Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

HAM: a deep collaborative ranking method incorporating textual information*

Cheng-wei WANG[1,3], Teng-fei ZHOU[3], Chen CHEN[1,3], Tian-lei HU[1,3], Gang CHEN[‡2,3]

[1]The Key Laboratory of Big Data Intelligent Computing of Zhejiang Province, Hangzhou 310027, China
[2]State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310027, China
[3]College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China
E-mail: rr@zju.edu.cn; zhoutengfei@zju.edu.cn; cc33@zju.edu.cn; htl@zju.edu.cn; cg@zju.edu.cn

Received July 28, 2019; Revision accepted Dec. 20, 2019; Crosschecked July 13, 2020

Abstract: The recommendation task with a textual corpus aims to model customer preferences from both user feedback and item textual descriptions. It is highly desirable to explore a very deep neural network to capture the complicated nonlinear preferences. However, training a deeper recommender is not as effortless as simply adding layers. A deeper recommender suffers from the gradient vanishing/exploding issue and cannot be easily trained by gradient-based methods. Moreover, textual descriptions probably contain noisy word sequences. Directly extracting feature vectors from them can harm the recommender's performance. To overcome these difficulties, we propose a new recommendation method named the HighwAy recoMmender (HAM). HAM explores a highway mechanism to make gradient-based training methods stable. A multi-head attention mechanism is devised to automatically denoise textual information. Moreover, a block coordinate descent method is devised to train a deep neural recommender. Empirical studies show that the proposed method outperforms state-of-the-art methods significantly in terms of accuracy.

Key words: Deep learning; Recommendation system; Highway network; Block coordinate descent
https://doi.org/10.1631/FITEE.1900382 **CLC number:** TP181

1 Introduction

Recommendation system (RS) is an important application of artificial intelligence. It infers the preferences of customers from historical feedback, and then recommends personalized items to customers (McLaughlin and Herlocker, 2004; Salakhutdinov and Mnih, 2007). An RS improves users' satisfaction with the e-commerce platform and brings extra profits to vendors (Linden et al., 2003). With the development of e-commerce, millions and trillions of items are available online. In such circumstances, a modern RS can assist customers in filtering out unwanted items. RS has become an indispensable part of modern e-commerce platforms.

RSs have been comprehensively studied in recent decades. The content-based RS adopts a nearest-neighbor strategy that recommends similar users with similar items (Linden et al., 2003). These simple methods have small generalization errors. They are commonly used in the early cold-start stage of an RS. With the accumulation of feedback datasets, RSs based on matrix factorization outperform content-based methods substantially in terms of accuracy (Srebro et al., 2004; Adomavicius and Tuzhilin, 2005; Paterek, 2007). Recently, evidence shows that customers' preferences are highly nonlinear (Wang H et al., 2015). Thus, deep neural

---

‡ Corresponding author

* Project supported by the Key R&D Program of Zhejiang Province, China (No. 2020C01024) and the National Key R&D Program (No. 2016YFB1001503)

ORCID: Cheng-wei WANG, https://orcid.org/0000-0002-6514-2603; Gang CHEN, https://orcid.org/0000-0002-7483-0045

networks (DNNs), well known for their powerful nonlinear modeling ability, have become mainstream methods (Wang H et al., 2015; Devooght and Bersini, 2016). However, research on DNN-based methods is far from finished. Two challenges still exist.

First, a DNN-based recommender can work only with a shallow depth, which limits its expressive power. Generally speaking, the depth of neural models perhaps plays the most crucial role in their success (Mhaskar et al., 2017). The expressive power of a neural network increases as its depth grows (Raghu et al., 2017). Moreover, a deeper network usually enjoys better generalization ability because of its over-parameterization property (Neyshabur et al., 2017). Thus, it is desirable to use very deep neural networks to design an accurate recommender. However, the optimization of deep networks is substantially more difficult. Gradients of existing DNN-based RSs are prone to vanishing or exploding when they are very deep (Goodfellow et al., 2016). This makes gradient-based methods unable to train an RS effectively.

Second, it is difficult to effectively incorporate textual information into a neural recommender because of data noise and the document's multi-topic property. In practice, user feedback is extremely sparse (Grčar et al., 2005). RSs that are trained only from feedback data usually have inferior accuracy because of information insufficiency. To handle this issue, many researchers exhibited that textual information, such as movie plots and item descriptions, is a valuable complement to feedback data. Informative feature vectors can be extracted from the textual data to boost recommendation performance. Nevertheless, designing a textual encoder is a challenging task. Textual information usually contains noisy sub-sequences. Ignoring such a fact would make neural networks overfit into noise, which leads to performance degradation (Wang H et al., 2016). Moreover, a document usually involves multiple topics. A neural encoder unaware of these characteristics cannot fully exploit textual information.

To overcome the above difficulties, we propose a new recommendation method named the HighwAy recoMmender (HAM). HAM tackles the gradient vanishing/exploding issue from both architecture construction and algorithm design perspectives. We integrate highway networks (Srivastava et al., 2015) in HAM to extract deep features. The highway networks use skip-connections and gating mechanisms to regulate the information flow. Through such regulation, gradients can flow across several neural layers without attenuation. Furthermore, we propose a new training method based on block coordinate descent (BCD). BCD is capable of decomposing a highly coupled deep network training problem into several independent subproblems on shallow networks. By such reduction, the unstable backward propagation of gradients can be avoided. In addition, HAM explores the multi-head attention mechanism to deal with the noisy document and model multiple topics. HAM learns to assign large weights to relevant parts of a document and give small weights to irrelevant ones. Then, it generates the feature vector by averaging each textual part. In this manner, the noisy word sequences are filtered out from the extracted features. By concentrating on different textual segments, each attention head can be interpreted as a semantic vector of topics.

In summary, the contributions of our paper are listed as follows:

1. We design a new neural recommendation framework based on the highway network to stabilize the gradient flow of a deep recommender.
2. We propose a novel BCD method to train the deeper recommender effectively.
3. A multi-head attention encoder is designed to exploit textual information to enhance recommendation performance.

2 Related work

State-of-the-art recommendation methods can be roughly divided into matrix factorization (MF) based methods and DNN-based methods. MF-based methods work by decomposing the user-item interactions into the product of latent vectors. Salakhutdinov and Mnih (2007) reinterpreted MF as a latent probabilistic model and used statistical inference to train the model. Koren (2008) improved MF by incorporating neighborhood information into the scoring function. Koren et al. (2009) proposed a temporal extension for MF and showed that MF has promising performance in the Netflix prize task (Bennett and Lanning, 2007). Rendle et al. (2009) proposed the seminal Bayesian personal ranking (BPR) method to predict personalized ranking using users' implicit feedback.

DNN-based methods try to characterize

nonlinearity in a user-item relationship via neural architectures. Salakhutdinov et al. (2007) proposed a restricted Boltzmann machine to mine users' preferences. Strub and Mary (2015) showed that a stack denoising auto-encoder has robust recommendation performance for a noisy feedback dataset. Devooght and Bersini (2016) used a recurrent neural network (RNN) to model the temporal dynamics of user preferences. He et al. (2017) modeled user ratings by a deep forward network. Cai et al. (2018) proposed an adversarial personalized ranking approach to enhance the BPR method. Note that none of the above neural recommenders encompasses a gradient stabilization module. Their gradients are prone to vanishing or exploding as the number of layers increases. This makes them untrainable when their depths are large.

The above methods have promising accuracy when customers' behavior data are sufficient. However, in many real-world scenarios, behavior data are very sparse. To enrich training data, much research incorporates text content data into the traditional recommendation data. Wang C and Blei (2011) proposed the well-known collaborative topic regression (CTR) method that seamlessly combines MF with latent Dirichlet allocation (LDA). Gopalan et al. (2014) proposed to use Poisson MF to model both user clicks and the word count matrix. Wang H et al. (2015) proposed collaborative deep learning (CDL), which uses a stack denoising auto-encoder to extract content features. Kim et al. (2016) showed that the convolution network can generate informative content features. Bansal et al. (2016) introduced the RNN to transform the text documents into feature vectors. Jin et al. (2018) proposed the long short term memory (LSTM) topic MF (LTMF), which combines LSTM and the topic modeling method to understand textual information. Shoja and Tabrizi (2019) developed an LDA-based attribute extractor to filter out useful product information in reviews and then built a deep neural network to transform such information into feature vectors.

3 Preliminaries and notations

In this paper, we use bold capital letters $\boldsymbol{A}$, $\boldsymbol{B}$, ... to denote matrices. Matrix slicing is expressed in a Numpy manner. We use $\boldsymbol{A}[i]$ to denote the i^{th} row of matrix $\boldsymbol{A}$. $\boldsymbol{A}[i_1, i_2, \ldots, i_k]$ is a submatrix of $\boldsymbol{A}$ containing $\boldsymbol{A}$'s rows i_1, i_2, ..., i_k. The size of a set S is denoted by $|S|$. Some other notations used in the paper are listed in Table 1. Assume that user feedback D and item textual content C are available for training an RS. The feedback dataset D contains users' historical choices. More precisely, $D = \{(i, j, j')|1 \leq i \leq m, j \in \rho_i^+, j' \in \rho_i^-\}$, where i is a user, ρ_i^+ consists of the user's wanted items, and ρ_i^- contains his/her unwanted products. The textual content $C = \{\text{doc}_j\}_{j=1}^{j=n}$ is a collection of documents doc_j which describes item j in the text.

Table 1 Notations used in this paper

Variable	Description
m	Number of users
n	Number of items
doc	A textual document
n_{vocab}	Number of unique words
n_{doc}	Length of a document
ω	Word ID
$\boldsymbol{P}$	Embedding matrix of users
$\boldsymbol{E}$	Embedding matrix of words
$\boldsymbol{W}_{A_k}$	Parameter of the k^{th} attention head
$\boldsymbol{W}_L$	Parameter of the last layer of encoder
$\boldsymbol{G}_l, \boldsymbol{K}_l$	Parameters of the user's deep transformer
$\tilde{\boldsymbol{G}}_l, \tilde{\boldsymbol{K}}_l$	Parameters of the item's deep transformer
$H(\cdot\|\boldsymbol{G}, \boldsymbol{K})$	Highway cell with parameters $\boldsymbol{G}$ and $\boldsymbol{K}$
$\text{Enc}(\cdot\|\theta_e)$	Textual encoder with parameter θ_e

The recommendation problem can be formulated as reconstructing the preference function $f(i, \text{doc}_j)$ of user i on item j from the available dataset. Intuitively, the function $f(i, \text{doc}_j)$ will give higher values to a user's wanted items and lower scores to unwanted ones. Such intuition can be formulated by the following area under the receiver operating characteristic curve (AUC) maximization problem:

$$\max_{\theta_f} \sum_{(i,j,j')} \mathcal{I}\{f(i, \text{doc}_j) > f(i, \text{doc}_{j'})\}, \tag{1}$$

where θ_f is the parameter set of function $f(i, \text{doc}_j)$ and $\mathcal{I}\{\text{condition}\}$ is the indicator function which equals 1 if "condition" is true and 0 otherwise. However, problem (1) is a combinatorial optimization problem that is computationally prohibitive to solve. To reduce the computational cost, one can relax the discrete function $\mathcal{I}\{f(i, \text{doc}_j) > f(i, \text{doc}_{j'})\}$ via the following smoothing function (Rendle et al., 2009):

$$\ln \sigma(f(i, \text{doc}_j) - f(i, \text{doc}_{j'})) \triangleq l(i, j, j'), \tag{2}$$

with $\sigma(\cdot)$ being the sigmoid function. By such relaxation, the recommendation task can be recast as the following minimization problem:

$$\min_{\theta_f} -\frac{1}{|D|}\sum_{(i,j,j')} l(i,j,j') \triangleq \mathcal{L}_{\text{rank}}(\theta_f). \tag{3}$$

4 Architecture of the scoring function

To model the nonlinear correlation between a user and an item, we devise a deep neural network to parameterize the function $f(\cdot,\cdot)$. We depict the neural architecture in Fig. 1. The architecture has four modules: embedding module, content encoder, deep transformer, and scoring producer.

4.1 Embedding module

The embedding layer maps users and words into low-dimensional latent spaces. More precisely, let $\boldsymbol{P}\in\mathbb{R}^{m\times d}$ and $\boldsymbol{E}\in\mathbb{R}^{n_{\text{vocab}}\times d}$ be users' embedding matrix and words' embedding matrix, respectively, with m being the number of users, n_{vocab} the vocabulary size, and d the model dimension. The embedding vectors of user i and word ω are $\boldsymbol{P}[i]$ and $\boldsymbol{E}[\omega]$, respectively.

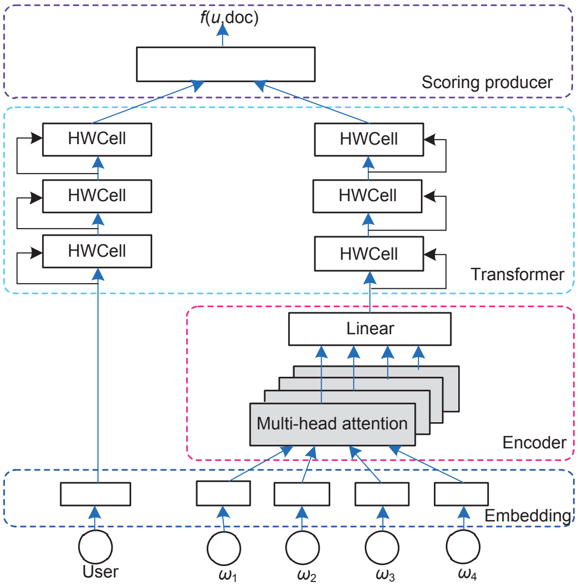


Fig. 1 Architecture of the HighwAy recoMmender (HAM)

4.2 Content encoder

The content encoder extracts feature vectors from textual documents. Let doc $=\{\omega_i\}_{i=1}^{i=n_{\text{doc}}}$ be a textual document of some items. Let $\boldsymbol{E}[\text{doc}]\in\mathbb{R}^{n_{\text{doc}}\times d}$ be formed by stacking embedding vectors of word ω_i horizontally. The content encoder takes $\boldsymbol{E}[\text{doc}]$ as input and outputs the feature vector of doc. The content encoder has two layers. The lower layer is the multi-head attention mechanism, which represents doc by multiple semantic vectors. The k^{th} attention head is formulated as follows:

$$\textbf{head}_k=(\boldsymbol{E}[\text{doc}])^{\text{T}}\,\text{softmax}(\boldsymbol{E}[\text{doc}]\boldsymbol{W}_{A_k}\boldsymbol{W}_{A_k}^{\text{T}}\textbf{query}), \tag{4}$$

where $\boldsymbol{W}_{A_k}\in\mathbb{R}^{d\times l}$ aligns the k^{th} head to a specific topic, softmax$(\cdot)$ is the softmax function, and

$$\textbf{query}=\frac{\sum_{\omega\in\text{doc}}\boldsymbol{E}[w]}{n_{\text{doc}}}. \tag{5}$$

It has been validated by much research (Chorowski et al., 2015; Vaswani et al., 2017) that attention signals can assign larger values to the relevant parts of a sequence and smaller values to the irrelevant parts. This property helps our encoder concentrate on the informative part of the textual document while ignoring the noise. Each attention head of the encoder focuses on representing one part of the semantic information in its textual content. This helps the encoder understand different topics in the content.

In the upper layer, we use a dense linear layer followed by dropout units to summarize the attention heads $\textbf{head}_k$ into the feature vector of doc. In precise terms, the feature vector is given by the following equation:

$$\textbf{Ouput}=\text{Dropout}(\text{Concate}([\textbf{head}_k]_{k=1}^{K})\boldsymbol{W}_L), \tag{6}$$

where Dropout$(\cdot)$ is the dropout operator, Concate$(\cdot)$ concatenates vectors $\textbf{head}_k$ into a matrix of size $K\times d$ with K being the number of attention heads. For brevity, in what follows, we denote the content encoder by Enc$(\boldsymbol{E}[\text{doc}]|\theta_e)$ with θ_e being a parameter set including $\boldsymbol{W}_{A_k}$ and $\boldsymbol{W}_L$.

4.3 Deep transformer

The deep transformer enhances the expressive power of lower-layer neural features. To stabilize the gradient flow, our transformer regulates the information flow through the gating mechanism. Specifically, the transformer is a stack of highway cells. Denote a cell by $H(\boldsymbol{x}|\boldsymbol{G},\boldsymbol{K})$, where $\boldsymbol{x}\in\mathbb{R}^d$ is the input and $\boldsymbol{G},\boldsymbol{K}$ are the cell parameters. The cell can be computed by the following equation:

$$\begin{cases} g=\sigma(\boldsymbol{G}\boldsymbol{x}), \\ H(\boldsymbol{x}|\boldsymbol{G},\boldsymbol{K})=g\odot\boldsymbol{x}+(1-g)\odot\tanh(\boldsymbol{K}\boldsymbol{x}), \end{cases} \tag{7}$$

where $\sigma(\cdot)$ is the element-wise sigmoid function and $\odot$ is the element-wise multiplication operator. The term g acts as an information gate. When $g=0$, the highway cell cuts off the skip-connection and becomes a feed-forward unit; when $g=1$, the highway cell blocks the result of the feed-forward unit and degenerates into an identity map. Thus, the highway cell can be viewed as a smooth interpolation between a nonlinear cell and an identity map. Stacking L layers of highway cells, we obtain the transformer. We summarize the computing process of the deep transformer in Algorithm 1.

Algorithm 1 Transformer($\boldsymbol{x}|\{\boldsymbol{G}_l,\boldsymbol{K}_l\}_{l=1}^{L}$)

```
$\boldsymbol{h}_0=\boldsymbol{x}$
for $l=1$ to $L$ do
    $g=\sigma(\boldsymbol{G}_l\boldsymbol{h}_{l-1})$
    $\boldsymbol{h}_l=g*\boldsymbol{h}_{l-1}+(1-g)*\tanh(\boldsymbol{K}_l\boldsymbol{h}_{l-1})$
end for
return $\boldsymbol{h}_L$
```

4.4 Scoring producer

The scoring producer produces the score $f(i,\mathrm{doc}_j)$ of user i and item j. Given that $f(i,\mathrm{doc}_j)$ reflects the correlation between user i and item j, we formulate the score as the inner product between the user deep feature $\boldsymbol{u}_i$ and the item deep feature $\boldsymbol{v}_j$:

$$f(i,\mathrm{doc}_j)=\boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{v}_j. \tag{8}$$

The deep features are generated by applying the deep transformer to low-level embedding vectors, that is,

$$\begin{cases} \boldsymbol{u}_i=\mathrm{Transformer}\left(\boldsymbol{P}[i]\mid\{\boldsymbol{G}_l,\boldsymbol{K}_l\}_{l=1}^{L}\right), \\ \boldsymbol{v}_j=\mathrm{Transformer}\left(\mathrm{Enc}(\boldsymbol{E}[\mathrm{doc}]|\theta_e)\mid\{\tilde{\boldsymbol{G}}_l,\tilde{\boldsymbol{K}}_l\}_{l=1}^{L}\right), \end{cases} \tag{9}$$

where e_{doc_j} is the output of document doc_j.

5 Parameter inference

According to the framework of multi-task learning (Ruder, 2017), training a deep network from multiple learning tasks can improve the network's generalization ability. Intuitively, learning word embeddings from the textual corpus is a natural choice for an auxiliary task. Given the textual corpus $\{\mathrm{doc}_j\}$ and a word ω, several surrounding words of ω are selected as context c. We denote the set containing all such word context pairs as "pairs." The word embedding task can be formulated as the following MF problem (Levy and Goldberg, 2014):

$$\min_{\boldsymbol{C},\boldsymbol{E}}\underbrace{\sum_{(c,\omega)\in\mathrm{pairs}}\frac{\left((\boldsymbol{C}[c])^{\mathrm{T}}\boldsymbol{E}[\omega]-\mathrm{PMI}(c,\omega)\right)^2}{|\mathrm{pairs}|}}_{\mathcal{L}_{\mathrm{embd}}(\boldsymbol{C},\boldsymbol{E})}, \tag{10}$$

where $\boldsymbol{C}$ is the context embedding matrix. Note that $\mathrm{PMI}(c,\omega)$ is the pointwise mutual information defined as follows:

$$\mathrm{PMI}(c,\omega)=\ln\left(\frac{|\mathrm{pairs}|\#(c,\omega)}{\#(c)\#(\omega)}\right), \tag{11}$$

where $\#(c,\omega)$ is the number of occurrences of the context-word pair, $\#(c)$ is the number of occurrences of context c, and $\#(\omega)$ is the frequency of word ω. Training an RS and word embedding simultaneously can be formulated as the following composite objective minimization problem:

$$\min_{\theta_f,\boldsymbol{C}}\mathcal{L}_{\mathrm{rank}}(\theta_f)+\lambda\mathcal{L}_{\mathrm{embd}}(\boldsymbol{C},\boldsymbol{E}), \tag{12}$$

where $\lambda>0$ is a regularization parameter, and θ_f is the parameter set of neural network $f(\cdot,\cdot)$.

To further stabilize the gradient flow, we design a BCD method to solve the optimization problem (12). Problem (12) can be rewritten as the following constrained minimization problem according to Eqs. (3), (8), (9), and (12):

$$\begin{aligned} \min_{\theta_{\mathrm{opt}}} & -\frac{1}{N}\sum_{(i,j,j')}\ln\sigma\left((\boldsymbol{P}_L[i])^{\mathrm{T}}(\boldsymbol{Q}_L[j]-\boldsymbol{Q}_L[j'])\right) \\ & +\lambda L_{\mathrm{embd}}(\boldsymbol{C},\boldsymbol{E}) \end{aligned}$$

s.t.

$$\begin{cases} \boldsymbol{P}_l[i]=H(\boldsymbol{P}_{l-1}[i]|\boldsymbol{G}_l,\boldsymbol{K}_l), & i\le m, 1\le l\le L, \\ \boldsymbol{Q}_l[j]=H(\boldsymbol{Q}_{l-1}[j]|\tilde{\boldsymbol{G}}_l,\tilde{\boldsymbol{K}}_l), & j\le n, 1\le l\le L, \\ \boldsymbol{P}_0=\boldsymbol{P}, & \\ \boldsymbol{Q}_0[j]=\mathrm{Enc}(\boldsymbol{E}[\mathrm{doc}]|\theta_e), & j\le n, \end{cases} \tag{13}$$

where θ_{opt} includes embedding matrices $\boldsymbol{P}$ and $\boldsymbol{E}$, intermediate matrices $\{\boldsymbol{P}_l, \boldsymbol{Q}_l\}_{l=1}^{l=L}$, parameters of transformers $\{\boldsymbol{G}_l, \boldsymbol{K}_l, \tilde{\boldsymbol{G}}_l, \tilde{\boldsymbol{K}}_l\}_{l=1}^{L}$, and parameters of encoders θ_{enc}. Using the constraints as a penalty, the minimization problem (13) is converted to the following unconstrained optimization problem:

$$
\begin{aligned}
&\min_{\theta_{\text{opt}}} \underbrace{-\frac{1}{N}\sum_{(i,j,j')} \ln\sigma\left((\boldsymbol{P}_L[i])^{\mathrm{T}}(\boldsymbol{Q}_L[j]-\boldsymbol{Q}_L[j'])\right)}_{\mathcal{L}_O(\boldsymbol{P}_L,\boldsymbol{Q}_L)} \\
&+\lambda\mathcal{L}_{\text{embd}}(\boldsymbol{C},\boldsymbol{E})+\sum_{l=1}^{L}\Bigg\{\underbrace{\frac{\gamma}{m}\|\boldsymbol{P}_l-H_l(\boldsymbol{P}_{l-1})\|^2}_{\mathcal{L}_{H(l)}(\boldsymbol{P}_l,\boldsymbol{P}_{l-1},\boldsymbol{G}_l,\boldsymbol{K}_l)} \\
&+\underbrace{\frac{\gamma}{n}\|\boldsymbol{Q}_l-\tilde{H}_l(\boldsymbol{Q}_{l-1})\|^2}_{\mathcal{L}_{H(l)}(\boldsymbol{Q}_l,\boldsymbol{Q}_{l-1},\tilde{\boldsymbol{G}}_l,\tilde{\boldsymbol{K}}_l)}\Bigg\}+\underbrace{\frac{\gamma}{m}\|\boldsymbol{P}_0-\boldsymbol{P}\|^2}_{\mathcal{L}_S(\boldsymbol{P}_0,\boldsymbol{P})} \\
&+\underbrace{\frac{\gamma}{n}\|\boldsymbol{Q}_0-\boldsymbol{Q}-\boldsymbol{E}_{\theta_e}(\{\text{doc}_j\})\|^2}_{\mathcal{L}_E(\boldsymbol{Q}_0,\boldsymbol{Q},\boldsymbol{E}_{\text{doc}_j},\theta_e)},
\end{aligned}
\tag{14}
$$

where $H_l(\boldsymbol{P}_{l-1})$ is obtained by stacking vectors $H(\boldsymbol{P}_{l-1}[i]|\boldsymbol{G}_l, \boldsymbol{K}_l)$ into a matrix of size $m \times d$, and $\boldsymbol{E}_{\theta_e}(\{\text{doc}_j\})$ is a matrix of size $n \times d$ obtained by stacking vectors $\text{Enc}(\boldsymbol{E}[\text{doc}]|\theta_e)$. We solve the optimization problem (14) through Algorithm 2, which optimizes the objective with respect to one block of variables while fixing the remaining ones.

Algorithm 2 Block coordinate descent for an RS

```
Require: $\lambda, \gamma, T, \eta$
$\theta_{\text{opt}}$ = RandomInitialization()
for $k = 1$ to $T$ do
  // Compute stochastic approximations of losses
    Randomly sample a tuple $(i, j, j')$ from $D$
    Randomly sample a user $u$ from $\{1, 2, \ldots, m\}$
    Randomly sample an item $v$ from $\{1, 2, \ldots, n\}$
    $\hat{\mathcal{L}}_O = -\ln\sigma\left((\boldsymbol{P}_L[i])^{\mathrm{T}}(\boldsymbol{Q}_L[j]-\boldsymbol{Q}_L[j'])\right)$
    $\hat{\mathcal{L}}_O = \hat{\mathcal{L}}_O + \gamma\|\boldsymbol{P}_L[i] - H_L(\boldsymbol{P}_{L-1}[i])\|^2$
    $\hat{\mathcal{L}}_O = \hat{\mathcal{L}}_O + \gamma\|\boldsymbol{Q}_L[j] - \tilde{H}_L(\boldsymbol{Q}_{L-1}[j])\|^2/2$
    $\hat{\mathcal{L}}_O = \hat{\mathcal{L}}_O + \gamma\|\boldsymbol{Q}_L[j'] - \tilde{H}_L(\boldsymbol{Q}_{L-1}[j'])\|^2/2$
    for $l = L$ to 1 do
        $\hat{\mathcal{L}}_{H(l)} = \gamma\|\boldsymbol{P}_l[u] - H_l(\boldsymbol{P}_{l-1}[u])\|^2$
        $\hat{\mathcal{L}}_{\tilde{H}(l)} = \gamma\|\boldsymbol{Q}_l[v] - \tilde{H}_l(\boldsymbol{Q}_{l-1}[v])\|^2$
    end for
    pairs = GenerateContextWordPairs(doc$_v$)
    $\hat{\mathcal{L}}_{\text{embd}} = \sum_{(c,\omega)\in\text{pairs}} \left(\boldsymbol{C}[c]^{\mathrm{T}}\boldsymbol{E}[\omega] - \text{PMI}(c,\omega)\right)^2/|\text{pairs}|$
    $\hat{\mathcal{L}}_E = \gamma\|\boldsymbol{Q}_0[v] - \boldsymbol{Q}[v] - \text{Enc}(\boldsymbol{E}[\text{doc}_v]|\theta_e)\|^2$
    // Compute stochastic gradients
    $\text{grad}_{\boldsymbol{P}_L[i]} = \partial\hat{\mathcal{L}}_O/\partial\boldsymbol{P}_L[i]$, $\text{grad}_{\boldsymbol{Q}_L[j]} = \partial\hat{\mathcal{L}}_O/\partial\boldsymbol{Q}_L[j]$,
    $\text{grad}_{\boldsymbol{Q}_L[j']} = \partial\hat{\mathcal{L}}_O/\partial\boldsymbol{Q}_L[j']$
    $\text{grad}_{\boldsymbol{G}_L} = \partial\hat{\mathcal{L}}_{H(L)}/\partial\boldsymbol{G}_L$, $\text{grad}_{\boldsymbol{K}_L} = \partial\hat{\mathcal{L}}_{H(L)}/\partial\boldsymbol{K}_l$
    $\text{grad}_{\tilde{\boldsymbol{G}}_L} = \partial\hat{\mathcal{L}}_{\tilde{H}(L)}/\partial\tilde{\boldsymbol{G}}_L$, $\text{grad}_{\tilde{\boldsymbol{K}}_L} = \partial\hat{\mathcal{L}}_{\tilde{H}(L)}/\partial\tilde{\boldsymbol{K}}_l$
    for $l = L - 1$ to 1 do
        $\text{grad}_{\boldsymbol{P}_l[u]} = \partial(\hat{\mathcal{L}}_{H(l)} + \hat{\mathcal{L}}_{H(l+1)})/\partial\boldsymbol{P}_l[u]$
        $\text{grad}_{\boldsymbol{Q}_l[v]} = \partial(\hat{\mathcal{L}}_{\tilde{H}(l)} + \hat{\mathcal{L}}_{\tilde{H}(l+1)})/\partial\boldsymbol{Q}_l[v]$
        $\text{grad}_{\boldsymbol{G}_l} = \partial\hat{\mathcal{L}}_{H(l)}/\partial\boldsymbol{G}_l$, $\text{grad}_{\boldsymbol{K}_l} = \partial\hat{\mathcal{L}}_{H(l)}/\partial\boldsymbol{K}_l$
        $\text{grad}_{\tilde{\boldsymbol{G}}_l} = \partial\hat{\mathcal{L}}_{\tilde{H}(l)}/\partial\tilde{\boldsymbol{G}}_l$, $\text{grad}_{\tilde{\boldsymbol{K}}_l} = \partial\hat{\mathcal{L}}_{\tilde{H}(l)}/\partial\tilde{\boldsymbol{K}}_l$
    end for
    $\text{grad}_{\theta_e} = \partial\hat{\mathcal{L}}_E/\partial\theta_e$
    $W = \{w|w \in \text{pairs}\}$, $C = \{c|c \in \text{pairs}\}$
    $\text{grad}_{\boldsymbol{E}[W]} = \partial(\hat{\mathcal{L}}_E + \lambda\hat{\mathcal{L}}_{\text{embd}})/\partial\boldsymbol{E}[W]$
    $\text{grad}_{\boldsymbol{C}[C]} = \lambda\hat{\mathcal{L}}_{\text{embd}}/\partial\boldsymbol{C}[C]$
    // Update parameters
    $\boldsymbol{P}_L[i] \mathrel{-}= \eta\,\text{grad}_{\boldsymbol{P}_L[i]}$, $\boldsymbol{Q}_L[j] \mathrel{-}= \eta\,\text{grad}_{\boldsymbol{Q}_L[j]}$,
    $\boldsymbol{Q}_L[j'] \mathrel{-}= \eta\,\text{grad}_{\boldsymbol{Q}_L[j']}$
    $\tilde{\boldsymbol{G}}_L \mathrel{-}= \text{grad}_{\tilde{\boldsymbol{G}}_L}$, $\tilde{\boldsymbol{K}}_L \mathrel{-}= \text{grad}_{\tilde{\boldsymbol{K}}_L}$
    $\boldsymbol{G}_L \mathrel{-}= \eta\,\text{grad}_{\boldsymbol{G}_L}$, $\boldsymbol{K}_L \mathrel{-}= \text{grad}_{\boldsymbol{K}_L}$
    for $l = L - 1$ to 1 do
        $\boldsymbol{P}_l[u] \mathrel{-}= \eta\,\text{grad}_{\boldsymbol{P}_l[u]}$, $\boldsymbol{Q}_l[v] \mathrel{-}= \eta\,\text{grad}_{\boldsymbol{Q}_l[v]}$
        $\boldsymbol{G}_l \mathrel{-}= \eta\,\text{grad}_{\boldsymbol{G}_l}$, $\boldsymbol{K}_l \mathrel{-}= \text{grad}_{\boldsymbol{K}_l}$
        $\tilde{\boldsymbol{G}}_l \mathrel{-}= \eta\,\text{grad}_{\tilde{\boldsymbol{G}}_l}$, $\tilde{\boldsymbol{K}}_l \mathrel{-}= \text{grad}_{\tilde{\boldsymbol{K}}_l}$
    end for
    $\theta_e \mathrel{-}= \eta\,\text{grad}_{\theta_e}$
    $\boldsymbol{E}[W] \mathrel{-}= \eta\,\text{grad}_{\boldsymbol{E}[W]}$, $\boldsymbol{C}[C] \mathrel{-}= \eta\,\text{grad}_{\boldsymbol{C}[C]}$
    $\boldsymbol{P}[u] = \boldsymbol{P}_0[u]$
    $\boldsymbol{Q}[v] = \boldsymbol{Q}_0[v] - \text{Enc}(\boldsymbol{E}_{\text{doc}_v}|\theta_e)$
end for
```

6 Experiments

We conducted extensive experiments on benchmark datasets. We used the following state-of-the-art methods as baselines:

1. CML + Skip-Thought (CMLST)

We combined the collaborative filtering method CML (Hsieh et al., 2017) with the textual encoder Skip-Thought (Kiros et al., 2015) to learn from the feedback dataset and textual corpus.

2. CRAE (Wang H et al., 2016)

CRAE is a combination of Bayesian MF and RNN. The Bayesian MF module models user feedback. The Bayesian RNN module extracts feature vectors from the textual corpus.

3. CDL (Wang H et al., 2015)

CDL is a probabilistic graphics model that encompasses a collaborative filtering module with a probabilistic auto-encoder.

4. CDAE (Wu et al., 2016)

CDAE models user feedback with a stack denoising auto-encoder.

5. CTR (Wang C and Blei, 2011)

CTR combines MF and probabilistic topic modeling to produce a content-based recommendation.

All baselines and the proposed method HAM were run on the same machine with i7-5930K CPU, 64-GB RAM, and one TITAN Xp GPU.

6.1 Benchmark datasets

We used the following datasets to evaluate the performances of the different methods: CiteULike, M1M, and M10M. The CiteULike dataset (Wang C and Blei, 2011) is composed of user behavior of bookmarking research papers and their abstracts. M1M and M10M (Liu et al., 2017) contain five-star ratings of movies and textual documents recording movie plots. Basic statistics on the datasets are listed in Table 2. For these datasets, we treated every bookmarked paper (or rated movie) as "relevant." For each "relevant" user-item pair in all datasets, we sampled NS items from unbookmarked papers (or unrated movies) to form "irrelevant" pairs. We dub NS the negative sampling number (NS). The observed data were partitioned into 80% for training with the remaining 20% for testing.

6.2 Default hyperparameter settings

By default, a model's dimension was set to $d = 256$ for HAM. The depth of the transformer of HAM was set to 50. The negative sampling number was set to NS = 6. The regularization parameter λ was set to 0.2. γ was set to 0.3. Values of these hyperparameters were selected by five-fold cross validation. Parameters of baseline methods were set to their default values.

6.3 Evaluation metrics

To evaluate the performance of the different methods, we used the following measurements:

1. Area under the curve (AUC)

AUC reflects the performance of the recommender by counting the portion of incorrectly ordered pairs. It is formulated as follows:

$$\mathrm{AUC} = \frac{\sum_{(i,j,j')\in D'} \mathcal{I}\{f(i,\mathrm{doc}_j) > f(i,\mathrm{doc}_{j'})\}}{|D'|}, \tag{15}$$

where D' is the test set.

2. Recall@k

Recall@k is the proportion of relevant items in the top-k recommendation list. It is defined as follows:

$$\mathrm{Recall@}k = \sum_i \frac{|Y_i \cap Y_i^k|}{|Y_i|}, \tag{16}$$

where Y_i comprises the relevant items for user i and Y_i^k is the top-k recommendation list.

3. Precision@k

Precision@k is the fraction of relevant items from the top-k recommendation list:

$$\mathrm{Precision@}k = \sum_i \frac{|Y_i \cap Y_i^k|}{k}. \tag{17}$$

4. Mean average precision (MAP)

MAP is the mean value of Precision@k under different k, where Precision@k is the accuracy of top-k items in the recommendation list.

6.4 Accuracy comparison

In this subsection, we compared the accuracy of HAM with those of the baseline methods. All methods were tested five times. The average performance and standard deviations are reported in Table 3. We measured the ranking accuracy of the involved methods in terms of AUC and MAP. Table 3 shows that HAM outperforms the baselines on both AUC and MAP. This is because HAM is deeper and has a more robust encoder of textual information. In addition, we assessed the top-k recommendation performance of the involved methods via Recall@k and Precision@k. We report the means and standard deviations of both Recall@5 and Precision@5 of the

Table 2 Statistics of the benchmark datasets

Dataset	Number of users	Number of items	Number of feedbacks	Average number of words
CiteULike	5551	16 980	210 504	204.9
M1M	6040	3861	996 045	82.19
M10M	13 975	10 681	1 962 580	84.66

Table 3 Performance of the different methods on the collaborative ranking task

Method	AUC			MAP		
	CiteULike	M1M	M10M	CiteULike	M1M	M10M
CMLST	0.9313±0.003	0.9104±0.013	0.9370±0.013	0.1217±0.002	0.1881±0.004	0.1753±0.006
CRAE	0.9206±0.005	0.9157±0.023	**0.9682±0.024**	0.0580±0.007	0.1856±0.007	0.1585±0.005
CDL	0.9120±0.004	0.9188±0.023	0.9365±0.036	0.1124±0.003	0.1432±0.005	0.1547±0.004
CDAE	0.9348±0.003	0.9107±0.014	0.9366±0.017	0.1211±0.006	0.1630±0.002	0.1712±0.004
CTR	0.9043±0.007	0.9109±0.012	0.9211±0.021	0.0603±0.002	0.1008±0.009	0.1148±0.003
HAM	**0.9406±0.002**	**0.9239±0.009**	0.9671±0.010	**0.1481±0.008**	**0.2332±0.004**	**0.2249±0.007**

The best results are in bold

Table 4 Performance of the different methods on the top-*k* ranking task

Method	Recall@5			Precision@5		
	CiteULike	M1M	M10M	CiteULike	M1M	M10M
CMLST	0.0937±0.002	0.0722±0.003	0.0667±0.013	0.1100±0.006	0.2291±0.004	0.1822±0.006
CRAE	0.0825±0.003	0.0731±0.003	0.0729±0.024	0.1308±0.004	0.3132±0.002	0.2260±0.007
CDL	0.0895±0.007	0.0758±0.006	0.0840±0.036	0.1404±0.006	0.3782±0.010	0.2773±0.008
CDAE	0.0925±0.004	0.0858±0.002	0.0767±0.017	0.1511±0.002	0.3680±0.009	0.2445±0.003
CTR	0.0363±0.001	0.0434±0.002	0.0520±0.013	0.0430±0.001	0.2059±0.003	0.1356±0.005
HAM	**0.1151±0.002**	**0.0958±0.004**	**0.1038±0.010**	**0.1647±0.006**	**0.4142±0.006**	**0.3532±0.009**

The best results are in bold

different methods in Table 4. From the table, we can see that HAM has much better top-k recommendation accuracy than baseline methods. For general k, Recall@k and Precision@k of the different methods are displayed in Fig. 2. Fig. 2 shows that HAM consistently outperforms baseline methods in terms of both Recall@k and Precision@k, demonstrating that recommendation accuracy can be improved when a deeper network and a more robust encoder are used.

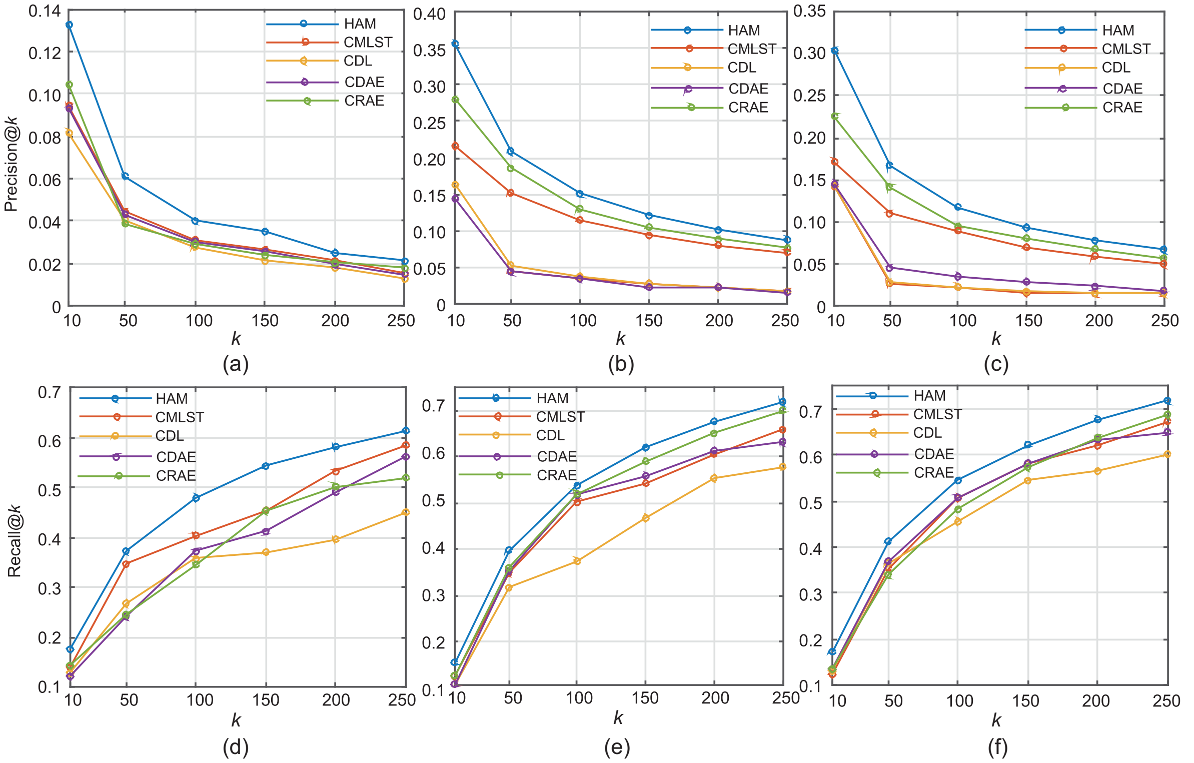
6.5 Performance under different model dimensions

We empirically observed that the model dimension d greatly influences the involved methods' recommendation precision. Thus, we examined the performance of the involved methods under different settings of model dimension. To see the parameters' impact, we report Recall@50 and Precision@50 of HAM and baseline methods in Fig. 3. Fig. 3 demonstrates that HAM has better accuracy under different settings of hyperparameters. The results show that a deeper neural structure can enhance recommendation performance regardless of model dimension.

7 Conclusions and future work

We have proposed a deep HighwAy recoMmender (HAM). HAM employs a highway mechanism to make gradient-based solvers stable. To automatically denoise textual information, HAM has been equipped with a multi-head attention architecture. Furthermore, HAM uses a novel block coordinate descent method to train its deep neural structure. Experimental results demonstrated that HAM notably outperforms state-of-the-art methods. We attribute the superior results to the following three factors: First, the proposed network is deeper than the baseline methods. A deeper network is more suitable for describing nonlinear preferences and generalizes better than shallow models. Second, the proposed block descent algorithm successfully trains the deep model without gradient vanishing/exploding. It is known that the backpropagation of a very deep neural network suffers from vanishing/exploding gradients. To avoid such backpropagation, the proposed algorithm decomposes the highly coupled DNN minimization problem into several loosely coupled single-layer subproblems. Third, our method effectively integrates useful information contained in the textual corpus by a multi-head attention mechanism.

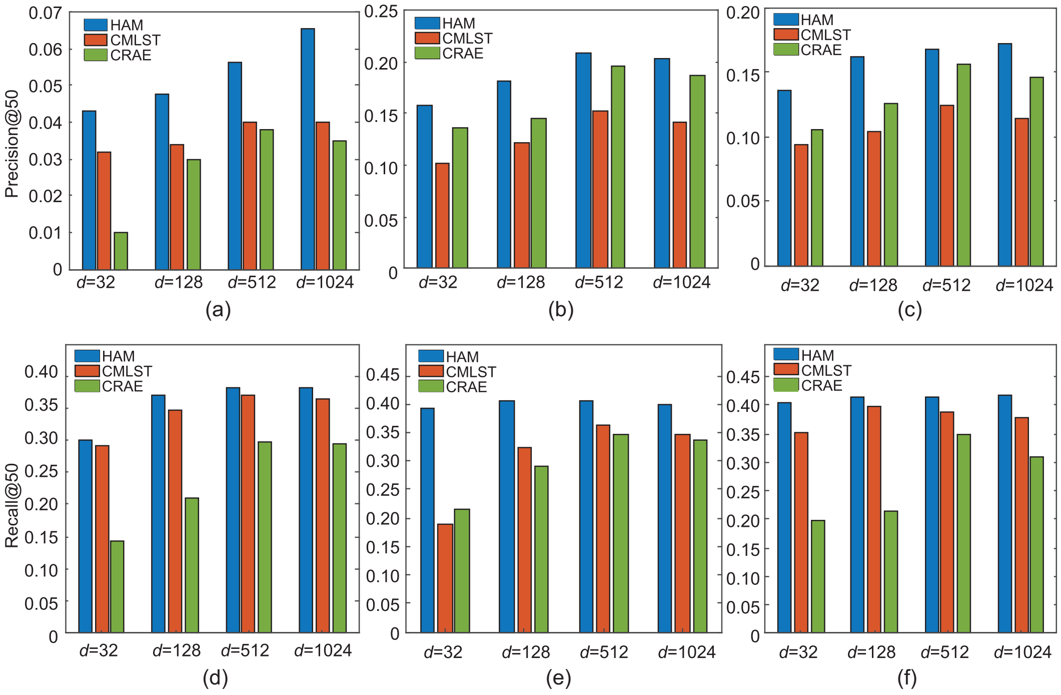
In this paper, HAM focuses on the recommendation task with textual side information available. Other types of side information, such as photos, videos, and social relations, are often accessible in real-world tasks. They can help a recommender better understand the user-item relationship and



(a) (b) (c)

(d) (e) (f)

Fig. 2 Precision@*k* of the different methods under different choices of *k* with respect to benchmark datasets CiteULike (a), M1M (b), and M10M (c), and Recall@*k* of the different methods under different choices of *k* with respect to benchmark datasets CiteULike (d), M1M (e), and M10M (f)



(a) (b) (c)

(d) (e) (f)

Fig. 3 Precision@50 of the different methods under different choices of dimension with respect to benchmark datasets CiteULike (a), M1M (b), and M10M (c), and Recall@50 of the different methods under different choices of dimension with respect to benchmark datasets CiteULike (d), M1M (e), and M10M (f)

are critical for improving recommendation accuracy. In the future, we will investigate the influence of all these information sources and further enhance the performance of HAM.

Contributors

Cheng-wei WANG, Teng-fei ZHOU, Chen CHEN, Tian-lei HU, and Gang CHEN discussed the idea. Cheng-wei WANG designed the research. Chen CHEN processed the data. Teng-fei ZHOU wrote the code and conducted the experiments. Cheng-wei WANG drafted the manuscript. Teng-fei ZHOU, Chen CHEN, and Tian-lei HU helped organize the manuscript. Gang CHEN revised and finalized the paper.

Compliance with ethics guidelines

Cheng-wei WANG, Teng-fei ZHOU, Chen CHEN, Tian-lei HU, and Gang CHEN declare that they have no conflict of interest.

References

Adomavicius G, Tuzhilin A, 2005. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Trans Knowl Data Eng*, 17(6):734-749. https://doi.org/10.1109/TKDE.2005.99

Bansal T, Belanger D, McCallum A, 2016. Ask the GRU: multi-task learning for deep text recommendations. Proc 10th ACM Conf on Recommender Systems, p.107-114. https://doi.org/10.1145/2959100.2959180

Bennett J, Lanning S, 2007. The Netflix prize. Proc KDD Cup and Workshop, p.35.

Cai XY, Han J, Yang L, 2018. Generative adversarial network based heterogeneous bibliographic network representation for personalized citation recommendation. 32nd AAAI Conf on Artificial Intelligence, p.5747-5754.

Chorowski JK, Bahdanau D, Serdyuk D, et al., 2015. Attention-based models for speech recognition. Proc 30th Int Conf on Neural Information Processing Systems, p.577-585.

Devooght R, Bersini H, 2016. Collaborative filtering with recurrent neural networks. https://arxiv.org/abs/1608.07400

Goodfellow I, Bengio Y, Courville A, 2016. Deep Learning. MIT Press, Cambridge, MA.

Gopalan PK, Charlin L, Blei D, 2014. Content-based recommendations with Poisson factorization. Proc Advances in Neural Information Processing Systems, p.3176-3184.

Grčar M, Mladenič D, Fortuna B, et al., 2005. Data sparsity issues in the collaborative filtering framework. Proc 7th Int Workshop on Knowledge Discovery on the Web, p.58-76.

He XN, Liao LZ, Zhang HW, et al., 2017. Neural collaborative filtering. Proc 26th Int Conf on World Wide Web, p.173-182. https://doi.org/10.1145/3038912.3052569

Hsieh CK, Yang L, Cui Y, et al., 2017. Collaborative metric learning. Proc 26th Int Conf on World Wide Web, p.193-201.

Jin M, Luo X, Zhu H, et al., 2018. Combining deep learning and topic modeling for review understanding in context-aware recommendation. Proc Conf of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, p.1605-1614.

Kim D, Park C, Oh J, et al., 2016. Convolutional matrix factorization for document context-aware recommendation. Proc 10th ACM Conf on Recommender Systems, p.233-240. https://doi.org/10.1145/2959100.2959165

Kiros R, Zhu Y, Salakhutdinov RR, et al., 2015. Skip-thought vectors. Advances in Neural Information Processing Systems, p.3294-3302.

Koren Y, 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. Proc 14th ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining, p.426-434. https://doi.org/10.1145/1401890.1401944

Koren Y, Bell R, Volinsky C, 2009. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30-37. https://doi.org/10.1109/MC.2009.263

Levy O, Goldberg Y, 2014. Neural word embedding as implicit matrix factorization. Proc 27th Int Conf on Neural Information Processing Systems, p.2177-2185.

Linden G, Smith B, York J, 2003. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Int Comput*, 7(1):76-80. https://doi.org/10.1109/MIC.2003.1167344

Liu CH, Jin T, Hoi SCH, et al., 2017. Collaborative topic regression for online recommender systems: an online and Bayesian approach. *Mach Learn*, 106(5):651-670. https://doi.org/10.1007/s10994-016-5599-z

McLaughlin MR, Herlocker JL, 2004. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. Proc 27th Annual Int ACM SIGIR Conf on Research and Development in Information Retrieval, p.329-336. https://doi.org/10.1145/1008992.1009050

Mhaskar H, Liao Q, Poggio T, 2017. When and why are deep networks better than shallow ones? Proc 31st AAAI Conf on Artificial Intelligence, p.2343-2349.

Neyshabur B, Bhojanapalli S, McAllester D, et al., 2017. Exploring generalization in deep learning. Proc 30th Conf on Advances in Neural Information Processing Systems, p.5947-5956.

Paterek A, 2007. Improving regularized singular value decomposition for collaborative filtering. Proc KDD Cup and Workshop, p.5-8.

Raghu M, Poole B, Kleinberg J, et al., 2017. On the expressive power of deep neural networks. Proc 34th Int Conf on Machine Learning, p.2847-2854.

Rendle S, Freudenthaler C, Gantner Z, et al., 2009. BPR: Bayesian personalized ranking from implicit feedback. Proc 25th Conf on Uncertainty in Artificial Intelligence, p.452-461.

Ruder S, 2017. An overview of multi-task learning in deep neural networks. https://arxiv.org/abs/1706.05098

Salakhutdinov R, Mnih A, 2007. Probabilistic matrix factorization. Proc 20th Int Conf on Neural Information Processing Systems, p.1257-1264.

Salakhutdinov R, Mnih A, Hinton G, 2007. Restricted Boltzmann machines for collaborative filtering. Proc 24th Int Conf on Machine Learning, p.791-798. https://doi.org/10.1145/1273496.1273596

Shoja BM, Tabrizi N, 2019. Customer reviews analysis with deep neural networks for e-commerce recommender systems. *IEEE Access*, 7:119121-119130. https://doi.org/10.1109/ACCESS.2019.2937518

Srebro N, Rennie J, Jaakkola TS, 2004. Maximum-margin matrix factorization. Conf on Neural Information Processing Systems, p.1329-1336.

Srivastava RK, Greff K, Schmidhuber J, 2015. Training very deep networks. Advances in Neural Information Processing Systems, p.2377-2385.

Strub F, Mary J, 2015. Collaborative filtering with stacked denoising autoencoders and sparse inputs. NIPS Workshop on Machine Learning for e-Commerce, p.1-8.

Vaswani A, Shazeer N, Parmar N, et al., 2017. Attention is all you need. 31st Conf on Neural Information Processing Systems, p.5998-6008.

Wang C, Blei DM, 2011. Collaborative topic modeling for recommending scientific articles. Proc 17th ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining, p.448-456. https://doi.org/10.1145/2020408.2020480

Wang H, Wang NY, Yeung DY, 2015. Collaborative deep learning for recommender systems. Proc 21st ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining, p.1235-1244. https://doi.org/10.1145/2783258.2783273

Wang H, Shi XJ, Yeung DY, 2016. Collaborative recurrent autoencoder: recommend while learning to fill in the blanks. Proc 30th Int Conf on Neural Information Processing Systems, p.415-423.

Wu Y, DuBois C, Zheng AX, et al., 2016. Collaborative denoising auto-encoders for top-N recommender systems. Proc 9th ACM Int Conf on Web Search and Data Mining, p.153-162. https://doi.org/10.1145/2835776.2835837