HAM: Hybrid Associations Model with Pooling for Sequential Recommendation

Bo Peng[1] , **Zhiyun Ren**[2] , **Srinivasan Parthasarathy**[1,2] and **Xia Ning**[1,2*]

[1]Department of Computer Science and Engineering, The Ohio State University, Columbus, USA
[2]Department of Biomedical Informatics, The Ohio State University, Columbus, USA
peng.707@buckeyemail.osu.edu, {ren.685,ning.104}@osu.edu, srini@cse.ohio-state.edu

Abstract

We developed a hybrid associations model (HAM) to generate sequential recommendations using two factors: 1) users' long-term preferences and 2) sequential, both high-order and low-order association patterns in the users' most recent purchases/ratings. HAM uses simplistic pooling to represent a set of items in the associations. We compare HAM with three the most recent, state-of-the-art methods on six public benchmark datasets in three different experimental settings. Our experimental results demonstrate that HAM significantly outperforms the state of the art in all the experimental settings, with an improvement as high as 27.90%.

1 Introduction

Sequential recommendation aims to identify and recommend the next few items for a user that the user is most likely to purchase/review, given the user's purchase/rating history. It becomes an effective tool to chronologically help users select favorite items from a variety of options. A key challenge in sequential recommendation is to identify, learn or represent the patterns and dynamics in users' purchase/rating sequences that are most pertinent to inform their future interactions with other items, and also to capture the relations between such patterns and future interactions. With the prosperity of deep learning, many deep models, particularly based on recurrent neural networks [Hidasi *et al.*, 2015; Hidasi and Karatzoglou, 2018] and with attention mechanisms, have been developed for sequential recommendation purposes. These methods typically model users' sequential behaviors and their long-term/short-term preferences, and have significantly improved recommendation performance.

However, given the notoriously sparse nature of most recommendation data without any side information on the items or users, a question on these deep learning methods, particularly those with attention mechanisms, is that whether the sparse recommendation data is sufficient to enable well-learned attention weights that play effective roles in identifying important information leading to accurate recommendations. Recent studies [Dacrema *et al.*, 2019; Ludewig *et al.*, 2019] bring such concerns on recommendation algorithms in general, demonstrating that complicated deep recommendation methods may not always outperform simple ones.

We propose a hybrid associations model (HAM) with a simplistic pooling mechanism to better model users historical purchase/rating sequences. HAM generates recommendations for the next items using two factors: 1) users' general/long-term preferences and 2) sequential, association patterns in the users' most recent purchases/ratings. The users' general preferences are learned by leveraging their all historical purchases/ratings and represented in user embeddings. The item associated patterns used in HAM include both high-order associations (i.e., more items induce the next, a few items) and low-order associations (i.e., fewer items induce the next, a few items). We use simplistic pooling to represent a set of items in the associations, and recommend the next items based on their recommendation scores aggregated from users' general preferences and item association patterns. We compare the HAM models with 3 the most recent, state-of-the-art methods on 6 public benchmark datasets in 3 different experimental settings. Our experimental results show that HAM models significantly outperform the state-of-the-art methods in all the experimental settings, with a best improvement 27.90%. [1]

The major contributions in this paper are as follows:

- To the best of our knowledge, HAM is the first method that explicitly models both high-order and low-order sequential associations among items for sequential recommendation. HAM significantly outperforms the state-of-the-art methods.
- HAM uses simplistic pooling instead of learned attentions to represent a set of items.
- We investigated the attention weights learned in benchmark datasets and studied the potential scenarios in which pooling could outperform attention mechanisms in sequential recommendation.
- We studied various experimental settings in which sequential recommendation performance is evaluated, and discussed the potential issues in the most widely used experimental setting in literature.

2 Literature Review

Numerous sequential recommendation methods have been developed, particularly using Markov Chains (MCs), Recur-

*Contact Author

[1]we will publish the source code once this paper is accepted.

Table 1: Notations

notations	meanings
m/n	number of users /tems
S_i	purchase/rating sequence of user i
$S_i(t,l)$	a length-l subsequence of S_i starting from the t-th purchase/rating
U	user embedding matrix
V	item embedding matrix
W	candidate item embedding matrix
$\hat{r}_{ij}$	the recommendation score of user i on item j
n_h/n_l	the number of items of high-order/ low-order sequential association
n_p	the number of next items to be recommended during training

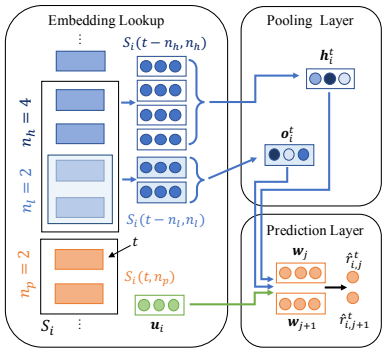


Figure 1: HAM Model Architecture

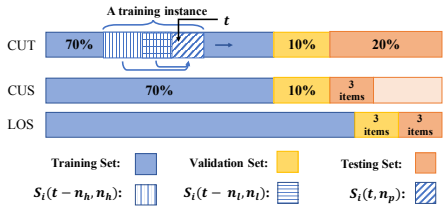


Figure 2: Experimental Settings.

rent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), attention and gating mechanisms, etc. Specifically, MCs-based methods, such as factorized personalized Markov chains (FPMC) [Rendle *et al.*, 2010], use MCs to capture pairwise item-item transition relations to recommend the next item for each user. Based on FPMC, He *et al.* [He and McAuley, 2016a] developed a factorized sequential prediction model (fossil), which uses high-order MCs to capture the impact from all historical purchases/ratings on the next item. Recently, many RNN-based methods have been developed to model the sequential patterns in users' purchase/rating sequences. For instance, Hidasi *et al.* [Hidasi *et al.*, 2015; Hidasi and Karatzoglou, 2018] used gated recurrent units (GRUs) to capture the users' short-term preferences. Kang *et al.* [Kang and McAuley, 2018] developed a self-attention based sequential model (SASRec) to capture a few, most informative items in users' purchase/rating sequences to generate recommendations. Recent work also adapts CNNs for sequential recommendation. For example, Tang *et al.* [Tang and Wang, 2018] developed a convolutional sequence embedding recommendation model (Caser), which uses multiple convolutional filters on the most recent purchases/ratings to extract sequential features from a set of recent items. Ma *et al.* [Ma *et al.*, 2019] developed a hierarchical gating network (HGN), which captures item-item transition relations and user long-term preferences, and uses gating mechanisms to identify important items and their latent features from users' historical purchases/ratings. HGN has been demonstrated as the state of the art, and outperforms an extensive set of existing methods including Caser and SASRec.

3 Methods

Figure 1 presents the HAM architecture and Table 1 presents the key notations. In this paper, the historical purchases or ratings of user i in chronological order are represented as a sequence S_i=$\{s_i(1), s_i(2), \cdots\}$, where $s_i(t)$ is the t-th purchased/rated item. A length-l subsequence of S_i starting at the t-th purchase/rating is denoted as $S_i(t,l)$, that is, $S_i(t,l) = \{s_i(t), s_i(t+1), \cdots, s_i(t+l-1)\}$. When no ambiguity arises, we will eliminate i in $S_i/S_i(t,l)$.

HAM generates recommendations for the next items for each user using two factors: 1) the user's general/long-term preferences, and 2) the sequential, association patterns in the user's most recent purchases/ratings. These two factors will be used to calculate a recommendation score for each item candidate in order to prioritize and recommend the next items.

3.1 Modeling Users' General Preferences

It has been shown that users' general preferences play an important role in their purchases/ratings [He and McAuley, 2016a]. Therefore, in HAM, we learn users' general preferences using an embedding matrix $U \in \mathbb{R}^{m \times d}$. U serves as a lookup table, in which the i-th row, denoted as $\mathbf{u}_i$, represents the general preferences of user i.

3.2 Modeling Sequential Hybrid Associations

Previous study [Tang and Wang, 2018] has shown the existence of sequential associations in recommendation data. Here, we denote the sequential association at time t from its previous n_h purchases/ratings to the next n_p subsequent purchases/ratings as $S(t-n_h, n_h) \rightarrow S(t, n_p)$, and the number of the involved n_h+n_p items as the order of the association. It has also been shown [Tang and Wang, 2018] that the sequential associations among items in benchmark recommendation datasets (used in experiments as in Section 4.2) have different orders. For example, about 50% significant associations have n_h=2 and $n_p \leq 2$ (i.e., previous 2 purchases/ratings have immediate effects on the next 1 or 2 purchases/ratings), and about 15% significant associations have n_h=4 and $n_p \leq 2$.

We propose to explicitly model the item associations of different orders in HAM such that the aggregated information from the previous different numbers of purchases/ratings will contribute to the recommendation scores of all the subsequent item candidates. Particularly, we include a high-order association $S(t-n_h, n_h) \rightarrow S(t, n_p)$ and a low-order association $S(t-n_l, n_l) \rightarrow S(t, n_p)$ ($n_l < n_h$, $S(t-n_l, n_l) \subset S(t-n_h, n_h)$) to recommend the next n_p items. During training, we split the historical purchase/rating sequence of each user into multiple subsequences (training instances) of length n_h+n_p (these subsequences will overlap), and learn the individual item embeddings, and thus the embeddings of n_h/n_l successive items, from all the length-(n_h+n_p) subsequences. Note that HAM can be a general framework, in which arbitrary numbers of various-order associations can be incorporated.

Mean and Max Pooling from Previous Items In order to represent the information from the previous n_h/n_l purchased/rated items as a whole, we use pooling mechanisms, and learn one embedding **h/o** for the n_h/n_l successive purchases/ratings using 1) mean pooling and 2) max pooling, respectively, from individual item embeddings of the n_h/n_l

purchased/rated items. Given the sparse nature of the recommendation data, it might be difficult to learn and differentiate the contributions of different items. The mean pooling is a simplistic solution to average the effects from each individual item. HAM with mean pooling is denoted as HAM_m. The hypothesis of max pooling is that the purchased/rated items contribute in various dimensions in determining the next n_p purchases/ratings. HAM with max pooling is denoted as HAM_x.

3.3 Heterogeneous Item Embeddings

It has also been shown [Rendle *et al.*, 2010; Kang and McAuley, 2018] that the item transitions may be asymmetric: item j might be frequently purchased/rated after item k, but not vise versa. To model such asymmetry in HAM, we follow the ideas of heterogeneous item embeddings [Kang and McAuley, 2018] and learn two item embedding matrices, denoted as $V \in \mathbb{R}^{n \times d}$ and $W \in \mathbb{R}^{n \times d}$, respectively. V and W are lookup tables, in which the j-th row, denoted as $\mathbf{v}_j$ and $\mathbf{w}_j$, respectively, represent item j. If item j is used to recommend the next items, it is represented by $\mathbf{v}_j$. If item j is a candidate to be recommended, it is represented by $\mathbf{w}_j$.

3.4 Recommendation Scores

The recommendation scores are calculated from the user embedding $\mathbf{u}$, the embedding of previous n_h purchases/ratings $\mathbf{h}$ and the embedding of previous n_l purchases/ratings $\mathbf{o}$. For user i, given the subsequence $S_i(t{-}n_h, n_h)$, the estimated recommendation score of user i on item candidate j, denoted as $\hat{r}_{ij}^t$, is calculated as follows:

$$\hat{r}_{ij}^t = \underbrace{\mathbf{u}_i \mathbf{w}_j^\top}_{\text{user's general preferences}} + \underbrace{\mathbf{h}_i^t \mathbf{w}_j^\top}_{\text{high-order association}} + \underbrace{\mathbf{o}_i^t \mathbf{w}_j^\top}_{\text{low-order association}}, \quad (1)$$

where $\mathbf{w}_j$ is embedding of item j, $\mathbf{h}_i^t$ is the embedding for $S_i(t{-}n_h, n_h)$ and $\mathbf{o}_i^t$ is the embedding for $S_i(t{-}n_l, n_l)$; $\mathbf{u}_i \mathbf{w}_j^\top$ measures how user i's general preferences match item candidate j; $\mathbf{h}_i^t \mathbf{w}_j^\top$ measures how $S_i(t{-}n_h, n_h)$ induces item candidate j, and $\mathbf{o}_i^t \mathbf{w}_j^\top$ measures how $S_i(t{-}n_l, n_l)$ induces item candidate j. For each user, we recommend the items of top-k largest recommendation scores. Note that we do not explicitly weight the three factors, as their weights can be learned as part of the user/sequence/item embeddings.

3.5 Objective Function

We adapt the Bayesian personalized ranking objective [Rendle *et al.*, 2012] and minimize the loss that occurs when the truly purchased/rated items are ranked below those not purchased/rated. The objective function is as follows:

$$\min_{\mathbf{\Theta}} \sum_{i=1}^{m} \sum_{\substack{S_i(t,n_p) \\ \subset S_i}} \sum_{\substack{j \in S_i(t,n_p) \\ k \notin S_i(t,n_p)}} -\log \sigma(\hat{r}_{ij} - \hat{r}_{ik}) + \lambda(\|\mathbf{\Theta}\|^2), \quad (2)$$

where $\mathbf{\Theta} = \{U, V, W\}$ is the set of the parameters, σ is the sigmoid function, $S_i(t, n_p)$ is a sequence of n_p items in S_i, j denotes an item in $S_i(t, n_p)$, and k denotes an item not in $S_i(t, n_p)$. Given the huge number of items not in $S_i(t, n_p)$, following the ideas in literature [Tang and Wang, 2018; Ma *et al.*, 2019], we randomly sample a non-purchased/rated item k for each purchased/rated item j. Please note that, in HAM, the recommendation scores of purchased/rated item are not necessarily close to their ground-truth ratings, as long as the scores of purchased/rated items are higher than scores of those not purchased/rated. Also note that each of the items in $S_i(t, n_p)$ will be recommended and evaluated independently from $S_i(t{-}n_h, n_h)$, following the literature [Ma *et al.*, 2019].

Table 2: Dataset Statistics

dataset	#users	#items	#intrns	#i/u	#u/i
Books	52,406	41,264	1,856,747	35.4	45.0
CDs	17,052	35,118	472,265	27.7	13.4
Children	48,296	32,871	2,784,423	57.6	84.7
Comics	34,445	33,121	2,411,314	70.0	72.8
ML-20M	129,780	13,663	9,926,480	76.5	726.5
ML-1M	5,950	3,125	573,726	96.4	183.6

The columns "#users"/"#items"/"#intrns" represent the number of users/items/user-item interactions, respectively. The column of "#i/u" represents the average number of interactions (average length of purchase/rating sequence) of each user. The column of "#u/i" represents the average number of purchases/ratings on each item.

4 Materials

4.1 Baseline Methods

We compare HAM with following state-of-the-art methods.

Caser [Tang and Wang, 2018] uses multiple convolutional filters on the most recent purchases/ratings of a user to extract the user's sequential features and items' group features. These two features and the users' long-term preferences are used to calculate item recommendation scores.

SASRec [Kang and McAuley, 2018] uses self-attention mechanisms to capture the most informative items in users' purchase/rating sequences to recommend the next item.

HGN [Ma *et al.*, 2019] uses gating mechanisms to identify important items and their latent features from users' historical purchases/ratings to recommend next items. HGN has been compared with a comprehensive set of other methods and has been demonstrated as the state of the art. Thus, we compare HAM with HGN instead of the methods that HGN outperforms.

4.2 Datasets

We evaluate the methods on 6 public benchmark datasets: Amazon-Books (Books) and Amazon-CDs (CDs) [He and McAuley, 2016b], Goodreads-Children (Children) and Goodreads-Comics (Comics) [Wan and McAuley, 2018], and MovieLens-1M (ML-1M) and MovieLens-20M (ML-20M) [Harper and Konstan, 2016]. The Books and CDs datasets are from Amazon reviews [Amazon, 2020], which contain users' 1-5 star ratings and reviews on books and CDs, respectively. The Children and Comics datasets are from goodreads website [GoodReads, 2020]. These two datasets contain users' implicit feedback (i.e., if a user has read the book or not), explicit feedback (i.e., ratings) and reviews on children and comics books. The ML-1M and ML-20M datasets are from the MovieLens website [MovieLens, 2020] with user-movie ratings. Following the data preprocessing protocol in HGN [Ma *et al.*, 2019], among the 6 datasets, we only kept the users with at least 10 ratings, and items with at least 5 ratings. We converted the rating values into binary values by setting rating 4 and 5 to value 1, and the lower ratings to value 0. Table 2 presents the statistics of the 6 datasets after the preprocessing.

Table 3: Performance Comparison in CUT (n_l=2)

	Dataset	Caser	SASRec	HGN	$\texttt{HAM}_\text{x}$	$\texttt{HAM}_\text{m}$	improv
Recall@10	CDs	0.0291	0.0376	0.0445	0.0430	**0.0508**	14.16%
	Books	0.0295	0.0365	0.0426	0.0441	**0.0477**	11.97%
	Children	0.1161	0.1204	0.1254	0.1332	**0.1350**	7.66%
	Comics	0.1580	0.1599	0.1733	0.1789	**0.1823**	4.67%
	ML-20M	0.1235	0.1073	0.1264	0.1220	**0.1284**	1.58%
	ML-1M	**0.1317**	0.1236	0.1233	0.1210	0.1258	-4.48%
NDCG@10	CDs	0.0176	0.0202	0.0233	0.0217	**0.0259**	11.16%
	Books	0.0235	0.0260	0.0316	0.0305	**0.0324**	2.53%
	Children	0.1118	0.1130	0.1126	0.1173	**0.1195**	6.13%
	Comics	0.1886	0.1874	0.1910	0.1938	**0.1959**	2.57%
	ML-20M	**0.1266**	0.1072	0.1184	0.1149	0.1213	-4.19%
	ML-1M	**0.1655**	0.1579	0.1521	0.1477	0.1537	-7.13%

Table 4: Performance Comparison in CUS (n_l=2, n_p=3)

	Dataset	Caser	SASRec	HGN	$\texttt{HAM}_\text{x}$	$\texttt{HAM}_\text{m}$	improv
Recall@10	CDs	0.0340	0.0402	0.0508	0.0488	**0.0583**	14.76%
	Books	0.0384	0.0473	0.0573	0.0596	**0.0625**	9.08%
	Children	0.1622	0.1773	0.1980	0.1961	**0.2008**	1.41%
	Comics	0.2012	0.2801	0.3055	0.3003	**0.3065**	0.33%
	ML-20M	**0.1779**	0.1368	0.1655	0.1605	0.1690	-5.00%
	ML-1M	0.1830	0.1963	0.2045	0.1950	**0.2083**	1.86%
NDCG@10	CDs	0.0133	0.0149	0.0211	0.0198	**0.0240**	13.74%
	Books	0.0163	0.0188	0.0241	0.0249	**0.0260**	7.88%
	Children	0.0703	0.0846	**0.1054**	0.1036	0.1051	0.28%
	Comics	0.1070	0.1621	**0.2040**	0.1942	0.1982	-2.84%
	ML-20M	**0.0713**	0.0530	0.0674	0.0665	0.0707	0.84%
	ML-1M	0.0763	0.0801	0.0852	0.0802	**0.0883**	3.64%

Table 5: Performance Comparison in LOS (n_l=2, n_p=3)

	Dataset	Caser	SASRec	HGN	$\texttt{HAM}_\text{x}$	$\texttt{HAM}_\text{m}$	improv
Recall@10	CDs	0.0306	0.0412	0.0493	0.0497	**0.0565**	14.60%
	Books	0.0327	0.0506	0.0517	0.0550	**0.0576**	11.41%
	Children	0.1241	0.1349	0.1473	0.1467	**0.1503**	2.04%
	Comics	0.1984	0.2196	**0.2366**	0.2271	0.2331	-1.48%
	ML-20M	0.1393	0.1259	0.1461	0.1420	**0.1489**	1.92%
	ML-1M	0.1794	0.1635	0.1762	0.1736	**0.1818**	1.34%
NDCG@10	CDs	0.0112	0.0156	0.0205	0.0203	**0.0239**	16.59%
	Books	0.0131	0.0211	0.0211	0.0226	**0.0236**	11.85%
	Children	0.0622	0.0646	0.0753	0.0737	**0.0763**	6.13%
	Comics	0.1219	0.1321	**0.1564**	0.1458	0.1486	-4.99%
	ML-20M	0.0561	0.0484	0.0601	0.0587	**0.0624**	3.83%
	ML-1M	0.0769	0.0670	0.0761	0.700	**0.0780**	1.43%

In the above three tables, the best performance in each dateset is **bold**. The second best performance in each dataset is underlined. The column "improv" presents the improvement of best performance of HAM-based methods over the best performance of non-HAM methods in each row.

4.3 Experimental Settings

We use the following three experimental settings to evaluate the methods. Figure 2 presents the three settings.

80-20-cut-off setting (CUT): We extract the first 70% of each user's sequence as training set, the next 10% as validation set for parameter tuning, and the remaining 20% as testing set [2]. CUT is the most widely used experimental setting in sequential recommendation literature [Yuan *et al.*, 2014; Zhao *et al.*, 2016; Tang and Wang, 2018; Ma *et al.*, 2019].

80-3-cut-off setting (CUS): We use the same training and validation set as in CUT, but only the next 3 items after the validation set as the testing set. Compared to CUT, CUS recommends the immediate next few items, not potentially many items that might be only purchased/rated much later (e.g., in CUT, 20% of a long user sequence may have many items).

Leave-out setting (LOS): We use only the last 3 items in each user sequence for testing and all the previous items for training and validation. The validation set contains only the 3 items before the testing items. Thus, LOS maximizes the data for training and recommends the immediate next few items.

[2] We used the data splits from https://github.com/allenjack/HGN.

4.4 Evaluation Metrics

Following the literature [Ma *et al.*, 2019], we use Recall@k and NDCG@k to evaluate the different methods. For each user, Recall@k measures the proportion of all the ground-truth purchased/rated items in the testing set that are correctly recommended. The overall Recall@k value is calculated as the average over all the users. Higher Recall@k indicates better recommendation performance. NDCG@k is the normalized discounted cumulative gain for among top-k ranking, in which gain $\in \{0, 1\}$, indicating whether a ground-truth purchased/rated item has been recommended (i.e., 1) or not (i.e., 0). Thus, NDCG@k measures the positions of the correctly recommended items among the top-k recommendations. Higher NDCG@k indicates better performance.

5 Experimental Results

5.1 Overall Performance in CUT Setting

Table 3 presents the results on all the six datasets in CUT. Note that CUT is the setting used in Caser and HGN. In CUT, for HGN, we used the parameters reported by its authors on CDs, Books, Children, Comics and ML-20M; on ML-1M we tuned the parameters using grid search. For $\texttt{HAM}_\text{m}$, $\texttt{HAM}_\text{x}$ and other baseline methods, we also tuned their parameters using grid search and report the best results. For HGN, we achieved similar results as reported; for other baseline methods, the results are slightly better than those reported [Ma *et al.*, 2019].

Table 3 shows that in terms of Recall@10, $\texttt{HAM}_\text{m}$ achieves the best performance on 5 out of 6 datasets, and the second best performance on the rest ML-1M dataset; $\texttt{HAM}_\text{x}$ achieves the second best performance on 3 out of 6 datasets. In terms of NDCG@10, $\texttt{HAM}_\text{m}$ achieves the best performance on 4 out of 6 datasets, and the second best performance on the ML-20M dataset; $\texttt{HAM}_\text{x}$ achieves the second best performance on 2 of 6 datasets. On average, $\texttt{HAM}_\text{m}$ achieves 27.90%, 18.90% and 7.10% improvement in terms of Recall@10 , and 14.08%, 12.27% and 4.31% improvement in terms of NDCG@10 over all 6 datasets compared to Caser, SASRec and HGN, respectively. This indicates that HAM outperforms the state of the art on most benchmark datasets with significant improvement.

Table 3 also shows that $\texttt{HAM}_\text{m}$ outperforms HGN on all the datasets. The difference between $\texttt{HAM}_\text{m}$ and HGN is that $\texttt{HAM}_\text{m}$ uses mean pooling over the last n_h items and last n_l items to recommend the next items, whereas HGN uses gating mechanisms over the last items and also over their latent features to differentiate the importance of such items and features.

However, as Table 2 shows, each user typically has only a few items (compared to all the possible items), and each item is typically only purchased/rated by a few users (compared to all the possible users). Therefore, the data sparsity issue may lead to less meaningful gating weights learned by parameterized gating mechanisms, whereas equal weights from mean pooling would suffice. Similarly, $\texttt{HAM}_\text{x}$, which uses max pooling to capture item difference, could also be effective, demonstrated by its relatively similar performance as $\texttt{HAM}_\text{m}$ in Table 3. In addition, HAM combines both high-order and low-order sequential patterns, conforming to the discovery in [Tang and Wang, 2018], which may also contribute to the superior performance.

In addition, $\texttt{HAM}_\text{m}$ outperforms SASRec on all the datasets in terms of Recall@10, and in terms of NDCG@10 on 5 out 6 datasets except on ML-1M. A key difference between HAM and SASRec is that HAM leverages item associations in the most recent purchases/ratings, whereas SASRec only uses the long-term user preferences. As demonstrated in literature [Zhou *et al.*, 2019], user preferences may shift over time and thus preferences from the most recent purchases/ratings might provide more pertinent information for the next recommendations. Moreover, $\texttt{HAM}_\text{m}$ outperforms $\texttt{HAM}_\text{x}$ across all datasets consistently. This might be due to a similar reason as for HGN, that is, the sparse data does not substantially enable well-learned difference among latent item features. Table 3 also shows that $\texttt{HAM}_\text{m}$ outperforms Caser except on the relatively dense datasets ML-1M and ML-20M, which may exhibit strong local patterns in the latent item features that Caser learns from sufficient data and utilizes for recommendation.

Overall, HAM outperforms the best baseline methods with very high percentage improvement when the dataset is very sparse (e.g., 14.16% on CDs on Recall@10), and significant improvement when the datasets is moderately sparse (e.g., 4.67% on Comics on Recall@10). When the datasets are dense (e.g., ML-1M as in Table 2), HAM could be slightly worse than the other baseline methods. However, most of the recommendation problems always have very sparse datasets, on which HAM will be effective. More detailed analysis on the data sparsity aspect is available later in Section 5.4.

5.2 Overall Performance in CUS Setting

Table 4 presents the results in CUS. In CUS, for $\texttt{HAM}_\text{m}$, $\texttt{HAM}_\text{x}$ and all the baseline methods, the parameters are tuned by grid search on the validation sets, and the best results are reported. Overall, the performance comparison between HAM and the baseline methods has a similar trend as in CUT. Particularly, $\texttt{HAM}_\text{m}$ achieves overall the best performance compared to the other methods. In terms of NDCG@10, although $\texttt{HAM}_\text{m}$ is slightly worse than HGN on Children and Comics, it still achieves the second best performance.

Table 3 and 4 together show that in terms of Recall@10, all the methods have in general better performance in CUS (evaluating on the immediate next few items) than in CUT (evaluating on the rest 20% items). Note that CUT and CUS have the same training sets but different testing sets. The results in the two settings correspond to our intuition that the historical purchase/rating sequences are most informative for the immediately next few items compared to the items purchased/rated much later. In terms of NDCG@10, however, the results in CUT are better. It may be due to that although the recall rate is low in CUT, the number of accurately recommended items is larger, which increases NDCG@10.

5.3 Overall Performance in LOS Setting

Table 5 presents the results in LOS. Overall, $\texttt{HAM}_\text{m}$ outperforms the other methods, achieving the best performance in terms of both Recall@10 and NDCG@10 on 5 datasets, and the second best performance on the rest dataset. Table 4 and 5 together show that in terms of Recall@10 and NDCG@10, all the methods have in general better performance in CUS (i.e., the next 3 items after the validation set of each user are used for testing and the first 80% sequence are used for training and validation) than in LOS (i.e., the last 3 items of each user are used for testing and all the previous items are used for training and validation). Compared to CUS setting, the training sets in LOS setting contain more early purchases/ratings (i.e., purchases/ratings that occurred long time ago before the testing items). These early purchases/ratings may not accurately represent users' preferences at the time of the testing items as such preferences may shift [Zhou *et al.*, 2019].

5.4 HGN Attention Weight Analysis

It has been shown that the learned attention weights may not always be meaningful [Jain and Wallace, 2019; Serrano and Smith, 2019]. Therefore, we further investigate the attention weights in HGN to interpret their significance and to understand why instead the simplistic mean pooling in HAM would suffice. We use datasets CDs, Comics, ML-1M and ML-10M in the investigation, because these datasets, as Table 2 shows, represent different data sparsities (i.e., CDs is highly sparse, Comics is moderately sparse, and ML-1M and ML-10M are dense). Figure 3 (the x-axis in the figure is logarithmized item frequencies and then normalized into [0,1]) shows that most of the items in CDs and Comics are very infrequent, whereas in ML-1M infrequent items are fewer; in ML-20M, the infrequent items (compared to other frequent items in ML-20M) still have many purchases/ratings (Table 2).

Figure 4a, 4b, 4c and 4d present the distributions of attention weights from the best performing HGN models on the CDs, Comics, ML-1M and ML-20M, respectively. Note that a same item can have different weights in different user sequences; we use all the weights of a same item from all the users in the figures. The distributions for CDs and Comics datasets show very similar pattens: for very infrequent items, their attention weights are highly centered around 0.5 (i.e., the initialization value); for the most frequent items, their attention weights are slightly off 0.5 and have some different values than 0.5. Given that as Figure 3 shows, most of the items in CDs and Comics are infrequent, the weight distribution over infrequent items indicates that the weights might not be well learned to differentiate the importance of infrequent items. The weights on frequent items might be relatively better learned. However, unfortunately, frequent items are not many and their weights may not substantially affect recommendations. The distributions for ML-1M and ML-20M datasets also show similar pattens: the weights for both infrequent and frequent items are closely centered at 0.5, indicat-

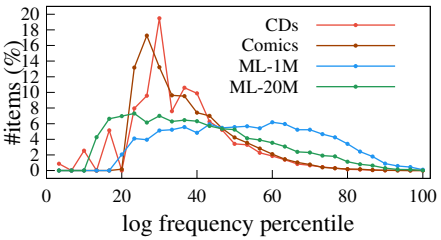


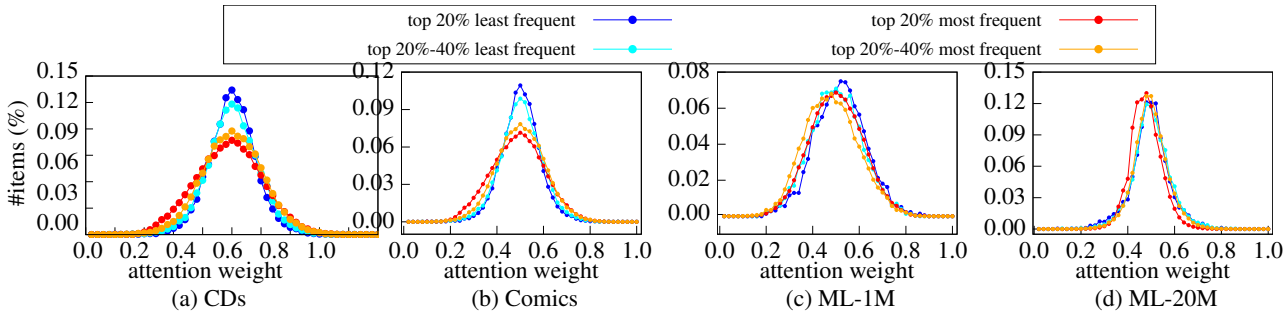
Figure 3: Item Frequency Distribution

Table 6: HAM$_m$ Parameter Study (CUS, n_l=2, n_p=3)

n_h	CDs		Comics	
	Recall@5	Recall@10	Recall@5	Recall@10
3	0.0358	0.0556	0.2384	0.3022
4	**0.0369**	**0.0583**	0.2402	0.3043
5	0.0366	0.0567	**0.2414**	**0.3065**
6	0.0360	0.0572	0.2392	0.3063

Table 7: HAM$_m$ Parameter Study (LOS, n_l=2, n_p=3)

n_h	CDs		Comics	
	Recall@5	Recall@10	Recall@5	Recall@10
4	0.0343	0.0544	0.1818	0.2317
5	**0.0357**	**0.0565**	**0.1823**	**0.2331**
6	0.0355	0.0550	0.1821	0.2314



(a) CDs (b) Comics (c) ML-1M (d) ML-20M

Figure 4: HGN Attention Weight Distributions

Table 8: Recall@10 for HAM$_m$ Parameter Study (CUT, n_l=2)

n_h	n_p	CDs	Comics
4	1	0.0507	0.1676
	2	0.0507	0.1763
	3	0.0505	0.1795
	4	0.0499	0.1792
5	2	0.0507	0.1786
	3	**0.0508**	0.1814
	4	0.0496	0.1813
6	2	0.0507	0.1787
	3	0.0498	0.1813
	4	0.0487	**0.1823**
	5	0.0498	0.1807

In these two tables, the best performance for each n_h is underlined. The best performance across all n_h values is **bold**.

Table 9: Recall@10 for HAM$_m$ Ablation Study (CUT, n_l=0)

n_h	n_p	CDs	Comics
2	2	0.0487	0.1588
	3	0.0482	0.1613
	4	0.0472	0.1620
	5	0.0470	0.1605
3	1	0.0484	0.1486
	2	**0.0492**	0.1585
	3	0.0481	0.1607
	4	0.0475	**0.1624**
	5	0.0477	0.1623
4	2	0.0491	0.1584
	3	0.0481	0.1608
	4	0.0463	0.1618
	5	0.0461	0.1600

ing that such weights might not well differentiate item importance. Such HGN weight distributions from both sparse and dense datasets indicate that the learned weights may not play an effective role in recommendation. Instead, a special case of weights, that is, equal weights as we have in HAM, should also achieve comparable performance as HGN. As a matter of fact, equal weights on better learned item representations as in HAM actually improve the recommendation performance.

5.5 Parameter and Ablation Study

Table 8 presents the parameter study on HAM$_m$ in the CUT on two sparse datasets CDs and Comics. Recall that n_h/n_l and n_p are the number of items in high-order/low-order associations, and the number of items to be recommended used for training. Table 8 shows that as more items are used to learn high-order item associations (i.e., larger n_h) and more items are recommended during training (i.e., larger n_p), the recommendation performance over the remaining 20% items is better. Still, the best performance is achieved when n_h and n_p are small (n_h is 4 or 5; n_p is 2 to 4), indicating that the most recent associations among a few items are effective in recommending next items.

Table 9 presents the results when low-order associations among items are not included in HAM$_m$ (i.e., $n_l = 0$) in CUT. Table 9 shows that when only a small number of items are used to learn item associations (e.g., n_h=3) and a small number of items are recommended during training (e.g., n_p=2), HAM$_m$ without low-order item associations achieves its best performance. Comparing Table 8 and Table 9, it is noticed that when low-order associations are not used, the recommendation performance (Table 9) is significantly worse than that when low-order associations are used (Table 8). It indicates that explicitly modeling the low-order associations, together with high-order associations, enables HAM to better capture the hybrid impact from the previous, different number of items, and thus to improve the performance.

Table 6 and 7 present the parameter study of HAM$_m$ in CUS and LOS, respectively. Similarly as in Table 8, a small number of items in high-order associations (e.g., n_h = 4) is sufficient for the best performance on sparse datasets.

6 Discussions on Experimental Settings

CUT (i.e., to evaluate the last 20% items) and NDCG@k (e.g., k=10) are the most commonly used experimental setting and evaluation metric. NDCG@k in CUT could overestimate the sequential recommendation performance, particularly for long sequences in which the last 20% items include many. In such long sequences, NDCG@k could be high when items that are purchased/rated very late are recommended on top. However, such recommendations would have limited use scenarios. Meanwhile, many testing items will increase the chances that the recommended k items will be included in the many items, and thus inflate the NDCG@k values. CUS and LOS mitigate the over-estimation issue because always a same number of items will be tested and NDCG@k calculated over a same number of testing items will not be affected by the number of testing items.

References

[Amazon, 2020] Amazon. Amazon review dataset. http://jmcauley.ucsd.edu/data/amazon/, 2020.

[Dacrema *et al.*, 2019] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. Are we really making much progress? a worrying analysis of recent neural recommendation approaches. 2019.

[GoodReads, 2020] GoodReads. Goodreads dataset. https://www.goodreads.com/, 2020.

[Harper and Konstan, 2016] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):19, 2016.

[He and McAuley, 2016a] Ruining He and Julian McAuley. Fusing similarity models with markov chains for sparse sequential recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 191–200. IEEE, 2016.

[He and McAuley, 2016b] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517. International World Wide Web Conferences Steering Committee, 2016.

[Hidasi and Karatzoglou, 2018] Balázs Hidasi and Alexandros Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 843–852. ACM, 2018.

[Hidasi *et al.*, 2015] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[Jain and Wallace, 2019] Sarthak Jain and Byron C Wallace. Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, 2019.

[Kang and McAuley, 2018] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 197–206. IEEE, 2018.

[Ludewig *et al.*, 2019] Malte Ludewig, Noemi Mauro, Sara Latifi, and Dietmar Jannach. Performance comparison of neural and non-neural approaches to session-based recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 462–466, New York, NY, USA, 2019. Association for Computing Machinery.

[Ma *et al.*, 2019] Chen Ma, Peng Kang, and Xue Liu. Hierarchical gating networks for sequential recommendation. *arXiv preprint arXiv:1906.09217*, 2019.

[MovieLens, 2020] MovieLens. Movielens dataset. https://movielens.org/, 2020.

[Rendle *et al.*, 2010] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 811–820. ACM, 2010.

[Rendle *et al.*, 2012] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.

[Serrano and Smith, 2019] Sofia Serrano and Noah A Smith. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951, 2019.

[Tang and Wang, 2018] Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 565–573. ACM, 2018.

[Wan and McAuley, 2018] Mengting Wan and Julian McAuley. Item recommendation on monotonic behavior chains. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 86–94. ACM, 2018.

[Yuan *et al.*, 2014] Quan Yuan, Gao Cong, and Aixin Sun. Graph-based point-of-interest recommendation with geographical and temporal influences. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 659–668. ACM, 2014.

[Zhao *et al.*, 2016] Shenglin Zhao, Tong Zhao, Haiqin Yang, Michael R Lyu, and Irwin King. Stellar: spatial-temporal latent ranking for successive point-of-interest recommendation. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[Zhou *et al.*, 2019] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5941–5948, 2019.