the interaction history, which is the weighted average of embeddings of items in the users' historical actions. The attention layer identifies the items whose presence in the basket is expected to highly impact the selection of the next item. Subsequently, these items are offered to the RNN to enhance its prediction accuracy.

Prediction Layer: We use the output of the RNN block at each step to predict the next item. The output of the RNN layer is multiplied by $\mathbf{E}^V$ to provide a similarity vector of size $|V|$. The indices with higher values represent items that are more likely to be interacted with next. We can rank this vector and make recommendations based on that. The objective loss function will convert the output vector to a probability vector using a softmax layer, which is discussed in the next subsection.

4.4 Loss Function

We adopt cross entropy loss function as our objective function:

$$-log\Big(\frac{exp(y[target])}{\sum_k exp(y[k])}\Big) \tag{1}$$

where y is the output of ARNN layer and $target$ is the ground truth item to be predicted v'_{j+1}. This objective function combines LogSoftmax and negative log-likelihood loss. The loss is averaged across observations of each mini-batch. We also ignore any padded items in training explained further in detail.

4.5 Training

We train the model to predict the items in the current session S_i^u sequentially given a user and their historical behavior. A basket of size n_i^u will have n_i^u prediction steps. Following a common practice in sequential recommender systems [21] and machine translation models [38], we benefit from teacher enforcing during the training. However, since we do not assume any natural order for items in the session, it is beneficial to know which items in the basket are the most related items to predict the next item j. Some recommenders use bidirectional learning that utilizes both left and right items for prediction [36]. The idea is to randomly mask an item (or items) and use remaining information on both left and right to predict masked item(s) [1, 5]. To alleviate the same issue, we take a different approach and modify our teacher enforcing at each step as follows:

Table 1: Post preprocessing datasets statistics

Dataset	#users	#items	#actions	avg. actions/user	Density
Synthetic	8,192	2,000	5.8M	710	0.10
Nielsen	12,800	1,302	15.1M	1,179	0.21

$$v'_j = \begin{cases} SOB & j = 0 \\ \begin{cases} \hat{v}_j & if\ \hat{v}_j \in \{v'_{j+1}, ..., v'_{n_i^u}\} \\ rand(\{v'_{j+1}, ..., v'_{n_i^u}\}) & if\ \hat{v}_j \notin \{v'_{j+1}, ..., v'_{n_i^u}\} \end{cases} & 0 < j < n_i^u \\ EOB & j \geq n_i^u \end{cases} \tag{2}$$

where *SOB* and *EOB* are **S**tart and **E**nd **O**f **B**asket tokens. In other words, we start with *SOB* as input of step 0, and if the predicted output in each step is in the rest of the basket, we add that to inputs of the next step. Otherwise, we randomly pick an item from the remaining items in the basket to perform the teacher enforcing. This method of teacher enforcing has the benefit of bringing related items within a session closer to each other in position and the attention layer can capture those relationships. For instance, a basket in the dataset might be *(cereal, toothbrush, milk, toothpaste)*. The aforementioned teacher enforcing method brings *(cereal, milk)* and *(toothbrush, toothpaste)* closer in position by accurately predicting the item-item relationships.

To perform training in mini-batches, we randomly put baskets of similar sizes in the same mini-batch to avoid unnecessary calculation steps. We pad all baskets on the left with *SOB* and if some baskets within the mini-batch are of different sizes, we pad them with *EOB* on the right. We use *Adam* and *Sparse Adam* optimizers to optimize network's non-sparse and embedding parameters, respectively. Both optimizers adaptively estimate the moments. We also examined Stochastic Gradient Descent (SGD) optimizer and found similar results.

Causality: To avoid leaking information from the future to the prediction of the current step, we only feed the model with the inputs seen at previous steps. The only added item will be either the prediction of the last step or a randomly selected item from the remaining basket. We do not use any extra information from the remaining of the session (including the item we are predicting at the current step). Our modified teacher enforcement makes sure no future information is leaked to the past.

5 EXPERIMENTAL RESULTS

5.1 Datasets

We evaluate our model on two datasets: a controlled synthetic dataset and a real-world dataset. We discuss the details of each data preprocessing next.

Data Synthesis: Verifying a novel method for capturing item-item relationships in an empirical setting is challenging due to the lack of ground truth. Often these relationships are not known and could vary drastically from user to user. Owing to these issues, most studies primarily evaluate the face validity of the model [4, 21]. The synthetic data allows us to test various effects and evaluate our

model under a controlled environment. We start by examining our proposed model on synthetic data, comparing it to benchmarks, and evaluating the characteristics of the model (e.g., personalized and universal item-item relationships). Retail market basket data is one of the well-studied areas [2]. We follow common practice in synthesizing sessions to recreate heterogeneous user behaviors and various item-item relationships [8, 29]. We extract personalized user behaviors from our model using only the basket data, withholding any information regarding items' relations or user choice models. We compare the extracted user behaviors to that of known values in our simulation. The simulation results validate the model hypothesis and provide an additional basis for performance under empirical data, where results are similar and consistently outperform state-of-the-art models. To the best of our knowledge, no proposed sequential recommender model provides a validated study on personalized item-item relationship. We next discuss the data generation schema.

The simulation of baskets is as follows: for a basket, S_t^u, with size n_t^u purchased by user u at time t, first the user chooses n_t^u categories from a set of available categories. Then from each chosen category the user chooses a product j. A category c is a set of similar items (e.g., various types of cereal, milk) and categories are disjoint. For simplicity, each basket can contain at most one item from a category and not all categories need to be purchased in a single basket. We use the multivariate normal distribution for simulating category choice model to capture various types of category-category relationships (e.g., complements vs. substitutes):

$$p_{ct}^u = \alpha_c + \epsilon_{ct}^u \tag{3}$$

where p_{ct}^u is probability of purchasing from category c by user u at time t, α_c is a constant category specific utility, and $\epsilon_{ct}^u \sim \mathcal{N}(0, \Sigma)$. The user chooses n_t^u categories with highest probability to purchase from at time t. Next, we use multinomial probit model [10] to simulate product choice within each category, explained in Appendix A. A positive value in the covariance matrix Σ indicates two categories are purchased together frequently (e.g., milk and cereal), while a negative value represents contrary of that (e.g., fresh vs. frozen meat). We manually chose a block diagonal form for the covariance matrix Σ. Each block represents categories that have relations, while categories from separate blocks are independent of each other. We chose various sizes for different blocks (2-3-4) to represent low-mid-high order relationships between categories. For off-diagonal values of each block, we manually chose various positive/zero/negative values to represent complementary/coincidence/substitute product relationships. Further, we divide users into multiple disjoint groups. Each group follows a different Σ to illustrate a deterministic user heterogeneity on top of the unknown user specific choice error terms. Heatmaps of Σ are illustrated in Fig. 3. We synthesize a less dense dataset compared to the real-world data with 8,192 users, 2,000 items, and 5.8M million user-item interactions. Data statistics are summarized in Table 1. Further details of parameter values are presented in Appendix A.

Empirical: We use the Nielsen Consumer Panel[1] dataset. The dataset comprises a representative panel of households. It includes all household purchases (from any outlet) intended for personal and at-home use [19]. This is a particularly challenging dataset in terms of sequential recommendation. Households are geographically dispersed over all states and demographically balanced to accurately represent the market in each area. The sessions are from different retailers. Over 4.3 million products in the dataset cover a variety of items including groceries, health and beauty aids, and alcohol. The majority of sequential recommenders evaluate their models on a limited set of users (e.g., within a state), a single retailer (e.g., a clothing retail or content provider), and a related set of items (e.g., all movies, or all clothing). We use data from 2014~2019 and randomly select 12,800 actively participating users with 15M+ user-item interactions from 50,000+ retailers. The products are grouped into hierarchical categories: 10 departments ⟶ 118 groups ⟶ 1,305 modules ⟶ 4.3+ million UPCs. We use *Module* level as items in our experiment to reduce the sparsity of purchase patterns, and will refer to it as products in the rest of the paper. We keep one copy of the same item purchased in multiple quantities in each basket. We exclude items and users that have less than 10 records in the entire dataset. The statistics of data are summarized in Table 1.

5.2 Experimental Setup

The optimal architecture of PARSRec includes $N = 1$ attention layer with $h = 2$ heads. We observe that adding extra attention layers does not increase the performance significantly, presumably because the session lengths are short and a single transformer is able to induce all the necessary relationships within a session. Item and user embeddings are the size of $d_v = d_u = 128$. We initialize the item and user embedding matrices with uniform distributions of range $[-\frac{1}{\sqrt{|V|}}, \frac{1}{\sqrt{|V|}}]$ and $[-\frac{1}{\sqrt{|U|}}, \frac{1}{\sqrt{|U|}}]$, respectively. Other matrix parameters with size $n \times m$ are initialized with $\sim \mathcal{N}(0, \frac{2}{n+m})$. Dropout rate is set to 0.1 and mini-batch size is 256. We used Adam and Sparse Adam optimizers with learning rate 0.001, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. The gradient is clipped at 30. We use PyTorch=v1.10.1 to implement the model[2].

5.3 Evaluation Metrics

We use users' most recent session $S^u_{t_{|S_u|}}$ for testing, their second to last session $S^u_{t_{|S_u|-1}}$ for validation, and the rest for training. Note that the test set will have validation in its historical data as well. We predict items in the user's test basket sequentially (basket size > 1). For consistency, we adopt a similar strategy to [15, 21, 36, 37], where they use a set of randomly sampled negative items plus ground truth item for evaluation purposes. We use a set of 100 randomly sampled items (negative and positive) plus ground truth items and rank these items to get the highest recommendations for the next item. Negative sampling is suitable in applications where users are less likely to interact with an item more than once (e.g., watching movies). However, in applications where users frequently interact with an item (e.g., grocery shopping, listening to music), evaluating the model on a pool of negative items (items that the user never interacts with) would make the prediction task more trivial. We observe that in the real-world dataset, about 81% of test basket items also appear in the user's purchase history. We include positive items in the sampling to resolve this issue.

[1] https://www.chicagobooth.edu/research/kilts/datasets/nielseniq-nielsen

[2] Codes for PARSRec: https://github.com/ehgh/PARSRec

For evaluation metrics, we report common top-N metrics [21, 36]: Hit Ratio (HR@k) and Normalized Discounted Cumulative Gain (NDCG@k) with $k \in \{1, 5, 10\}$. NDCG is a rank-aware metric that penalizes the lower-ranked recommendations. Note that NDCG@1 equals HR@1. Additionally, we report another metric for session-level evaluation: session precision (**Sess-Prec@k**), the average ratio of items predicted accurately within a session:

$$session\ precision = \frac{|\{\hat{v}_j | \hat{v}_j \in S_t^u\}|}{|S_t^u|} \tag{4}$$

In other words, session precision is the number of items that are recommended accurately normalized by the basket size.

5.4 Benchmarks

To validate the effectiveness of our proposed model, we benchmark it against the following baselines:

- **POPRec**: A basic benchmark that recommends items based on their frequency of interactions by users.
- **SASRec** [21]: It uses one-directional transformers for sequential recommendation. We modified negative sampling to sampling explained in Section 5.3 for a fair comparison.
- **BERT4Rec** [36]: It applies bi-directional transformers with Cloze objective to sequential recommendation, and achieves state-of-the-art performance on many datasets. We applied a similar sampling technique in Section 5.3 for a fair comparison.

We omit comparison with some other benchmarks like GRU4Rec [18], GRU4Rec$^+$ [17], BPR [33], NCF[15], and FPMC [34] that have been outperformed by above methods on various real-world datasets [21, 36]. We use the code provided by the corresponding authors for benchmark models. We consider latent dimension sizes $d_v \in \{16, 32, 64, 128, 256\}$ for methods that include embeddings. The dropout rate is chosen from $\{0, 0.1, ..., 0.5\}$. We set $N = 820$ (full sequence length) for synthetic dataset and $N = 1000$ (~median of sequence length) for Nielsen dataset in SASRec and BERT4Rec that require setting sequence length. We either tuned all hyper-parameters on the validation set or referred to benchmarks' authors suggestions for optimal values. The reported results are under their optimal set of hyper-parameter values.

Performance Comparison: Table 2 summarizes the recommendation performance of all benchmarks as well as our proposed model on synthetic and Nielsen datasets. The last column shows the statistically significant improvements of the best model to the second-best model. We observe that POPRec has the lowest performance, presumably because it does not look into user-specific behaviors. SASRec uses transformers and performs lower than BERT4Rec because it only looks at item relationships from left to right. BERT4Rec also uses transformers but in a bidirectional setting that allows learning from both right and left. PARSRec performs consistently better than all methods on both datasets. It is likely because PARSRec differentiates between interactions within a session and interactions in the past sessions by using RNN blocks. It also uses explicit user queries in its attention blocks to learn personalized item-item relationships which is explained more in detail in Section 5.5. PARSRec gains an average of 19.1% HR@5, 20.9% NDCG@5, and 21.1% Sess-Prec@5 improvement over the second-best benchmark.

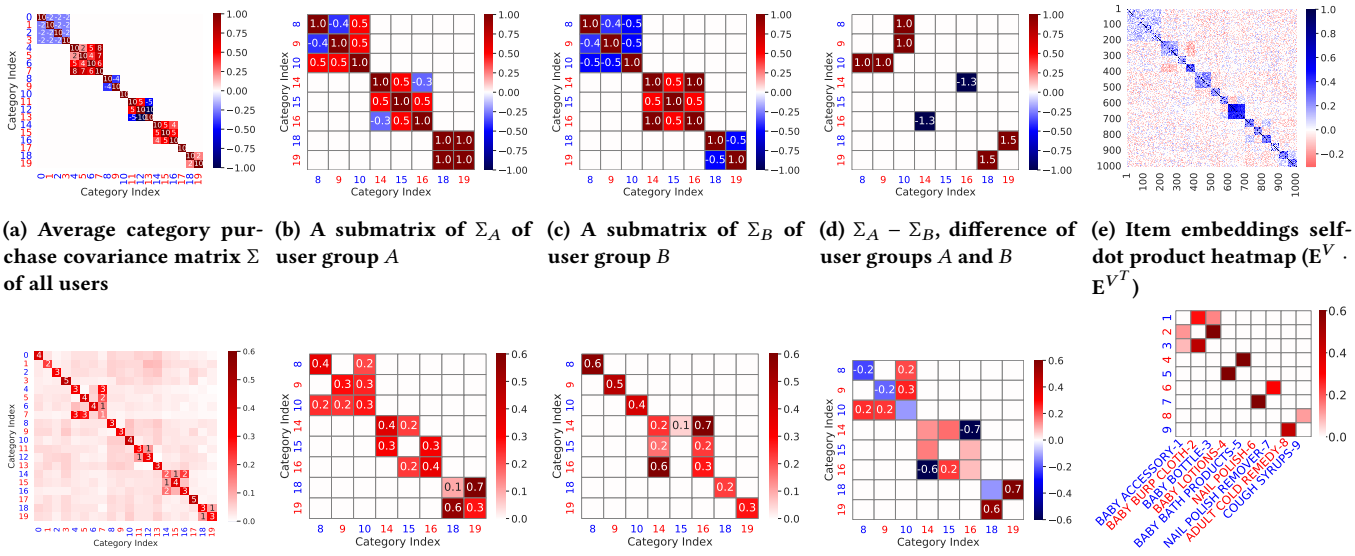
Table 2: Performance comparison of all benchmarks. Boldface and underlined values in each row represent the best and second best performances, respectively. Improvements of best to second best model are presented in the last column.

Dataset	Metric	POPRec	SASRec	BERT4Rec	PARSRec	Improvement
Synthetic	HR@1	0.0005	0.1816	<u>0.1963</u>	**0.2136**	8.8%
	HR@5	0.0028	0.3919	<u>0.4150</u>	**0.4791**	15.4%
	HR@10	0.0066	0.5133	<u>0.5441</u>	**0.5825**	7.0%
	NDCG@5	0.0013	0.2872	<u>0.3061</u>	**0.3489**	13.9%
	NDCG@10	0.0026	0.3253	<u>0.3426</u>	**0.3800**	10.9%
	Sess-Prec@1	0.0005	0.1826	<u>0.1974</u>	**0.2249**	13.9%
	Sess-Prec@5	0.0029	0.3931	<u>0.4163</u>	**0.4784**	14.9%
	Sess-Prec@10	0.0068	0.5151	<u>0.5357</u>	**0.5792**	8.1%
Nielsen	HR@1	0.0008	0.1663	<u>0.1761</u>	**0.2444**	38.8%
	HR@5	0.0921	0.4235	<u>0.4833</u>	**0.5934**	22.7%
	HR@10	0.1620	0.5771	<u>0.6580</u>	**0.7355**	11.7%
	NDCG@5	0.0454	0.2991	<u>0.3290</u>	**0.4208**	27.9%
	NDCG@10	0.0680	0.3497	<u>0.3987</u>	**0.4632**	16.1%
	Sess-Prec@1	0.0008	0.1669	<u>0.1856</u>	**0.2753**	48.3%
	Sess-Prec@5	0.0919	0.4197	<u>0.4790</u>	**0.6093**	27.2%
	Sess-Prec@10	0.1632	0.5729	<u>0.6344</u>	**0.7439**	17.2%

5.5 Discussion

The key deliverable of our study was to develop a personalized recommendation system that accounted for patterns of individual preferences over time and item relationships. The model developed here can be applied to multiple scenarios, from populating song lists to basket completion exercises. The model's success hinges on its ability to address two key questions: (1) how does PARSRec capture personalized item-item relationships? and (2) how does learning personalized item-item relationships affect the recommendation?

5.5.1 Extracting Personalized Item Relationships. Recommender systems follow one of the two common methods to capture heterogeneous user behaviors: *i)* learn an explicit user representation based on the user behavior [34, 37]; *ii)* implicitly represent users by aggregating the embeddings of user interacted items [14, 17, 18, 21, 36]. PARSRec takes the former approach by learning an explicit user embedding. Recently, the attention layers of transformers in the state-of-the-art recommender systems provide an explainable visual for item-item relationships. These models (e.g., BERT4Rec [36], and SASRec [21]) often use a self-attention layer where all key, value, and query parameters consist of item embeddings (implicit user representation). Hence, they learn global item-item relationships that differentiate users via their purchase history. However, in PARSRec the query consists of explicit user embeddings (plus user purchase history, if available). This allows for the network to explicitly learn and visualize the personalized item-item relationships. To validate this capability, we conduct a controlled simulation on the synthetic dataset. We split users into various groups and each group has a different category covariance matrix, Σ, during data synthesis. Figures 3a to 3d illustrate the heatmaps (of a subset) of Σ for two different user groups A, and B (along with their average and difference). We then extract the highest attention weights of each user at every step of the training phase (averaged on attention heads). Note that we only extract weights of



(a) Average category purchase covariance matrix Σ of all users

(b) A submatrix of Σ_A of user group A

(c) A submatrix of Σ_B of user group B

(d) $\Sigma_A - \Sigma_B$, difference of user groups A and B

(e) Item embeddings self-dot product heatmap ($\mathbf{E}^V \cdot \mathbf{E}^{V^T}$)

(f) Average category attention weights learned by PARSRec

(g) Attention weights of corresponding categories in (3b) of user group A

(h) Attention weights of corresponding categories in (3c) of user group B

(i) Category attention weights difference of user groups A and B

(j) Empirical data: Average attention weights of items for users with babies

Figure 3: Simulation and empirical heatmaps: (a) Average covariance matrix Σ of category purchases $p^u_{ct} \sim \mathcal{N}(\alpha_c, \Sigma)$ in data synthesis. A positive value in Σ indicates two categories are purchased together frequently (e.g. milk and cereal), while a negative value represents contrary of that. (b, c) A subset of covariance matrix for user groups A and B, (d) difference of (b) and (c), (e) Item embedding self-dot-product shows embedding has learned global item categories and relationships. (f) Average category attention weights learned by the model. Higher values correspond to positive values in (a) while lower values correspond to zero/negative values in (a), (g, h) Category attention weights for selected categories in (b, c), the matching of (b<->g) and (c<->h) illustrates that model can learn personalized attention. Values < 0.05 are filtered for clarity, (i) difference between (g) and (h). The high values matches Fig. (d). (j) Attention weights of sample products in empirical data for a group of users with babies. The network learns correct relations without prior knowledge of users or products (Color represents sign and intensity represents value. Annotations in (a) and (f) are scaled by 10 for readability. All other annotations illustrate the true values of the cell. Figure best viewed in color. Axis label colors are for readability)

the valid non-padding items. We aggregate the item-item attention weights to category level for ease of comparison and readability. We perform row-wise sum followed by column-wise average within each category, then row-wise normalization over all items for each user individually. Figures 3f to 3i show the heatmaps of the category level attention weights for two different user groups A and B, and their average and difference. Further, Figure 3e represents the heatmap of dot-product of item embedding with itself ($\mathbf{E}^V \cdot \mathbf{E}^{V^T}$). We observe following traits from the figures:

- (3a) vs. (3f): We expect the attention weights in (3f) to capture the average category relations for all users corresponding to Figure (3a). We observe that there is a correlation between attention weights and covariance matrices. Higher attention weights correspond to larger positive covariance values, and near-zero attention weights correspond to the negative covariance values. We also observe that attention weights capture category independence in the form of block diagonal matrices. Note that the network has no prior knowledge of item categories, and user-item interactions are the only information introduced to the network.
- (3b, 3c, 3d) vs. (3g, 3h, 3i): We observe that PARSRec can learn user heterogeneity by accurately extracting different user groups' attention weights. We see a one-to-one match between user groups' covariance matrices in (3b, 3c) and their corresponding attention weights in (3g, 3h). For easier comparison, we include their differences in (3i). The cells with higher absolute values in Figure (3i) match the differences in Figure (3d). Note that the network has no prior knowledge of user groups and infers their difference via user behavior.
- (3a) vs. (3e): The global item-item similarity (i.e. dot-product) matches the average relationships of items. We observe the similarity of items within each category illustrated by blue colors. This is because in data synthesis, a user interacts with at most one item per category during a session, and items within a category are expected to have similar embedding. (3e) also shows a matching pattern with global category correlations where red cells match high positive category covariance values and blue cells match low positive or negative covariace values. Note that unlike attention weights, item embeddings are learned relative to other item embeddings. E.g., categories 14 and 16 show similarities in Figure (3e) (blue color) even though they have a positive

(a) Sales estimation of different categories for user groups A and B

(b) Sales estimation MAPE

Figure 4: Estimating the influence of removing category C_0 (present at training phase) from the assortment at testing phase on sales of categories C_1 and C_2 for two user groups. Cov(C_0,C_2)=0.6 for user group A. Other covariances are zero. PARSRec captures category spillover effects.

but relatively lower correlation compared to category pairs 14-15 and 15-16 (red color). However, attention weights of categories 14 and 16 are captured positively in Figure (3f). (3e) shows that the model learns the item similarities globally to some extent. However, item representations alone cannot present personalized item relationships like the attention layer does.

Empirical Attention Between Items: The empirical dataset lacks the ground truth for user behavior. However, we can look into item relationships learned by the network. Figure (3j) shows the average of a subset of attention weights between items for a group of users who have babies in the Nielsen dataset. We observe that the network learns to separate different product groups related to each other without prior knowledge of users or item categories. The network also identifies subgroups of related items within a category (e.g., {baby bath products, baby lotion} vs. {baby bottle, baby burp cloth, baby accessory}) even though they occur in the same sessions.

5.5.2 Effect of Personalized Recommendation. To investigate the importance of learning personalized item-item relationships in making recommendations, we conduct a controlled simulation on the synthetic dataset. We drop a product category from the pool of products and observe its consequences on other categories' sales. For example, it is observed when a retailer discontinues tobacco and cigarettes, they face a cross-category spillover effect on other products such as alcoholic beverages [9]. We mimic the real-world behavior in our simulation as follows: 1) we assume the assortment contains all products during training and validation phase, 2) we remove baskets that contain the dropped category C_0 from the *test set* and also from the pool of products *during testing phase*, and 3) we estimate the unit sales of two types of categories for two user groups during testing phase using top 10 recommendations. Category C_1 is independent of C_0 for both user groups. Category C_2 is highly correlated with C_0 for user group A (cov=0.6), but independent of C_0 for user group B (cov=0). Figure 4a shows the estimates of category sales for each user group, and Figure 4b shows Mean Absolute Percentage Error (MAPE) of sales. We observe that PARSRec accurately distinguishes the spill over effect of C_0 removal on categories C_1 and C_2. PARSRec estimates the cross-category drop in sales of category C_2 for user group A (MAPE=3.5%) and uninfluenced sales of C_2 for user group B (MAPE=1%). It also accurately estimates the sales of the independent category C_1 for both user groups (MAPE=3.8% and 3.7%). It is important to note that all training/validation sets include category C_0 and only the test set is subject to change. Without any prior training on category removal effects, the proposed model can predict the user behavior by learning personalized item-item relationships. This allows retailers to conduct simulated experiments on assortment modifications and estimate their effect on different customers without the costs of applied experiments.

6 CONCLUSION AND FUTURE WORK

We proposed PARSRec, a sequential recommender model that combines attention mechanism and RNN. PARSRec learns personalized item relationships by using explicit user embeddings in the query of attention mechanism. We conduct a controlled simulation on a synthetic dataset to validate user behavior learning. Empirical results on Nielsen's Consumer Panel dataset show that PARSRec outperforms state-of-the-art self-attention models. One future direction is incorporating the attention mechanism with GRU/LSTM networks for longer sessions and exploring the impact of a different loss function such as Bayesian personalized ranking. Another direction would be to leverage item features (e.g., textual information, price, flavor) in the attention mechanism.

ACKNOWLEDGMENTS

"Researcher(s)' own analyses calculated (or derived) based in part on data from Nielsen Consumer LLC and marketing databases provided through the NielsenIQ Datasets at the Kilts Center for Marketing Data Center at The University of Chicago Booth School of Business." "The conclusions drawn from the NielsenIQ data are those of the researcher(s) and do not reflect the views of NielsenIQ. NielsenIQ is not responsible for, had no role in, and was not involved in analyzing and preparing the results reported herein." The authors thank Erin Nannery, Ashutosh Nayak, Jörn Boehnke, and Kourosh Vali for their help and assistance.

REFERENCES

[1] Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. 2019. Cloze-driven pretraining of self-attention networks. *arXiv preprint arXiv:1903.07785* (2019).
[2] Pedro Bordalo, Nicola Gennaioli, and Andrei Shleifer. 2013. Salience and consumer choice. *Journal of Political Economy* 121, 5 (2013), 803–843.
[3] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. 2019. Behavior sequence transformer for e-commerce recommendation in alibaba. In *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*. 1–4.
[4] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 108–116.
[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[6] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. 2017. Sequential user-based recurrent neural network recommendations. In *Proceedings of the eleventh ACM conference on recommender systems*. 152–160.

[7] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. 2019. Deep session interest network for click-through rate prediction. *arXiv preprint arXiv:1905.06482* (2019).
[8] Sebastian Gabel, Daniel Guhl, and Daniel Klapper. 2019. P2V-MAP: Mapping market structures for large retail assortments. *Journal of Marketing Research* 56, 4 (2019), 557–580.
[9] Ali Goli and Pradeep K Chintagunta. 2021. What happens when a retailer drops a product category? investigating the consequences of ending tobacco sales. *Marketing Science* 40, 6 (2021), 1169–1198.
[10] William H Greene. 2003. *Econometric analysis*. Pearson Education India.
[11] Lei Guo, Yu Han, Haoran Jiang, Xinxin Yang, Xinhua Wang, and Xiyu Liu. 2020. Learning to make document context-aware recommendation with joint convolutional matrix factorization. *Complexity* 2020 (2020).
[12] Casper Hansen, Christian Hansen, Lucas Maystre, Rishabh Mehrotra, Brian Brost, Federico Tomasi, and Mounia Lalmas. 2020. Contextual and sequential user embeddings for large-scale music recommendation. In *Fourteenth ACM Conference on Recommender Systems*. 53–62.
[13] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based recommendation. In *Proceedings of the eleventh ACM conference on recommender systems*. 161–169.
[14] Ruining He and Julian McAuley. 2016. Fusing similarity models with markov chains for sparse sequential recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 191–200.
[15] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.
[16] Balázs Hidasi. 2019. Cutting-Edge Collaborative Recommendation Algorithms: Deep Learning. In *COLLABORATIVE RECOMMENDATIONS: Algorithms, Practical Challenges and Applications*. World Scientific, 79–126.
[17] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 843–852.
[18] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).
[19] https://www.chicagobooth.edu/research/kilts/datasets/nielseniq nielsen. 2020. Nielsen Consumer Panel Dataset. Retrieved Jan 15, 2021 from https://www.chicagobooth.edu/research/kilts/datasets/nielseniq-nielsen
[20] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y Chang. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 505–514.
[21] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.
[22] Yehuda Koren and Robert Bell. 2015. Advances in collaborative filtering. *Recommender systems handbook* (2015), 77–118.
[23] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.
[24] Daniel Lewandowski, Dorota Kurowicka, and Harry Joe. 2009. Generating random correlation matrices based on vines and extended onion method. *Journal of multivariate analysis* 100, 9 (2009), 1989–2001.
[25] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1419–1428.
[26] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time interval aware self-attention for sequential recommendation. In *Proceedings of the 13th international conference on web search and data mining*. 322–330.
[27] Zhi Li, Hongke Zhao, Qi Liu, Zhenya Huang, Tao Mei, and Enhong Chen. 2018. Learning from history and present: Next-item recommendation via discriminatively exploiting user behaviors. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1734–1743.
[28] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: short-term attention/memory priority model for session-based recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1831–1839.
[29] Puneet Manchanda, Asim Ansari, and Sunil Gupta. 1999. The "shopping basket": A model for multicategory purchase incidence decisions. *Marketing science* 18, 2 (1999), 95–114.
[30] Andriy Mnih and Russ R Salakhutdinov. 2007. Probabilistic matrix factorization. *Advances in neural information processing systems* 20 (2007).
[31] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G Azzolini, et al. 2019. Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091* (2019).
[32] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *proceedings of the Eleventh ACM Conference on Recommender Systems*. 130–137.
[33] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).
[34] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. 811–820.
[35] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. Introduction to recommender systems handbook. In *Recommender systems handbook*. Springer, 1–35.
[36] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 1441–1450.
[37] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 565–573.
[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.
[39] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2015. Learning hierarchical representation model for nextbasket recommendation. In *Proceedings of the 38th International ACM SIGIR conference on Research and Development in Information Retrieval*. 403–412.
[40] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 950–958.
[41] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A dynamic recurrent model for next basket recommendation. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 729–732.
[42] Wenhui Yu, Xiangnan He, Jian Pei, Xu Chen, Li Xiong, Jinfei Liu, and Zheng Qin. 2021. Visually aware recommendation with aesthetic features. *The VLDB Journal* 30, 4 (2021), 495–513.
[43] Weihua Yuan, Hong Wang, Xiaomei Yu, Nan Liu, and Zhenghao Li. 2020. Attention-based context-aware sequential recommendation model. *Information Sciences* 510 (2020), 122–134.

Table 3: Ablation analysis (NDCG@10 and Sess-Prec@10) of synthetic and empirical dataset. An increase in performance from the default PARSRec setting is emphasized in bold.

Architecture	Synthetic		Nielsen	
	Sess-Prec@10	NDCG@10	Sess-Prec@10	NDCG@10
PARSRec default	0.5792	0.3800	0.7439	0.4632
Remove LN	0.5756	0.3746	0.6775	0.4183
Remove Dropout	0.5780	0.3775	0.7356	0.4553
Remove Q in LN	0.5643	0.3745	0.7400	0.4601
#Att Layers=2	0.5506	0.3622	0.7320	0.4543
#heads=1	0.5769	0.3781	0.7399	0.4602
#heads=4	0.5790	0.3799	0.7431	0.4628
FFN pre-RNN	0.5723	0.3775	0.7426	**0.4637**
FFN post-RNN	0.5009	0.3362	0.6900	0.4189
Extra Features	-	-	**0.7591**	**0.4759**

A DATA SYNTHESIS

We use multinomial probit model [10] to simulate product choice within each category, $j_t^{u,c}$:

$$j_t^{u,c} = \underset{j \in c}{\operatorname{argmax}}\, \eta_{jt}^{u,c}$$
$$\eta_{jt}^{u,c} = \omega_j^{u,c} - \beta^c v_j^c + \gamma_{jt}^{u,c} \tag{5}$$

where $\eta_{jt}^{u,c}$ is utility of item j in category c for user u at time t, $\omega_j^{u,c} \sim \mathcal{N}(0, \Omega^c)$ is the product base utility, $\beta^c v_j^c$ is a disutility for paying price v_j^c, and $\gamma_{jt}^{u,c} \sim \mathcal{N}(0, \sigma^c)$ is a random term to capture any uncontrollable parameter affecting the product choice. The simulation first chooses categories based on Eq. (3) and the user chooses the product with highest utility in each category based on Eq. (5). We use $C = 20$ categories, each with $|V^c| = 100$ products. Basket size n_t^u is sampled from Weibull(0.80, 1.47) to represent the basket size distribution in our real-world data. Single item baskets and large (>10) baskets (tail of the distribution) are filtered. We set $\sigma^c = 1, \beta^c = 0.1$, and $\alpha_c = -0.5$. Category specific covariance matrix $\Omega^c = \tau_c^2 \Omega_0^c$ where $\tau_c = 2$ is the standard deviation and Ω_0^c is the correlation matrix. Ω_0^c is a positive-semidefinite matrix generated using vine method [24] under Beta(0.2, 1) distribution to simulate various degrees of product competition. Product prices $v_j^c \sim Uniform(\frac{v^c}{2}, 2v^c)$ where category base price $v^c \sim LogNormal(0.5, 0.1)$ is set to mimic the real-world data prices.

B MATHEMATICAL MODEL OF PARSREC ARCHITECTURE

Embedding Layer

We denote user and item embedding matrices with $\mathbf{E}^U \in \mathbb{R}^{|U| \times d_u}$ and $\mathbf{E}^V \in \mathbb{R}^{|V| \times d_v}$, respectively.

- We convert item indices of the basket $S_{t_i}^u = [v_1^{(S)}, ..., v_{|n_i^u|}^{(S)}]$ to input embedding vectors $\mathbf{K}_i^u \in \mathbb{R}^{n_i^u \times d_v}$ where the j-th row of $\mathbf{K}_i^u$ is $\mathbf{E}^V_{v_j^{(S)}}$. We use subsets of $\mathbf{K}_i^u$ as keys and values to attention layer. The details are explained in section 4.2.
- We convert item indices of the previous sessions, $(S_{t_1}^u, ..., S_{t_{i-1}}^u)$ to their embedding representations by $\mathbf{E}^V$ and reduce them using the weighted sum operator to a single representation vector, $\mathbf{H}_{i-1}^u \in \mathbb{R}^{1 \times d_v}$:

$$\mathbf{H}_{i-1}^u = \sum_{v \in \{w \in s | s \in \{S_{t_1}^u, ..., S_{t_{i-1}}^u\}\}} \mathbf{E}_v^V \tag{6}$$

We use $\mathbf{H}_{i-1}^u$ in the initial hidden state h_0 to the **ARNN** block.

- In cases where we have extra features for users, we can merge user features $\mathbf{F}_u^U$ with the user embedding $\mathbf{E}_u^U$ to create a combined user input to our model:

$$\hat{\mathbf{E}}_u^U = combine(\mathbf{E}_u^U, \mathbf{F}_u^U) \tag{7}$$

where $\hat{\mathbf{E}}_u^U, \mathbf{E}_u^U, \mathbf{F}_u^U$ are combined input, embedding and features of user u, respectively, and $combine(.)$ is any function that merges two vectors into one. Examples are (weighted) addition, concatenation, and element-wise multiplication. In our ablation study, we investigated the choice of concatenation.

Attention Layer

The multi-head attention with h heads can be summarized as:

$$MultiHead(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = Concat(head_1, ..., head_h)\mathbf{W}^O$$
$$head_i = Attention(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V)$$
$$= softmax\Big(\frac{(\mathbf{Q}\mathbf{W}_i^Q)(\mathbf{K}\mathbf{W}_i^K)^T}{\sqrt{d_k}}\Big)(\mathbf{V}\mathbf{W}_i^V)$$

where the projection matrices $\mathbf{W}^O \in \mathbb{R}^{hd_v \times d_v}$, $\mathbf{W}_i^Q \in \mathbb{R}^{d_q \times d_q}$, $\mathbf{W}_i^K \in \mathbb{R}^{d_k \times d_q}$, and $\mathbf{W}_i^V \in \mathbb{R}^{d_v \times d_q}$ are learned parameters, query $\mathbf{Q}$, key $\mathbf{K}$, and value $\mathbf{V}$ are matrices, d_k is the dimension of the key and value, and d_q is the dimension of the query. Key, value, and query are:

$$\mathbf{Q} = Concat(\mathbf{E}_u^U, h_j)$$
$$m\text{-th row}(\mathbf{K}) = m\text{-th row}(\mathbf{V}) = \mathbf{E}^V_{v_m^{(S)}}$$

Recurrent Layer

The RNN and prediction layers combined are:

$$h_{j+1} = ReLU(\tilde{v}_j \mathbf{W}^{(1)} + \tilde{h}_j \mathbf{W}^{(2)} + \mathbf{b}^{(1)})$$
$$\hat{v}_{j+1} = \underset{}{\operatorname{argmax}}\,((\tilde{v}_j \mathbf{W}^{(3)} + \tilde{h}_j \mathbf{W}^{(4)} + \mathbf{b}^{(2)}).\mathbf{E}^{V^T})$$

where $\mathbf{W}^{(1)} \in \mathbb{R}^{d_v \times d_v}$, $\mathbf{W}^{(2)} \in \mathbb{R}^{(d_u+d_v) \times d_v}$, $\mathbf{W}^{(3)} \in \mathbb{R}^{d_v \times |V|}$, $\mathbf{W}^{(4)} \in \mathbb{R}^{(d_u+d_v) \times |V|}$ are matrices, $\mathbf{b}^{(1)}$ is d_v, and $\mathbf{b}^{(2)}$ is $|V|$ dimensional vectors.

C ABLATION STUDY

We conduct an ablation study to understand the impact of various components of PARSRec and some variations in the design of the network. Table 3 summarizes the performance (NDCG@10 and Sess-Prec@10) of the optimal model and eight variants on both synthetic and empirical datasets. All other hyper-parameters are unchanged. Because the HR metric has a similar pattern to the NDCG metric, for parsimony and readability, we do not include the metric in Table 3. Results are available from the authors upon request. The variants are as follows:

(1,2) *Remove LN, Dropout*: Including both components help improve the performance, with LN being more prominent on empirical dataset.

(3) *Remove **Q** from LN*: Adding **Q** at LN increases the accuracy. It carries the user history and is beneficial to add it to RNN input.

(4) *Number of Attention Layers*: Stacking more attention layers achieves similar performance on the empirical dataset but performs lower on the synthetic dataset. Presumably, because the sessions are short and single layer attention is sufficient to learn the relationship of items within a session. Longer sequences might pose more complex patterns and require more layers.

(5) *Number of heads in Multi-Head Attention*: The results show that increasing the number of heads beyond two does not significantly boost performance, presumably because of small session lengths. Longer sessions might benefit from more number of heads.

(6) *FFN pre-RNN input*: We explore adding layers of Feed-Forward Network right before the input of RNN. The results show that the added FFN has a similar performance. It is likely that the projection layer of the attention mechanism extracts the required features, and adding extra layers is redundant.

(7) *FFN post-RNN output*: Adding FFN after RNN output significantly decreases the performance, possibly due to overfitting. The optimal dimension of RNN is satisfactory for output prediction.

(8) *Extra User Features*: We explore adding extra user and session features from the empirical dataset. We pass continuous features through a Multi-Layer Perceptron and further concatenate them with the user and categorical features' embeddings to use as the initial hidden state. Features include income, age, gender, location, retailer id, day of the week of session, household size and composition, user education, marital status, and race. These features contribute slightly to improving performance (~2% increase). This confirms that the personalized model learns user features that contribute to the recommendation task via user embedding. However, in cases like the *cold start* problem where we do not have sufficient user purchase history, the impact of explicit user characteristics might become more prominent.