Signed Distance-based Deep Memory Recommender

Thanh Tran
Department of Computer Science
Worcester Polytechnic Institute
Massachusetts, USA
tdtran@wpi.edu

Xinyue Liu
Department of Computer Science
Worcester Polytechnic Institute
Massachusetts, USA
xliu4@wpi.edu

Kyumin Lee
Department of Computer Science
Worcester Polytechnic Institute
Massachusetts, USA
kmlee@wpi.edu

Xiangnan Kong
Department of Computer Science
Worcester Polytechnic Institute
Massachusetts, USA
xkong@wpi.edu

ABSTRACT

Personalized recommendation algorithms learn a user's preference for an item by measuring a distance/similarity between them. However, some of the existing recommendation models (e.g., matrix factorization) assume a linear relationship between the user and item. This approach limits the capacity of recommender systems, since the interactions between users and items in real-world applications are much more complex than the linear relationship. To overcome this limitation, in this paper, we design and propose a deep learning framework called *Signed Distance-based Deep Memory Recommender*, which captures non-linear relationships between users and items *explicitly* and *implicitly*, and work well in both general recommendation task and shopping basket-based recommendation task. Through an extensive empirical study on six real-world datasets in the two recommendation tasks, our proposed approach achieved significant improvement over ten state-of-the-art recommendation models.

KEYWORDS

Memory recommender; signed distance; metric-based attention.

ACM Reference Format:
Thanh Tran, Xinyue Liu, Kyumin Lee, and Xiangnan Kong. 2019. Signed Distance-based Deep Memory Recommender. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3308558.3313460

1 INTRODUCTION

Recommender systems [1] have been deployed in many online applications such as e-commerce, music/video streaming services, social media, *etc.* They have played a vital role for users to explore new items and for companies to increase their revenues. Most of recommendation algorithms model user preferences and item properties based on observed interactions (*e.g.,* clicks, reviews, ratings) between users and items [20, 21, 30]. In a perspective, we can view most of the recommendation models as a measurement of similarity or distance between a user and an item. For instance, the well known latent factor (i.e., matrix factorization) models [19] usually employ an inner product function to approximate the similarity between the user and the item. Although the latent factor models achieved competitive performance in some datasets, they did not correctly capture complex (i.e., non-linear) relationships between users and items because the inner product function follows limited linear nature.

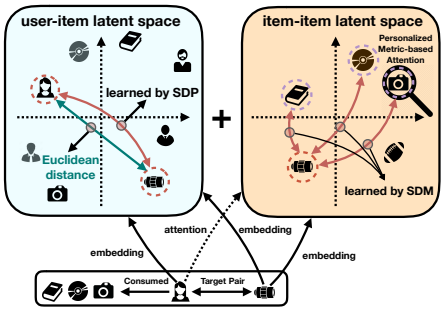


Figure 1: We consider a recommender as a signed distance approximator, and decompose the signed distance between a user and an item into two parts: the left box learns an explicitly signed distance between the user and item (i.e., the *camera lens*), the right box learns an implicitly signed distance between the user and the item via the user's recently consumed items (i.e., the *book*, *CD* and *camera*). Our novel personalized metric-based soft attention is applied to the consumed items to optimize their contributions to the output signed distance score. Then the two parts are combined to obtain a final score. Most of linear latent factor models are equivalent to simply measuring the linear Euclidean distance in the user-item latent space (shown as the green line).

Existing recommendation algorithms faced difficulties in finding good kernels for different data patterns [30], only focused on user-item latent space without considering the item-item latent space together [12, 24, 25, 43, 57], or required additional auxiliary information (e.g., item description, music content, reviews) [4, 17, 29, 31, 53]. To overcome the drawbacks, in this paper we aim to propose and build a deep learning framework to learn a non-linear relationship

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313460

between a user and a target item by measuring a distance from the observed data. In particular, we propose *Signed Distance-based Deep Memory Recommender* (SDMR), which captures non-linear relationship of the user and item *explicitly* and *implicitly*, combines *explicitly* and *implicitly* measured relationship to produce a final distance score for the recommendation, and performs well in both general recommendation task and shopping basket-based recommendation task.

SDMR internally combines two signed distances, each of which is measured by our proposed *Signed Distance-based Perceptron* (SDP) and *Signed Distance-based Memory Network* (SDM). On one hand, SDP explicitly measures a non-linear signed distance between the user and the item. Many existing models [13, 16] rely on a pre-defined metric such as Euclidean distance (the green line in Figure 1) which is much more limited than the customized non-linear signed distance learned from the data (the red curves in Figure 1). On the other hand, SDM implicitly measures a non-linear signed distance between the user and the item via the user's recently consumed items. SDM is similar to the item neighborhood-based recommender [36, 42] in nature. However, it is more advanced in several aspects, as shown in the right side of Figure 1. First, SDM only focuses on a set of recently consumed items of the target user (*e.g.,* the *book*, *CD* and *camera* in Figure 1) as context items. Second, it employs additional memories to learn a novel personalized metric-based attention on the consumed items. The goal of our proposed attention is to compute weights of each consumed item *w.r.t.* the target item (i.e., the *camera lens*). In the example, the attention module assigns higher weights on the *camera* and lower weights on the *book* and *CD*. Unlike our approach, most of the existing neighborhood-based models consider contribution of consumed items to the target item equally, leading to suboptimal results. Last but not the least, we update the attention weights via a gated multi-hop to build a long-term memory within SDM. This multi-hop design helps refine our attention module and produces more accurate attentive scores.

The contributions of this work are summarized as follows:

- We design a deep learning framework which can tackle both general recommendation task and shopping basket-based recommendation task.
- We propose SDMR that combines two signed distance scores internally measured by SDP and SDM, which capture non-linear relationship between a user and an item explicitly and implicitly.
- To better balance the weights among consumed items of the user, we propose a novel multi-hop memory network with a personalized metric-based attention mechanism in SDM.
- Extensive experiments on six datasets in two different recommendation tasks demonstrate the effectiveness of our proposed methods against ten baselines.

2 RELATED WORK

Latent Factor Models (LFM) have been extensively studied in the literature, which include Matrix Factorization [16], Bayesian Personalized Ranking [39], fast matrix factorization for implicit feedbacks (eALS) [13], *etc.* Despite their success, LFM suffer from several limitations. First, LFM overlook associations between the user's previously consumed items and the target item (e.g. mobile phones and phone cases). Second, LFM usually rely on inner product function, whose linearity limits the capability of modeling complex user-item interactions. To address the second issue, several non-linear latent factor models have been proposed, with the help of Gaussian process [23] or kernels [30, 62]. However, they either require expensive hyper-parameter tuning or face difficulties in finding good kernels for different data patterns.

Neighborhood-based models [36, 42] are usually based on the principle that similar users prefer similar items. The problem turns into finding the neighbors of a user or an item based on a pre-defined distance/similarity metric, such as *cosine vector similarity* [3, 22], *Person Correlation similarity* [6], *etc.* The recommendation quality highly depends on a chosen metric, but finding a good pre-defined metric is usually very challenging. Furthermore, these models are also sensitive to the selection of neighbors. Our proposed SDM is similar to neighborhood-based models in nature, but it exploits a novel personalized metric-based attention for assigning attentive weights to context items. Therefore, our approach is more robust and less sensitive than conventional neighborhood-based models.

NeuMF [12] is a neural network that generalizes matrix factorization via Multi Layer Perceptron (MLP) for learning non-linear interaction functions. Similarly, some other works [24, 25, 43, 57] substitute MLP with auto-encoder architecture. It is worth noting that all these approaches are limited by only considering the user-item latent space, and overlook the correlations in the item-item latent space. Besides, some deep learning based works [32, 34, 45, 51] employ auxiliary information such as item description [17], music content [53], item visual features [4, 29], reviews [31] to address the cold-start problem. However, this auxiliary information is not always available, and it limits their applicability in many real-world systems. Another line of works use deep neural networks to model temporal effects of consumed items [14, 37, 49, 56]. Although our proposed methods do not explicitly consider the temporal effects, SDM utilizes the time information to select a set of recently consumed items as the context items of the target item.

The most closely related work to our work is recently proposed (*Collaborative Memory Network* (CMN) [7]). In this work, Memory Network [48] is adapted to measure similarities between users and user neighbors. Key differences between our work and CMN are as follows: (i) First, we follow an item neighborhood based design, whereas CMN follows a user neighborhood based design. The prior work showed that item neighborhood based models slightly outperformed user neighbor based models [27, 42]; (ii) Second, our proposed SDM model uses our proposed personalized metric-based attention mechanism and produces signed distance scores as output, whereas CMN exploited a traditional inner product based attention; (iii) Third, we use a gated multi-hop architecture [28], which was shown to perform better than the original multi-hop design [48].

3 PROBLEM STATEMENT

In this section, we describe two recommendation problems: (i) general recommendation task; and (ii) shopping basket-based recommendation task. In following sections, we focus on solving them.

General recommendation task: Given a whole item set $V = \{v_1, v_2, ..., v_{|V|}\}$, and a whole user set $U = \{u_1, u_2, ..., u_{|U|}\}$. Each user $u_i \in U$ may consume several items $\{v_{i1}, v_{i2}, ..., v_{ik}\}$ in V,

denoted as a set of context items c. In this task, given previously consumed items of a user u_i, a recommendation model predicts a next target item v_j that u_i may prefer, denoting this task as estimating $P(u_i, v_j|c)$. Note that some existing works assume independent relationships between v_j and context items in the set $\boldsymbol{c}$, leading to $P(u_i, v_j|c) = P(u_i, v_j)$ [12, 13]. In our work, we model the u_i's preference on v_j in two steps: (i) an explicit preference of u_i on v_j in a signed distance based perceptron, and (ii) an implicit preference of u_i on v_j via summing attentive effects of context items toward target item v_j in a signed distance based memory network.

Shopping Basket-based recommendation task: This problem is based on the fact that users go shopping offline/online and add some items into a basket/cart together. Each shopping basket/cart is seen as a transaction, and each user may shop once or multiple times, leading to one or multiple transactions. Let $T^{(u)} = \{t_1, t_2, ..., t_{|T^{(u)}|}\}$ as a set of the user u's transactions, where $|T^{(u)}|$ denotes the number of user u's transactions. Each transaction $t_i = \{v_1, v_2, ..., v_{|t_i|}\}$ consists of several items in the whole item set V. In this problem, it is assumed that all the items in t_i are inserted into the same basket at the same time, ignoring the actual order of the items being inserted and considering t_i's transaction time as each item's insertion time. Given a target item $v_j \in t_i$, the rest of the items in t_i will be seen as the context items of v_j, denoted as c (i.e. $c = t_i \backslash \{v_j\}$). Then, given the set of context items c, a recommendation model predicts a conditional probability $P(u, v_j|c)$, which is interpreted as the conditional probability that u will add the item v_j into the same basket with the other items c.

Both of the recommendation tasks above are popular in the literature [8, 12, 37, 40]. The *general recommendation task* differs from the *shopping basket-based recommendation task* because there is no specific context items of the target item in the *general recommendation task*. Note that the two tasks are *personalized* recommendation problems. In fact, there are *non-personalized* recommendation problems such as *session-based recommendation* [14], where users (i.e. user IDs) are not available in transactions. However, in this paper, we focus on *personalized* recommendation tasks because they are more preferred in the literature [8, 37, 40].

4 PROPOSED METHODS

Our proposed *Signed Distance-based Deep Memory Recommender* (SDMR) consists of two major components: *Signed Distance-based Perceptron* (SDP) and *Signed Distance-based Memory network* (SDM). We first describe an overview of our models as follows:

- Given a target user i and a target item j as two one-hot vectors, we pass the two vectors through the user and item embedding spaces to get user embedding u_i and item embedding v_j.
- On one hand, our proposed *Signed Distance-based Perceptron* (SDP) will measure a signed distance score between u_i and v_j by a multi-layer perceptron network.
- On the other hand, given target user i, target item j, and the user i's recently consumed context items s as the input, our *Signed Distance-based Memory network* (SDM) will measure a signed distance score between user i and item j via attentive distances between context items s and target item j.

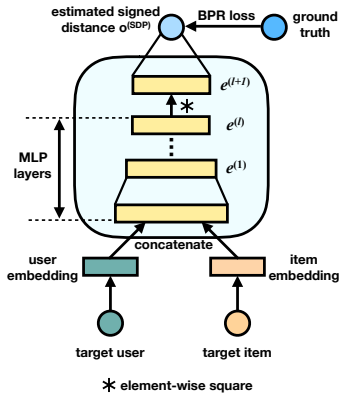


Figure 2: The illustration of our SDP model.

- Then, the *Signed Distance-based Deep Memory Recommender* (SDMR) model will measure a total distance between user i and item j by learning a combination of SDP and SDM. The smaller the total distance is, the more likely user i will consume item j.

Next, we describe SDP, SDM, and SDMR in detail.

4.1 Signed Distance-based Perceptron (SDP)

We first propose *Signed Distance-based Perceptron* (SDP) that *explicitly* learns a signed distance between a target user i and a target item j. An illustration of SDP is shown in Figure 2. Let the embedding of a target user i be $\boldsymbol{u}_i \in \mathbb{R}^d$, and the embedding of a target item j be $\boldsymbol{v}_j \in \mathbb{R}^d$, where d is the number of dimensions in each embedding. First, SDP takes a concatenation of these two embeddings as the input and proceeds as follows:

$$\boldsymbol{e}^{(1)} = f_1(\mathbf{W}^{(1)} \begin{bmatrix} \boldsymbol{u}_i \\ \boldsymbol{v}_j \end{bmatrix} + \boldsymbol{b}^{(1)}) \tag{1}$$

$$\boldsymbol{e}^{(2)} = f_2(\mathbf{W}^{(2)} \boldsymbol{e}^{(1)} + \boldsymbol{b}^{(2)}) \tag{2}$$

$$\dots \tag{3}$$

$$\boldsymbol{e}^{(\ell)} = f_\ell(\mathbf{W}^{(\ell)} \boldsymbol{e}^{(\ell-1)} + \boldsymbol{b}^{(\ell)}) \tag{4}$$

$$\boldsymbol{e}^{(\ell+1)} = square(\boldsymbol{e}^{(\boldsymbol{\ell})}) \tag{5}$$

$$o^{(SDP)} = \boldsymbol{w}^{(o)\top} \boldsymbol{e}^{(\ell+1)} + \boldsymbol{b}^{(o)} \tag{6}$$

where $f_l(\cdot)$ refers to a non-linear activation function at the layer l^{th} (e.g. `sigmoid`, `ReLu` or `tanh`), and $square(\cdot)$ denotes an element-wise square function (e.g $square([2, 3]) = [6, 9]$). Through experimental results, we choose `tanh` as the activation function because it yields slightly better results than `ReLu`. From now on, we will use $f(\cdot)$ to denote the `tanh` function. It can be easily observed that Eq. (1) – (4) form a trivial Multi-Layer Perceptron (MLP) network, which is a popular design [12, 60] to learn a complex and non-linear interaction between user embedding $\boldsymbol{u}_i$ and item embedding $\boldsymbol{v}_j$. Our new design starts at Eq. (5) – Eq. (6). In Eq. (5), we apply the element-wise squared function $square(\cdot)$ to the output vector $\boldsymbol{e}^{(l)}$ of the MLP and obtain a new output vector $\boldsymbol{e}^{(l+1)}$. Next, in Eq. (6), we use a fully connected layer $\boldsymbol{w}^{(o)}$ to combine different dimensions in $\boldsymbol{e}^{(l+1)}$ and yields a final distance value $o^{(SDP)}$. Our idea of using $\boldsymbol{w}^{(o)}$ in here is that after applying the element-wise square function $square(\cdot)$ in Eq. (5), all the dimensions in $\boldsymbol{e}^{(l+1)}$ will be

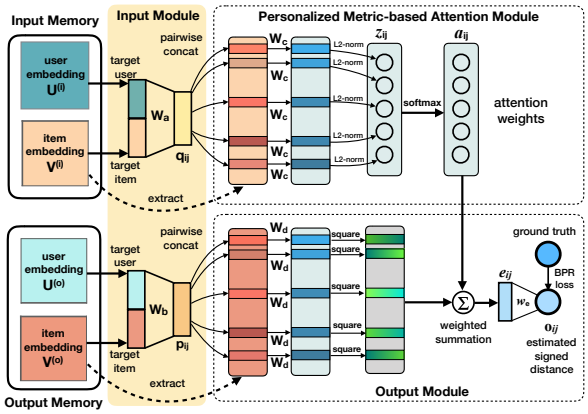


Figure 3: The illustration of a single-hop SDM, which consists of a memory module, an input module, an attention module, and an output module.

non-negative. Thus, we consider each dimension of $\boldsymbol{e}^{(l+1)}$ as a distance value. The edge weights $\boldsymbol{w}^{(o)}$ will then be used to combine those distant dimensions to provide a more fine-grained distance.

We note that SDP can be reduced to a squared Euclidean distance with the following setting: at Eq. (1), $\mathbf{W}^{(1)} = [\mathbb{1}, -\mathbb{1}]$ with $\mathbb{1}$ denotes an identity matrix and so $\mathbf{W}^{(1)} \begin{bmatrix} \boldsymbol{u}_i \\ \boldsymbol{v}_j \end{bmatrix} = \boldsymbol{u}_i - \boldsymbol{v}_j$; the activation $f(\cdot)$ is an identity function; the number of MLP layers $\ell = 1$; the edge-weights layer at Eq. (6): $\boldsymbol{w}^{(o)} = \mathbf{1}$ (e.g. the all-ones matrix), bias $\boldsymbol{b}^{(o)} = \mathbf{0}$. Note that if $\boldsymbol{w}^{(o)}$ in Eq. (6) is an all-negative layer, it will yield a negative value, which we name as a signed distance[1] score. If we see each user $\boldsymbol{i}$ as a point in multi dimensional space, and the user's preference space is defined by a boundary Ω, we can interpret this signed distance score as follows: When the item $\boldsymbol{j}$ is out of the user $\boldsymbol{i}$'s preference boundary Ω, the distance $d(\boldsymbol{i}, \boldsymbol{j})$ between them is positive (i.e. $d(\boldsymbol{i}, \boldsymbol{j}) > 0$) and it reflects that user $\boldsymbol{i}$ does not prefer item $\boldsymbol{j}$. When the distance between user $\boldsymbol{i}$ and item $\boldsymbol{j}$ is shortened and $\boldsymbol{j}$ is right on the boundary Ω, the distance between them is zero and it indicates user $\boldsymbol{i}$ likes item $\boldsymbol{j}$. As $\boldsymbol{j}$ is coming inside Ω, the distance between them becomes negative and reflects a higher preference of user $\boldsymbol{i}$ on item $\boldsymbol{j}$. In short, we can see SDP as a signed distance function, which could learn a complex signed distance between a user and an item via a MLP architecture with non-linear activations and an element-wise square function $square(\cdot)$. In the recommendation domain, the signed distances will provide more fine-grained distance values, thus, reflecting users' preferences on items more accurately.

4.2 Signed Distance-based Memory Network (SDM)

We propose a multi-hop memory network, *Signed Distance-based Memory network* (SDM), to model *implicit* preference of a user on the target item via the user's previously consumed items (i.e., context items). The implicit preference is represented as a signed distance. First, we describe a single-hop SDM, and then describe how to extend it into a multi-hop design. Following the traditional architecture of a memory network [28, 48, 58], our proposed single-hop SDM has four main components: a memory module, an input module, an attention module, and an output module. The overview of SDM's architecture is presented in Figure 3. We will go into details of each SDM's module as follows:

4.2.1 Memory Module: We maintain two memories called input memory and output memory. The input memory contains two embedding matrices $\mathbf{U}^{(i)} \in \mathbb{R}^{M \times d}$ and $\mathbf{V}^{(i)} \in \mathbb{R}^{N \times d}$, where M and N are the number of users and the number of items in the system, respectively. d denotes the embedding size of each user and each item. Similarly, the output memory also contains two embedding matrices $\mathbf{U}^{(o)} \in \mathbb{R}^{M \times d}$ and $\mathbf{V}^{(o)} \in \mathbb{R}^{N \times d}$. As shown in Figure 3, the input memory will be used to calculate attention weights of a user's consumed items (i.e., context items), whereas the output memory will be used to measure a final signed distance between the target user and the target item via the user's context items.

Given a target user i, a target item j and a set of user i's consumed items as context items $\mathcal{T}_j^{\,i}$, the output of this module is the embeddings of user i, item j, and all context items $k \in \mathcal{T}_j^{\,i}$: $(\boldsymbol{u}_i, \boldsymbol{v}_j, <\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_k>)$. Since this module has a separated input memory and output memory, we obtain $(\boldsymbol{u}_i^{(i)}, \boldsymbol{v}_j^{(i)}, <\boldsymbol{v}_1^{(i)}, \boldsymbol{v}_2^{(i)}, ..., \boldsymbol{v}_k^{(i)}>)$ as the output of the input memory, and $(\boldsymbol{u}_i^{(o)}, \boldsymbol{v}_j^{(o)}, <\boldsymbol{v}_1^{(o)}, \boldsymbol{v}_2^{(o)}, ..., \boldsymbol{v}_k^{(o)}>)$ as the output of the output memory. It is obvious that $\boldsymbol{u}_i^{(i)}$ is the i-th row of $\mathbf{U}^{(i)}$, $\boldsymbol{v}_j^{(i)}$ and $\boldsymbol{v}_k^{(i)}$ are the corresponding j-th and k-th row of $\mathbf{V}^{(i)}$. A similar explanation is applied to $\boldsymbol{u}_i^{(o)}$ $\boldsymbol{v}_j^{(o)}$, and $\boldsymbol{v}_k^{(o)}$.

4.2.2 Input Module: The goal of the input module is to form a non-linear combination between the target user embedding and the target item embedding. Given the target user embedding $\boldsymbol{u}_i^{(i)}$ and the target item embedding $\boldsymbol{v}_j^{(i)}$ from the input memory in the memory module, following the widely adopted design in multimodal deep learning work [47, 61], the input module simply concatenates the two embeddings, and then applies a fully connected layer with a non-linear activation $f(\cdot)$ (i.e. `tanh` function) to obtain a coherent hidden feature vector as follows:

$$\boldsymbol{q}_{ij} = f\left(\mathbf{W}_a \begin{bmatrix} \boldsymbol{u}_i^{(i)} \\ \boldsymbol{v}_j^{(i)} \end{bmatrix} + \boldsymbol{b}_a\right) \tag{7}$$

where $\mathbf{W}_a \in \mathbb{R}^{d \times 2d}$ is the weights of input module. Note that $q_{ij} \in \mathbb{R}^d$ can be seen as a query embedding in Memory Network [48].

Similarly, if the inputs of the input module are the target user embeddings $\boldsymbol{u}_i^{(o)}$ and the target item embeddings $\boldsymbol{v}_j^{(o)}$ from the output memory, we can form a non-linear combination between $\boldsymbol{u}_i^{(o)}$ and $\boldsymbol{v}_j^{(o)}$ (i.e. an output query), denoted as $\boldsymbol{p}_{ij}$, as follows:

$$\boldsymbol{p}_{ij} = f\left(\mathbf{W}_b \begin{bmatrix} \boldsymbol{u}_i^{(o)} \\ \boldsymbol{v}_j^{(o)} \end{bmatrix} + \boldsymbol{b}_b\right) \tag{8}$$

4.2.3 Attention Module: The goal of the attention module is to assign attentive scores to different context items (or candidates) given the combined vector (or a query) $\boldsymbol{q}_{ij}$ of the target user i and target item j obtained in Eq. (7). First, we calculate the squared $\mathcal{L}2$ distance between $\boldsymbol{q}_{ij}$ and each candidate item $\boldsymbol{v}_k^{(i)}$ as follows:

[1] https://en.wikipedia.org/wiki/Signed_distance_function

$$z_{ijk} = \left\| f\left(\mathbf{W}_c \begin{bmatrix} \boldsymbol{q}_{ij} \\ \boldsymbol{v}_k^{(i)} \end{bmatrix} + \boldsymbol{b}_c\right)\right\|_2^2 \quad (9)$$

where $||\cdot||_2$ refers to the $\mathcal{L}2$ distance (or Euclidean distance), which is widely used in previous works to measure similarity among items [8] or between users and items [15]. To better understand our intuition in Eq. (9), we will break it into smaller parts and explain them. First, similar to the intuition of Eq. (7), we have $f\left(\mathbf{W}_c \begin{bmatrix} \boldsymbol{q}_{ij} \\ \boldsymbol{v}_k^{(i)} \end{bmatrix} + \boldsymbol{b}_c\right)$ component to define a non-linear combination between the input query $\boldsymbol{q}_{ij}$ and each context item embeddings $\boldsymbol{v_k}^{(i)}$. Then, $||\cdot||_2^2$ will measure the squared $\mathcal{L}2$ distance of the combined vector. It is worth to note that with a following setting: $\boldsymbol{W}_a = [\mathbf{0}, \mathbb{I}]$ where $\mathbb{I}$ refers to an identity matrix and $\mathbf{0}$ is an all-zeros matrix; $f(\cdot)$ is an identity function; $\boldsymbol{W}_c = [\mathbb{I}, -\mathbb{I}]$; bias terms $\boldsymbol{b}_a = \boldsymbol{b}_c = 0$. Then, in Eq. (7), $\boldsymbol{q}_{ij} = f\left(\mathbf{W}_a \begin{bmatrix} \boldsymbol{u}_i^{(i)} \\ \boldsymbol{v}_j^{(i)} \end{bmatrix} + \boldsymbol{b}_a\right) = \boldsymbol{v}_j^{(i)}$; in Eq. (9), $f\left(\mathbf{W}_c \begin{bmatrix} \boldsymbol{q}_{ij} \\ \boldsymbol{v}_k^{(i)} \end{bmatrix} + \boldsymbol{b}_c\right) = \boldsymbol{v}_j^{(i)} - \boldsymbol{v}_k^{(i)}$, and $z_{ijk} = ||(\boldsymbol{v}_j^{(i)} - \boldsymbol{v}_k^{(i)})||_2^2$, which simply generalizes a squared $\mathcal{L}2$ distance between the target item j and the context item k. Additionally, with another setting: $\boldsymbol{W}_a = [\mathbb{I}, -\mathbb{I}]$; $f(\cdot)$ is an identity function; $\boldsymbol{W}_c = [\mathbb{I}, \mathbb{I}]$; bias terms $\boldsymbol{b}_a = \boldsymbol{b}_c = 0$. Then, in Eq. (7), $\boldsymbol{q}_{ij} = f\left(\mathbf{W}_a \begin{bmatrix} \boldsymbol{u}_i^{(i)} \\ \boldsymbol{v}_j^{(i)} \end{bmatrix} + \boldsymbol{b}_a\right) = \boldsymbol{u}_i^{(i)} - \boldsymbol{v}_j^{(i)}$, in Eq. (9), $f\left(\mathbf{W}_c \begin{bmatrix} \boldsymbol{q}_{ij} \\ \boldsymbol{v}_k^{(i)} \end{bmatrix} + \boldsymbol{b}_c\right) = \boldsymbol{u}_i^{(i)} - \boldsymbol{v}_j^{(i)} + \boldsymbol{v}_k^{(i)}$, and $z_{ijk} = ||(\boldsymbol{v}_k^{(i)} + \boldsymbol{u}_i^{(i)} - \boldsymbol{v}_j^{(i)})||_2^2$, which simply generalizes a squared $\mathcal{L}2$ distance between the target item j and the context item k where the user i plays as a translator [9]. The two examples above show that our proposed design can learn a more generalized distance between target and context items.

The output squared $\mathcal{L}2$ distance in Eq. (9) will show how similar the target item j and the context item k are. The lower the distance score is, the more similar two items j and k are. Next, we use the Softmax function to normalize and obtain attentive score between j and k as follows:

$$a_{ijk} = \frac{exp(-z_{ijk})}{\sum_{p \in \mathcal{T}_j^i} exp(-z_{ijp})} \quad (10)$$

where $\mathcal{T}_j^i$ is the set of user $\boldsymbol{i}$'s neighborhood items. The **minus** sign in Eq. (10) is used to assign a higher attention score for a lower distance between two items (j, k).

We note that the $\mathcal{L}2$ distance (or Euclidean distance) satisfies four conditions of a metric [2]. While the crucial triangle inequality property of a metric was shown to provide a better performance compared to the inner product [15, 38, 46] in recommendation domains, to our best of knowledge, most of existing attention designs [2, 5, 26, 33, 44, 54, 59] adopted the inner product for measuring attentive scores. Hence, this proposed attention design is the **first attempt** to bring metric properties into the attention mechanism.

Similar to [50], we limit the number of considering context items by choosing the user $\boldsymbol{i}$'s $\boldsymbol{s}$ most recently consumed items before target item $\boldsymbol{j}$ as the context items of target item $\boldsymbol{j}$. Here, $\boldsymbol{s}$ can be selected via tuning with a development dataset. The soft attention vector containing attentive contribution scores of $\boldsymbol{s}$ context items toward the target item $\boldsymbol{j}$ of a user $\boldsymbol{i}$ is given as follows:

$$\boldsymbol{a}_{ij} = \left[a_{ij1}, \cdots, a_{ijs}\right]^T \quad (11)$$

4.2.4 Output Module: Given the attentive scores $\boldsymbol{a}_{ij}$ in Eq.(11) and the combined vector $\boldsymbol{p}_{ij} \in \mathbb{R}^d$ of the user embedding $\boldsymbol{u}_i^{(o)}$ and item embedding $\boldsymbol{v}_j^{(o)}$ from the output memory $\boldsymbol{U}^{(o)}$ and $\boldsymbol{V}^{(o)}$, the goal of this output module is to measure a total output distance $\boldsymbol{o}_{ij}^{(SDM)}$ between the output target item embeddings $\boldsymbol{v}_j^{(o)}$ and all the user $\boldsymbol{i}$'s output context item embeddings $\boldsymbol{v}_k^{(o)} (k \in T_j^i)$ using attention weights $\boldsymbol{a}_{ij}$ and the output query $\boldsymbol{p}_{ij}$ as follows:

$$o_{ij}^{(SDM)} = \boldsymbol{w}_e^\top \boldsymbol{e}_{ij} + b_e \quad (12)$$

where $\boldsymbol{e}_{ij} \in \mathbb{R}^d$ is calculated as follows:

$$\boldsymbol{e}_{ij} = \sum_{k \in \mathcal{T}_j^i} \boldsymbol{a}_{ijk} \times square\left(f\left(\mathbf{W}_d \begin{bmatrix} \boldsymbol{p}_{ij} \\ \boldsymbol{v}_k^{(o)} \end{bmatrix} + \boldsymbol{b}_d\right)\right) \quad (13)$$

In here, let $\boldsymbol{r}_{ijk} = f\left(\mathbf{W}_d \begin{bmatrix} \boldsymbol{p}_{ij} \\ \boldsymbol{v}_k^{(o)} \end{bmatrix} + \boldsymbol{b}_d\right)$. Similar to the previously discussed intuition in Eq (9), $\boldsymbol{r}_{ijk}$ is a flexible combination between $\boldsymbol{p}_{ij}$ and each output context item embeddings $\boldsymbol{v}_k^{(o)}$; $square(\cdot)$ is an element-wise squared function. Our idea in Eq. (12), (13) is similar to the idea in Eq. (5), (6) of the SDP model. First, in Eq. (13), each context item k will attentively contribute to the target item j via a squared Euclidean measure. Second, in Eq. (12), each non-negative dimension in e_{ij} will be considered as a distance dimension and we use an edge-weights layer $\boldsymbol{w}_e$ to combine them flexibly. When there is only one context item in $\mathcal{T}_j^i$, then in Eq. (13), the attention score $\boldsymbol{a}_{ijk}$=1.0, leading to $\boldsymbol{e}_{ij} = square(\boldsymbol{r}_{ijk})$, which is similar to Eq. (5). In this case, SDM will measure the distance between target item j and context item k in the same way as SDP model does. Note that Eq. (13) is similar to Eq. (6) so SDM can also learn a signed distance value, which also provides a more fine-grained distance compared to a general distance value.

4.2.5 Multi-hop SDM:. Inspired by previous work [48] where the multi-hop design helped to refine the attention module in Memory Network, we also integrate multiple hops to further extend our SDM model to build a deeper network (Figure 4). As the gated multi-hop

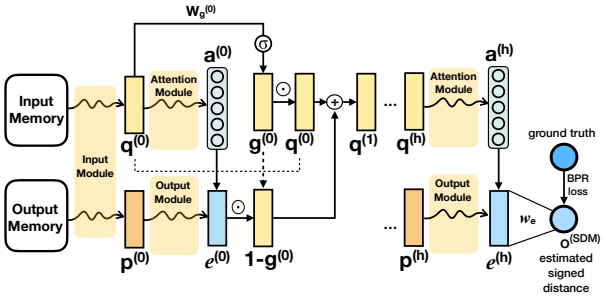


Figure 4: The illustration of our multi-hop SDM.

[2]https://en.wikipedia.org/wiki/Metric_(mathematics)

design [28] was shown to perform better than the original multi-hop design with a simple residual connection in [48], we employ this gated memory update from hop to hop as follows:

$$\boldsymbol{g}^{(h-1)} = \sigma(\mathbf{W}_g^{(h-1)}\boldsymbol{q}^{(h-1)} + \boldsymbol{b}_g^{(h-1)}) \quad (14)$$

$$\boldsymbol{q}^{(h)} = (1 - \boldsymbol{g}^{(h-1)}) \odot \boldsymbol{e}^{(h-1)} + \boldsymbol{g}^{(h-1)} \odot \boldsymbol{q}^{(h-1)} \quad (15)$$

where $\boldsymbol{q}^{(h-1)}$ is the input query embedding as shown in Eq. (7) at hop $h-1$, $\mathbf{W}_g^{(h-1)}$ and bias $\boldsymbol{b}_g^{(h-1)}$ are hop-specific parameters, σ is the sigmoid function, $e^{(h-1)}$ is the output of Eq. (13) at hop $h-1$, $\boldsymbol{q}^{(h)}$ is the input query embedding at the next hop h. So the attention could be updated at hop h accordingly using $\boldsymbol{q}^{(t)}$ as follows:

$$\alpha_{ijk}^{(h)} = \frac{exp(-z_{ijk}^{(h)})}{\sum_{p \in \mathcal{T}_j{}^i} exp(-z_{ijp}^{(h)})}, \text{ where } z_{ijk}^{(h)} = \left\| f\left(\mathbf{W}_c^{(h)} \begin{bmatrix} \boldsymbol{q}_{ij}^{(h)} \\ \boldsymbol{v}_k^{(i)} \end{bmatrix} + \boldsymbol{b}_c\right)\right\|_2^2 \quad (16)$$

The multi-hop architecture with gated design further refines the attention for different users based on the previous output from hop to hop. Hence, if the final hop is h then the SDM model with h hops, denoted as *SDM-h*, will use $\boldsymbol{a}_{ij}^{(h)}$ to yield a final signed distance score as follows:

$$o_{ij}^{(SDM-h)} = \boldsymbol{w}_e^\top \boldsymbol{e}_{ij}^{(h)} + b_e^{(h)} \quad (17)$$

where $\boldsymbol{e}_{ij}$ is calculated as:

$$\boldsymbol{e}_{ij}^{(h)} = \sum_{k \in \mathcal{T}_j{}^i} \boldsymbol{a}_{ijk}^{(h)} \times square\left(f\left(\mathbf{W}_d^{(h)} \begin{bmatrix} \boldsymbol{p}_{ij}^{(h)} \\ \boldsymbol{v}_k^{(o)} \end{bmatrix} + \boldsymbol{b}_d^{(h)}\right)\right) \quad (18)$$

Weight constraints in multi-hop SDM model: To save memory, we use the global weight constraint in multi-hop SDM. Particularly, input memory $\boldsymbol{U^{(i)}}, \boldsymbol{V^{(i)}}$ and output memory $\boldsymbol{U^{(o)}}, \boldsymbol{V^{(o)}}$ are shared among different hops. All the weights are shared from hop to hop $\boldsymbol{W}_a^{(1)} = \boldsymbol{W}_a^{(2)} = ... = \boldsymbol{W}_a^{(h)}; \boldsymbol{W}_b^{(1)} = \boldsymbol{W}_b^{(2)} = ... = \boldsymbol{W}_b^{(h)}; \boldsymbol{W}_c^{(1)} = \boldsymbol{W}_c^{(2)} = ... = \boldsymbol{W}_c^{(h)}; \boldsymbol{W}_d^{(1)} = \boldsymbol{W}_d^{(2)} = ... = \boldsymbol{W}_d^{(h)}$; and so do all bias terms. The gate weights are also global weights: $\boldsymbol{W}_g^{(1)} = \boldsymbol{W}_g^{(2)} = ... = \boldsymbol{W}_g^{(h)}$.

4.3 Signed Distance-based Deep Memory Recommender (SDMR)

Now we propose Signed Distance-based Deep Memory Recommender (SDMR), a hybrid network that combines SDP and SDM. The first approach to combine them is to employ a weighted summation of the output scores from SDP and SDM as follows:

$$o = \beta o^{(\text{SDP})} + (1 - \beta) o^{(\text{SDM})} \quad (19)$$

where $o^{(\text{SDP})}$ is the signed distance score obtained at Eq. (6), $o^{(\text{SDM})}$ is the signed distance score obtained at Eq. (17), and $\beta \in [0, 1]$ is a hyper-parameter to control the contribution of SDP and SDM. When β=0, SDMR becomes SDM. When β=1, SDMR becomes SDP.

However, to avoid tuning an additional hyper-parameter β, we do not use Eq. (19) for SDMR. Instead, we let SDMR self-learns the combination of SDM and SDM as follows:

$$o = ReLU\left(\boldsymbol{w}_u^\top \begin{bmatrix} \boldsymbol{e}^{(\ell+1)} \\ \boldsymbol{e}^{(h)} \end{bmatrix} + b_u\right) \quad (20)$$

where $\boldsymbol{e}^{(\ell+1)}$ is the final layer embedding from SDP and is obtained at Eq. (5), $\boldsymbol{e}^{(h)}$ is the final hop output from the multi-hop SDM obtained at Eq. (18). We note that SDP and SDM are first pre-trained separately using the BPR loss function (see the next section). Then, we obtain $\boldsymbol{e}^{(\ell+1)}$ from SDP, and $\boldsymbol{e}^{(h)}$ from SDM, and keep them fixed in Eq. (20) to learn $\boldsymbol{w}_u$ and b_u. We use ReLU in Eq. (20) because ReLU encourages sparse activations and helps to reduce over-fitting when combining the two components SDP and SDM.

4.4 Loss Functions

We adopt the Bayesian Personalized Ranking (BPR) as our loss function, which is similar to the idea of AUC (area under the curve):

$$\mathcal{L} = \underset{\theta}{\text{argmin}} \left(- \sum_{(u, i^+, i^-)} \log \sigma(o_{ui^-} - o_{ui^+}) + \lambda \|\theta\|^2 \right) \quad (21)$$

where we uniformly sample tuples in a form of (u, i^+, i^-) for user u with positive item (consumed) i^+ and negative item (unconsumed) i^-. λ is a hyper-parameter to control the regularization term, and $\sigma(\cdot)$ is the sigmoid function. Note that other pairwise probability functions could be plugged in Eq. (21) to replace $\sigma(\cdot)$. Both SDP and SDM are end-to-end differentiable since we uses soft attention over the output memory. Hence, we can utilize back-propagation to learn our models with stochastic gradient descent or Adam [18].

5 EMPIRICAL STUDY

We evaluate our SDP, SDM, SDMR models against ten state-of-the-art baselines in two recommendation tasks: (i) *general recommendation task*, and (ii) *shopping basket-based recommendation task*. We mainly aim to answer the following research questions (RQs):

- **RQ1:** How do SDP, SDM, and SDMR perform compared to other state-of-the-art models in both general recommendation task and shopping basket-based recommendation task?
- **RQ2:** Why/How does the multi-hop design help to improve the proposed models' performance?

5.1 Datasets

General recommendation task: In this task, we evaluate our proposed models and state-of-the-art methods using different datasets with various density levels as follows:

- **Movielens** [41]: It is a widely adopted benchmark dataset for collaborative filtering evaluation. We use two versions of this benchmark dataset, namely MovieLens100k (or ML-100k) and MovieLens1M (or ML-1M).
- **Netflix Prize** [3]: It is a real-world dataset collected by Netflix. This dataset was collected from 1999 to 2005, and consists of 463,435 users and 17,769 items with 56.9M of interactions. Since the dataset is extremely large, we subsample the Netflix dataset by randomly picking one-month data for evaluation.
- **Epinions** [35] [4]: It is an online rating dataset where users can share product feedback by giving explicit ratings and reviews.

In preprocessing preparation, we adopted a popular k-core preprocessing step [11, 25, 52] (with *k-core* = 5) to filter out inactive users with less than five ratings and items which are consumed

[3] https://www.netflixprize.com/
[4] http://www.trustlet.org/downloaded_epinions.html

Table 1: Statistics of the four datasets in the general recommendation task.

Statistics	ML-100k	ML-1M	Netflix	Epinions
# of users	943	6,040	1,888	23,137
# of items	1,682	3,706	3,724	23,585
# of interactions	100,000	1,000,209	103,254	461,982
Density (%)	6.3%	4.5%	1.5%	0.08%

Table 2: Statistics of the two real-world transactional datasets in the shopping basket-based recommendation task.

Statistics	IJCAI-15	Tafeng
# of users	2,433	22,851
# of items	4,534	22,291
avg # of items in a transaction	6.28	9.28
# of generated instances	15,422	523,653
Density (%)	0.14%	0.10%

by less than five users. Since ML-100k and ML-1M are already preprocessed, we only apply 5-core preprocessing step on the Netflix and Epinions datasets. We also binarize the rating scores as implicit feedback by converting all observed rating scores as positive interactions and the remaining as negative interactions. The statistics of the four datasets are summarized in Table 1.

Shopping basket-based recommendation task: We use two real-world transaction datasets as follows:

- **IJCAI-15** [5]: It consists of shopping logs of users from Tmall [6]. Since the original dataset is extremely large scale. We subsample IJCAI-15 by randomly picking 20k transactions for evaluation.
- **Tafeng** [7]: It is a grocery store transaction data. It contains four month transaction data from November 2000 to February 2001 by T-Feng supermarket.

Users in both IJCAI-15 and Tafeng datasets are logged under four types of actions: *click*, *add-to-cart*, *purchase*, and *add-to-favourite*. We consider all the four types as the *click* action. We only keep transactions with at least five items. This is because we will take one item out for testing, another item for development. In the remaining three items, one will be taken out as a target item and the two items will be used as the context items. Attentive scores will be assigned to the context items. In each of original transactions, we generate data instances of the format $< \mathbf{c}, v_c >$ where v_c is the target/predicting item and $\mathbf{c}$ is a set of all other items in the same transaction with v_c. In particular, in each transaction t, each time we pick one item out as a target item and leave the rest of items in t as corresponding context items. Subsequently, for each transaction t containing $|t|$ items, we can generate $|t|$ data instances. The statistics of the two transactional datasets are summarized in Table 2.

For an easy reference, we call (ML-100k, ML-1M, Netflix, Epinions) as *Group-1 dataset* and (IJCAI-15, Ta-Feng) as *Group-2 datasets*.

[5]https://tianchi.aliyun.com/datalab/dataSet.htm?id=1
[6]https://www.tmall.com
[7]http://stackoverflow.com/questions/25014904/download-link-for-ta-feng-grocery-dataset

5.2 Baselines and State-of-the-art Methods

We compared our proposed models against several strong baselines in the general recommendation task as follows:

- **Item**KNN [42]: It is an item neighborhood-based collaborative filtering method. It exploited cosine item-item similarities to produce recommendation results.
- **Bayesian Personalized Ranking (MF-BPR)** [39]: It is a state-of-the-art pairwise matrix factorization method for implicit feedback datasets. It minimizes $\sum_i \sum_{j^+,j^-} -log\sigma(u_i^T v_{j^+} - u_i^T v_{j^-}) + \lambda(||u_i||^2 + ||v_{j^+}||^2)$ where (u_i, v_{j^+}) is a positive interaction and (u_i, v_{j^-}) is a negative sample.
- **Sparse LInear Method (**SLIM**)** [36]: It learns a sparse item-item similarity matrix by minimizing the squared loss $||A - AW||^2 + \lambda_1||W|| + \lambda_2||W||^2$, where A is a $m \times n$ user-item interaction matrix and W is a $n \times n$ sparse matrix of aggregation coefficients of context items.
- **Collaborative Metric Learning (CML)** [15]: It is a state-of-the-art collaborative metric-based model that utilizes Euclidean distance to measure similarities between users and items. For fair comparison, we learn CML with BPR loss by minimizing $-\sum_{i,j^+,j^-} log(\sigma(||u_i - v_{j^-}||_2^2 - ||u_i - v_{j^+}||_2^2))$, where $||\cdot||_2^2$ is a squared Euclidean distance, (u_i, v_{j^+}) is a positive interaction and (u_i, v_{j^-}) is a negative sample.
- **Neural Collaborative Filtering (NeuMF++)** [12]: It is a state-of-the-art matrix factorization method using deep learning architecture. We use a pre-trained NeuMF to achieve its best performance, and denote it as NeuMF++.
- **Collaborative Memory Network (CMN++)** [7]: It is a state-of-the-art memory network based recommender. Its architecture follows traditional user neighborhood based collaborative filtering approaches. It adopts a memory network to assign attentive weights for other similar users.

Even though our approaches do not model the order of consumed items in the user's purchase history (e.g. rigid orders of items), since we consider latest s items as the context items to predict the next item, we still compare our models with some key sequential models to further show our models' effectiveness as follows:

- **Personalized Ranking Metric Embedding (PRME)** [8]: Given a user u, a target item j, and a previous consumed item k, it models a personalized first-order Markov behavior with two components: $d_{ujk} = \alpha||v_u - v_j||^2 + (1-\alpha)||v_k - v_j||^2$, where $||\cdot||_2^2$ is a squared $\mathcal{L}2$ distance. Then PRME is learned by minimizing BPR loss.
- **PRME_s:** It is our extension of PRME, where the distance between the target item j and the previous consumed item k is replaced by the average distance between j and each of previous s items: $d_{ujs} = \alpha||v_u - v_j||^2 + (1-\alpha)\frac{1}{|s|}\sum_{k \in s}||v_k - v_j||^2$. We use BPR loss to learn PRME_s.
- **Translation-based Recommendation (TransRec)** [9]: It uses first-order Markov and considers a user u as a translator of his/her previous consumed item k to a next item j. In another word, $prob(j|u,k) \propto \beta_j - d(u + v_k - v_j)$ where β_j is an item bias term, d is a distance function (e.g. $\mathcal{L}1$ or $\mathcal{L}2$ distance). We use $\mathcal{L}2$ distance because it was shown to

perform better than $\mathcal{L}1$ [9]. TransRec is then learned with BPR loss.

- **Convolutional Sequence Embedding Recommendation (Caser)** [49]: It is a state-of-the-art sequential model. It uses convolution neural network with many horizontal and vertical kernels to capture the complex relationships among items.

The strong sequential baselines above surpassed many other sequential models such as: TransRec outperformed FMC[40], FPMC [40], HRM [55]; Caser surpassed GRU4Rec [14] and Fossil [10], so we exclude them in our evaluation.

Comparison: In the general recommendation task, we compare our proposed models with all **ten** strong baselines listed above. In the shopping basket-based recommendation task, since the sequential models often work better than general recommendation-based models (see Table 3), we only compared our proposed models with sequential baselines. We name general recommendation baselines (i.e. ItemKNN, BPR, SLIM, CML, NeuMF++, CMN++) as *Group-1 baselines*, and call sequential baselines (i.e. PRME, PRME_s, TransRec, Caser) as *Group-2 baselines* for an easy reference.

5.3 Experimental Settings

Protocol: We adopt the widely used *leave-one-out* setting [12, 60], in which for each user, we reserve her last interaction as the test sample. If there are no timestamps available in the dataset, then the test sample is randomly drawn. Among the remaining data, we randomly hold one interaction for each user to form the development set, while all others are utilized as the training set. Since it is very time-consuming and unnecessary to rank all the unobserved items for each user, we follow the standard strategy to randomly sample 100 unobserved items for each user. Then, we rank them together with the test item [12, 19].

Assigning item orders: Sequential models need rigid orders of consumed items but consumed items in the same transaction (in IJCAI-15 and TaFeng datasets) are assigned the same timestamp of the transaction containing these items. Hence, we assigned the item timestamps where the orders of items are kept as in the original dataset. This may give credits to sequential models but not our methods (because our methods will use all consumed items in the same transaction as context items and do not model the item orders).

Hyper-parameters selection: We perform a grid search for the embedding size d from $\{8, 16, 32, 64, 128\}$ and regularization terms from $\{0.1, 0.01, 0.001, 0.0001, 0.00001\}$ in all the models. We select the best number of hops for CMN++ and our SDM from $\{1, 2, 3, 4\}$. In NeuMF++, we select the best number of MLP layers from $\{1, 2, 3\}$. In our models, we fix the batch size to 256. We adopt Adam optimizer [18] with a fixed learning rate of 0.001. Similar to CMN++ and NeuMF++, the number of negative samples is set to 4. We use one layer perceptron for SDP (more complex datasets may need more than one layer to get better results). In the four datasets used in general recommendation task (e.g ML-100k, ML-1M, Netflix, Epinions), to avoid too many *zero paddings* for users with a smaller number of consumed items or too many context items are kept in the memory, which unnecessarily slow down the model's execution, we follow [50] to limit the number of context items using latest s consumed items. We search s in {5, 10, 20}. In the two shopping basket-based recommendation datasets (i.e. IJCAI-15 and TaFeng), since the maximum number of items in a transaction is small (e.g. 13 in IJCAI-15, and 18 in TaFeng), we consider all the other items in the same transaction with the target item as its context items. All the hyper-parameters are tuned using the development dataset. Our source code is available at: *https://github.com/thanhdtran/SDMR*.

Evaluation Metrics: We evaluate all models' performance by two widely used metrics: Hit Ratio (*hit@k*), and Normalized Discounted Cumulative Gain (NDCG@k), where k is a truncated number or *top-k* item recommendation. Intuitively, *hit@k* shows whether the test item is in the *top-k* list or not, while *NDCG@k* accounts for the position of the hits by assigning higher scores to the hits at top ranks and downgrading the scores to hits by log_2 at lower ranks.

5.4 Experimental Results

RQ1: Overall results in general recommendation task: The performance of our proposed models and the baselines are shown in Table 3. First, we observe that SDP significantly outperformed BPR in all four datasets in *Group-1 datasets*, improving *hit*@10 from 8.33~41.19%, and NDCG@10 from 10.44~44.56%. Although SDP and BPR shared the same loss function, the difference between them is SDP measured a signed distance score between a target user and a target item via a MLP which modeled a non-linear interaction between them, while BPR used Matrix Factorization with inner product. This result confirms the effectiveness of using signed distance based similarity over inner product in the general recommendation task. Second, we compare SDP with CML. CML worked by trying to minimize the squared Euclidean distance scores between target users and target items. Our SDP, in another hand, works by minimizing signed distance scores of non-linear interactions (via non-linear activation functions) between target users and target items. We observe that SDP performed better than CML in all *Group-1 datasets*, improving *hit*@10 from 8.33~11.19%, and NDCG@10 from 7.06~12.24%. On average, SDP improved *hit*@10 by 7.5% and *NDCG*@10 by 9.5% compared to CML. Our SDP even gain competitive results compared to NeuMF++ and CMN++. On average, SDP is just slightly worse than NeuMF++ and CMN++ by -2.67% for *hit*@10, and -1.68% for *NDCG*@10. All of these results show the effectiveness of using signed distance in our SDP model.

Next, we compare SDM with neighborhood-based baselines. Both SLIM and item-KNN used previously consumed items of a user to make the prediction for the next item. SDM significantly outperformed both baselines, improving *hit*@10 from 20.53~130.92% and NDCG@10 from 39.05~106.35% compared with SLIM. It is an obvious result because the neighborhood-based baselines barely measured linear similarities between the target item and the user's consumed items. In contrast, our SDM produced signed distance scores and assigned personalized metric-based attention weights to each of consumed items that contribute to the target item.

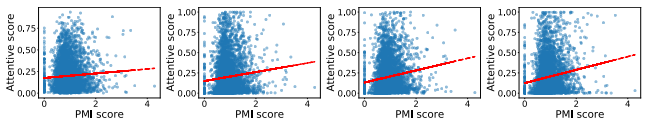
We then compare SDM with CMN++ and NeuMF++. SDM outperformed CMN++ in all *Group-1 datasets*, improving *hit*@10 from 11.71~35.93% and NDCG@10 from 26.51~43.38%. On average, it improves *hit*@10 by 18.63% and *NDCG@10* by 32.84% compared to CMN++. This result shows the effectiveness of our personalized metric-based attention with signed distance and item-based neighborhood design over the traditional inner product-based attention

Table 3: General Recommendation Task: Overall performance of the baselines, and our proposed SDP, SDM, and SDMR on four datasets. The last four lines show the relative improvement of the SDM and SDMR over the best baseline method in General Recommenders (Group 1) and Sequential Recommenders (Group 2), respectively.

Method type	Method	ML-100k		ML-1M		Netflix		Epinions	
		hit@10	NDCG@10	*hit*@10	NDCG@10	*hit*@10	NDCG@10	*hit*@10	NDCG@10
General Recommenders (Group 1)	Item-KNN	0.166	0.073	0.235	0.110	0.039	0.019	0.121	0.096
	SLIM	0.520	0.298	0.677	0.420	0.358	0.212	0.249	0.189
	MF-BPR	0.554	0.316	0.595	0.352	0.352	0.193	0.384	0.232
	CML	0.596	0.326	0.662	0.390	0.447	0.254	0.376	0.237
	NeuMF++	0.623	0.341	0.716	0.438	0.509	0.279	0.428	0.274
	CMN++	0.620	0.344	0.729	0.442	0.523	0.293	0.423	0.272
Sequential Recommenders (Group 2)	PRME	0.638	0.381	0.724	0.486	0.509	0.329	0.538	0.346
	PRME_s	0.674	0.398	0.734	0.491	0.539	0.348	0.380	0.244
	TransRec	0.684	0.402	0.770	0.524	0.511	0.345	0.551	0.357
	Caser	0.674	0.386	**0.826**	0.606	0.480	0.253	0.326	0.268
Ours	SDP	0.616	0.349	0.694	0.424	0.497	0.279	0.416	0.266
	SDM	**0.713**	0.435	**0.816**	0.584	**0.584**	0.379	**0.575**	0.390
	SDMR	**0.695**	**0.562**	**0.810**	**0.662**	**0.592**	**0.449**	**0.568**	**0.423**
Compared to Group 1	Imprv. of SDM	14.54%	26.51%	11.93%	32.13%	11.71%	29.32%	34.35%	42.34%
	Imprv. of SDMR	11.65%	63.44%	11.11%	49.77%	13.24%	53.20%	32.71%	54.38%
Compared to Group 2	Imprv. of SDM	4.24%	8.21%	-1.21%	-3.63%	8.35%	8.91%	4.36%	9.24%
	Imprv. of SDMR	1.61%	39.80%	-1.94%	9.24%	9.83%	29.02%	3.09%	18.49%

Table 4: Shopping basket-based Recommendation Task: Overall performance of the baselines, and our proposed models on two datasets. The last two lines show the relative improvement of the SDM and SDMR over the best baseline.

Method	IJCAI-15		Ta-Feng	
	hit@10	NDCG@10	*hit*@10	NDCG@10
PRME	0.276	0.177	0.594	0.365
PRME_s	0.229	0.133	0.590	0.355
TransRec	0.262	0.168	0.622	0.401
Caser	0.173	0.096	0.605	0.373
SDP	0.323	0.201	0.633	0.401
SDM	0.316	0.189	**0.646**	0.439
SDMR	**0.336**	**0.222**	0.627	**0.559**
Imprv. of SDM	14.49%	6.78%	3.86%	9.48%
Imprv. of SDMR	21.74%	25.42%	0.80%	39.40%



(a) Hop 1. **(b) Hop 2.** **(c) Hop 3.** **(d) Hop 4.**

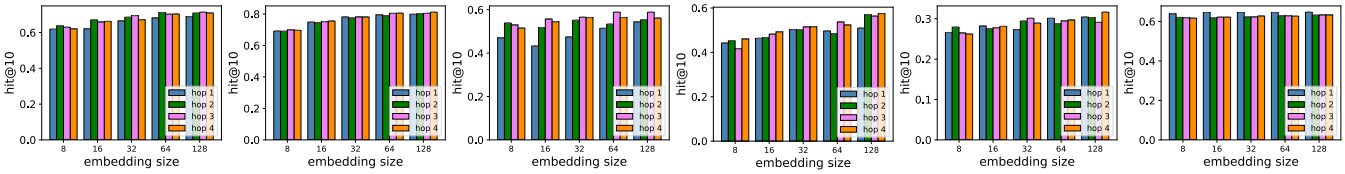
Figure 5: ML-100K: Scatter plots of PMI scores and attentive scores generated by SDM with h hops (h={1, 2, 3, 4} from left to right). The red lines are the linear trend lines. The Pearson correlation between two scores increases when h increased.

in a user-based neighborhood design in CMN++. SDM also outperformed NeuMF++, improving *hit*@10 from 13.97~34.35%, and NDCG@10 from 27.42~42.34%. On average, in all *Group-1 datasets*, SDM outperformed all the baselines in "General Recommenders" (Group 1), improved *hit*@10 by 18.13% and *NDCG*@10 by 32.58% compared to the best baseline in *Group 1*.

Finally, we look at the performance of SDMR model, which is the proposed fusion of SDP and SDM. Compared to SDM, our SDMR insignificantly downgrades SDM on *hit*@10 measurement with a very small amount, but it does help a lot in refining the ranking of items and boosting *NDCG*@10 results. As shown in Table 3, SDMR improved from 8.46~29.20% for *NDCG*@10, and by 17.37% for *NDCG*@10 on average compared to SDM in *Group-1 datasets*. SDMR also surpassed all the methods in *Group 1*. On average, SDMR improved *hit*@10 by 17.18% and NDCG@10 by 55.20% compared to the best model in *Group 1*.

We also compared our models with some strong sequential models in Table 3. Sequential models exploited consuming time of items and model their rigid orders, which often lead to a much improved performance compared to general recommendation models in *Group-1 baselines*. As such, compared to the best sequential baseline model, on average, SDM improves *hit*@10 by 3.94% and *NDCG*@10 by 5.68% , and SDMR improves *hit*@10 by 3.15% and *NDCG*@10 by 24.14% compared to the best sequential model reported in Table 3.

Overall results in shopping basket-based recommendation task: Table 4 shows the performance of our models and sequential baselines in *Group-2 datasets*. Again, our models outperformed all the sequential baselines. On average, SDM improved *hit*@10 by 9.2% and *NDCG*@10 by 8.1%, SDMR improved *hit*@10 by 11.3% and *NDCG*@10 by 32.4% compared to the best reported baseline.



(a) ML-100K. (b) ML-1M. (c) Netflix. (d) Epinions. (e) IJCAI-15. (f) TaFeng.

Figure 6: Comparison of varying the number of hops regarding different embeddings sizes in the six datasets.

5.5 RQ2: Understanding our multi-hop personalized metric-based attention design?

In the previous section, we see that our models outperformed many strong baselines in **six** different datasets of the **two** different recommendation problems. In this part, we explore why did we achieve those better results? As "attention is all you need" [54], the core reason brought us an surpassed performance accredit to the metric-based attention which are further refined via multi-hop design. Therefore, we want to explore quantitatively and qualitatively how our attention with multi-hop design worked by answering two smaller research questions: (i) what did our metric-based attention with multi-hop design learn?, (ii) did the metric-based attention with multi-hop design improve recommendation results? Without a special mention, since our SDMR model just learned a combination between SDP and SDM without re-learning the learned-already parameters in SDP and SDM, we explore SDM in this section to understand how attention with multi-hop design works. Note that we conduct this analysis for ML-100k only due to space limitation and the availability of movies genre in ML-100k (for visualization in Figure 7).

What did our metric-based attention with multi-hop design learn? To answer this question, we first measure the *point-wise mutual information* (PMI) between two certain items j and k as:

$$PMI(j,k) = log\frac{P(j,k)}{P(j) \times P(k)} \tag{22}$$

where $P(j,k)$ is the joint probability between two items j and k, which shows how likely j and k are co-preferred ($P(j,k) = \frac{\#(j,k)}{|D|}$, where D denotes a collection of all item-item pairs, and $|D|$ is the total number of item-item co-occurrence pairs in D). Similarly, $P(j)$ and $P(k)$ are the probabilities of the item j and k in D, respectively (e.g. $P(j) = \frac{\#(j)}{|D|}$, $P(k) = \frac{\#(k)}{|D|}$). Intuitively, a PMI score between two items shows how likely the two items are co-purchased/co-preferred. The higher the PMI score between j and k is, the more likely the user will purchase j if k was purchased before.

We denote SDM-h is the SDM model with h hops. Now, given a target item j and the user's context items k, *SDM-h* will assign attentive scores for all (j,k) pairs. We also get PMI scores (from Eq. (22)) of (j,k) pairs. Next, we plot a scatter plot of PMI scores and attentive scores for all (j,k) pairs to see the relationship between the two scores. Our results for ML-100k dataset is shown in Figure 5.

In Figure 5, the Pearson correlation between PMI scores and attentive scores are 0.059, 0.097, 0.143, and 0.146 for SDM-1, SDM-2, SDM-3 SDM-4, respectively. It indicates that as we increase the number of hops in SDM model, PMI scores and attentive scores are more positively correlated. In another word, as we increase the number of hops, our metric-based attention with multi-hop design will assign higher weights for co-purchased items, which is what we desire. Furthermore, scatter plots in Figure 5a presents that there is a high density of points with small attentive scores. This indicates that attention in SDM-1 is distributed to several items (which is somewhat close to equally focusing on context items). However, when we increase the number of hops h, the density spreads up to the top, indicating that the model tends to give a higher attention to some context items, which can be more relevant than others. This observation is consistent with "learning to attend" in [2, 59].

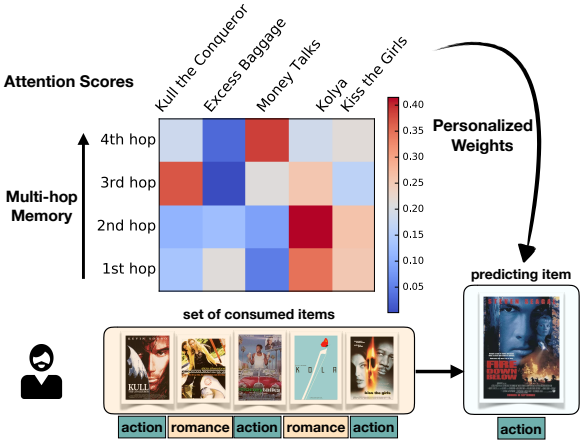


Figure 7: Multi-hop Attention visualization.

Did the metric-based attention with multi-hop design improve recommendation results? We answer this research question by showing the results of SDM model when varying number of hops h from {1, 2, 3, 4} with different embedding sizes and visualize attention scores of SDM-h with a random observation as follows:

Varying number of hops with different embedding sizes: The performance of SDM-h regarding *hit*@10 with h from {1, 2, 3, 4} and embedding size from {8, 16, 32, 64, 128} is presented in Figure 6. We see that more hops tend to give additional improvement in all 6 datasets, except in Tafeng dataset where SDM with more hops over-fitted. In ML-100k and ML-1M, the optimal number of hops are 3 or 4. In Netflix, SDM with 3 hops performed well. In Epinions and IJCAI-15, SDM-4 tends to achieve better results. Overall, the selection of the number of hops depends on the dataset complexity, and it varies from datasets to datasets.

Attention Visualization: Lastly, to visualize how the personalized metric-based attention with multi-hop design works, we chose one user from ML-100K data. The learned weights at each hop of SDM is shown in Figure 7. The target item in this example is an action movie called *Fire Down Below* (1997). The first two hops of SDM

assigned high weights to two romance movies, and the lowest score to the action movie *Money Talks* (1997). The 3rd-hop and 4th-hop attention refined the weights of movies to better reflect the correlations and similarities *w.r.t* the target movie. At last, *Money Talks* (1997) was assigned with the highest weight 0.386, and the total weights of two romance movies decreased to less than 0.2. This result shows the effectiveness of our multi-hop SDM model.

6 CONCLUSION

In this paper, we have studied the top-k recommendation problem in a signed distance learning perspective. Different from previous works, we have considered two independent signed distance models for measuring user-item and item-item similarities respectively via deep neural networks. Extensive experiments have been performed on six real-world datasets in general recommendation and shopping basket-based recommendation task. We presented that our proposed SDMR outperformed **ten** baselines in all two recommendation tasks. To an extension, future works can integrate position embeddings [54] in our models, which give our models a sense of which position they are dealing with, which can further improve our model's performance when rigid orders of items are available.

ACKNOWLEDGMENT

This work was supported in part by NSF grant CNS-1755536, Google Faculty Research Award, Microsoft Azure Research Award, AWS Cloud Credits for Research, and Google Cloud. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsors.

REFERENCES

[1] Charu C Aggarwal. 2016. *Recommender systems*. Springer.
[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).
[3] Daniel Billsus and Michael J Pazzani. 2000. User modeling for adaptive news access. *User modeling and user-adapted interaction* 10, 2-3 (2000), 147–180.
[4] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. 335–344.
[5] Heeyoul Choi, Kyunghyun Cho, and Yoshua Bengio. 2018. Fine-grained attention mechanism for neural machine translation. *Neurocomputing* 284 (2018), 171–176.
[6] Mukund Deshpande and George Karypis. 2004. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems* 22, 1 (2004), 143–177.
[7] Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative Memory Network for Recommendation Systems. In *Proceedings of the 41st ACM International Conference on Research and Development in Information Retrieval*.
[8] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. 2015. Personalized Ranking Metric Embedding for Next New POI Recommendation. In *International Joint Conference on Artificial Intelligence*, Vol. 15. 2069–2075.
[9] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. 161–169.
[10] Ruining He and Julian McAuley. 2016. Fusing similarity models with markov chains for sparse sequential recommendation. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. 191–200.
[11] Ruining He and Julian McAuley. 2016. 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*. 507–517.
[12] 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.
[13] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 549–558.
[14] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).
[15] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative metric learning. In *Proceedings of the 26th International Conference on World Wide Web*. 193–201.
[16] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. 263–272.
[17] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. 2016. Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 233–240.
[18] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[19] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 426–434.
[20] Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 447–456.
[21] Yehuda Koren. 2010. Collaborative filtering with temporal dynamics. *Commun. ACM* 53, 4 (2010), 89–97.
[22] Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *International Conference on Machine Learning*. 331–339.
[23] Neil D Lawrence and Raquel Urtasun. 2009. Non-linear matrix factorization with Gaussian processes. In *Proceedings of the 26th Annual International Conference on Machine Learning*. 601–608.
[24] Sheng Li, Jaya Kawale, and Yun Fu. 2015. Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 811–820.
[25] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. (2018).
[26] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130* (2017).
[27] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 1 (2003), 76–80.
[28] Fei Liu and Julien Perez. 2017. Gated end-to-end memory networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, Vol. 1. 1–10.
[29] Qiang Liu, Shu Wu, and Liang Wang. 2017. DeepStyle: Learning user preferences for visual recommendation. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 841–844.
[30] Xinyue Liu, Chara Aggarwal, Yu-Feng Li, Xiaugnan Kong, Xinyuan Sun, and Saket Sathe. 2016. Kernelized matrix factorization for collaborative filtering. In *Proceedings of the 2016 SIAM International Conference on Data Mining*. 378–386.
[31] Yichao Lu, Ruihai Dong, and Barry Smyth. 2018. Coevolutionary Recommendation Model: Mutual Learning between Ratings and Reviews. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. 773–782.
[32] Yichao Lu, Ruihai Dong, and Barry Smyth. 2018. Convolutional Matrix Factorization for Recommendation Explanation. In *Proceedings of the 23rd International Conference on Intelligent User Interfaces Companion*. 34.
[33] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* (2015).
[34] Chen Ma, Peng Kang, Bin Wu, Qinglong Wang, and Xue Liu. 2019. Gated Attentive-Autoencoder for Content-Aware Recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 519–527.
[35] Paolo Massa and Paolo Avesani. 2007. Trust-aware recommender systems. In *Proceedings of the 2007 ACM conference on Recommender systems*. 17–24.
[36] Xia Ning and George Karypis. 2011. Slim: Sparse linear methods for top-n recommender systems. In *2011 11th IEEE International Conference on Data Mining*. 497–506.
[37] 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.
[38] Parikshit Ram and Alexander G Gray. 2012. Maximum inner-product search using cone trees. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 931–939.
[39] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. 452–461.

[40] 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.

[41] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. 175–186.

[42] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. 285–295.

[43] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*. 111–112.

[44] Paul Hongsuck Seo, Zhe Lin, Scott Cohen, Xiaohui Shen, and Bohyung Han. 2016. Hierarchical attention networks. *arXiv preprint arXiv:1606.02393* (2016).

[45] Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. 2017. Interpretable convolutional neural networks with dual local and global attention for review rating prediction. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. 297–305.

[46] Anshumali Shrivastava and Ping Li. 2014. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). In *Advances in Neural Information Processing Systems*. 2321–2329.

[47] Nitish Srivastava and Ruslan R Salakhutdinov. 2012. Multimodal learning with deep boltzmann machines. In *Advances in Neural Information Processing Systems*. 2222–2230.

[48] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*. 2440–2448.

[49] 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.

[50] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Latent relational metric learning via memory-based attention for collaborative ranking. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. 729–739.

[51] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Multi-Pointer Co-Attention Networks for Recommendation. *arXiv preprint arXiv:1801.09251* (2018).

[52] Thanh Tran, Kyumin Lee, Yiming Liao, and Dongwon Lee. 2018. Regularizing Matrix Factorization with User and Item Embeddings for Recommendation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 687–696.

[53] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*. 2643–2651.

[54] 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*. 6000–6010.

[55] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2015. Learning hierarchical representation model for nextbasket recommendation. In *ACM SIGIR Conference on Research and Development in Information Retrieval*. 403–412.

[56] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *Proceedings of the tenth ACM international conference on web search and data mining*. 495–503.

[57] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. 153–162.

[58] Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *International Conference on Machine Learning*. 2397–2406.

[59] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*. 2048–2057.

[60] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep Matrix Factorization Models for Recommender Systems. In *Proceeding of the 26th International Joint Conference on Artificial Intelligence*. 3203–3209.

[61] Hanwang Zhang, Yang Yang, Huanbo Luan, Shuicheng Yang, and Tat-Seng Chua. 2014. Start from scratch: Towards automatically identifying, modeling, and naming visual attributes. In *Proceedings of the 22nd ACM International Conference on Multimedia*. 187–196.

[62] Tinghui Zhou, Hanhuai Shan, Arindam Banerjee, and Guillermo Sapiro. 2012. Kernelized probabilistic matrix factorization: Exploiting graphs and side information. In *Proceedings of the 2012 SIAM International Conference on Data Mining*. 403–414.