Collaborative Memory Networks for Recommendation with Unstructured Text

AAAI Press
Association for the Advancement of Artificial Intelligence
2275 East Bayshore Road, Suite 160
Palo Alto, California 94303

Abstract

Collaborative filtering (CF) is the key technique for recommender systems. Pure CF approaches exploit the user-item interaction data (e.g., clicks, likes, and views) only and hence suffer from the data sparsity issue. Items are usually associated with content information such as abstracts of articles and reviews of products. CF methods can be extended to integrate unstructured text, leading to hybrid filtering. In this paper, we develop a unified neural framework to exploit both user-item interactions and unstructured text seamlessly. The proposed framework, Collaborative Memory Networks (CoMem), estimates the probability that a user will like an item by two terms: 1) behavior factors — the personalized preference of the user to the given item, and 2) semantic factors — the high-level representation attentively extracted from the text which matches word semantics with user preferences. On real-world datasets, CoMem shows better performance by comparing with various baselines in terms of three ranking metrics.

Introduction

Recommender systems are widely used in various domains and e-commerce platforms, such as to help consumers buy products at Amazon, watch videos on Youtube, and read articles on Google News. They are useful to alleviate the information overload and improve user satisfaction. Given the history records of consumers such as the product transactions, Web page browsing and movie watching, collaborative filtering (CF) is among the most effective approaches based on the simple intuition that if users rated items similarly in the past then they are likely to rate items similarly in the future (Sarwar et al 2001).

The history records include both implicit (e.g., purchase and clicks) and explicit (e.g., likes/dislikes and ratings) feedback which can be represented as a user-item interaction matrix. Usually, the observed user-item interactions are partial with a large portion remaining not recorded. The goal of recommendation is to predict the user preferences on the missing item interactions. This setting requires to complete the partial observed rating matrix. Matrix factorization (MF) techniques which can learn the latent factors for users and items are the main cornerstone for CF (Mnih and Salakhutdinov 2008; Koren et al 2008; Koren et al 2009). It is effective and flexible to be integrated with additional data sources. Recently, neural networks like multilayer perceptron (MLP) are used to learn the interaction function from data (Dziugaite and Roy 2015; Cheng et al 2016; He et al 2017) with the power of learning highly nonlinear relationships between users and items. MF and neural CF exploit the user-item behavior interactions only and hence suffer from the data sparsity and cold-start issues.

Items are usually associated with content information such as unstructured text, like the news articles and product reviews. These additional sources which provide independent and diverse information are essential for recommendation beyond user-item interaction data, and hence can alleviate the data sparsity issue (Ganu et al 2009). For application domains like recommending research papers and news articles, the unstructured text associated with the item is its text content (Wang and Blei 2011; Wang et al 2015; Bansal et al 2016). Other domains like recommending products, the unstructured text associated with the item is its user reviews which justify the rating behavior (McAuley and Leskovec 2013; He et al 2016; Hu and Dai 2017). These methods adopt topic modelling techniques or neural networks to exploit the item content leading to performance improvement.

The typical way of exploiting text is to extract a feature vector from the text document by averaging the word embeddings pre-trained from a large corpus such as Wikipedia (Hu and Dai 2017; He et al 2016; Ma et al 2018). These methods separate the extraction of text feature from the learning of user-item interaction. As a result the two processes can not benefit from each other. Other approaches learn a topic vector using topic modelling (Wang and Blei 2011; McAuley and Leskovec 2013; Bao et al 2014) by aligning behavior factors and topic factors with a link function such as softmax and offset. Recently, neural networks are used to learn a representation from the text using autoencoders (Wang et al 2015; Zhang et al 2016), recurrent networks (Bansal et al 2016), and convolutional networks (Zheng et al 2017; Catherine and Cohen 2017). These methods treat different words in the document as equal importance and do not match word semantics with the specific user.

In this paper, we propose a novel neural framework to exploit interaction data and content information seamlessly. The proposed framework, Collaborative Memory Networks (CoMem), fuses semantic representations learnt from unstructured text with behavior representations learnt from user-item interactions for effective estimation on user preferences' items. CoMem estimates the probability that a user will like an item by two factors. The *behavior factor* is to capture the personalized preference of the user to the given item. And the *semantic factor* is to capture the high-level representation attentively extracted from the unstructured text matching word semantics with user preferences. These two effective representations are used to learn user preferences on items.

To model the behavior factor, we adopt the same approach as neural CF, which learns the user-item nonlinear interaction relationships using a neural network (CFNet). To model the semantic factor, we adopt memory networks to match word semantics with the specific user via the attention mechanism inherent in the memory module (MemNet), determining which words are highly relevant to the user preferences. As far as we know, CoMem is the first deep model that integrates interactions data with unstructured text by bridging neural CF and memory networks.

Related Work

We review the related works on collaborative filtering and hybrid filtering which integrates unstructured text.

Matrix factorization Matrix factorization (MF) technique is the main cornerstone for collaborative filtering (CF) since it can learn the latent factors for users and items. MF techniques have gained popularity and become the standard recommender approaches due to their accuracy and scalability (Koren et al 2008; Koren et al 2009). MF has probabilistic interpretation with Gaussian noise (Mnih and Salakhutdinov 2008. MF is flexible to add side data sources for recommender such as text (Wang and Blei 2011), speech (Van den Oord et al 2013), and visual images (He et al 2016). Bayesian personalized ranking (BPR) (Rendle 2009) uses a pair-wise loss to learn from the implicit feedback (Hu et al 2008; Pan et al 2008). MF uses a fixed dot product to compute the matching degree between users and items. It can not capture the transitivity of user-user and item-item (Hsieh et al 2017) and can not model the highly nonlinear relationships between users and items (He et al 2017).

To address the data sparsity, an item-item matrix (SPPM) is constructed from the user-item interaction matrix in the CoFactor model (Liang et al 2016). It then simultaneously factorizes the interaction matrix and the SPPMI matrix in a shared item latent space, enabling the usage of co-click information to regularize the learning of user-item matrix. In contrast with our method, CoMem uses independent unstructured text to alleviate the data sparsity issue in the user-item matrix and at the same time uses neural CF to exploit the interaction data.

Neural collaborative filtering Recently, neural networks (NN) have been proposed to directly parameterize the interaction function between users and items. The MF Autoencoder (van Baalen 2016) and NNMF model (Dziugaite and Roy 2015) parameterize the interaction function by a multilayer feedforward (FF) NN to implement the function of matrix factorization. The wide & deep model (Cheng et al 2016), multilayer perceptron (MLP) (He et al 2017), and deep MF (Xue et al 2017) also use FFNNs by firstly embedding the discrete indices of user and item to continuous vectors and then going through several nonlinear hidden layers to learn high-level representations. The basic MLP architecture is extended to regularize the factors of users and items by additional data source such as social and geographical information (Yang et al 2017). Autoencoders (Sedhain et al 2015), convolutional networks (Zheng et al 2017; Catherine and Cohen 2017), recurrent networks (Wu et al 2017) are also proposed to learn from explicit feedback for rating prediction. In contrast with our method, CoMem uses memory networks to model the unstructured text to alleviate the data sparsity issues and evaluates on the top-K recommendation setting (Cremonesi et al 2010; Wu et al 2016).

Topic modelling Combining matrix factorization and topic modelling technique (Topic MF) is an effective way to integrate ratings with item contents (Wang and Blei 2011; Ling et al, 2014). Item reviews justify the rating behavior of a user, and item ratings are associated with their attributes hidden in reviews (Ganu et al 2009). Topic MF methods combine item latent factors in ratings with latent topics in reviews (McAuley and Leskovec 2013; Bao et al 2014). The behavior factors and topic factors are aligned with a link function such as softmax transformation in the hidden factors and hidden topics (HFT) model (McAuley and Leskovec 2013) or an offset deviation in the collaborative topic regression (CTR) model (Wang and Blei 2013). The CTR model assumes the item latent vector learnt from the interaction data is close to the corresponding topic proportions learnt from the text content, but allows them to be divergent from each other if necessary. In contrast with our method, these methods treat different words in the document as equal importance and do not match word semantics with the specific user.

Neural hybrid filtering Additional sources of information are integrated into CF to alleviate the data sparsity issues. Convolutional networks (CNNs) have been used to extract the features from audio signals for music recommendation (Van den Oord et al 2013) and from image for product (He et al 2016) and multimedia (Chen et al 2017) recommendation. Autoencoders are used to learn an intermediate representations from text (Wang et al 2015; Zhang et al 2016). Recurrent networks (Bansal et al 2016) and convolutional networks (Kim et al 2016; Zheng et al 2017; Catherine and Cohen 2017) can exploit the word order when learning the text representations. In contrast with our method, CoMem integrates interactions data with unstructured text by bridging neural CF and memory networks which can match word semantics with the specific user.

The CoMem Framework

We propose Collaborative Memory Networks (CoMem) that explains users' preferences by capturing both behavior preferences of users for items and semantic matching of words with the specific user. We firstly give the notations and problem formulation, and then we will introduce the architecture of CoMem followed by the memory component. We will describe how to learn the model parameters in the end.

Table 1: Notation.

Symbol	Description
$\mathcal{U}, \mathcal{I}$	The set of users and of items
m, n	The number of users ($m = \|\mathcal{U}\|$) and of items ($n = \|\mathcal{I}\|$)
r_{ui}	The rating of item i by user u
$\boldsymbol{R}$	The user-item rating matrix
d_{ui}	The document of item i by user u
$\mathcal{V}$	The word vocabulary
$\boldsymbol{x}_u, \boldsymbol{x}_i$	The embedding of user u and of item i
$\boldsymbol{P}, \boldsymbol{Q}$	The embedding matrix of user u and of item i
$\boldsymbol{A}, \boldsymbol{C}$	The internal and external matrices in the memory network
$\boldsymbol{W}$	Weight matrix of the hidden layer in the CF network
$\boldsymbol{h}$	Weight vector of the softmax layer in the output layer

Problem formulation

For collaborative filtering with implicit feedback, there is a binary matrix $\boldsymbol{R} \in \mathbb{R}^{m \times n}$ to describe user-item interactions where each entry $r_{ui} \in \{0, 1\}$ is 1 (called observed entries) if user u has an interaction with item i and 0 (unobserved) otherwise:

$$r_{ui} = \begin{cases} 1, & \text{if user-item interaction } (u, i) \text{ exists;} \\ 0, & \text{otherwise.} \end{cases}$$

Denote the set of m-sized users by $\mathcal{U}$ and n items by $\mathcal{I}$. Usually the interaction matrix is very sparse since a user $u \in \mathcal{U}$ only consumed a very small subset of all items. Similarly for the task of item recommendation, each user is only interested in identifying top-K items. The items are ranked by their predicted scores:

$$\hat{r}_{ui} = f(u, i|\Theta), \tag{1}$$

where f is the interaction function and Θ denotes model parameters.

For MF-based CF approaches, the interaction function f is fixed and computed by a dot product between the user and item vectors. For neural CF, neural networks are used to parameterize function f and learn it from interaction data:

$$f(\boldsymbol{x}_{ui}|\boldsymbol{P}, \boldsymbol{Q}, \theta_f) = \phi_o(...(\phi_1(\boldsymbol{x}_{ui}))...), \tag{2}$$

where the input $\boldsymbol{x}_{ui} = [\boldsymbol{P}^T \boldsymbol{x}_u, \boldsymbol{Q}^T \boldsymbol{x}_i] \in \mathbb{R}^d$ is vertically concatenated from that of user and item embeddings, which are projections of their one-hot encodings $\boldsymbol{x}_u \in \{0, 1\}^m$ and $\boldsymbol{x}_i \in \{0, 1\}^n$ by embedding matrices $\boldsymbol{P} \in \mathbb{R}^{m \times d/2}$ and $\boldsymbol{Q} \in \mathbb{R}^{n \times d/2}$, respectively ($d = d/2 + d/2$). The output and hidden layers are computed by ϕ_o and $\{\phi_l\}$ in a multilayer neural network.

Items are associated with unstructured text like reviews of products. For the document of item i by user u, denote the words in it as $d_{ui} = \{w_1, ..., w_j\}$ where the words come from a vocabulary $\mathcal{V}$. Neural CF can be extended to leverage text and then the interaction function has the form of $f(u, i, d_{ui}|\Theta)$.

Architecture

The architecture for the proposed CoMem model is illustrated in Figure 1. Besides the layers of input, embedding, and output, CoMem consists of a CF network (CFNet) to learn nonlinear interaction relationships between users and items

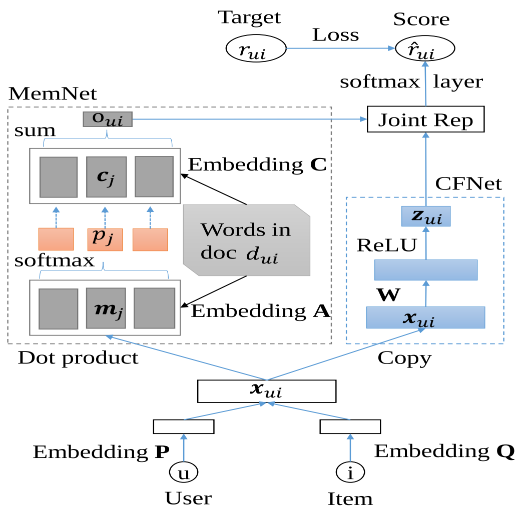

Figure 1: The architecture of CoMem.

and of a memory network (MemNet) to learn text representations matching word semantics with specific users.

The information flow in CoMem goes from the input (u, i) to the output $\hat{r}_{ui}$ through the following layers:

- **Input:** $(u, i) \rightarrow \boldsymbol{x}_u, \boldsymbol{x}_i$ This module encodes user-item interaction indices. We adopt the one-hot encoding. It takes user u and item i, and maps them into one-hot encodings $\boldsymbol{x}_u \in \{0, 1\}^m$ and $\boldsymbol{x}_i \in \{0, 1\}^n$ where only the element corresponding to that index is 1 and all others are 0.
- **Embedding:** $\boldsymbol{x}_u, \boldsymbol{x}_i \rightarrow \boldsymbol{x}_{ui}$ This module embeds one-hot encodings into continuous representations and then concatenates them as
$$\boldsymbol{x}_{ui} = [\boldsymbol{P}^T \boldsymbol{x}_u, \boldsymbol{Q}^T \boldsymbol{x}_i], \tag{3}$$
to be the input of following building blocks.
- **CFNet:** $\boldsymbol{x}_{ui} \rightsquigarrow \boldsymbol{z}_{ui}$ This module is a pure CF approach to exploit user-item interaction data. It takes the continuous representations from the embedding module and then transforms to a final *behavior factor* representation:
$$\boldsymbol{z}_{ui} = \text{ReLU}(\boldsymbol{W}\boldsymbol{x}_{ui} + \boldsymbol{b}), \tag{4}$$
where $\text{ReLU}(x) = \max(0, x)$ is the activation function and $\boldsymbol{W}$ and $\boldsymbol{b}$ are the weight and bias parameters.
- **MemNet:** $\boldsymbol{x}_{ui} \rightsquigarrow \boldsymbol{o}_{ui}$ This module is to model the item content with the guidance of interaction data. The item content is modelled by memories. It takes both representations from the embedding module and the text associated with the corresponding user-item to a final *semantic factor* representation:
$$\boldsymbol{o}_{ui} = \text{MemNet}(\boldsymbol{x}_{ui}, d_{ui}), \tag{5}$$
where MemNet denotes the computing function in the memory module (described later).
- **Output:** $[\boldsymbol{z}_{ui}, \boldsymbol{o}_{ui}] \rightarrow \hat{r}_{ui}$ This module predicts the score $\hat{r}_{ui}$ for the given user-item pair based on the concatenated representation from behavior factor and semantic factor:
$$\boldsymbol{y}_{ui} = [\boldsymbol{z}_{ui}, \boldsymbol{o}_{ui}] \tag{6}$$

from the CFNet and MemNet. The output is the probability that the input pair is a positive interaction. This can be achieved by a softmax/logistic layer:

$$\hat{r}_{ui} = \text{softmax}(\boldsymbol{y}_{ui}) = \frac{1}{1 + \exp(-\boldsymbol{h}^T \boldsymbol{y}_{ui})}, \tag{7}$$

where $\boldsymbol{h}$ is the parameter.

MemNet: Integrating Unstructured Text

We have used a single hidden layer to learn nonlinear interaction relationships between users and items in the CFNet. CF captures the behavior factor of user preferences on the items by learning from the interaction data. To model the unstructured text, we use a memory network (MemNet) to attentively extract useful content to match the word semantics with specific user where different words in the text document have different weights in the semantic factor.

Memory networks (Sukhbaatar et al 2015) are firstly proposed to address the question answering (QA) task where memories are a short story and the query is a question related to the text in which the answer can be reasoned by the network. We can think of the recommendation with text as a QA problem: the question to be answered is to ask how likely a user prefers an item. And the unstructured text is analogue to the story and the query is analogue to the user-item interaction.

Memory networks have been used in recommendation to model item content (Hu et al 2018; Huang et al 2017), model users' neighborhood (Ebesu et al 2018), and learn latent relationships (Tay et al 2018). We use memory networks to attentively extract important information from the text content via the attention mechanism which can match word semantics with the specific user and determine which words are highly relevant to the user preferences.

Recall that a memory network has one internal memory matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times d}$ where typically the size of vocabulary is reserved as $|\mathcal{V}| = 8000$ (Wang & Blei 2011) and d is the dimension of each memory slot. And another external memory matrix $\mathbf{C} \in \mathbb{R}^{|\mathcal{V}| \times d}$ is the same dimensions as $\mathbf{A}$. The computation path $\boldsymbol{o}_{ui} = \text{MemNet}(\boldsymbol{x}_{ui}, d_{ui})$ in the MemNet goes through $(\boldsymbol{x}_{ui}, d_{ui}) \overset{\boldsymbol{A}}{\to} \boldsymbol{a}^{ui} \overset{\boldsymbol{C}}{\to} \boldsymbol{o}_{ui}$ described in the following, and the function of the two memory matrices can work as we expect to effectively model unstructured text by matching the word semantics with specific users.

Given a doc $d_{ui} = (w_1, ..., w_j)$ corresponding to the user-item (u, i) interaction, we form the memory slots $\boldsymbol{m}_j \in \mathbb{R}^d$ by mapping each word w_j into an embedding vector with matrix $\mathbf{A}$. We get a preference vector $\boldsymbol{a}^{u,i} = [a_j^{u,i}]$ corresponding to the given document d_{ui} and the user-item interaction (u, i) where each element $a_j^{u,i}$ encodes the relevance of user u to these words $\{w_j\}$ given item i:

$$a_j^{u,i} = \boldsymbol{x}_u^T \boldsymbol{m}_j^{(u)} + \boldsymbol{x}_i^T \boldsymbol{m}_j^{(i)}. \tag{8}$$

On the right hand of the above equation, the first term captures the matching between preferences of user u and word semantics. The second term computes the support of item i to the words. Together, the content-based addressing scheme can determine internal memories with highly relevance to the specific user u regarding the words d_{ui} given item i.

We compute the attentive weights over words for a given user-item interaction to infer the importance of each word's unique contribution:

$$p_j^{u,i} = \text{softmax}(a_j^{u,i}) = \frac{\exp(a_j^{u,i})}{\sum_{k'=1}^{l} \exp(a_{j'}^{u,i})}, \tag{9}$$

which produces a probability distribution over the words in d_{ui}. The neural attention mechanism allows the MemNet to focus on specific words while to place little importance on other words which may be less relevant.

We construct the high-level semantic factor representations by interpolating the external memories with the attentive weights as the output of the memory module:

$$\boldsymbol{o}_{ui} = \sum\nolimits_j p_j^{u,i} \boldsymbol{c}_j, \tag{10}$$

where external memory slot $\boldsymbol{c}_j \in \mathbb{R}^d$ is another embedding vector for word $w_j \in d_{ui}$ by mapping it with matrix $\mathbf{C}$.

Learning

Due to the nature of the implicit feedback and the task of item recommendation, the squared loss $(\hat{r}_{ui} - r_{ui})^2$ may be not suitable since it is usually for rating prediction. Instead, we adopt the binary cross-entropy loss:

$$\mathcal{L} = -\sum\nolimits_{(u,i) \in \mathcal{S}} r_{ui} \log \hat{r}_{ui} + (1 - r_{ui}) \log(1 - \hat{r}_{ui}), \tag{11}$$

where $\mathcal{S} = \boldsymbol{R}^+ \cup \boldsymbol{R}^-$ are the union of observed interaction matrix and randomly sampled negative pairs. This objective function has a probabilistic interpretation and is the negative logarithm likelihood of the following likelihood function:

$$L(\Theta|\mathcal{S}) = \prod\nolimits_{(u,i) \in \boldsymbol{R}^+} \hat{r}_{ui} \prod\nolimits_{(u,i) \in \boldsymbol{R}^-} (1 - \hat{r}_{ui}), \tag{12}$$

where model parameters $\Theta = \{\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{A}, \boldsymbol{C}, \boldsymbol{W}, \boldsymbol{b}, \boldsymbol{h}\}$. This objective function can be optimized by the stochastic gradient descent (SGD) algorithm and its variants. The update equation is: $\Theta^{new} \leftarrow \Theta^{old} - \eta \partial L(\Theta)/\partial \Theta$, where η is the learning rate. Typical deep learning library like TensorFlow (`https://www.tensorflow.org`) provides automatic differentiation and hence we omit the gradient equations $\partial L(\Theta)/\partial \Theta$ which can be computed by chain rule in back-propagation.

Connections to Existing Approaches

In this section, we reveal the relations between the proposed model and two existing approaches including MF and LCMR.

We firstly show that the CFNet of CoMem generalizes matrix factorization.

Let $\boldsymbol{P}^T \boldsymbol{x}_u$ be that of latent user factors $\underline{\boldsymbol{x}_u}$ in MF, and $\boldsymbol{Q}^T \boldsymbol{x}_i$ be that of latent item factors $\underline{\boldsymbol{x}_i}$ in MF. MF computes the predicted score by $\hat{r}_{ui} = \underline{\boldsymbol{x}_u}^T \underline{\boldsymbol{x}_i}$. We replace the concatenation (see Eq. (3)) in the embedding module with element-wise multiplication:

$$\boldsymbol{x}_{ui} = (\boldsymbol{P}^T \boldsymbol{x}_u) \odot (\boldsymbol{Q}^T \boldsymbol{x}_i). \tag{13}$$

It requires that the dimensions of latent features of users and items are the same in this case (denoted as d). We have one fixed weight $\boldsymbol{W} \equiv \mathbf{1}$ which is a d-dimensional vector of all 1-s and the bias $\boldsymbol{b} \equiv \mathbf{0}$ is all 0-s. And we do not perform nonlinear ReLU activation and insted use the identity mapping (see Eq.(4)). In this way, the prediction is the same whatever it is computed by MF or CoMem.

This allows CoMem to have the nonlinear modelling power beyond the standard matrix factorization and mimic a class of factorization models, e.g., 2-way factorization machines (Rendle 2010). As we will see in the experiments, CoMem outperforms MF methods on real-world datasets.

We now show that the CoMem is a more reasonable architecture design than an existing neural hybrid filtering method.

Theorem 1 Collaborative memory networks and LCMR (Hu et al 2018) have the same function by replacing the activation ReLU with softmax in the CFNet.

Proof Since the input, embedding, MemNet, and output layers of both CoMem and LCMR are the same, we only need to show that the CFNet of CoMem have the same function with the centralized memory blocks in LCMR. Assume that there is only one hop centralized memory block in LCMR (If there are multi-hop, we can prove it by induction).

Denote the input as $\boldsymbol{x} \in \mathbb{R}^{d_1}$ and the output as $\boldsymbol{y} \in \mathbb{R}^{d_2}$. Our CFNet has two hidden layers $\mathcal{L}_1$ and $\mathcal{L}_2$, which are called a pair $\mathcal{P} = \{\mathcal{L}_1, \mathcal{L}_2\}$, and the corresponding connection matrices are $\boldsymbol{W}_1 \in \mathbb{R}^{N \times d_1}$ and $\boldsymbol{W}_2 \in \mathbb{R}^{d_2 \times N}$ where N is the number of units in the first layer $\mathcal{L}_1$. Then the output of layer $\mathcal{L}_1$ is $\boldsymbol{o} = \boldsymbol{W}_1 \boldsymbol{x}$. After a *softmax activation*, the input to layer $\mathcal{L}_2$ is $\boldsymbol{a} = \text{Softmax}(\boldsymbol{o})$ and the final output $\boldsymbol{y} = \boldsymbol{W}_2 \boldsymbol{a}$.

For the LCMR, it has one building block $\mathcal{B} = \{\mathcal{A}, \mathcal{C}\}$ which consists of key and value memories $\mathcal{A}$ and $\mathcal{C}$, where dimensions $\boldsymbol{A} \in \mathbb{R}^{N \times d_1}$ and $\boldsymbol{C} \in \mathbb{R}^{N \times d_2}$ are compatibility with the input and output, respectively (where the size of memories is N). The attention weights $\boldsymbol{a}$ are firstly computed by $\boldsymbol{o} = \boldsymbol{A}\boldsymbol{x}$ and then normalized by *softmax function* $\boldsymbol{a} = \text{Softmax}(\boldsymbol{o})$ to be a simplex vector. The output is a sum over the values weighted by attentive weights $\boldsymbol{y} = \boldsymbol{C}^T \boldsymbol{a}$. We can see that if we let $\boldsymbol{A} = \boldsymbol{W}_1$ and $\boldsymbol{C} = \boldsymbol{W}_2^T$, then the learning functions of CoMem and LCMR are the same. □

It seems unusual to use a softmax function mapping between two hidden layer connections. The values would be fairly small causing a vanishing gradient since it would be normalized to a probability distribution. A ReLU function would typically be used in this case. As a result, we can expect that the CoMem is a more reasonable architecture than LCMR, empirically demonstrated in the experiments.

Experiments

In this section, we conduct empirical study to answer the following questions: 1) how does the proposed CoMem model perform compared with state-of-the-art recommender systems; and 2) how does the unstructured text contribute to the proposed framework. We firstly introduce the evaluation protocols and experimental settings, and then we compare the performance of different recommender systems.

Experimental Settings

Datasets We evaluate on two real-world cross-domain datasets. The public **Amazon** (`http://snap.stanford.edu/data/web-Amazon.html`) dataset has been widely used to evaluate the performance of collaborative filtering approaches (McAuley and Leskovec 2013; He et al 2016). We use the category of Amazon Men. The original ratings are from 1 to 5 where five stars indicate that the user shows a positive preference on the item while the one stars are not. We convert the ratings of 4-5 as positive samples. The dataset we used contains 56K positive ratings on Amazon Men. There are 8.5K users and 28K Men products. We aim to improve the recommendation with the product reviews. The data sparsity is over 99.7%. We filter stop words and use tf-idf to choose the top 8,000 distinct words as the vocabulary (Wang and Blei 2011). The average number of words per review is 32.9. The second dataset, **Mobile**, is provided by a large internet company, i.e., Cheetah Mobile (`http://www.cmcm.com/en-us/`). The information contains logs of user reading news, the history of app installation, and some metadata such as news publisher and user gender collected in one month in the US. We removed users with fewer than 10 feedbacks. For each item, we use the news title as its text content. We filter stop words and use tf-idf to choose the top 8,000 distinct words as the vocabulary. This yields a corpus of 612K words. The average number of words per news is less than 10. The dataset we used contains 477K user-news reading records. There are 15.8K users and 84K news. We aim to improve the news recommendation by exploiting news titles. The data sparsity is over 99.6%. The statistics of the two datasets are summarized in Table 3.

Table 3: Datasets and Statistics.

Dataset	Amazon	Mobile
#Users	8,514	15,890
#Items	28,262	84,802
#Feedback	56,050	477,685
#Words	1,845,387	612,839
Rating Density (%)	0.023%	0.035%
Avg. Words per Item	65.3	7.2

Evaluation protocols For item recommendation task, the leave-one-out (LOO) evaluation is widely used and we follow the protocol in (He et al 2017). That is, we reserve one interaction as the test item for each user. We determine hyperparameters by randomly sampling another interaction per user as the validation set. We follow the common strategy which randomly samples 99 (negative) items that are not interacted by the user and then evaluate how well the recommender can rank the test item against these negative ones.

Since we aim at top-K item recommendation, the typical evaluation metrics are hit ratio (HR), normalized discounted cumulative gain (NDCG), and mean reciprocal rank (MRR), where the ranked list is cut off at $topK = \{5, 10, 20\}$. HR intuitively measures whether the reserved test item is present on the top-K list, defined as:

$$HR = \frac{1}{|\mathcal{U}|} \sum\nolimits_{u \in \mathcal{U}} \delta(p_u \le topK),$$

where p_u is the hit position for the test item of user u, and

Table 2: Comparison results of different methods on Amazon dataset. The best baselines are marked with asterisks(*) and the best results are boldfaced. The last two columns are the relative improvements of our method.

Cutoff	Metric	Methods						Improvement of CoMem vs.		
		BPRMF	HFT	TextBPR	MLP	LCMR	CoMem	MLP	TextBPR	LCMR
topK=5	HR	0.0810	0.1077	0.1517	0.2100*	0.2024	**0.2352**	12.00%	55.04%	16.20%
	NDCG	0.0583	0.0815	0.1208	0.1486*	0.1451	**0.1646**	7.61%	28.87%	9.63%
	MRR	0.0509	0.0729	0.1104	0.1283*	0.1263	**0.1413**	6.19%	20.36%	7.41 %
topK=10	HR	0.1204	0.1360	0.1777	0.2836*	0.2836*	**0.3186**	12.34%	79.29%	12.34%
	NDCG	0.0710	0.0907	0.1291	0.1697*	0.1678	**0.1915**	7.68%	35.11%	8.35%
	MRR	0.0561	0.0767	0.1138	0.1371*	0.1356	**0.1524**	5.39%	21.72%	5.92%
topK=20	HR	0.1821	0.2782	0.2268	0.3820	0.3951*	**0.4221**	10.49%	86.11%	6.83%
	NDCG	0.0864	0.1252	0.1414	0.1899	0.1918*	**0.2175**	7.22%	33.55%	6.50%
	MRR	0.0602	0.0854	0.1171	0.1426*	0.1420	**0.1595**	4.42%	18.69%	4.42%

$\delta(\cdot)$ is the indicator function. NDCG and MRR also account for the rank of the hit position, respectively defined as:

$$NDCG = \frac{1}{|\mathcal{U}|} \sum\nolimits_{u \in \mathcal{U}} \frac{\log 2}{\log(p_u + 1)},$$

and

$$MRR = \frac{1}{|\mathcal{U}|} \sum\nolimits_{u \in \mathcal{U}} \frac{1}{p_u}.$$

NDCG provides a good discriminative power (Valcarce et al 2018). A higher value with lower cutoff indicates better performance.

Baselines We compare with various baselines as summarized in the following table.

Baselines	Shallow method	Deep method
Pure CF	BPRMF	MLP
Hybrid	HFT, TextBPR	LCMR, CoMem (ours)

- **BPRMF**, Bayesian personalized ranking (Rendle et al 2009), is a latent factor model based on matrix factorization and pair-wise loss. It learns on the target domain only.
- **HFT**, Hidden factors and hidden Topics (McAuley and Leskovec 2013), adopts topic distributions to learn latent factors from text reviews. It is a hybrid method.
- **TextBPR** extends the basic BPRMF model by integrating text content. It computes the prediction scores by two parts: one is the standard latent factors, same with the BPRMF; and the other is the text factors learned from the text content. It has two implementations, the VBPR model (He et al 2016) and the TBPR model (Hu and Dai 2017) which are the same in essence.
- **MLP**, multilayer perceptron (He et al 2017), is a neural CF approach which learns the nonlinear interaction function using neural networks. It is a deep model learning on the target domain only.
- **LCMR**, Local and Centralized Memory Recommender (Hu et al 2018), is a deep model for collaborative filtering with unstructured Text. The local memory module is similar to our MNet except that we only have one layer. This is a deep hybrid method.

Implementation For BPRMF, we use LightFM's implementation[1] which is a popular CF library. For HFT and TextBPR, we use the code released by their authors[2]. The word embeddings used in the TextBPR are pre-trained by GloVe[3]. For latent factor models, we vary the number of factors from 10 to 100 with step size 10. For MLP, we use the code released by its authors[4]. The LCMR model is similar to our CoMem model and thus implemented in company. Our methods are implemented using TensorFlow. Parameters are randomly initialized from Gaussian $\mathcal{N}(0, 0.01^2)$. The optimizer is Adam with initial learning rate 0.001. The size of mini batch is 128. The ratio of negative sampling is 1. The MLP follows a tower pattern, halving the layer size for each successive higher layer. Specifically, the configuration of hidden layers in the base MLP network is $[64 \to 32 \to 16 \to 8]$ as reference in the original paper (He et al 2017). The dimension of embeddings of users and items is both default 75 (and hence $d = 150$).

Comparisons of Different Recommender Systems

The comparison results are shown in Table 2 and Table 4 and we have the following observations.

Firstly, CoMem outperforms the neural CF method MLP on the two datasets in terms of three ranking metrics. On the Amazon dataset, CoMem obtains a large improvement in performance gain with relative 12.34% HitRatio@10, 7.68% NDCG@10, and 6.19% MRR@5. On the Mobile dataset, CoMem obtains a large improvement in performance gain with relative 4.98% HitRatio@5, and 4.16% NDCG@5, and 3.88% MRR@5. Since the CFNet component of CoMem is a neural CF method, the results show the benefit of exploiting unstructured text to alleviate the data sparsity issue faced by the pure methods BPRMF and MLP.

Secondly, CoMem also outperforms the traditional hybrid methods HFT and TextBPR on the two datasets in terms of three ranking metrics. On the Amazon dataset, CoMem obtains a significantly large improvement in performance gain with relative 55.04% HitRatio@5, 28.87% NDCG@5, and 20.36%MRR@5. On the Mobile dataset, CoMem still obtains

[1]`https://github.com/lyst/lightfm`

[2]`http://cseweb.ucsd.edu/~jmcauley/`

[3]`https://nlp.stanford.edu/projects/glove/`

[4]`https://github.com/hexiangnan/neural_collaborative_filtering`

Table 4: Comparison results of different methods on Mobile dataset. The best baselines are marked with asterisks(*) and the best results are boldfaced. The last two columns are the relative improvements of our method.

Cutoff	Metric	Methods						Improvement of CoMem vs.		
		BPRMF	HFT	TextBPR	MLP	LCMR	CoMem	MLP	TextBPR	LCMR
topK=5	HR	0.4380	0.4966	0.4948	0.5380	0.5476*	**0.5648**	4.98%	14.14%	3.14%
	NDCG	0.3971	0.3617	0.4298*	0.4121	0.4189	**0.4345**	4.16%	0.95%	2.85%
	MRR	0.3606	0.3175	0.3826*	0.3702	0.3762	**0.3911**	3.88%	1.72%	2.72 %
topK=10	HR	0.4941	0.5580	0.5466	0.6176	0.6311*	**0.6424**	4.02%	17.52%	1.79%
	NDCG	0.4182	0.4093	0.4499*	0.4381	0.4460	**0.4598**	3.51%	1.81%	2.19%
	MRR	0.3694	0.3365	0.3913*	0.3810	0.3874	**0.4016**	3.34%	1.88%	2.25%
topK=20	HR	0.5398	0.6547	0.6123	0.6793	0.6927*	**0.6952**	2.34%	13.53%	0.36%
	NDCG	0.4316	0.4379	0.4682*	0.4529	0.4619	**0.4732**	2.99%	0.82%	1.63%
	MRR	0.3730	0.3445	0.3958*	0.3851	0.3918	**0.4053**	2.97%	1.55%	1.95%

reasonably large improvements with relative 17.52% HitRatio@10, 1.81% NDCG@10, and 1.88%MRR@10. Compared with traditional hybrid methods which integrate the text using topic modelling or word embeddings, the results show the benefit of integrating text information through memory networks (and exploiting the interaction data through neural CF). In detail, the TextBPR extracts the text feature $\boldsymbol{f_i}$ by averaging the word embeddings $\boldsymbol{e}_w$ in the document: $\boldsymbol{f_i} = \frac{1}{|d_{ui}|}\sum_{w\in d_{ui}} \boldsymbol{e}_w$. We can see that it treats different words in the document as equal importance and does not match word semantics with the specific user.

Lastly, CoMem outperforms the neural hybrid method LCMR by a large margin on the Amazon dataset with relative improvements of 16.20% HitRatio@5, 9.63% NDCG, and 7.41%MRR@5. CoMem still obtains reasonabe improvements on the Mobile dataset with relative improvements of 3.14% HitRatio@5, 2.85% NDCG, and 2.72% MRR. As we have revealed the relationships between CoMem and LCMR in the theorem 1, the design of CFNet of CoMem is more reasonable than that of centralized memory module of LCMR. The results show the effectiveness of CoMem to exploit unstructured text via the MemNet and the interaction data via the CFNet.

Comparing pure CF method MLP on the two datasets, the results show that the MLP model is not much more effective for recommending news articles on the Mobile dataset than recommending products on Amazon. Comparing hybrid method TextBPR on the two datasets, the results show that the TextBPR model is much more effective for recommending news articles on the Mobile dataset than recommending products on Amazon.

Analysis

We first evaluate the effects of the dimensionality of the embedding space on the top-K recommendations. The x-axis in Figure 2 (left) is the dimension of user/item and hence the dimensionality of input to CFNet and MemNet is double since we adopt concatenation (see Eq.(3)). It clearly indicates that the embedding should not be too small due to the possibility of information loss and the limits of expressiveness.

We next show optimization curves of performance@10 and loss (averaged over all examples) against iterations on the Mobile in Figure 2 (right). The model learns quickly in the first 20 iterations and improves slowly until 50, though training losses continue to go down and valid losses stabilize. The average time per epoch of CoMem takes 68.1s and as a reference it is 34.5s for MLP using one NVIDIA TITAN Xp GPU.

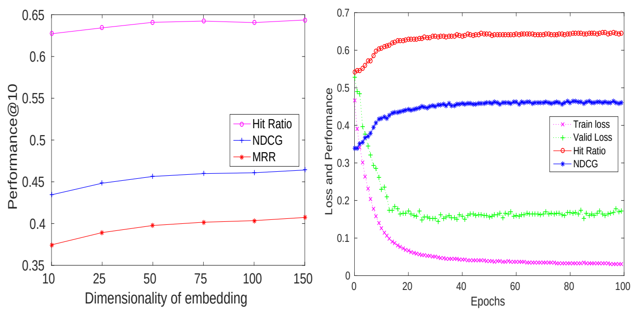

Figure 2: Left: Embedding dimensionality. Right: Loss and performance@10 with iterations. Dataset: Mobile.

Conclusion

It is shown that the text content can be effectively integrated with interaction data under a neural architecture to help improve recommendation performance. The sparse user-item interaction matrix can be reconstructed with the information guidance from unstructured text to alleviate the data sparse issue. We proposed a novel deep neural model, CoMem, for recommendation with unstructured text. CoMem consists of a memory component which can attentively focus relevant words to match user preferences (semantic factor) and of a CF component which can model nonlinear relationships between users and items to infer users' personalize preferences (behavior factor). These two representations of behavior and semantic factors are used to learn user preferences on items in an end-to-end manner. By building the connections of CoMem to existing matrix factorization (MF) and neural hybrid method (LCMR), we theoretically show the higher expressive power of CoMem than MF and the more reasonable architecture design of CoMem than LCMR. In practice, CoMem shows better performance than various baselines on two real-world datasets in terms of three ranking metrics under different settings.

Reference

Yang Bao, Hui Fang, and Jie Zhang. TopicMF: Simultaneously Exploiting Ratings and Reviews for Recommendation. In *AAAI*, 2014.

Trapit Bansal, David Belanger, and Andrew McCallum. Ask the gru: Multi-task learning for deep text recommendations. In *RecSys*, 2016.

Rose Catherine and William Cohen. Transnets: Learning to transform for recommendation. *arXiv preprint arXiv:1704.02298*, 2017.

Heng-Tze Cheng, Levent Koc, and Jeremiah Harmsen et al. Wide & deep learning for recommender systems. In *RecSys Workshop*, 2016.

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

Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys*, 2010.

Travis Ebesu, Bin Shen, and Yi Fang. Collaborative Memory Network for Recommendation Systems. In *SIGIR*, 2018.

G. Dziugaite and D. Roy. Neural network matrix factorization. *arXiv preprint arXiv:1511.06443*, 2015.

Gayatree Ganu, Noemie Elhadad, and Amelie Marian. Beyond the stars: Improving rating predictions using review text content. In *WebDB*, 2009.

Ruining He and Julian McAuley. Vbpr: visual bayesian personalized ranking from implicit feedback. In *AAAI*, 2016.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *WWW*, 2017.

Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin Collaborative Metric Learning. In *WWW*, 2017.

Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, 2008.

G.-N. Hu and X. Dai. Integrating reviews into personalized ranking for cold start recommendation. In *PAKDD*, 2017

Guangneng Hu, Yu Zhang and Qiang Yang. LCMR: Local and Centralized Memories for Collaborative Filtering with Unstructured Text. *arXiv preprint arXiv:1804.06201*, 2018

Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. Convolutional matrix factorization for document context-aware recommendation. In *RecSys*, 2016.

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

Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 2009.

D. Liang, J. Altosaar, L. Charlin, and D. Blei. Factorization Meets the Item Embedding: Regularizing Matrix Factorization with Item Co-occurrence. In *RecSys*, 2016.

G. Ling, M. Lyu, and I. King. Ratings meet reviews, a combined approach to recommend. In *RecSys*, 2014.

Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*, 2013.

Andriy Mnih and Ruslan R Salakhutdinov. Probabilistic matrix factorization. In *NIPS*, 2008.

Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *ICDM*, 2008.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, 2009.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 2001.

Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *WWW*, 2015.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *NIPS*, 2015.

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

Daniel Valcarce, Alejandro Bellogín, Javier Parapar, and Pablo Castells On the Robustness and Discriminative Power of IR Metrics for Top-N Recommendation. In *RecSys*, 2017

M. van Baalen. Chapter 3: Autoencoding variational matrix factorization. *Deep Matrix Factorization for Recommendation (Master Thesis)*, 2016.

A. Van den Oord, S. Dieleman, and B. Schrauwen. Deep content-based music recommendation. In *NIPS*, 2013.

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

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

Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *WSDM*, 2016.

C. Wu, A. Ahmed, A. Beutel, A. Smola, and H. Jing. Recurrent recommender networks. In *WSDM*, 2017.

Hong-Jian Xue, Xin-Yu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. Deep matrix factorization models for recommender systems. In *IJCAI*, 2017.

Carl Yang, Lanxiao Bai, Chao Zhang, Quan Yuan, and Jiawei Han. Bridging collaborative filtering and semi-supervised learning: A neural approach for poi recommendation. In *SIGKDD*, 2017.

Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *SIGKDD*, 2016.

Lei Zheng, Vahid Noroozi, and Philip S Yu. Joint deep modeling of users and items using reviews for recommendation. In *WSDM*, 2017.