Collaborative Memory Network for Recommendation Systems

Travis Ebesu
Santa Clara University
Department of Computer Engineering
Santa Clara, CA, USA
tebesu@scu.edu

Bin Shen
Google
New York, NY, USA
bshen@google.com

Yi Fang
Santa Clara University
Department of Computer Engineering
Santa Clara, CA, USA
yfang@scu.edu

ABSTRACT

Recommendation systems play a vital role to keep users engaged with personalized content in modern online platforms. Deep learning has revolutionized many research fields and there is a recent surge of interest in applying it to collaborative filtering (CF). However, existing methods compose deep learning architectures with the latent factor model ignoring a major class of CF models, neighborhood or memory-based approaches. We propose Collaborative Memory Networks (CMN), a deep architecture to unify the two classes of CF models capitalizing on the strengths of the global structure of latent factor model and local neighborhood-based structure in a nonlinear fashion. Motivated by the success of Memory Networks, we fuse a memory component and neural attention mechanism as the neighborhood component. The associative addressing scheme with the user and item memories in the memory module encodes complex user-item relations coupled with the neural attention mechanism to learn a user-item specific neighborhood. Finally, the output module jointly exploits the neighborhood with the user and item memories to produce the ranking score. Stacking multiple memory modules together yield deeper architectures capturing increasingly complex user-item relations. Furthermore, we show strong connections between CMN components, memory networks and the three classes of CF models. Comprehensive experimental results demonstrate the effectiveness of CMN on three public datasets outperforming competitive baselines. Qualitative visualization of the attention weights provide insight into the model's recommendation process and suggest the presence of higher order interactions.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → **Neural networks**;

KEYWORDS

deep learning; memory networks; collaborative filtering; implicit feedback

SIGIR '18, July 8–12, 2018, Ann Arbor, MI, USA

ACM ISBN 978-1-4503-5657-2/18/07…$15.00
https://doi.org/10.1145/3209978.3209991

1 INTRODUCTION

Recommendation systems are vital to keeping users engaged and satisfied with personalized recommendations in the age of information explosion. Users expect personalized content in modern E-commerce, entertainment and social media platforms but the effectiveness of recommendations are restricted by existing user-item interactions and model capacity. The ability to leverage higher order reasoning may help alleviate the problem of sparsity. A popular and successful technique, collaborative filtering (CF), establishes the relevance between users and items from past interactions (e.g., clicks, ratings, purchases) by assuming similar users will consume similar items.

CF can generally be grouped in three categories: memory or neighborhood-based approaches, latent factor models and hybrid models [17, 26]. Memory or neighborhood-based methods form recommendations by identifying groups or neighborhoods of similar users or items based on the previous interaction history. The simplicity of these models such as item K nearest neighbor (KNN) have shown success in production systems at Amazon [21]. Latent factor models such as matrix factorization project each user and item into a common low dimensional space capturing latent relations. Neighborhood methods capture local structure but typically ignore the mass majority of ratings available due to selecting at most K observations from the intersection of feedback between two users or items [17]. On the other hand, latent factor models capture the overall global structure of the user and item relationships but often ignore the presence of a few strong associations. The following weaknesses between the local neighborhood-based and global latent factor models lead to the development of hybrid models such as SVD++ [17] and generalizations such as Factorization Machines [24] which integrate both neighborhood-based approaches and latent factor models to enrich predictive capabilities.

Recently, deep learning has made massive strides in many research areas obtaining state of the art performance in computer vision [9], question answering [18, 30, 35, 39], learning programs [8], machine translation [1] and many other domains. The successful integration of deep learning methods in recommendation systems have demonstrated the noticeable advantages of complex nonlinear transformations over traditional linear models [40]. However, existing composite architectures incorporate the latent factor model ignoring the integration of neighborhood-based approaches in a nonlinear fashion. Hence, we propose to represent the neighborhood-based component with a Memory Network [30, 35] to capture higher order complex relations between users and items. An external memory permits encoding rich feature representations while the neural attention mechanism infers the user specific contribution from the community.

We propose a unified hybrid model which capitalizes on the recent advances in Memory Networks and neural attention mechanisms for CF with implicit feedback. The memory component allows read and write operations to encode complex user and item relations in the internal memory. An associative addressing scheme acts as a nearest neighborhood model finding semantically similar users based on an adaptive user-item state. The neural attention mechanism places higher weights on specific subsets of users who share similar preferences forming a collective neighborhood summary. Finally, a nonlinear interaction between the local neighborhood summary and the global latent factors[1] derives the ranking score. Stacking multiple memory components allows the model to reason and infer more precise neighborhoods further improving performance.

Our primary contributions can be summarized as follows:

- We propose Collaborative Memory Network (CMN) inspired by the success of memory networks to address implicit collaborative filtering. CMN is augmented with an external memory and neural attention mechanism. The associative addressing scheme of the memory module acts as a nearest neighborhood model identifying similar users. The attention mechanism learns an adaptive nonlinear weighting of the user's neighborhood based on the specific user and item. The output module exploits nonlinear interactions between the adaptive neighborhood state jointly with the user and item memories to derive the recommendation.
- We reveal the connection between CMN and the two important classes of collaborative filtering models: the latent factor model and neighborhood-based similarity model. Furthermore, we reveal the advantages of the nonlinear integration fusing the two types of models yielding a hybrid model.
- Comprehensive experiments on three public datasets demonstrate the effectiveness of CMN against seven competitive baselines. Multiple experimental configurations confirm the added benefits of the memory module [2].
- Qualitative visualizations of the attention weights provide insight into the memory component providing supporting evidence for deeper architectures to capture higher order complex interactions.

2 RELATED WORK

2.1 Deep Learning in Recommendation Systems

Recently, a surge of interest in applying deep learning to recommendation systems has emerged. Among the early works, Salakhutdinov et al. [27] address collaborative filtering by applying a two layer Restricted Boltzmann Machine modeling tabular movie ratings. Autoencoders have been a popular choice of deep learning architecture for recommender systems [20, 28, 33, 37, 41]. The autoencoder acts as a nonlinear decomposition of the rating matrix replacing the traditional linear inner product. For example, AutoRec [28] decomposes the rating matrix with an autoencoder followed by reconstruction to directly predict ratings obtaining competitive results on numerous benchmark datasets. Collaborative denoising autoencoders (CDAE) [37] address top-n recommendation by integrating a user-specific bias into an autoencoder demonstrating CDAE can be seen as a generalization of many existing collaborative filtering methods. Li et al. [20] adopt a marginalized denoising autoencoder to diminish the computational costs associated with deep learning. Employing two autoencoders, one for item content and the other for user content bridged with user and item latent factors. AutoSVD++ [41] extends the original SVD++ model with a contrastive autoencoder to capture auxiliary item information. A hierarchical Bayesian model [33] bridges matrix factorization with the deepest layer of a stacked denoising autoencoder leveraging item content in the process.

Neural Matrix Factorization [11] address implicit feedback by jointly learning a matrix factorization and a feedforward neural network. The outputs are then concatenated before the final output to produce an interaction between the latent factors and the nonlinear factors. Ebesu and Fang [4] address the item cold-start problem by tightly coupling a deep neural network (DNN) for item content and pairwise matrix factorization for rating decomposition. The DNN constructs a robust representation of the item content and item latent factor for new items. Cheng et al. [3] jointly train a logistic regression and a feedforward neural network to leverage the generalization aspects of deep learning and specificity of generalized linear models for mobile app recommendations in the Google Play store.

Convolutional neural networks (CNN) in recommendation systems have been used to capture localized item feature representations of music [31], text [16, 29] and images [40]. Previous methods represent text as bag-of-words representations, CNN overcomes this limitation by learning weight filters to identify the most prominent phrases within the text. The sequential nature of recurrent neural networks (RNNs) provides desirable properties for time-aware [36] and session-based recommendation systems [12]. For example, Recurrent Recommender Networks [36] capture temporal aspects with a user and item Long Short Term Memory (LSTM) [7] cell coupled with stationary factors to identify movie popularity fluctuations caused by external events such as film awards. Jannach and Ludewig [14] interpolate KNN with a session-based RNN [12] demonstrating further performance gains. However, the interpolation scheme is a fixed weighting hyperparameter and lacks a nonlinear interaction to capture more complex relations. Wang et al. [34] unify the generative and discriminative methodologies under the generative adversarial network [7] framework for web search, item recommendation, and question answering.

Attention mechanisms have been recently explored in recommender systems. Gong and Zhang [6] perform hashtag recommendation with a CNN augmented with an attention channel to concentrate on the most informative (trigger) words. However, a hyperparameter must be carefully set to control the threshold of triggering the word to be informative. Huang et al. [13] tackle the same task with an End-to-End Memory Network [30] integrating a hierarchical attention mechanism over the user's previous tweets on a word and sentence level. Chen et al. [2] incorporate multimedia content with an item level attention representing the user preferences and a component level attention to isolate item specific visual features. Similarly, Seo et al. [29] introduce a local and global

[1] We use the terms user/item latent factors, memories and embeddings interchangeably.
[2] Source code available at: http://github.com/tebesu/CollaborativeMemoryNetwork

attention mechanism over convolutions to model review text. Xiao et al. [38] extend Factorization Machines [24] with an attention mechanism to learn the importance of each pairwise interaction rather than treating them uniformly. Most existing neural attention based methods rely on additional content or context information while our task is to study collaborative filtering. To the best of our knowledge, no prior work has employed the memory network architecture to address implicit feedback in the collaborative filtering setting.

2.2 Memory Augmented Neural Networks

We first provide a brief overview of the inner workings of memory-based architectures. Memory augmented neural networks, generally consist of two components: an external memory typically a matrix and a controller which perform operations on the memory (e.g., read, write, and erase). The memory component increases model capacity independent of the controller (typically a neural network) while providing an internal representation of knowledge to track long-term dependencies and perform reasoning. The controller manipulates these memories with either content-based or location-based addressing. Content-based or associative addressing finds a scoring function between the given question (query) and a passage of text, typically the inner product followed by the softmax operation leading to softly reading each memory location [1, 18, 30, 35, 39]. Performing a soft read over the memory locations allows the model to maintain differentiation hence can be trained via backpropagation. The latter type of addressing (usually combined with content-based) performs sequential reads or random access [8].

The initial framework proposed by Weston et al. [35] demonstrated promising results to track long-term dependencies and perform reasoning over synthetic question answering tasks. Sukhbaatar et al. [30] alleviated the strong levels of supervision required to train the original memory network becoming an End-to-End system. The notion of attention is biologically motivated how humans do not uniformly process all information in a given task but focus on specific subsets of information. Attention mechanisms also provide a level of insight into the deep learning black box by visualizing the attention weights [1]. Kumar et al. [18] improve upon the existing architecture by introducing an episodic memory component allowing for multiple passes or consultations of the memory before producing the final answer. The flexibility of the memory network architecture allows it to perform visual question answering [39] and joint task learning for identifying the sentiment and the relation to target entity [19].

3 COLLABORATIVE MEMORY NETWORK

In this section, we introduce our proposed model Collaborative Memory Network (CMN), see Figure 1a for a visual depiction of the architecture. At a high level, CMN maintains three memory states: an internal user-specific memory, an item-specific memory, and a collective neighborhood state. The architecture allows for the joint nonlinear interaction of the specialized local structure of neighborhood-based methods and the global structure of latent factor models. The associative addressing scheme acts as a nearest neighbor similarity function that learns to select semantically similar users based on the current item. The neural attention mechanism permits learning an adaptive nonlinear weighting function for the neighbor model, where the most similar users contribute higher weights at the output module. We later extend the model to a deeper architecture by stacking multiple hops in Section 3.4 depicted in Figure 1b.

3.1 User Embedding

The memory component consists of a user memory matrix $\mathbf{M} \in \mathbb{R}^{P \times d}$ and an item memory matrix $\mathbf{E} \in \mathbb{R}^{Q \times d}$, where P and Q represents the number of users and items respectively and d denotes the size (dimensionality) of each memory cell. Each user u is embedded in a memory slot $\mathbf{m}_u \in \mathbf{M}$ storing her specific preferences. Similarly, each item i corresponds to another memory slot $\mathbf{e}_i \in \mathbf{E}$ encoding the item's specific attributes. We form a user preference vector $\mathbf{q}_{ui}$ where each dimension q_{uiv} is the similarity of the target user u's level of agreement with user v in the neighborhood given item i as:

$$q_{uiv} = \mathbf{m}_u^\top \mathbf{m}_v + \mathbf{e}_i^\top \mathbf{m}_v \quad \forall\, v \in N(i) \tag{1}$$

where $N(i)$ represents the set of all users (neighborhood) who have provided implicit feedback for item i. We would like to point out $N(i)$ could be replaced or combined with $R(i)$ to handle the case of explicit feedback where $R(i)$ denotes the set of all users who provided explicit feedback for item i. The intuition is as follows, the first term computes compatibility between the target user and the users who have rated item i. The second term introduces the level of confidence user v supports the recommendation of item i. Hence, the associative addressing scheme identifies the internal memories with the highest similarity of the target user u with respect to neighborhood of users given the specific item.

3.2 Neighborhood Attention

The neural attention mechanism learns an adaptive weighting function to focus on a subset of influential users within the neighborhood to derive the ranking score. Traditional neighborhood methods predefine a heuristic weighting function such as Pearson correlation or cosine similarity and require specifying the number of users to consider [26]. While factorizing the neighborhood partially alleviates this problem, it is still linear in nature [15]. Instead by learning a weighting function over the entire neighborhood, we no longer need to empirically predefine the weighting function or number of neighbors to consider. Formally, we compute the attention weights for a given user to infer the importance of each user's unique contribution to the neighborhood:

$$p_{uiv} = \frac{\exp(q_{uiv})}{\sum_{k \in N(i)} \exp(q_{uik})} \quad \forall\, v \in N(i) \tag{2}$$

which produces a distribution over the neighborhood. The attention mechanism allows the model to focus on or place higher weights on specific users in the neighborhood while placing less importance on user's who may be less similar. Next we construct the final neighborhood representation by interpolating the external neighborhood memory with the attention weights:

$$\mathbf{o}_{ui} = \sum_{v \in N(i)} p_{uiv} \mathbf{c}_v \tag{3}$$

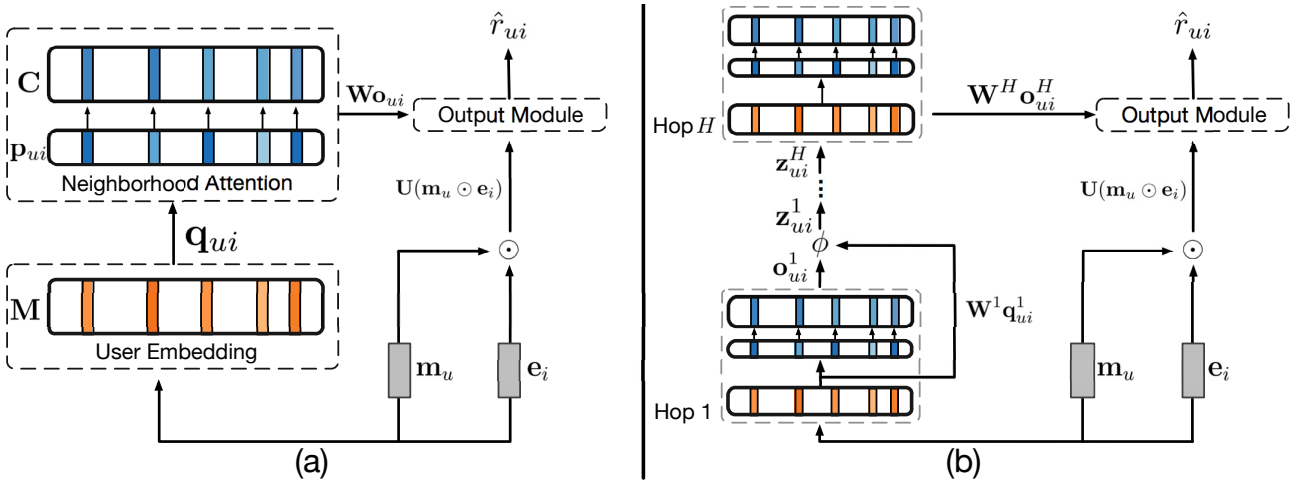


Figure 1: Proposed architecture of Collaborative Memory Network (CMN) with a single hop (a) and with multiple hops (b).

where $\mathbf{c}_v$ is another embedding vector for user v which is called external memory in the original memory network framework [35]. Denoting the v^{th} column of the embedding matrix $\mathbf{C}$ with the same dimensions as $\mathbf{M}$. The external memory allows the storage of long-term information pertaining specifically to each user's role in the neighborhood. In other words, the associative addressing scheme identifies similar users within the neighborhood acting as a key to weight the relevant values stored in the memory matrix $\mathbf{C}$ via the attention mechanism. The attention mechanism selectively weights the neighbors according to the specific user and item. The final output $\mathbf{o}_{ui}$ represents a weighted sum over the neighborhood composed of the relations between the specific user, item and the neighborhood.

CMN captures the similarity of users and dynamically assigns the degrees of contribution to the collective neighborhood based on the target item rather than a predefined number of neighbors which may restrict generalization capacity. Furthermore, the attention mechanism reduces the bottleneck of encoding all information into each individual memory slot and allows the joint exploitation of the user and item observations.

3.3 Output Module

As noted earlier neighborhood models capture the local structure from the rating matrix via the neighbors while latent factor models identify the global structure of the rating matrix [17]. Hence we consider the collective neighborhood state to capture localized user-item relations and the user and item memories to capture the global user-item interactions. The output module smoothly integrates a nonlinear interaction between the local collective neighborhood state and the global user and item memories. Existing models lack the nonlinear interaction between the two terms potentially limiting the extent of captured relations [2, 41]. For a given user u and item i the ranking score is given as:

$$\hat{r}_{ui} = \mathbf{v}^{\mathsf{T}}\phi\big(\mathbf{U}(\mathbf{m}_u \odot \mathbf{e}_i) + \mathbf{W}\mathbf{o}_{ui} + \mathbf{b}\big) \tag{4}$$

where $\odot$ is the elementwise product; $\mathbf{W}, \mathbf{U} \in \mathbb{R}^{d\times d}$; and $\mathbf{v}, \mathbf{b} \in \mathbb{R}^d$ are parameters to be learned. We first apply the elementwise product between the user and item memories followed by a linear projection with $\mathbf{U}$, subsequently introducing a skip-connection thus reducing the longest path from the output to input. Skip-connections have been shown to encourage the flow of information and ease the learning process [10]. In this way, the model can better correlate the specific target addresses (user and item memories) with the ranking score to propagate the appropriate error signals. We further motivate this choice by demonstrating its connection to the latent factor model (Section 3.7.1). Similarly, the final neighborhood representation $\mathbf{o}_{ui}$ is projected to a latent space with $\mathbf{W}$ then combined with the previous term followed by a nonlinear activation function $\phi(\cdot)$. Empirically we found the rectified linear unit (ReLU) $\phi(x) = \max(0, x)$ to work best due to its nonsaturating nature and suitability for sparse data [7, 10].

Our proposed model provides the following advantages. First, consider the case where the amount of feedback for a given user is sparse, we can leverage all users who have rated the item to gain additional insight about the existing user and item relations. Second, the neural attention mechanism adjusts the confidence of each user's contribution to the final ranking score dependent on the specific item. Finally, the nonlinear interaction between the local neighborhood and global latent factors provide a holistic view of the user-item interactions.

3.4 Multiple Hops

We now extend our model to handle an arbitrary number of memory layers or hops. Figure 1b (right) illustrates CMN's architecture with multiple hops. Each hop queries the internal user memory and item memory followed by the attention mechanism to derive the next collective neighborhood state vector. The first hop may introduce the need to acquire additional information. Starting from the second hop, the model begins to take into consideration the collective user neighborhood guiding the search for the representation of the community preferences. Each additional hop repeats this step considering the previous hop's newly acquired information before

producing the final neighborhood state. In other words, the model has the chance to look back and reconsider the most similar users to infer more precise neighborhoods. More specifically, multiple memory modules are stacked together by taking the output from the h^{th} hop as input to the $(h+1)^{th}$ hop. Similar to [19, 30, 39] we apply a nonlinear projection between hops:

$$\mathbf{z}_{ui}^{h} = \phi(\mathbf{W}^{h}\mathbf{q}_{ui}^{h} + \mathbf{o}_{ui}^{h} + \mathbf{b}^{h}) \tag{5}$$

where $\mathbf{W}^{h}$ is a square weight matrix mapping the user preference query $\mathbf{q}_{ui}^{h}$ to a latent space coupled with the existing information from the previous hop followed by a nonlinearity. Intuitively, the initial consultation of the memory may introduce the need for additional information to infer more precise neighborhoods. The nonlinear transformation updates the internal state then solicits the user neighborhood:

$$q_{uiv}^{h+1} = (\mathbf{z}_{ui}^{h})^{\mathsf{T}}\mathbf{m}_{v} \quad \forall\, v \in N(i) \tag{6}$$

The newly formed user preference vector then recomputes the compatibility between the target user and the neighborhood followed by the adaptive attention mechanism producing an updated collective neighborhood summary. This process is repeated for each hop yielding an iterative refinement. The output module receives the weighted neighborhood vector from the last (H^{th}) hop to produce the final recommendation.

3.5 Parameter Estimation

Since our objective is to study implicit feedback which is more pervasive in practice and can be collected automatically (e.g. clicks, likes). In the case of implicit feedback, the rating matrix contains a 1 if the item is observed and 0 otherwise. We opt for the pairwise assumption, where a given user u prefers the observed item i^{+} over unobserved or negative item i^{-}. The traditional pointwise approach assumes the user is not interested in the item i^{-} but in reality may not be aware of the item. We can form triplet preferences (u, i^{+}, i^{-}) since the number of preference triplets is quadratic in nature we uniformly sample a ratio of positive items to negative items which we further investigate in Section 4.7. We leverage the Bayesian Personalized Ranking (BPR) optimization criterion [25] as our loss function which approximates AUC (area under the ROC curve):

$$\mathcal{L} = -\sum_{(u,i^{+},i^{-})} \log \sigma(\hat{r}_{ui^{+}} - \hat{r}_{ui^{-}}) \tag{7}$$

where $\sigma(x) = 1/\big(1+\exp(-x)\big)$ is the logistic sigmoid function. It is worth noting we are not restricted to setting $\sigma(x)$ as the logistic sigmoid function. Other pairwise probability functions such as the Probit function can be used as in [4]. Since the entire architecture is differentiable, CMN can be efficiently trained with the backpropagation algorithm. To reduce the number of parameters we perform layerwise weight tying sharing all embedding matrices across hops [19, 30, 39].

3.6 Computational Complexity

The computational complexity for a forward pass through CMN for a user is $O\big(d|N(i)| + d^2 + d\big)$ where $|N(i)|$ denotes the size of the neighborhood for item i and d is the embedding size. The first term $O\big(d|N(i)|\big)$ is the cost for computing the user preference vector and the latter terms correspond to the final interactions in the output module. Each additional hop introduces $O\big(d|N(i)| + d^2\big)$ complexity. During training two forward passes are computed one for the observed positive item and the second for the negative unobserved item. Parameters can be updated via backpropagation with the same complexity. In real-world datasets, $|N(i)|$ is usually slightly larger than or comparable to d, and thus the primary computational complexity is computing $O(d|N(i)|)$. The cost is reasonable since other deep learning methods such as CDAE [37] compute a forward pass in $O\big(Qd\big)$ where Q is the total number of items. The proposed memory module only computes the similarity with the target user's neighbors (not over all users) and $|N(i)|$ is often less than or comparable to Q. Thus, the training in CMN is quite efficient. In practice, prefiltering techniques can be used to limit the number of neighbors to the top-K because not all neighbors may be indicative in contributing to the final prediction [26]. Since the purpose of our study is to understand the characteristics of CMN, we leave prefiltering techniques to future work.

Recommendation can be performed by computing the predicted ranking score (Eqn. 4) for a given user and item with a single pass through the network. The item with the highest value is recommended to the user. The computational complexity for runtime recommendation is the same with that of the single forward pass during training.

3.7 Relation to Existing Models

CMN consists of components which can be interpreted in terms of the three classes of collaborative filtering methods. We show the connection with the latent factor model and neighborhood-based similarity models, and finally the relation to hybrid models such as SVD++. We conclude the section by drawing parallels between memory networks and CMN.

3.7.1 Latent Factor Model. The latent factor model discovers hidden relations by decomposing the ratings matrix into two lower rank matrices. By omitting the neighborhood term and bias, and further setting $\mathbf{U}$ to the identity matrix Eqn. 4 becomes the following:

$$\hat{r}_{ui} = \mathbf{v}^{\mathsf{T}}\phi(\mathbf{m}_{u} \odot \mathbf{e}_{i}) \tag{8}$$

which leads to a generalized matrix factorization (GMF) model [11]. Removing the nonlinearlity by setting $\phi(\cdot)$ to the identity function and constraining $\mathbf{v}$ to $\mathbb{1}$ vector of all ones, we recover matrix factorization. Under our pairwise loss function (Eqn. 7) we recover BPR [25].

3.7.2 Neighborhood-based Similarity Model. The objective of neighborhood-based similarity models are to estimate a user-user[3] similarity matrix $\mathbf{S} \in \mathbb{R}^{P \times P}$. For each user who rated item i an aggregated similarity score produces the confidence of recommending the item. The general form of neighborhood similarity models are:

$$\hat{r}_{ui} = \alpha \sum_{v \in N(i)} \mathbf{S}_{uv} \tag{9}$$

where α is a normalization term to weight the ranking score. In the simplest case, the normalization term is set to $|N(i)|^{-\rho}$ where

[3] Equivalently switching users with items yields item-based methods.

Dataset	Ratings	Users	Items	Sparsity
Epinions	664,823	40,163	139,738	99.98%
citeulike-a	204,987	5,551	16,980	99.78%
Pinterest	1,500,809	55,187	9,916	99.73%

Table 1: Dataset statistics.

ρ is a hyperparameter controlling the level of similarity required to obtain a high score. In KNN, the neighborhood $N(i)$ is restricted to be the weighted combination of the K most similar users and the similarity matrix $\mathbf{S}$ is approximated with a heuristically predefined function such as Pearson correlation or cosine similarity. Another approach is to learn the similarity function by approximating $\mathbf{S}$ [15, 23]. In our case, the attached memory module from Eqn. 3 acts as the neighborhood similarity matrix. If we designate the attention mechanism as a predefined normalization term the user-user similarity matrix is then factorized as $\mathbf{S} = \mathbf{C}\mathbf{C}^\mathsf{T}$. Using the prediction rule from Eqn. 9 the memory module yields a user-based variant of FISM and under the BPR loss function we recover FISMauc [15].

3.7.3 Hybrid Model. We have shown the connection between the components of CMN and the two classes of collaborative filtering models. Hybrid models such as SVD++ [17] contains two general terms, a user-item latent factor interaction and a neighborhood component. The output module (Eqn. 4) smoothly integrates the latent factors and the similarity or neighborhood terms together leading to a hybrid model.

$$\hat{r}_{ui} = \mathbf{v}^\mathsf{T}\phi\Big(\overbrace{\mathbf{m}_u \odot \mathbf{e}_i}^{\text{Latent Factors}} + \underbrace{\sum_{v \in N(i)} p_{uiv}\mathbf{c}_v}_{\text{Neighborhood}} \Big) \tag{10}$$

We remove the projection matrices and bias terms for clarity. We can see the global interaction from the latent factors consist of the user and item memories. The memory module represents the localized neighborhood component and the neighborhood normalization term is replaced with the adaptive attention mechanism pushed inside the summation becoming a user-item specific weighting scheme. Unlike SVD++, our hybrid model allows for complex non-linear interactions between the two terms to model the diverse tastes of users.

3.7.4 Memory Networks. Traditional memory networks address the task of question answering. A short story or passage of text is provided along with a question for which the answer can be derived by leveraging some form of reasoning. If we pose recommendation as a question answering problem we are asking how likely will this user enjoy the item where the user neighborhood is the story and the output ranking score is the answer. Continuing our analogy, each word in the story acts as a user in the neighborhood providing supporting evidence for the recommendation.

4 EXPERIMENTAL RESULTS

4.1 Datasets

We study the effectiveness of our proposed approach on three publicly available datasets. The first dataset is collected from *Epinions*[4] [22] which provides an online service for users to share product feedback in the form of explicit ratings (1-5) and reviews. We convert the explicit ratings to implicit feedback as a 1 if the user has rated the item and 0 otherwise. The second dataset is *citeulike-a*[5] [32] collected from CiteULike an online service which provides users with a digital catalog to save and share academic papers. User preferences are encoded as 1 if the user has saved the paper (item) in their library. The third dataset from *Pinterest*[6] [5] allows users to save or pin an image (item) to their board indicating a 1 or positive interaction otherwise a 0 and preprocessed according to [11]. Table 1 summarizes the statistics of the datasets.

4.2 Evaluation

We validate the performance of our proposed approach using the leave-one-out evaluation method following the prior work [2, 11, 25]. Closely following the setup from He et al. [11], for each user we randomly hold out one item the user has interacted with and sample 100 unobserved or negative items to form the test set. The remaining positive examples form the training set. If the user has only rated a single item we keep it in the training set to prevent the cold-start setting. We rank the positive item along with the 100 negative items and adopt two common ranking evaluation metrics *Hit Ratio (HR)* and *Normalized Discounted Cumulative Gain (NDCG)* [26]. Intuitively, HR measures the presence of the positive item within the top N and NDCG measures the items position in the ranked list and penalizes the score for ranking the item lower in the list.

4.3 Baselines and Settings

We compare our proposed approach against seven competitive baselines representing neighborhood-based; traditional latent factor models; hybrid model and deep learning-based models.

- KNN [26] is a neighborhood-based approach computing the cosine item-item similarity to provide recommendations.
- Factored Item Similarity Model (FISM) [15] is a neighborhood-based approach factorizing the item-item similarity matrix into two low rank matrices optimizing the BPR loss function.
- Bayesian Personalized Ranking (BPR) [25] is a competitive pairwise matrix factorization for implicit feedback.
- SVD++ [17] is a hybrid model combining the neighborhood-based similarity and the latent factor model.
- Generalized Matrix Factorization (GMF) [11] is a nonlinear generalization of the latent factor model. We use the ReLU activation function and optimize the BPR loss function.
- Collaborative Denoising Auto Encoder (CDAE) [37] is an item-based deep learning model for item ranking with a user specific bias.

[4] http://www.trustlet.org/downloaded_epinions.html
[5] http://www.cs.cmu.edu/~chongw/data/citeulike/
[6] http://sites.google.com/site/xueatalphabeta/

	Epinions				*citeulike-a*				*Pinterest*			
	HR@5	HR@10	NDCG@5	NDCG@10	HR@5	HR@10	NDCG@5	NDCG@10	HR@5	HR@10	NDCG@5	NDCG@10
KNN	0.1549	0.1555	0.1433	0.1435	0.6990	0.7348	0.5789	0.5909	0.5738	0.8376	0.3450	0.4310
FISM	0.5542	0.6717	0.4192	0.4573	0.6727	0.8072	0.5106	0.5545	0.6783	0.8654	0.4658	0.5268
BPR	0.5584	0.6659	0.4334	0.4683	0.6547	0.8083	0.4858	0.5357	0.6936	0.8674	0.4912	0.5479
SVD++	0.5628	0.6754	0.4112	0.4477	0.6952	0.8199	0.5244	0.5649	0.6951	0.8684	0.4796	0.5362
GMF	0.5365	0.6562	0.4015	0.4404	0.7271	0.8326	0.5689	0.6034	0.6726	0.8505	0.4737	0.5316
CDAE	0.5666	0.6844	0.4333	0.4715	0.6799	0.8103	0.5106	0.5532	0.7008	0.8722	0.4966	0.5525
NeuMF	0.5500	0.6660	0.4214	0.4590	0.7629	0.8647	0.5985	0.6316	0.7041	0.8732	0.4978	0.5530
CMN-1	0.5962	0.6943	0.4684	0.5003	0.6692	0.7809	0.5213	0.5575	0.6984	0.8662	0.4960	0.5507
CMN-2	0.6017†	**0.7007**†	0.4724†	0.5045†	**0.7959**†	**0.8921**†	0.6185†	0.6500†	0.7267†	0.8904†	**0.5180**†	0.5714†
CMN-3	**0.6020**†	0.6985†	**0.4748**†	**0.5062**†	0.7932†	0.8901†	**0.6234**†	**0.6551**†	**0.7277**†	**0.8931**†	0.5175†	**0.5715**†

Table 2: Experimental results for different methods on the *Epinions*, *citeulike-a* and *Pinterest* datasets. Best results highlighted in bold. † indicates the improvement over baselines is statistically significant on a paired t-test ($p < 0.01$).

- Neural Matrix Factorization (NeuMF) [11] is a composite matrix factorization jointly coupled with a multilayer perceptron model for item ranking.

We would like to note that FISM [15] improves upon Sparse Linear Methods (SLIM) [23] by factorizing the item-item similarity matrix to handle missing entries hence we do not compare to SLIM. We exclude baselines utilizing additional information for fair comparison since our objective is to study implicit collaborative filtering without content or contextual information.

All hyperparameters are tuned according to the validation set. The validation set is formed by holding out one interaction per user from the training set [11]. We perform a grid search over each model's latent factors from $\{10, 20, 30, 40, 50\}$ and regularization terms $\{0.1, 0.01, 0.001\}$. In addition, we varied CDAE's corruption ratio from $\{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ and NeuMF's layers from $\{1, 2, 3\}$. The number of negative samples is set to 4.

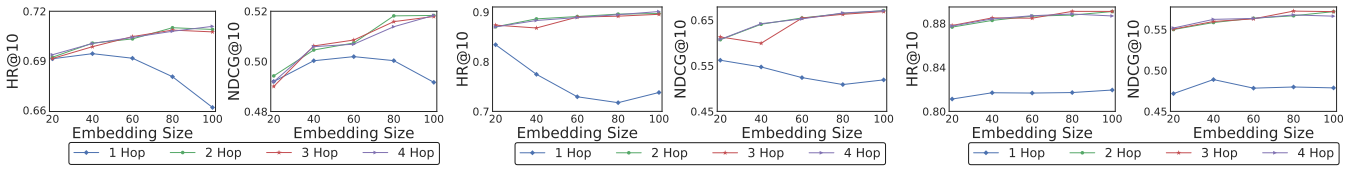
Careful initialization of deep neural networks is crucial to avoid saddle points and poor local minima [7]. Thus, CMN initializes the user (**M**) and item (**E**) memory embeddings from a pretrained model according to Eqn. 8. Remaining parameters are initialized according to [9] which adapts the variance preserving Xavier initialization [7] for the ReLU activation function. The gradient is clipped if the norm exceeds 5; l_2 weight decay of 0.1 with the exception of the user and item memories; and the mini-batch size is set to 128 for *Epinions*, *citeulike-a* and 256 for *Pinterest*. We adopt the RMSProp [7] optimizer with a learning rate of 0.001 and 0.9 for both decay and momentum. The default number of hops is set to 2 and memory or embedding size d to 40, 50 and 50 for the *Epinions*, *citeulike-a* and *Pinterest* datasets respectively. The effects of these hyperparameters are further explored in Sections 4.5 and 4.7.

4.4 Baseline Comparison

Table 2 lists results of CMN with one, two and three hops along with the baselines for HR and NDCG with cut offs at 5 and 10 on the *Epinions*, *citeulike-a* and *Pinterest* datasets. We denote CMN with one hop as 'CMN-1', two hops with 'CMN-2' and so on. At a high-level CMN variants obtain the best performance across both HR and NDCG at all cut offs for all datasets. We now provide a detailed breakdown of our results on the *Epinions* dataset. All baselines with the exception of KNN show competitive performance with each other across all metrics and cut offs. Since CMN shares the same loss function with BPR, FISM and GMF, we can attribute the performance increase to the memory component. The application of a nonlinear transformation does not necessarily help as evident from BPR outperforming its nonlinear counterpart GMF. KNN demonstrated the poorest performance particularly due to the restrictive ability to handle sparse data when only a few neighbors are present and a large number (139k) of items. However, FISM's learned similarity function performs better since it can address missing entries in the item-item similarity matrix. On the other hand, CMN can leverage the global structure of the latent factors encoded in the memory vectors and the additional memory component to infer complex user preferences. Furthermore, CMN's performance gains over CDAE, GMF and NeuMF portray the successful integration of the memory component and attention mechanism over existing nonlinear and deep learning-based methods.

The denser *citeulike-a* dataset contains fewer items than the *Epinions* dataset leading to competitive performance from the item-based KNN method obtaining the strongest baseline NDCG@5. SVD++ outperforms the neighborhood-based FISM and latent factor BPR revealing the effectiveness of combining two approaches into a single hybrid model. The linear decomposition of the item-item similarity matrix in FISM may lack the expressiveness to capture complex preferences as suggested by the nonlinear GMF outperforming the linear BPR. CMN demonstrates improved performance over the fixed neighborhood-based weighting of KNN and the learned linear similarity scheme of FISM indicating the additional nonlinear transformation and adaptive attention mechanism captures more complex semantic relations between users. CMN can be viewed as NeuMF by replacing the memory network component with a multilayer perceptron. CMN outperforming NeuMF further establishes the advantage of the memory network component to identify complex interactions and iteratively update the internal neighborhood state.

In the *Pinterest* dataset, CMN with a single hop demonstrates competitive performance to baseline methods but with additional hops performance is further enhanced. SVD++ demonstrates competitive performance but may lack the full expressiveness of nonlinearity found in deep learning-based models to capture latent user-item relations. The larger dataset helps the two deep learning baselines to outperform the non-deep learning-based methods but the hybrid nature of CMN allows the joint nonlinear exploitation of



(a) *Epinions* dataset **(b) *citeulike-a* dataset** **(c) *Pinterest* dataset**

Figure 2: Experimental results for CMN varying the embedding size from 20-100 and hops from 1-4.

	Epinions				*citeulike-a*				*Pinterest*			
	HR@5	HR@10	NDCG@5	NDCG@10	HR@5	HR@10	NDCG@5	NDCG@10	HR@5	HR@10	NDCG@5	NDCG@10
CMN	0.6017	0.7007	0.4724	0.5045	0.7959	0.8921	0.6185	0.6500	0.7267	0.8904	0.5180	0.5714
CMN-Attn	0.5807	0.6948	0.4438	0.4809	0.7411	0.8589	0.5503	0.5887	0.6995	0.8773	0.4949	0.5530
CMN-Linear	0.5954	0.6977	0.4660	0.4992	0.7721	0.8665	0.5974	0.6282	0.6992	0.8777	0.4951	0.5534
CMN-Linear-Attn	0.5830	0.6937	0.4457	0.4816	0.7649	0.8676	0.5922	0.6256	0.6996	0.8775	0.4947	0.5527

Table 3: CMN variants without attention (CMN-Attn); linear activation with attention (CMN-Linear); and linear without attention (CMN-Linear-Attn).

the local neighborhood and global latent factors yielding additional performance gains. Overall, CMN with two hops outperforms CMN with a single hop but supplementing additional hops greater than two did not provide significant advantages.

4.5 Embedding Size

We illustrate the effect of varying the size of memory slots or embeddings and the number of hops for HR@10 and NDCG@10 on the *Epinions* dataset in Figure 2a. Since HR and NDCG show similar patterns, we focus our analysis on NDCG. The general trend shows a steady improvement as the embedding size increases with the exception of a single hop where an embedding size of 40 shows peak HR@10 performance followed by a degradation potentially due to overfitting. A single hop confines the model's expressiveness to infer and discriminate between the relevant and irrelevant information encoded in each memory cell. With a small embedding size of 20 increasing the number of hops provides negligible benefits but as the embedding size increases multiple hops show significant improvement over a single hop.

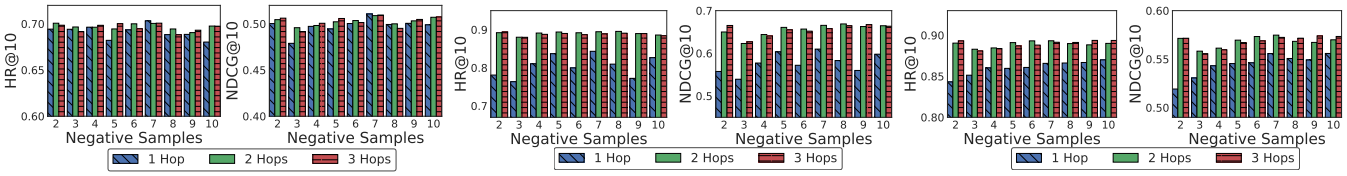
For the *citeulike-a* dataset, Figure 2b portrays the best performance of a single hop at an embedding size of 20 followed by a degradation as model capacity increases which is somewhat similar to the *Epinions* dataset. At three hops and an embedding size of 40 shows an unusual drop in performance potentially from finding a poor local minima due to the nonconvex nature of neural networks. Two and four hops show almost identical performance with at most a deviation of 0.3% from each other on HR and NDCG. In general, two hops demonstrates competitive performance against the three and four hop models across all embedding sizes.

The *Pinterest* dataset in Figure 2c shows a similar trend of gradual performance gains as the embedding size increases but a single hop shows insufficient capacity to model complex user-item interactions. Unlike the results from the previous datasets the performance of a single hop does not degrade as the embedding size increases. The larger dataset may provide some implicit form of regularization to prevent overfitting. Two, three and four hops show similar performance and incremental with larger embedding sizes.

Identifying a sufficient number of hops initially takes precedence over the size of the embeddings. With a sufficient number of hops the embedding size can be increased yielding a trade off between computational cost and performance. By introducing additional hops, CMN can better manipulate the memories and internal state to represent more complex nonlinear relations. Consistent with previous results, the addition of more than two hops do not show significant benefit.

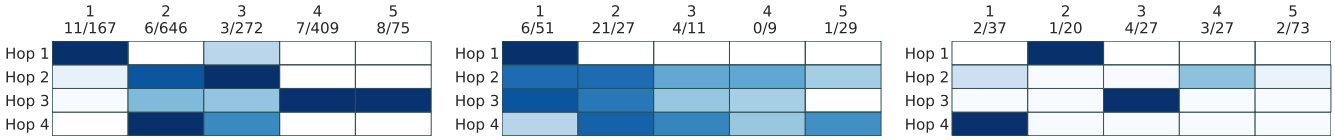
4.6 Effects of Attention and Nonlinearity

In this section, we seek to further understand the effect of individual components on performance. In Table 3, the results for CMN without attention denoted 'CMN-Attn' uniformly performs worse than with attention hinting at the effectiveness of the attention mechanism. We also experimented with a linear version of CMN where all ReLU activation functions are set to the identity function denoted as 'CMN-Linear'. The linear version with the attention mechanism generally outperforms the nonlinear version without attention. Further illustrating the effectiveness of the attention mechanism. The final variation removes the attention mechanism from the linear version denoted as 'CMN-Linear-Attn' which generally performs worse than the linear version with the attention mechanism. In general, removing the nonlinear transformation and attention mechanism in some variation yield similar performance on the *Epinions* and *Pinterest* datasets. In the *citeulike-a* dataset the linear version with and without the attention mechanism show improvements over the nonlinear variant without the attention mechanism. This seems counter intuitive and may indicate a potential difficulty in finding a good local minima or a vanishing gradient problem which is consistent with the unusual drop in performance reported in Section 4.5. CMN requires a combination from both the attention mechanism and nonlinear transformations to yield the best performance.



(a) *Epinions* dataset (b) *citeulike-a* dataset (c) *Pinterest* dataset

Figure 3: Experimental results for CMN varying the number of negative samples from 2-10 and hops from 1-3.



(a) *Epinions* dataset (b) *citeulike-a* dataset (c) *Pinterest* dataset

Figure 4: Heatmap of the attention weights over four hops. The color scale indicates the intensities of the weights, darker representing a higher weight and lighter a lower weight. Each column represents a user in the neighborhood labeled with the number of items in common with the target user / number of ratings in training set. For example, 11/167 indicates user 1 has 11 items corated with the current user u and has rated a total of 167 items in the training set.

4.7 Negative Sampling

In this section, we study the characteristics of varying the negative samples for CMN reporting HR@10 and NDCG@10. We exclude the results of four hops since the results were consistent with that of three hops. We also omit 1 negative sample since CMN was unable to distinguish between positive and negative samples leading to random performance. Figure 3a illustrates the performance of CMN varying the negative samples from 2-10 on the *Epinions* dataset. A single hop shows fluctuations reporting low HR at 5 and 10 negative samples but outperforms the two and three hop versions with 7 negative samples. A single hop uniformly performs worse than the two and three hop counterparts in the *citeulike-a* dataset presented in Figure 3b. In Figure 3c, the results for a single hop on the *Pinterest* dataset describes a general upward trend where performance improves as the number of negative samples increase. In both the *citeulike-a* and *Pinterest* datasets we observe two and three hops show comparable results and more stability to the number of negative samples while outperforming a single hop. Overall, the performance of CMN is fairly stable with respect to the number of negative samples when at least two hops are present. Similar to the previous section on embedding size, we notice having at least two hops reduces the sensitivity to the hyperparameter.

4.8 Attention Visualization

Attention mechanisms allow us to visualize the weights placed on each user in the neighborhood with the hope of providing interpretable recommendations. We plot a heatmap of the weights from Eqn. 2 in Figure 4. The color scale represents the intensities of the attention weights, where a darker color indicates a higher value and lighter colors indicate a lower value. For ease of reference, we label each column representing a user in the neighborhood starting from 1 which may not necessarily reflect the true user id from the dataset. Furthermore, for each user we provide additional context in the form of user statistics. We denote 11/167 to indicate the user has rated a total of 167 items with 11 items observed in common with the target user in the training set. Since the size of the neighborhood can be large we limit the visualization to top 5 neighbors sorted by the highest aggregated attention values. We would like to point out that in some cases the attention weights can be small and hence not visually distinguishable.

The attention weights for a random user from the *Epinions* dataset is portrayed in Figure 4a. The user has a total of 49 neighbors thus we show only the top 5 neighbors due to space constraints. We can see all the top users attended to have at least a single item in common. The first hop places heavy levels of attention on user 1 and lightly on user 3. At two hops the attention on user 3 increases. Progressing to three hops the attention spread out across four users. Finally, at four hops user 2 and 3 have the highest weight with a balance of the number items in common with the target user and overall number of ratings observed suggesting these users may be the most influential in the recommendation process. As shown in previous sections performance generally increases with additional hops suggesting considering a combination of multiple users may be beneficial.

Figure 4b illustrates the attention weights over four hops for a random user from the *citeulike-a* dataset with a total of 9 neighbors. We observe the first hop places a large amount of weight on a single user which may explain the poorer performance of CMN with a single hop. User 1 has the highest number of observations out of the neighborhood which may be a reasonable choice but it ignores other information that may be present from other users. Examining the weights of the second hop we see the attention is spread out across five users with a higher emphasis on users 1 and 2 who have the most items in common with the target user. Four out of the five users have a common item with the target user providing a strong indicator of the successful integration of the attention mechanism.

Next, we focus on three hops which removes the attention over user 5 and reduces the intensities on user 4, 2 and 3. In the final hop, attention is returned to user 5 with stronger weights than in hop two. The overall attention levels shift around slightly but focus most heavily on user 2 which makes sense since it has the highest number of commonly rated items. Since user 4 has no items in common with the target user but large attention weights this warranted further investigation. We found user 4 to have at least one item in common with all other users in the neighborhood which may explain the attention placed on user 4 despite no corated items with the target user in the training data. This demonstrates the memory component captures higher level interactions within the neighborhood suggesting some form of transitive reasoning.

Figure 4c illustrates the attention weights over four hops for a random user from the *Pinterest* dataset. Similar to the previous visualization the first hop places heavy weights on a single user followed by a more dispersed weighting in the following hops. In hops two through four, a small amount of attention is placed upon each user which may not be visually distinguishable. Each hop allows CMN to examine the external memory and perhaps through some form of trial and error arrives at identifying the most useful neighbor as user 1 in hop four.

5 CONCLUSION

We introduced a novel hybrid architecture unifying the strengths of the latent factor model and neighborhood-based methods inspired by Memory Networks to address collaborative filtering (CF) with implicit feedback. We reveal the connection between components of Collaborative Memory Network (CMN) the two important classes of CF models and draw parallels with the original memory network framework. Comprehensive experiments under multiple configurations demonstrate significant improvements over competitive baselines. Qualitative visualization of the attention weights provide insight into the model's recommendation process and suggest higher order transitive relations may be present. In future work, we hope to extend CMN to incorporate content and context information, tackle dialogue-based systems, and perform adversarial training [7]. Adversarial training is particularly attractive since it enables the model learn an implicit similarity metric to characterize the data rather than a surrogate loss function approximating the AUC.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

[2] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive collaborative filtering: Multimedia recommendation with feature-and item-level attention. In *SIGIR*.

[3] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & Deep Learning for Recommender Systems. In *RecSys*.

[4] Travis Ebesu and Yi Fang. 2017. Neural Semantic Personalized Ranking for item cold-start recommendation. *Information Retrieval Journal* 20, 2, 109–131.

[5] Xue Geng, Hanwang Zhang, Jingwen Bian, and Tat-Seng Chua. 2015. Learning image and user features for recommendation in social networks. In *ICCV*.

[6] Yuyun Gong and Qi Zhang. 2016. Hashtag Recommendation Using Attention-Based Convolutional Neural Network.. In *IJCAI*.

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[8] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature* 538, 7626 (2016), 471–476.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *CVPR*.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

[11] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*.

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

[13] Haoran Huang, Qi Zhang, Yeyun Gong, and Xuanjing Huang. 2016. Hashtag Recommendation Using End-To-End Memory Networks with Hierarchical Attention.. In *COLING*.

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

[15] Santosh Kabbur, Xia Ning, and George Karypis. 2013. Fism: factored item similarity models for top-n recommender systems. In *SIGKDD*.

[16] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. 2016. Convolutional Matrix Factorization for Document Context-Aware Recommendation. In *RecSys*.

[17] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *SIGKDD*.

[18] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask Me Anything: Dynamic Memory Networks for Natural Language Processing. In *ICML*.

[19] Cheng Li, Xiaoxiao Guo, and Qiaozhu Mei. 2017. Deep Memory Networks for Attitude Identification. In *WSDM*.

[20] Sheng Li, Jaya Kawale, and Yun Fu. 2015. Deep Collaborative Filtering via Marginalized Denoising Auto-encoder. In *CIKM*.

[21] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.Com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing* 7, 1.

[22] Paolo Massa and Paolo Avesani. 2007. Trust-aware Recommender Systems. In *RecSys*.

[23] Xia Ning and George Karypis. 2011. Slim: Sparse linear methods for top-n recommender systems. In *ICDM*.

[24] Steffen Rendle. 2010. Factorization machines. In *ICDM*.

[25] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-thieme. 2009. BPR : Bayesian Personalized Ranking from Implicit Feedback. *UAI* (2009).

[26] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. *Introduction to recommender systems handbook*. Springer.

[27] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann machines for collaborative filtering. In *ICML*.

[28] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. AutoRec : Autoencoders Meet Collaborative Filtering. *WWW*.

[29] 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 *RecSys*.

[30] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-To-End Memory Networks. In *NIPS*.

[31] Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. *NIPS*.

[32] Chong Wang and David M Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *SIGKDD*.

[33] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *SIGKDD*.

[34] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models. In *SIGIR*.

[35] Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory Networks. In *ICLR*.

[36] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J. Smola, and How Jing. 2017. Recurrent Recommender Networks. In *WSDM*.

[37] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. 2016. Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. In *WSDM*.

[38] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *IJCAI*.

[39] Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *ICML*.

[40] Shuai Zhang, Lina Yao, and Aixin Sun. 2017. Deep Learning based Recommender System: A Survey and New Perspectives. *arXiv:1707.07435* (2017).

[41] Shuai Zhang, Lina Yao, and Xiwei Xu. 2017. AutoSVD++: An Efficient Hybrid Collaborative Filtering Model via Contractive Auto-encoders. In *SIGIR*.