Next Item Recommendation with Self-Attentive Metric Learning

Shuai Zhang[1], Yi Tay[2], Lina Yao[1], Aixin Sun[2], Jake An[1]
[1]University of New South Wales, Australia
[2]Nanyang Technological University, Singapore
shuai.zhang@unsw.edu.au,
ytay017@e.ntu.edu.sg,

Abstract

In this paper, we propose a novel sequence-aware recommendation model. Our model utilizes self-attention mechanism to infer the item-item relationship from user's historical interactions. With self-attention, it is able to estimate the relative weights of each item in user interaction trajectories to learn better representations for user's transient interests. The model is finally trained in a metric learning framework, taking both local and global user intentions into consideration. Experiments on a wide range of datasets on different domains demonstrate that our approach outperforms the state-of-the-art by a wide margin.

Introduction

Anticipating a user's next interaction lives at the heart of making personalized recommendations. The importance of such systems cannot be overstated, especially given the ever growing amount of data and choices that consumers are faced with each day (Quadrana *et al.* 2018). Across a diverse plethora of domains, a wealth of historical interaction data exists, *e.g.,* click logs, purchase histories, views etc., which have, across the years, enabled many highly effective recommender systems.

Exploiting historical data to make future predictions have been the cornerstone of many machine learning based recommender systems. After all, it is both imperative and intuitive that a user's past interactions are generally predictive of their next. To this end, many works have leveraged upon this structural co-occurrence, along with the rich sequential patterns, to make informed decisions. Our work is concerned with building highly effective sequential recommender systems by leveraging these auto-regressive tendencies.

This paper proposes a new neural sequential recommender system where sequential representations are learned via modeling not only user short term preferences but across her general interests. As such our model can be considered as a 'local-global' approach. Overall, our intuition manifests in the form of a self-attentive metric embedding model that explicitly invokes item-item interactions across user's recent behaviors as well as user-item interactions across all her past activities. This not only enables us to learn global/long-range representations, but also short-term information between consecutive items. With self-attention, we learn to attend over the interaction sequence to effectively select the most relevant items to form the representation of user short-term intentions. Our experiments show that the proposed model outperforms the state-of-the-art sequential recommendation models by a wide margin, demonstrating the effectiveness of not only modeling local dependencies but also going global.

Our model takes the form of a metric learning framework in which the distance between the self-attended representation of a user and the prospective (golden) item is drawn closer during training. To the best of our knowledge, this is the first proposed attention-based metric learning approach in the context of sequential recommendation. To recapitulate, the prime contributions of this work are as follows:

- We propose a novel framework for sequential recommendation task. Our model combines self-attention network with metric embedding to model user temporary as well as long-lasting intents.
- Our proposed framework demonstrates the utility of explicit item-item relationships during sequence modeling by achieving state-of-the-art performance across **twelve** well-established benchmark datasets. Our proposed model outperforms the current state-of-the-art models (*e.g.,* Caser, TransRec and RNN based approach) on all datasets by large margins.
- We conduct extensive hyper-parameter and ablation studies. We study the impacts of various key hyper-parameters and model architectures on model performance. We also provide a qualitative visualisation of the learned attention matrices.

Related Work

Sequence-aware Recommender Systems

In many real-world applications, user-item interactions are recorded over time with associated timestamps. The accumulated data enables modelling temporal dynamics and provides evidence for user preference refinement (Koren 2009; Quadrana *et al.* 2018; Rendle *et al.* 2010; He and McAuley 2016a; Chen *et al.* 2018). Koren *et al.* (Koren 2009) propose treating user and item biases as a function that changes over

time, to model both item transient popularity and user temporal inclinations. Xiong *et al.* (Xiong *et al.* 2010) introduce additional factors for time and build a Bayesian probabilistic tensor factorization approach to model time drifting. Wu *et al.* (Wu *et al.* 2017) use recurrent neural network to model the temporal evolution of ratings. Nonetheless, these methods are specifically designed for the rating prediction task.

To generate personalized ranking lists, Rendle *et al.* (Rendle *et al.* 2010) propose combining matrix factorization with Markov chains for next-basket recommendation. Matrix factorization can capture user's general preference while Markov chain is used to model the sequential behavior. He *et al.* (He and McAuley 2016a) describe a sequential recommendation approach which fuses similarity based methods with Markov chain. Apart from Markov Chain, metric embedding has also shown to perform well on sequence-aware recommendation. Feng *et al.* (Feng *et al.* 2015) introduce a Point-of-Interest recommender with metric embedding to model personalized check-in sequences. Then, He *et al.* improve this model by introducing the idea of translating embedding (Bordes *et al.* 2013; He *et al.* 2017a). This approach views user as the relational vector acting as the junction between items. The major advantage of using metric embedding instead of matrix factorization is that it satisfies the transitive property of inequality states (Hsieh *et al.* 2017; Tay *et al.* 2018a).

Neural Attention Models

The neural attention mechanism shares similar intuition with that of the visual attention found in humans. It learns to pay attention to only the most important parts of the target, and has been widely employed across a number of applications *e.g.,* natural language processing and computer vision. Standard vanilla attention mechanism can be integrated into CNN and RNN to overcome their shortcomings. Specifically, attention mechanism makes it easy to memorize very long-range dependencies in RNN, and helps CNN to concentrate on important parts of inputs. Several recent studies also investigated its capability in recommendation tasks such as hashtag recommendation (Gong and Zhang 2016), one-class recommendation (Chen *et al.* 2017; He *et al.* 2018b; Tay *et al.* 2018c; 2018b), and session based recommendation (Li *et al.* 2017).

Our work is concerned with a new concept known as 'self-attention', or 'intra-attention'. Different from the standard vanilla attention, self-attention focuses on co-learning and self-matching of two sequences whereby the attention weights of one sequence is conditioned on the other sequence, and vice versa. It has only started to gain exposure due to its recent successful application on machine translation (Vaswani *et al.* 2017). It can replace RNN and CNN in sequence learning, achieving better accuracy with lower computation complexity. In this work, we utilize self-attention to model dependencies and importance of user short term behavior patterns. Note that, the usage of self-attention in the context of recommender systems is far from straightforward, substantially contributing to the overall novelty of our work.

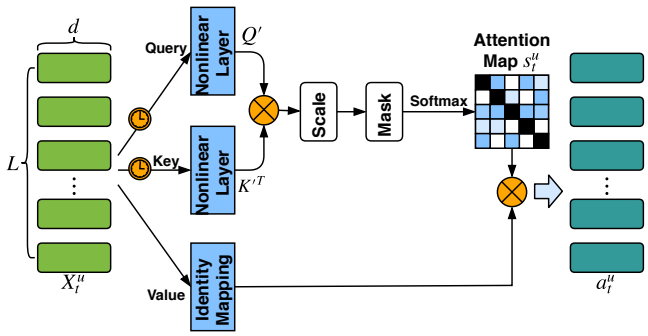


Figure 1: Illustration of the self-attention module. The input is the embedding matrix of the latest interacted L items, and the output is the self-attentive representations.

The Proposed Model: AttRec

We now present the proposed self-attentive sequential recommendation model, named **AttRec**. Our model consists of a *self-attention module* to model user short-term intent, and a *collaborative metric learning* component to model user long-term preference. Next, we formally define the task of sequential recommendation.

Sequential Recommendation

Sequential recommendation is very different from traditional one-class collaborative filtering recommendation. Let $\mathcal{U}$ be a set of users and $\mathcal{I}$ be a set of items, where $|\mathcal{U}| = M$ and $|\mathcal{I}| = N$. We use $\mathcal{H}^u = (\mathcal{H}_1^u, \cdots, \mathcal{H}_{|\mathcal{H}^u|}^u)$ to denote a sequence of items in chronological order that user u has interacted with before, where $\mathcal{H}^u \sqsubseteq \mathcal{I}$. The objective of sequential recommendation is to predict the next items that the user will interact with, given her former consumption trajectory.

Short-Term Intents Modelling with Self-Attention

User recent interactions reflect user's demands or intents in a near term. Modelling user short-term interaction therefore is an important task for better understanding of user's temporal preferences. To this end, we propose leveraging the recent success of self-attention mechanism in capturing sequential patterns, and use it to model user's recent interaction trail. Figure 1 illustrates the proposed self-attention module in our method.

Self-Attention Module. Self-attention is an special case of the attention mechanism and has been successfully applied to a variety of tasks. It refines the representation by matching a single sequence against itself. Unlike basic attention that learns representations with limited knowledge of the whole context, self-attention can keep the contextual sequential information and capture the relationships between elements in the sequence, regardless of their distance. Here, we apply self-attention to attend user's past behaviours.

The building block of self-attention is scaled dot-product attention. The input of the attention module consists of *query*, *key*, and *value*. The output of attention is a weighted

sum of the *value*, where the weight matrix, or affinity matrix, is determined by *query* and its corresponding *key*. In our context, all of these three components (*i.e., query*, *key*, and *value*) are the same and composed from user recent interaction histories (see Figure 1).

Suppose user's short-term intents can be derived from her recent L (*e.g.,* 5, 10) interactions. Assuming each item can be represented with a d-dimension embedding vector. Let $X \in \mathcal{R}^{N \times d}$ denote the embedding representations for the whole item set. The latest L items (*i.e.,* from item $t-L+1$ to item t) are stacked together in sequence to get the following matrix.

$$X_t^u = \begin{bmatrix} X_{(t-L+1)1} & X_{(t-L+1)2} & \cdots & X_{(t-L+1)d} \\ \vdots & \vdots & \vdots & \vdots \\ X_{(t-1)1} & X_{(t-1)2} & \cdots & X_{(t-1)d} \\ X_{t1} & X_{t2} & \cdots & X_{td} \end{bmatrix} \quad (1)$$

Here, the latest L items is a subset of $\mathcal{H}^u$. *Query*, *key*, and *value* for user u at time step t in the self-attention model equal to X_t^u.

First, we project *query* and *key* to the same space through nonlinear transformation with shared parameters.

$$Q' = ReLU(X_t^u W_Q) \quad (2)$$

$$K' = ReLU(X_t^u W_K) \quad (3)$$

where $W_Q \in \mathcal{R}^{d \times d} = W_K \in \mathcal{R}^{d \times d}$ are weight matrices for *query* and *key* respectively. *ReLU* is used as the activation function, to introduce some non-linearity to the learned attention. Then, the affinity matrix is calculated as follows:

$$s_t^u = \text{softmax}(\frac{Q' K'^T}{\sqrt{d}}) \quad (4)$$

The output is a $L \times L$ affinity matrix (or attention map) which indicates the similarity among L items. Note that the $\sqrt{d}$ is used to scale the dot product attention. As in our case, d is usually set to a large value (*e.g.,* 100), so the scaling factor could reduce the extremely small gradients effect. A masking operation (which masks the diagonal of the affinity matrix) is applied before the softmax, to avoid high matching scores between identical vectors of *query* and *key*.

Second, we keep the *value* equals to X_t^u unchanged. Unlike in other cases (Vaswani *et al.* 2017) where *value* is usually mapped with linear transformations, we found that it is beneficial to use identity mapping in our model. In other application domains, the *value* is usually pretrained feature embeddings such as word embedding or image features. In our model, the *value* is made up of parameters that need to be learned. Adding linear (or nonlinear) transformation will increase the difficulty in seeing the actual parameters. Note that *query* and *key* are used as auxiliary factors so that they are not as sensitive as *value* to transformations.

Finally, the affinity matrix and the *value* are multiplied to form the final weighted output of the self-attention module.

$$a_t^u = s_t^u X_t^u \quad (5)$$

Here, the attentive output $a_t^u \in \mathcal{R}^{L \times d}$ can be viewed as user's short-term intent representations. In order to learn a single attentive representation, we take the mean embedding of the L self-attention representations as user temporal intent. Note that other aggregation operation (*e.g.,* sum, max, and min) can also be used and we will evaluate their effectiveness in our experiments.

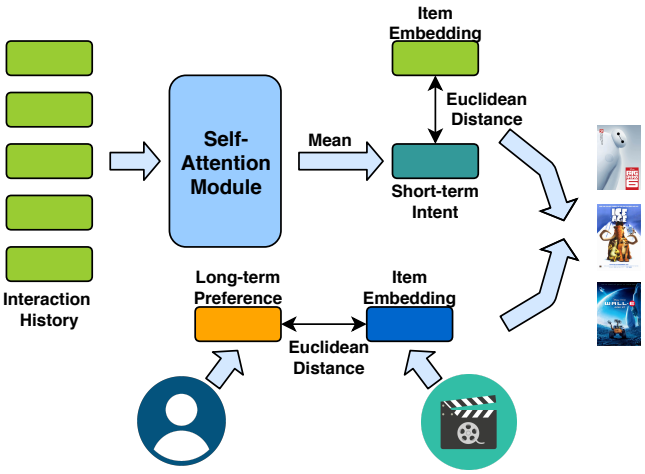


Figure 2: The architecture of the self-attentive metric learning approach for sequential recommendation. This model combine self-attention network with metric learning and considers both user's short-term and long-term preference.

$$m_t^u = \frac{1}{L} \sum_{l=1}^{L} a_{tl}^u \quad (6)$$

Input Embedding with Time Signals. The above attention model does not include time signals. Without time sequential signals, the input degrades to bag of embeddings and fails to retain the sequential patterns. Following the Transformer, we propose to furnish the *query* and *key* with time information by positional embeddings. Here, we use a geometric sequence of timescales to add sinusoids of different frequencies to the input. The time embedding (TE) consists of two sinusoidal signals defined as follows.

$$TE(t, 2i) = sin(t/10000^{2i/d}) \quad (7)$$

$$TE(t, 2i+1) = cos(t/10000^{2i/d}) \quad (8)$$

Here, t is the time step and i is the dimension. The TE is simply added to *query* and *key* before the nonlinear transformation.

User Long-Term Preference Modelling

After modelling the short-term effects, it is beneficial to incorporate general tastes or long-term preference of users. Same as latent factor approach, we assign each user and each item a latent factor. Let $U \in \mathcal{R}^{M \times d}$ and $V \in \mathcal{R}^{N \times d}$ denote the latent factors of users and items. We could use dot product to model the user item interaction as in latent factor model. However, recent studies (Hsieh *et al.* 2017; Tay *et al.* 2018a) suggest that dot product violate the important inequality property of metric function and will lead to

sub-optimal solutions. To avoid this problem, we adopt the Euclidean distance to measure the closeness between item i and user u.

$$\| U_u - V_i \|_2^2 \tag{9}$$

The distance is expected to be small if user u liked the item i, and large otherwise.

Model Learning

Objective Function. Given the short-term attentive intents at time step t and long-term preference, our task is to predict the item (denoted by $\mathcal{H}_{t+1}^u$) which the user will interact with at time step $t+1$. To keep consistency, we adopt Euclidean distance to model both short-term and long-term effects, and use their sum as the final recommendation score.

$$y_{t+1}^u = \omega \| U_u - V_{\mathcal{H}_{t+1}^u} \|_2^2 + (1-\omega) \| m_t^u - X_{t+1}^u \|_2^2 \tag{10}$$

In the above equation, the first term denotes the long-term recommendation score between user u and the next item $\mathcal{H}_{t+1}^u$, while the second term indicates the short-term recommendation score between user u and its next item. Note that both $V_{\mathcal{H}_{t+1}^u}$ and X_{t+1}^u are the embedding vectors for the next item, but V and X are two different parameters. The final score is a weighted sum of them with the controlling factor ω.

In some cases, we may want to predict the next several items instead of just one item. It enables our model to capture the skip behaviours in the sequence (Tang and Wang 2018). Let $\mathcal{T}^+$ denote the next T items that user liked in groundtruth. In this paper, we adopt a pairwise ranking method to learn the model parameters. Thus we need to sample T negative items that the user does not interact with (or dislike) and denote this set by $\mathcal{T}^-$. Apparently $\mathcal{T}^-$ is sampled from the set $\mathcal{I} \backslash \mathcal{T}^+$. To encourage discrimination between positive user item pair and negative user item pair, we use the following margin-based hinge loss.

$$\mathcal{L}(\Theta) = \sum_{(u,i) \in \mathcal{T}^+} \sum_{(u,j) \in \mathcal{T}^-} [y_i^u + \gamma - y_j^u]_+ + \lambda \| \Theta \|_2^2 \tag{11}$$

In the above equation, $\Theta = \{X, V, U, W_Q, W_K\}$ represents model parameters. γ is the margin that separates the positive and negative pairs. We use ℓ_2 loss to control the model complexity. Dropout can also be applied for nonlinear layer of the self-attention module. Because we use Euclidean distance in our method, for sparse datasets, we could also alternatively apply the norm clipping strategy to constrain X, V, U in a unit Euclidean ball.

$$\| X_* \|_2 \leq 1, \| V_* \|_2 \leq 1, \| U_* \|_2 \leq 1 \tag{12}$$

This regularization approach is useful for sparse dataset to alleviate the curse of dimensionality problem and prevent the data points from spreading too broadly.

Optimization and Recommendation. We optimize the proposed approach with adaptive gradient algorithm (Duchi *et al.* 2011) which could adapt the step size automatically; hence it reduces the efforts in learning rate tuning. In the recommendation stage, candidature items are ranked in ascending order based on the recommendation score computed by Equation (10) and the top ranked items are recommended to users.

Since we optimize the model in a pairwise manner, the training time complexity for each epoch is $\mathcal{O}(\mathcal{K}d)$ where K is the number of observed interactions. Note that $K \ll MN$ so that it can be trained efficiently. In the recommendation stage, we compute the recommendation scores for all user item pairs at once and generate the ranking lists with efficient sort algorithms. As such, the whole process can run very fast.

Figure 2 illustrates the architecture of our model. It includes not only user transient intents but also long-lasting preference. Both are added up to generate the final recommendation lists. The former is inferred by self-attention network from recent actions and the whole system is constructed under the metric learning framework.

Experiments

We evaluate the proposed model on a wide spectrum of real world datasets. We then conduct detailed ablation studies. In short, our experiments are designed to answer the following research questions: **RQ1**. Does the proposed self-attentive sequential recommendation model achieve state-of-the-art performance? Can it deal with sparse datasets? **RQ2**. What is the effect of the key hyper-parameters? For example, the aggregation method and the length of sequence for mining short-term intents.

Datasets Descriptions

We conduct experiments on the following datasets. All of them include time-stamps of interactions. **MovieLens**: This is a popular benchmark dataset for evaluating the performance of recommendation models. We adopt three well-established versions in our experiments: Movielens 100K, Movielens HetRec and Movielens 1M. **Amazon**: This is a user purchase and rating dataset collected from Amazon, a well-known e-commerce platform, by McAuley *et al.* (He and McAuley 2016b; McAuley *et al.* 2015). In this work, we adopt 7 sub-categories: Android Apps, Health/Care, Video Game, Tools/Home, Digital Music, Garden and Instant Video, due to limited space. textbfLastFM: This dataset contains user tag assignments collected from last.fm online music system[1]. **MovieTweetings**: It is obtained by scraping Twitter for well-structured tweets for movie ratings. This dataset is comparatively new and being updated[2]. The subset we used was downloaded in December 2016.

For datasets with explicit ratings, we convert it to implicit feedback following early studies (He *et al.* 2017b; Tay *et al.* 2018a). For Amazon, lastFM and MovieTweetings, we perform a modest filtering similar to (Rendle *et al.* 2010; Wang *et al.* 2015; Feng *et al.* 2015; He *et al.* 2017a; Tang and Wang 2018) to discard users with fewer than 10 associated actions and remove cold-start items. This is a common preprocess to reduce the noise of cold start issue as it is usually addressed separately (Tang and Wang 2018). Detail statistics

[1] http://www.lastfm.com

[2] https://github.com/sidooms/MovieTweetings

Table 1: Statistics of the datasets used in experiments.

Dataset	#Users	#Items	#Interactions	Density	#actions/ user	Time Interval
ML-100K	943	1,682	100,000	6.30%	106.04	Sept/1997 - Apr/1998
ML-HetRec	2,113	10,109	855,598	4.01%	404.92	Sept/1997 - Jan/2009
ML-1M	6,040	3,706	1,000,209	4.46%	165.57	Apr/2000 - Mar/2003
Android App	21,309	19,256	358,877	0.087%	16.84	Mar/2010 - Jul/2014
Health / Care	11,588	31,709	211,284	0.058%	18.23	Dec/2000 - Jul/2014
Video Game	7,220	16,334	140,307	0.119%	19.43	Nov/1999 - Jul/2014
Tools / Home	5,855	23,620	96,467	0.070%	16.48	Nov/1999 - Jul/2014
Digital Music	2,893	13,183	64,320	0.169%	22.23	May/1998 - Jul/2014
Garden	1,036	4,900	15,576	0.307%	15.03	Apr/2000 - Jul/2014
Instant Video	1,000	3,296	15,849	0.481%	15.85	Aug/2000 - Jul/2014
LastFM	951	12,113	83,382	0.724%	87.68	Aug/2005 - May/2011
MovieTweetings	9,608	14,220	461,970	0.338%	48.08	Mar/2013 - Dec/2016

of the datasets are presented in Table 1. It shows Movielens datasets are more dense than other datasets.

Evaluation Metrics

For each user, we use the most recent item for test and the second most recent item for hyper-parameter tuning. We evaluate the performance of all the models with hit ratio and mean reciprocal rank (MRR). Hit ratio measures the accuracy of the recommendation. We report the hit ratio with cut off value 50, defined as follows:

$$HR@50 = \frac{1}{M} \sum_{u \in \mathcal{U}} \mathbf{1}(R_{u,g_u} \leq 50) \tag{13}$$

Here, g_u is the item that user u interacted with at the most recent time step. R_{u,g_u} is the rank generated by the model for this groundtruth item. If a model ranks g_u among the top 50, the indicator function will return 1, otherwise 0.

Mean Reciprocal Rank indicates how well the model rank the item. Intuitively, ranking the groundtruth item higher is more preferable in practice. The definition of MRR is as follows:

$$MRR = \frac{1}{M} \sum_{u \in \mathcal{U}} \frac{1}{R_{u,g_u}} \tag{14}$$

R_{u,g_u} is the rank for groundtruth item. MRR cares about the position of the groundtruth item and calculates the reciprocal of the rank at which the groundtruth item was put.

Compared Models

Our model is dubbed as **AttRec** which can be considered as the abbreviation of "attentive recommendation". We compare AttRec with classic methods as well as recent state-of-the-art models. Specifically, the following baseline models are evaluated.

- **BPRMF** (Rendle *et al.* 2009). It optimizes the matrix factorization in a pairwise manner with Bayesian Personalized Ranking loss, which aims to maximize the difference between positive and negative items. It does not model the sequential signals.
- **FMC**. This is a simplified version of factorized personalized Markov Chain (FPMC) (Rendle *et al.* 2010) which does not include user personalized behaviours.
- **FPMC** (Rendle *et al.* 2010). This approach combines matrix factorization machine with Markov Chain for next item recommendations. The proposed approach captures both user-item preferences and user sequential behaviours.
- **HRM** (Wang *et al.* 2015). It is a Hierarchical Representation Model which captures both sequential and general user tastes by introducing both linear and nonlinear pooling operation for historical transaction aggregation. Here, the average aggregation is adopted.
- **PRME** (Feng *et al.* 2015). This model was originally proposed for POI recommendation. It utilizes metric embedding to learn user and item embeddings as well as the user check-in sequences.
- **TransRec** (He *et al.* 2018a; 2017a). This model applies the idea of translating embeddings (Bordes *et al.* 2013) to sequential recommendation. It views users as relation vectors and assumes that the next item is determined by user's recent interacted item plus the user relation vectors.
- **Caser** (Tang and Wang 2018). It models user past historical interactions with both hierarchical and vertical convolutional neural networks. It also considers the skip behaviors and the whole network is optimized by minimizing the cross entropy.
- **LSTM+**. To verify the efficacy of self-attention over RNN, we further design a variant of AttRec by replacing the self-attention module with LSTM (we also tested GRU, it turned out that LSTM slightly outperforms GRU).

Among all of these baselines, Caser and HRM are neural network based approach. PRME and TransRec are metric embedding based algorithms. Note that, in our experiments, we do not use pre-train for all models.

Table 2: Performance comparison in terms of hit ratio and MRR on all datasets. Best performance is in boldface.

Dataset	Metric	BPRMF	MC	FPMC	HRM	PRME	TransRec	Caser	LSTM+	AttRec
ML-100K	HR@50	0.3754	0.4115	0.4783	0.4821	0.4411	0.4634	0.4667	0.4867	**0.5273**
	MRR	0.0616	0.0662	0.0925	0.0889	0.0837	0.0827	0.0799	0.0856	**0.0981**
ML-HetRec	HR@50	0.1462	0.1903	0.2321	0.2380	0.2357	0.1912	0.2144	0.2376	**0.2964**
	MRR	0.0215	0.0359	0.0489	0.0486	0.0500	0.0337	0.0387	0.0414	**0.0592**
ML-1M	HR@50	0.2378	0.3419	0.4209	0.4311	0.4449	0.3358	0.4811	0.4776	**0.5223**
	MRR	0.0368	0.0654	0.1022	0.0873	0.1044	0.0561	0.0925	0.0961	**0.1172**
Android App	HR@50	0.1738	0.1802	0.1990	0.2001	0.1686	0.2016	0.1426	0.1684	**0.2187**
	MRR	0.0287	0.0355	0.0355	0.0295	0.0237	0.0306	0.0231	0.0275	**0.0415**
Health / Care	HR@50	0.0900	0.0786	0.1128	0.0965	0.0843	0.0962	0.0768	0.0756	**0.1272**
	MRR	0.0188	0.0245	0.0258	0.0183	0.0119	0.0232	0.0146	0.0142	**0.0277**
Video Game	HR@50	0.1630	0.1708	0.2226	0.2150	0.1855	0.2035	0.1438	0.1576	**0.2414**
	MRR	0.0277	0.0381	0.0451	0.0337	0.0263	0.0349	0.0248	0.0279	**0.0496**
Tools / Home	HR@50	0.0559	0.0384	0.0535	0.0488	0.0465	0.0658	0.0424	0.0550	**0.0775**
	MRR	0.0099	0.0093	0.0129	0.0086	0.0076	0.0112	0.0071	0.0121	**0.0148**
Digital Music	HR@50	0.1621	0.1307	0.1580	0.1998	0.1559	0.1894	0.1327	0.1324	**0.2205**
	MRR	0.0277	0.0320	0.0322	0.0310	0.0243	0.0300	0.0228	0.0240	**0.0467**
Garden	HR@50	0.0965	0.0946	0.1525	0.1593	0.1573	0.1486	0.1632	0.1727	**0.2177**
	MRR	0.0105	0.0333	0.0408	0.0255	0.0266	0.0257	0.0277	0.0304	**0.0459**
Instant Video	HR@50	0.2350	0.1650	0.2120	0.2430	0.1910	0.2570	0.1620	0.2020	**0.2790**
	MRR	0.0376	0.0426	0.0541	0.0414	0.0367	0.0441	0.0275	0.0351	**0.0634**
LastFM	HR@50	0.3659	0.1682	0.2808	0.3733	0.2503	0.3785	0.1756	0.0914	**0.3901**
	MRR	0.1062	0.0645	0.0869	0.1209	0.1276	0.1147	0.0343	0.0141	**0.1312**
MovieTweetings	HR@50	0.1749	0.3314	0.3417	0.3105	0.3286	0.2755	0.3332	0.3401	**0.3602**
	MRR	0.0231	0.0700	0.0674	0.0534	0.0476	0.0421	0.0585	0.0612	**0.0811**

Implementation Details

The former seven baselines were implemented in C++ based on (He *et al.* 2017a). We implemented Caser and our model with Tensorflow[3]. All experiments were conducted on a NVIDIA TITAN X Pascal GPU. For all baselines, Hyperparameters are tuned with grid search with validation set.

Since we adopt adaptive gradient optimizer for AttRec, the learning rate of AttRec for all datasets is set to 0.05 without further tuning. The number of latent dimensions d of all latent vectors (U, V, X) of AttRec and all other baselines (if exists.) is set to 100 for fair comparison. Note that the impact of d is also discussed in the following section. Due to the high sparsity of Amazon, LastFm and Movietweetings datasets, we use norm clipping to replace the ℓ_2 regularization for X, V, U. Weight matrices of nonlinear layer in self-attention module are regularized with ℓ_2 norm. Regularization rate λ is tuned amongst $\{0.1, 0.01, 0.001, 0.0001\}$. Dropout rate is tuned amongst $\{0, 0.3, 0.5, 0.7\}$. The weight factor ω is tuned amongst $\{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$. The length of sequence L is set to 5 for Movielens, 3 for MovieTweetings, and 2 for all other datasets. The target length T is set to 3 for Movielens and 1 for all other datasets. The margin γ of hinge loss is fixed to 0.5 for all datasets.

[3]https://www.tensorflow.org/

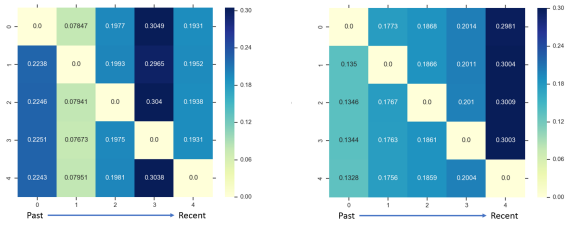
Performance Comparison

Table 2 reports the experimental results of the 8 baselines and our model on 12 benchmark datasets. Observe that **AttRec** always achieve the best performance on all datasets. This ascertains the effectiveness of the proposed approach. Notably, the performance gains over the strongest baselines is reasonably large in terms of both prediction accuracy and ranking quality. Our model performs well not only on dense datasets like Movielens but also on sparse datasets such as Amazon or MovieTweetings. The sequential intensity of sparse datasets is usually much lower than that of dense datasets.

Additionally, we make several observations on the comparison baselines. Markov Chain based models (FPMC and MC) achieve consistent performance on both dense and sparse data. TransRec and PRME, on the other hand, seems to be underperforming on some datasets. One important assumption of PRME and TransRec is that user's next item is only influenced by her latest action. This assumption might hold on sparse data as the interactions are extremely discrete along time but may not hold when user interacts with the system frequently. TransRec overcomes this shortcoming to

Table 3: HR@50 of AttRec with and without Self-Attention.

Dataset	100K	1M	Garden	Music
w/ Self-Att	0.5273	0.5223	0.2177	0.2205
w/o Self-Att	0.5015	0.5045	0.2026	0.2022



(a) User ID: 538 (Male) (b) User ID: 888 (Male)

Figure 3: Attention weights of two randomly selected users for prediction. The color scale represents the strength of the attention weights. Item information is not listed due to limited space. (*Best viewed in color.*)

some extent by introducing user specific relation vectors as intermediary. This claim can be demonstrated by HRM as it usually outperforms PRME, and their is no clear winner between HRM and TransRec. Caser achieves satisfactory performance on Movielens and MovieTweetings, but performs poorly on sparse datasets. In addition, our model also outperforms LSTM+ largely. As a final recapitulation, AttRec consistently outperforms all baselines by a wide margin, which clearly answers **RQ1**.

Model Analysis and Discussion

In this section, we dive into an in-depth model analysis, aiming to further understand behaviour of our model to answer the **RQ2**.

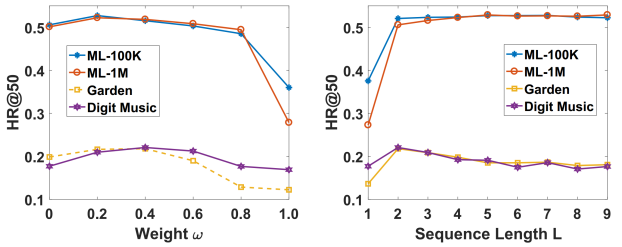
Impact of Self-Attention. Although we can infer the efficacy of self-attention implicitly from Table 2, we would like to verify the effectiveness of the self-attention mechanism explicitly. We remove the self-attention module from AttRec and replace m_t^u with the mean of X_t^u, that is: $m_t^u = \frac{1}{L}\sum_{l=1}^{L} X_{tl}^u$.

Table 3 shows the comparison between with and without self-attention. We observe that with self-attention indeed improves the performance. From both Tables 2 and Table 3, we find that even without self-attention, our model can still beat all baselines on these four datasets. This also justifies the method we use for preference modelling. Furthermore, in order to study the effect of self-attention, we visualize the self-attention matrix on Movielens 100K in Figure 3.

We make two observations from the results. First, the attention matrix is column distinguishable even they are unintentionally trained to achieve this. Each column represents the importance and weight for the corresponding action. Intuitively, the self-attention matrix allows us to inspect how much an action contributes to the overall short-term intent

Table 4: HR@50 of AttRec with different aggregation methods.

Dataset	ML-100K	ML-1M	Garden	Digit Music
Mean	0.5273	0.5223	0.2177	0.2205
Sum	0.4883	0.5201	0.2046	0.1908
Max	0.5254	0.5229	0.1892	0.1925
Min	0.5244	0.5267	0.1525	0.1548



(a) Effects of the weight ω (b) Effects of the sequence length L

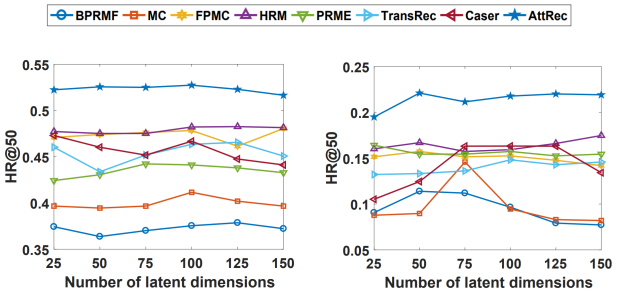
Figure 4: Effects of the weight ω and effects of the sequence length L on four datasets.

representation. Second, AttRec will not simply put more emphasis on recent actions, but learns to attend over previous actions with different weights automatically. For example, the most recent items are given higher weights for user "888", while higher weights are assigned to the first and fourth items for user "538". Evidently, analyzing the attention weights could lead to benefits in interpretability.

Impact of Aggregation Method. As aforementioned, we can use different aggregation strategies to get the representation of user short-term intents. Here, we explored four types of strategies to check their suitability. Table 4 shows the results of using different aggregation methods. We observe that "mean" achieves desirable performance on both sparse and dense datasets. The other three aggregation methods seem to be underperforming especially on sparse datasets. This is reasonable as m_t^u shall influence the embedding of the next item (X_{t+1}^u). Using mean aggregation could retain more information.

Impact of weight ω. The parameter ω controls the contribution of short-term and long-term effects. Observe from Figure 4a, considering only short term intents ($\omega = 0$) usually get better performance than considering only long-term preference ($\omega = 1.0$). Setting ω to a value between 0.2 and 0.4 is more preferable, which also indicates that short-term intent play a more important role in sequential recommendation. Additionally, the impact of ω also reflects the strength of sequential signal in the datasets. Datasets with higher sequential signal (dense datasets such as Movielens, detail sequential intensity evaluation can be found in (Tang and Wang 2018)) hit their best performance with a lower ω value.

Impact of sequence length L. Figure 4b shows the impact of the sequence length L. We observe that the proper L is



(a) Movielens 100K (b) Amazon Garden

Figure 5: HR@50 (y-axis) vs. the number of latent dimensions (x-axis) on Movielens 100K and Amazon Garden.

Table 5: Runtime comparison (second/epoch) between AttRec and Caser on four datasets.

Dataset	ML-100K	ML-1M	Garden	Digit Music
AttRec	1.315	15.429	0.169	1.525
Caser	1.309	15.328	0.120	0.956

highly dependent on the density of datasets. On MovieLens datasets where average number of actions per user is greater than a hundred, setting L to a larger value is beneficial to the performance. However, L should be set to a small value on sparse datasets, which is reasonable as increasing L will results in training sample decrease. Note that self-attention is capable of drawing dependencies between distant positions (Vaswani *et al.* 2017), which theoretically allows learning on very lengthy sequence.

Impact of Number of Latent Dimensions. Figure 5 shows the HR@50 for various d while keeping other optimal hyper-parameters unchanged. We make three observations from this figure. First, our model consistently outperforms all other baselines on all latent dimensions. Secondly, a larger latent dimension does not necessarily leads to better model performance. Overfitting could be a possible reason. Third, some models such as MC and Caser perform unstably, which might limit their usefulness.

Model Efficiency. Table 5 shows the runtime comparison with Caser. Other baselines are not listed here as the implementation cannot leverage the computation power of GPU. Experiments were run with batch size of 1000 on a NVIDIA TITAN X Pascal GPU. We observe that AttRec only incurs a small computational cost over Caser. This cost might be caused by the use of dual embedding and Euclidean distance calculation. Since both Caser and AttRec are trained in a pairwise manner, the difference in convergence speed is subtle. For example, it takes fewer than 30 epochs for AttRec to achieve its best performance on Movielens 100K.

Conclusion

In this paper, we proposed AttRec, a novel self-attention based metric learning approach for sequential recommendation. Our model incorporates both user short-term intent and long-term preference to predict her next actions. It utilizes the self-attention to learn user's short-term intents from her recent actions. Analysis shows that AttRec could accurately capture the importance of user recent actions. In addition, we generalize self attention mechanism into metric learning methodology for sequence prediction task, which has good effectiveness on sequential recommendation.

References

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, and et al. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795, 2013.

Jingyuan Chen, Hanwang Zhang, Xiangnan He, and et al. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *SIGIR*, pages 335–344. ACM, 2017.

Xu Chen, Hongteng Xu, Yongfeng Zhang, and et al. Sequential recommendation with user memory networks. In *WSDM*, pages 108–116. ACM, 2018.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

Shanshan Feng, Xutao Li, Yifeng Zeng, and et al. Personalized ranking metric embedding for next new poi recommendation. In *IJCAI*, IJCAI'15, pages 2069–2075. AAAI Press, 2015.

Yuyun Gong and Qi Zhang. Hashtag recommendation using attention-based convolutional neural network. In *IJCAI*, pages 2782–2788, 2016.

Ruining He and Julian McAuley. Fusing similarity models with markov chains for sparse sequential recommendation. In *ICDM*, pages 191–200. IEEE, 2016.

Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*, pages 507–517, 2016.

Ruining He, Wang-Cheng Kang, and Julian McAuley. Translation-based recommendation. In *RecSys*, pages 161–169. ACM, 2017.

Xiangnan He, Lizi Liao, Hanwang Zhang, and et al. Neural collaborative filtering. In *WWW*, pages 173–182, 2017.

Ruining He, Wang-Cheng Kang, and Julian McAuley. Translation-based recommendation: A scalable method for modeling sequential behavior. In *IJCAI*, pages 5264–5268, 2018.

X. He, Z. He, J. Song, and et al. Nais: Neural attentive item similarity model for recommendation. *TKDE*, pages 1–1, 2018.

Cheng-Kang Hsieh, Longqi Yang, Yin Cui, and et al. Collaborative metric learning. In *WWW*, pages 193–201, 2017.

Yehuda Koren. Collaborative filtering with temporal dynamics. In *SIGKDD*, pages 447–456. ACM, 2009.

Jing Li, Pengjie Ren, Zhumin Chen, and et al. Neural attentive session-based recommendation. In *CIKM*, pages 1419–1428. ACM, 2017.

Julian McAuley, Christopher Targett, and et al. Image-based recommendations on styles and substitutes. In *SIGIR*, pages 43–52. ACM, 2015.

Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. Sequence-aware recommender systems. *arXiv preprint arXiv:1802.08452*, 2018.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and et al. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461. AUAI Press, 2009.

Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *WWW*, pages 811–820. ACM, 2010.

Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*, pages 565–573. ACM, 2018.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Latent relational metric learning via memory-based attention for collaborative ranking. In *WWW*, pages 729–739, 2018.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. Couplenet: Paying attention to couples with coupled attention for relationship recommendation. *arXiv preprint arXiv:1805.11535*, 2018.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Multi-pointer co-attention networks for recommendation. *arXiv preprint arXiv:1801.09251*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, and et al. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

Pengfei Wang, Jiafeng Guo, Yanyan Lan, and et al. Learning hierarchical representation model for next basket recommendation. In *SIGIR*, pages 403–412. ACM, 2015.

Chao-Yuan Wu, Amr Ahmed, Alex Beutel, and et al. Recurrent recommender networks. In *WSDM*, pages 495–503. ACM, 2017.

Liang Xiong, Xi Chen, Tzu-Kuo Huang, and et al. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 211–222. SIAM, 2010.