Information Sciences 417 (2017) 72–87

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Deep hybrid recommender systems via exploiting document context and statistics of items☆

Donghyun Kim, Chanyoung Park, Jinoh Oh, Hwanjo Yu*

Pohang University of Science and Technology (POSTECH), Pohang, South Korea

ARTICLE INFO

Article history:
Received 2 December 2016
Revised 13 May 2017
Accepted 21 June 2017
Available online 23 June 2017

2016 MSC:
00-01
99-00

Keywords:
Collaborative filtering
Document modeling
Deep learning
Contextual information
Gaussian noise
Item statistics

ABSTRACT

The sparsity of user-to-item rating data is one of the major obstacles to achieving high rating prediction accuracy of model-based collaborative filtering (CF) recommender systems. To overcome the obstacle, researchers proposed hybrid methods for recommender systems that exploit auxiliary information together with rating data. In particular, document modeling-based hybrid methods were recently proposed that additionally utilize description documents of items such as reviews, abstracts, or synopses in order to improve the rating prediction accuracy. However, they still have two following limitations on further improvements: (1) They ignore *contextual information* such as word order or surrounding words of a word because their document modeling methods use bag-of-words model. (2) They do not explicitly consider *Gaussian noise* differently in modeling latent factors of items based on description documents together with ratings although Gaussian noise depend on *statistics of items*.

In this paper, we propose a robust document context-aware hybrid method, which integrates convolutional neural network (CNN) into probabilistic matrix factorization (PMF) with the statistics of items to both capture contextual information and consider Gaussian noise differently. Our extensive evaluations on three real-world dataset show that our variant recommendation models based on our proposed method significantly outperform the state-of-the-art recommendation models.

1. Introduction

Triggered by Netflix Prize[1] in 2006, model-based collaborative filtering (CF) recommender systems have received significant attention from academia and various industries, especially e-commerce service companies in the last decade [11,20]. However, recently, the exploding growth of the number of users and items in e-commerce service companies such as Amazon[2] and Netflix[3] causes an extremely high sparseness of relationships between users and items. In the end, this data spar-

☆ A preliminary version of the paper, "Convolutional Matrix Factorization for Document Context-Aware Recommendation", appeared in Proc. ACM RecSys 2016. However, this submission has been substantially extended from the previous paper and contains new and major-value added technical contribution and experimental contribution in comparison with the conference publication.

* Corresponding author.
E-mail addresses: kdh5377@postech.ac.kr (D. Kim), pcy1302@postech.ac.kr (C. Park), kurin@postech.ac.kr (J. Oh), hwanjoyu@postech.ac.kr (H. Yu).

[1] http://www.netflixprize.com/.
[2] https://www.amazon.com.
[3] https://www.netflix.com.

http://dx.doi.org/10.1016/j.ins.2017.06.026
0020-0255/

sity deteriorates the rating prediction accuracy of traditional CF recommender systems, because the recommender systems exploit the relationships (i.e. rating data) between users and items in order to model latent factors.

In order to alleviate this problem, several hybrid methods were actively proposed for recommender systems that leverage not only rating data but also auxiliary information such as demography of users [12], social networks [19], trust information of users [15,17], description documents of items [13,16,19,26,27], images of items [6], reviews of users about items [29], quality of users and items [23,24]. Particularly, since description documents of items such as reviews, abstracts or synopses are easily available from various sources without user privacy issues and they describe properties of items in detail, they were frequently used as highly informative features for enhancing the rating prediction accuracy, especially, on the items that have few ratings. To exploit such description documents of items in recommender systems, document modeling methods such as Latent Dirichlet Allocation (LDA) and Stacked Denoising Auto-Encoder (SDAE) were used to hybrid recommendation models [26,27].

However, existing recommendation models have two following limitations on further improvements. First, they ignore *contextual information* such as surrounding words and word orders by representing description documents as bag-of-words models. For example, suppose that the following two sentences are given in a document: "*people **trust** the man.*", "*people betray his **trust** finally.*" Since LDA and SDAE regard the document as a bag of distinguished words, they cannot understand syntactic difference between these words – a verb and a noun, respectively. Such differences within a document need to be taken into account for deeper understanding of the document more precisely when rating is extremely sparse. Second, they do not explicitly consider Gaussian noise differently in modeling latent factors of items based on description documents via the statistics of items (i.e., the number of ratings given to items). For clarity of exposition, suppose that item A has enough ratings to model accurate latent factors based on description documents, while item B does not. Then, we expect that latent factors of item based on description documents A will tend to be modeled so as to have less Gaussian noise (or error) than latent factors of item B. Therefore, such noise need to be carefully considered in order to model latent factors more accurately when rating data is not only extremely sparse but also skewed.

In our preliminary work [9], in order to address the former limitation of previous work, we exploited convolutional neural network (CNN) which enables us to effectively capture contextual information of documents. Furthermore, we proposed a document context-aware hybrid method that integrates CNN into PMF to apply CNN to recommendation tasks. By doing so, we developed a document context-aware recommendation model, *convolutional matrix factorization* (ConvMF), which draws on the recommendation objective of PMF while effectively exploiting both rating data and contextual information. As a result, ConvMF predicts unknown ratings more accurately even when datasets are extremely sparse. Our experiments showed that ConvMF significantly outperforms the state-of-the-art recommendation models. The superiority of ConvMF over various sparsenesses of the datasets verifies that ConvMF generates latent factors of items that reflect contextual information of description documents of items. Also, we investigated whether pre-trained word embedding model helps improve the rating prediction accuracy of ConvMF. Furthermore, we qualitatively demonstrated that ConvMF indeed captures subtle contextual differences of the same word in a document.

However, we do not still resolve the latter limitation of previous work. As an extension of our preliminary work to resolve the latter limitation, we propose a robust document context-aware hybrid method by introducing a new latent factor modeling method for items. In order to explicitly consider Gaussian noise differently, the new latent factor modeling method uses *the numbers of ratings given to items* from a probabilistic perspective with description documents. Finally, we develop a *robust convolutional matrix factorization* model (R-ConvMF), and thus, we make our preliminary model more robust to not only sparse but also skewed datasets by constructing latent factors of items even more accurately. To demonstrate the effectiveness of R-ConvMF, we report extensive new experimental results of R-ConvMF using three different real-world dataset obtained from MovieLens[4] and Amazon[5]. Our implementation and datasets are also available at http://dm.postech.ac.kr/ConvMF.

In this paper, our contributions are summarized as follows:

- We introduce a new latent factor modeling method from a probabilistic point of view to exploit not only contextual information but also the numbers of ratings given to items in order to consider Gaussian noise differently.
- We develop a robust convolutional matrix factorization model based on the new latent factor modeling method to generate more accurate latent factors, which leads to further enhance the rating prediction accuracy.
- We extensively evaluate variant models based on our proposed method using three real-world dataset and we demonstrate the effectiveness of our proposed method by comparing the state-of-the-art models.

The remainder of the paper is organized as follows. Section 2 briefly reviews preliminaries on the most representative collaborative filtering (CF) method, and summarizes recent methods using deep learning methods for CF. Section 3 explains our proposed method with an overview of R-ConvMF, and describes our CNN architecture and how to optimize R-ConvMF. Section 4 experimentally evaluates variant recommendation models based on our proposed method and discusses the evaluation results. Section 5 summarizes our contributions and gives future work.

[4] http://grouplens.org/datasets/movielens/.

[5] Preprocessed Amazon product data (http://jmcauley.ucsd.edu/data/amazon/).

2. Preliminary and related work

In this section, we briefly review matrix factorization (MF) as model-based CF method and introduce recent CF methods using deep learning methods.

2.1. Matrix factorization

Traditional CF methods are categorized into two categories [1]: memory-based CF methods (e.g. nearest neighborhood) [7,11] and model-based CF methods (e.g. latent factor model) [11,20]. In general, model-based CF methods are known to generate more accurate recommendation results [11]. Thus in this section, we describe MF, which is the most popular model-based CF method.

The goal of MF is to find latent factors of users and items on a shared latent space in which the strengths of user-item relationships (i.e., rating by a user given to an item) are computed. To be precise, suppose that we have N users, M items and a user-item rating matrix $R \in \mathbb{R}^{N\times M}$. In MF, latent factors of user i and item j are represented as k-dimensional vectors, $u_i \in \mathbb{R}^k$ and $v_j \in \mathbb{R}^k$. A rating r_{ij} of user i given to item j is approximated by the inner-product $\hat{r}$ of the corresponding latent vectors of user i and item j (i.e. $r_{ij} \approx \hat{r}_{ij} = u_i^T v_j$). A general way of training latent vectors is to minimize a loss function $\mathcal{L}$, which consists of sum-of-squared-error terms between the actual ratings and the predicted ratings and L_2 regularized terms that try to avoid the over-fitting problem as follows:

$$\mathcal{L} = \sum_i^N \sum_j^M I_{ij}(r_{ij} - u_i^T v_j)^2 + \lambda_u \sum_i^N \|u_i\|^2 + \lambda_v \sum_j^M \|v_j\|^2$$

where I_{ij} is an indicator function such that it is 1 if user i rated item j and 0 otherwise, and λs are the L_2 regularizer hyper-parameters.

2.2. Deep learning for collaborative filtering

Due to remarkable performance of deep learning methods in computer vision and NLP fields, several researchers recently adopted deep learning methods such as (de-noising) auto-encoder [12,22,27,28], recurrent neural network (RNN) [2] or CNN [6,25] to recommender systems.

Wu *et al.* [28] and Sedhain *et al.* [22] used (de-noising) auto-encoder to build user or an item-based CF methods that only consider rating data. However, like traditional CF methods, their models also suffer from the sparsity problem. To overcome this problem, Wang *et al.* proposed CDL [27], which is enhanced version of *collaborative topic regression* (CTR) [26] model by probabilistically combining topic modeling method (LDA) and model-based CF recommender system. CDL replaces LDA with the de-noising auto-encoder to encode bag-of-words representations of description documents of items into compressed representations. These compressed representations are used to help generate more accurate latent factors of items, especially for items with few ratings. As a variant of CDL, Li *et al.* proposed DCF [12] to deal with both user and item side auxiliary information by replacing the denoising auto-encoder of CDL with a marginalized denoising auto-encoder that enables to obtain closed form solutions. Nevertheless, their models cannot effectively encode *contextual information* due to the inherent limitation of the bag-of-word model as we explained in Section 1. Different from this work, our model reflects *contextual information* into latent factors of items. Furthermore, in order to generate more accurate latent factors of items even when datasets are extremely sparse and skewed, our model takes into account the statistic of datasets. We will discuss it in detail in Section 3.2 and Section 4.2.1.

Recently, Bansal *et al.* [2] used gated recurrent units (GRUs), a type of RNNs, in order to effectively encode documents for the multi-task learning with implicit feedback datasets. Besides, He and McAuley [6] used image features of items obtained from pre-trained CNN, which is trained by massive external image datasets in order to improve top-N recommendation. However, different from our work, their model is not an fully integrated model of CNN and CF. Moreover, since their pre-trained CNN is designed for image classification, it is not suitable to our task. van den Oord *et al.* [25] also applied CNN to music recommendation where they analyzed songs in a acoustic analysis perspective through CNN, and proposed a model that predicts the ratings based on the latent factors of items obtained by acoustic CNN. However, their CNN model, designed for acoustic signal processing, is not suitable for processing documents. Specifically, acoustic signals and documents have an inherent difference on the quality of surrounding features. An acoustic signal is inherently similar to its surrounding signals, i.e., the signals that have slight time difference, whereas a word in the document has a large semantical difference from the surrounding words. Such difference in the degree of similarity between surrounding features requires different CNN architectures. Furthermore, the model does not fully reflect rating data. In particular, latent factors of items are mainly determined by the results of audio signal analysis via CNN rather than rating data. Thus, overall recommendation performance falls short of that of weighted matrix factorization (WMF) [8], which is one of the conventional MF-based collaborative filtering techniques that deals with implicit feedback dataset.

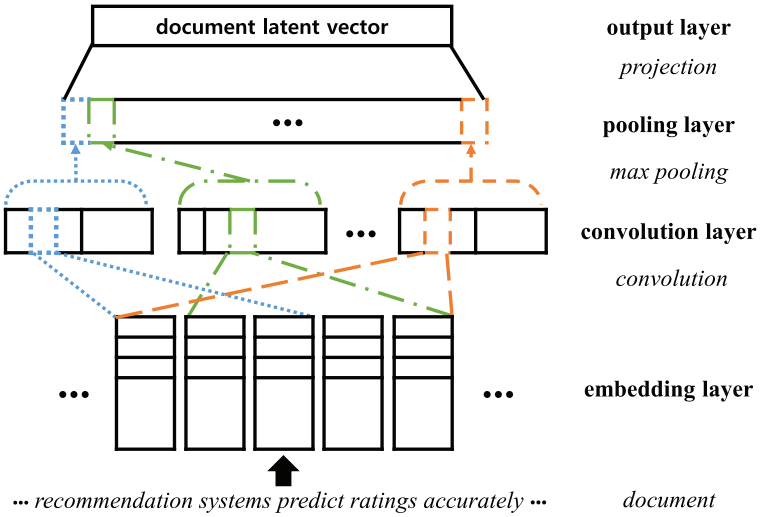


Fig. 1. Our CNN architecture of R-ConvMF.

3. Robust document context-Aware hybrid method

In this section, we introduce details of our proposed method through three steps: (1) We explain the detailed architecture of our CNN, which generates document latent vectors by analyzing description documents of items with capturing contextual information. (2) Beyond our preliminary work, we introduce our final recommendation model, *robust convolutional matrix factorization* (R-ConvMF). Also we describe the key idea to bridge PMF and CNN in order to utilize both ratings and description documents of items while considering Gaussian noise differently. (3) Finally, we describe how to optimize latent variables of R-ConvMF.

3.1. CNN Architecture of R-ConvMF

The objective of our CNN architecture is to generate document latent vectors from description documents of items with capturing contextual information. Obtained document latent vectors will be used in our recommendation model. Fig. 1 shows our CNN architecture that consists of four layers; (1) embedding layer, (2) convolution layer, (3) pooling layer, and (4) output layer.

3.1.1. Embedding layer

The embedding layer transforms an input document (an order-preserving sequence of words) into a dense numeric matrix that represents the document for the next convolution layer. In detail, regarding the document as a sequence of l words, we represent the document as a matrix by concatenating word embedding vectors of words in the document. The word embedding vectors are randomly initialized or initialized with pre-trained word embedding models such as Glove [18]. Then, the document matrix $D \in \mathbb{R}^{p \times l}$ becomes:

$$D = \begin{bmatrix} & | & | & | & \\ \cdots & w_{i-1} & w_i & w_{i+1} & \cdots \\ & | & | & | & \end{bmatrix}$$

where l is the length of the document, and p is the size of embedding dimension for each word w_i. The word embedding vectors will be further trained through the optimization process.

3.1.2. Convolution layer

The purpose of the convolution layer is to extract contextual features. Since documents are inherently different from signal processing or computer vision in the nature of contextual information, we use the convolution architecture in [4,10] to analyze documents properly. A contextual feature $c_i^j \in \mathbb{R}$ is extracted by jth shared weight $W_c^j \in \mathbb{R}^{p \times ws}$ whose window size ws determines the number of surrounding words:

$$c_i^j = f(W_c^j * D_{(:,i:(i+ws-1))} + b_c^j) \tag{1}$$

where $*$ is a convolution operator, $b_c^j \in \mathbb{R}$ is a bias for W_c^j and f is a non-linear activation function. Among non-linear activation functions such as sigmoid, tanh and rectified linear unit (ReLU), we use ReLU to avoid the problem of vanishing

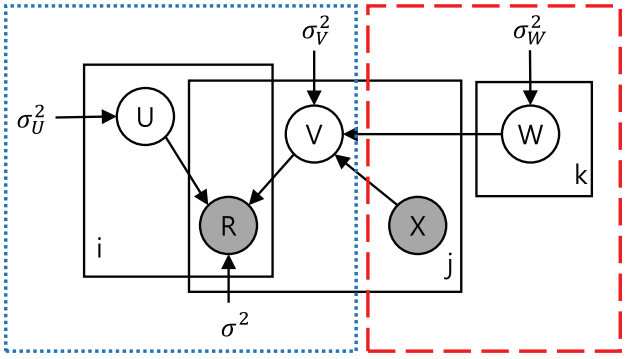


Fig. 2. The graphical model for R-ConvMF: PMF part in left (dotted-blue); CNN part in right (dashed-red). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

gradient which causes slow optimization convergence and may lead to a poor local minimum [5,14]. Then, a contextual feature vector $c^j \in \mathbb{R}^{l-ws+1}$ of a document with W_c^j is constructed by Eq. (1):

$$c^j = [\ c_1^j,\ c_2^j,\ \ldots,\ c_i^j,\ \ldots,\ c_{l-ws+1}^j] \tag{2}$$

However, one shared weight captures one type of contextual features. Thus, we use multiple shared weights to capture multiple types of contextual features, which enable us to generate contextual feature vectors as many as the number n_c of W_c. (i.e. W_c^j where $j = 1, 2, \ldots, n_c$).

3.1.3. Pooling layer

The pooling layer processes variable-length documents by extracting representative features from the convolution layer and constructing a fixed-length feature vector through a pooling operation. Precisely, after the convolution layer, a document is represented as n_c contextual feature vectors, where each contextual feature vector has a variable length (i.e., $l - ws + 1$ contextual feature). However, such representation imposes two problems: (1) there are too many contextual features c_i, where most contextual features might not help enhance the performance, (2) the length of contextual feature vectors varies, which makes it difficult to construct the following layers. Therefore, we exploit max-pooling, which reduces the representation of a document into a n_c fixed-length vector by extracting only the maximum contextual feature from each contextual feature vector as follows.

$$d_f = [\max(c^1), \max(c^2), \ldots, \max(c^j), \ldots, \max(c^{n_c})]$$

where c^j is a contextual feature vector of length $l - ws + 1$ extracted by jth shared weight W_c^j in Eq. (2).

3.1.4. Output layer

Generally, at output layer, high-level features obtained from the previous layer should be converted to fit a specific task. Thus, we project d_f on a k-dimensional space of user and item latent variables for our recommendation task, which finally produces a document latent vector by using conventional nonlinear projection:

$$s = \tanh(W_{f_2}\{\tanh(W_{f_1} d_f + b_{f_1})\} + b_{f_2}) \tag{3}$$

where $W_{f_1} \in \mathbb{R}^{f \times n_c}$, $W_{f_2} \in \mathbb{R}^{k \times f}$ are projection matrices, and $b_{f_1} \in \mathbb{R}^f$, $b_{f_2} \in \mathbb{R}^k$ is a bias vector for W_{f_1}, W_{f_2} with $s \in \mathbb{R}^k$.

Eventually, through the above processes, our CNN is simply regarded as a function that takes a document as an input, and returns a latent vector of each document as an output:

$$s_j = CNN_W(X_j) \tag{4}$$

where W denotes all the weight (W_c^j where $j = 1, 2, \ldots, n_c$, W_{f_1} and W_{f_2}) and bias (b_c^j where $j = 1, 2, \ldots, n_c$, b_{f_1} and b_{f_1}) variables in our CNN to prevent clutter, and X_j denotes an input document of item j, and s_j denotes a document latent vector of item j. In the next subsection, we will explain how document latent vectors are used for recommender systems.

3.2. Robust convolutional matrix factorization (R-ConvMF)

Fig. 2 shows an overview of how to integrate CNN into PMF in the form of probabilistic graphical model. Suppose we have N users and M items, and observed ratings are represented by $R \in \mathbb{R}^{N \times M}$ matrix. Then, our goal is to find user and item latent variables ($U \in \mathbb{R}^{k \times N}$ and $V \in \mathbb{R}^{k \times M}$) whose product ($U^T V$) approximates the rating matrix R. Note that U (or V) consists of N user (or M item) vectors that represent latent factors of users (or items).

3.2.1. Modeling ratings

In the probabilistic point of view, the conditional distribution of observed ratings as in [20] is given by

$$p(R|U,V,\sigma^2)=\prod_{i}^{N}\prod_{j}^{M}\mathcal{N}(r_{ij}|u_i^T v_j,\sigma^2)^{I_{ij}},$$

where $\mathcal{N}(x|\mu,\sigma^2)$ is the probability density function of Gaussian distribution with mean μ and variance σ^2; I_{ij} is an indicator function as mentioned in Section 2.1; u_i denotes an ith column vector of U, and v_j denotes a jth column vector of V. An inner-product of u_i and v_j is used as the mean of Gaussian distribution to model a rating r_{ij} of a user i given to an item j with σ^2.

3.2.2. Modeling users

For the generative model of the user latent variable, we place a conventional priori, a zero-mean spherical Gaussian prior on column vectors (i.e. independent users) of the user latent variable with variance σ_U^2.

$$p(U|\sigma_U^2)=\prod_{i}^{N}N(u_i|0,\sigma_U^2 I)$$

3.2.3. Modeling items

Exploiting contextual information: Unlike the generative model for the item latent variable in conventional PMF, we assume that each column vector of the item latent variable is generated with three variables in order to exploit contextual information in documents: (1) an internal weight variable W in our CNN, (2) an input description document X_j of an item j, and (3) an epsilon variable ϵ_j as Gaussian noise, which enables us to further optimize the column vector of the item latent variable. Thus, each column vector of the item latent variable is represented as the following equation.

$$v_j=CNN_W(X_j)+\epsilon_j, \quad \text{where } \epsilon_j\sim N(0,\sigma_V^2 I) \tag{5}$$

Accordingly, the conditional distribution of the item latent variable is given by

$$p(V|W,X,\sigma_V^2)=\prod_{j}^{M}N(v_j|CNN_W(X_j),\sigma_V^2 I), \tag{6}$$

where X is the set of description documents of items. A document latent vector from our CNN is used as the mean of Gaussian noise of an item. Thus, Eq. (6) plays an important role as a bridge between PMF and CNN that helps to analyze both description documents and ratings.

Considering Gaussian noise differently: Taking into account the results of [21], the number of ratings given to each item tends to affect the variance of the Gaussian noise of the corresponding column vector of the item latent variable. Specifically, we observe that the number of ratings and the variance for each item are inversely proportional. Therefore, for further enhancing rating prediction accuracy, we conclude that items with many ratings should be explicitly modeled to have less variance in Gaussian noise, while items with few ratings should be explicitly modeled to have more variance Gaussian noise. However, in addition to the limitation of existing work, in our preliminary work [9], we modeled each epsilon variable identically for all items (i.e. all items are modeled to have the same variance in Gaussian noise.). To address this limitation, we introduce the new latent factor modeling method for items, which explicitly reflects the number of ratings given to each item into Eq. (5) so as to consider Gaussian noise differently as follows:

$$v_j=CNN_W(X_j)+\epsilon_j, \quad \text{where } \epsilon_j\sim N(0,\frac{\sigma_V^2}{f(n_j)}I) \tag{7}$$

Note that n_j refers to the number of ratings given to item j, and $f(\cdot)$ refers to a monotonic increasing transformation function such as the square root or the logarithm to deal with the extremely skewed distribution of the number of ratings given to each item. Then, the conditional distribution of the item latent variable is given by

$$p(V|W,X,\sigma_V^2)=\prod_{j}^{M}N(v_j|CNN_W(X_j),\frac{\sigma_V^2}{f(n_j)}I) \tag{8}$$

This change of the distribution of the item latent variable eventually has the benefit of improving the optimization process of our preliminary model, and we discuss this impact in detail in Section 4.2.1.

For each weight w_k in W, we place a zero-mean spherical Gaussian prior, the most commonly used prior.

$$p(W|\sigma_W^2)=\prod_{k}^{|w_k|}N(w_k|0,\sigma_W^2)$$

By putting together, the posterior probability of trainable variables U, V and W in our proposed model is estimated as follows:

$$\begin{aligned} & p(U,V,W|R,X,\sigma^2,\sigma_U^2,\sigma_V^2,\sigma_W^2) \\ & \approx p(R|U,V,\sigma^2)p(U|\sigma_U^2)p(V|W,X,\sigma_V^2)p(W|\sigma_W^2) \end{aligned} \tag{9}$$

3.3. Optimization methodology

Given Eq. (9) in Section 3.2 and our CNN architecture, we use maximum a posteriori (MAP) estimation to optimize the variables U, V and W as follows:

$$\begin{aligned} & \max_{U,V,W} p(U,V,W|R,X,\sigma^2,\sigma_U^2,\sigma_V^2,\sigma_W^2) \\ & = \max_{U,V,W}[p(R|U,V,\sigma^2)p(U|\sigma_U^2)p(V|W,X,\sigma_V^2)p(W|\sigma_W^2)] \end{aligned} \tag{10}$$

By taking negative logarithm on Eq. (10), it is reformulated to minimize below loss function $\mathcal{L}$.

$$\begin{aligned} \mathcal{L} = & \sum_i^N \sum_j^M \frac{I_{ij}}{2}(r_{ij} - u_i^T v_j)^2 + \frac{\lambda_U}{2}\sum_i^N \|u_i\|^2 \\ & + \frac{\lambda_V}{2}\sum_j^M f(n_j)\|v_j - CNN_W(X_j)\|^2 + \frac{\lambda_W}{2}\sum_k^{|w_k|}\|w_k\|^2, \end{aligned} \tag{11}$$

where λ_U is σ^2/σ_U^2, λ_V is σ^2/σ_V^2, and λ_W is σ^2/σ_W^2. Note that although we use Eq. (8) for the conditional distribution of the item latent variable instead of Eq. (6), Eq. (11) can be simply reduced to our preliminary model (ConvMF) using Eq. (6) without using the number of ratings of each item (i.e. set $f(n_j) = 1$ for all items ($j = 1, \ldots, M$)).

To minimize $\mathcal{L}$, we adopt coordinate descent which iteratively updates a trainable variable while fixing the remaining variables. Specifically, Eq. (11) becomes a quadratic function with respect to U (or V) while temporarily assuming W and V (or U) to be constant. Then, the solution of U (or V) can be analytically computed in a closed form by simply differentiating the loss function $\mathcal{L}$ with respect to u_i (or v_j) as follows.

$$u_i \leftarrow (VI_iV^T + \lambda_U I_K)^{-1}VR_i \tag{12}$$

$$v_j \leftarrow (UI_jU^T + f(n_j)\lambda_V I_K)^{-1}(UR_j + f(n_j)\lambda_V CNN_W(X_j)) \tag{13}$$

where I_i is a diagonal matrix with I_{ij}, $j = 1, \ldots, M$ as its diagonal elements and R_i is a vector with $(r_{ij})_{j=1}^M$ for user i. For item j, I_j and R_j are similarly defined as I_i and R_i, respectively. Eq. (13) shows the effect of a document latent vector of CNN (i.e. $CNN_W(X_j)$) in updating v_j through λ_V as a balancing parameter that regulates how much document information is used as in [26].

However, W cannot be updated through building an analytic solution as we do for U and V, because W is closely related to the non-linearity in the CNN architecture such as the max-pooling layer and non-linear activation functions. Nonetheless, we observe that when U and V are temporarily constant, $\mathcal{L}$ can be interpreted as a weighted squared error function with L_2 regularized terms as follows:

$$\begin{aligned} \mathcal{E}(W) = & \quad \frac{\lambda_V}{2}\sum_j^M f(n_j)\|(v_j - CNN_W(X_j))\|^2 \\ & + \frac{\lambda_W}{2}\sum_k^{|w_k|}\|w_k\|^2 + \text{constant} \end{aligned} \tag{14}$$

We use the back-propagation algorithm which is typically used in neural networks in order to obtain Eq. (15), and we optimize W until convergence of $\mathcal{E}$ or reaching the pre-defined number of iteration.

$$\nabla_{w_k}\mathcal{E}(W) = -\lambda_V \sum_j^M f(n_j)(v_j - \nabla_{w_k}CNN_W(X_j)) + \lambda_W w_k \tag{15}$$

As shown in Algorithm 1, the overall optimization process (U, V and W are alternatively updated) is repeated until satisfying validation-based early stopping with the pre-defined patience parameter to avoid the over-fitting. With optimized U, V, and W, finally we can predict unknown ratings of users on items:

$$\begin{aligned} r_{ij} & \approx \mathbb{E}[r_{ij}|u_i^T v_j, \sigma^2] \\ & = u_i^T v_j = u_i^T(CNN_W(X_j) + \epsilon_j) \end{aligned}$$

Recall that $v_j = CNN_W(X_j) + \epsilon_j$.

3.4. Time complexity analysis

For each epoch, the user and item latent variables are updated in $O(k^2 n_R + k^3 N + k^3 M)$, where n_R is the number of observed ratings. To be specific, line 7 takes $O(k^2 n_i + k^3)$ time where n_i is the number of ratings of user i, and since document latent vectors are already computed while updating W, line 10 takes $O(k^2 n_j + k^3)$ time where n_j is the number of ratings of item j. Time complexity for updating all weight and bias variables of CNN (i.e. W) is typically dominated by the computation of convolution layer, and thus W is updated in $O(n_c \cdot p \cdot l \cdot M)$. As a result, the total time complexity per epoch is $O(k^2 n_R + k^3 N + k^3 M + n_c \cdot p \cdot l \cdot M)$, and this optimization process scales linearly with the size of given data.

Algorithm 1 R-ConvMF algorithm.

Require: R: user-item rating matrix, X: description documents of items
Ensure: optimized latent variables U, V, and W

Initialize U and W randomly
for $j \leq M$ **do**
  Initialize V by $v_j \leftarrow CNN_W(X_j)$
end for
repeat
  for $i \leq N$ **do**
    Update $u_i \leftarrow (VI_iV^T + \lambda_U I_K)^{-1} VR_i$
  end for
  for $j \leq M$ **do**
    Update $v_j \leftarrow (UI_jU^T + f(n_j)\lambda_V I_K)^{-1}(UR_j + f(n_j)\lambda_V CNN_W(X_j))$
  end for
  repeat
    for $j \leq M$ **do**
      Perform backpropagation and update W through Eqn. (15)
    end for
  until convergence
until satisfying early stopping using a validation set

4. Experiment

In this section, we evaluate the empirical performance of variant recommendation models based on our proposed method on three real-world datasets. Compared with our preliminary work [9], our new experiments are conducted to solve the following questions:

Q1. Does taking into account Gaussian noise differently for items improve the accuracy of our preliminary models?
Q2. How does the replacement of CNN to LSTM or GRU affect the performance (accuracy, training time) of R-ConvMF?
Q3. How does lemmatization affects the accuracy of R-ConvMF?
Q4. How hyper-parameters such as regularized parameters, the number of kernel, and max length of documents affect the accuracy of our models?

4.1. Experimental setting

4.1.1. Datasets

To demonstrate the effectiveness of our recommendation models in terms of rating prediction, we used three real-world datasets obtained from MovieLens and Amazon. These datasets consist of users' explicit ratings on items on a scale of 1 to 5. Since MovieLens datasets does not include description documents of items, we obtained documents (i.e., plot summary) of corresponding items from IMDB.[6] Amazon dataset includes reviews on items and thus we used reviews as description documents of items.

Similar to [26] and [27], we preprocessed description documents for all datasets as follows: (1) We set maximum length of input documents to 300; (2) We removed stop words; (3) We calculated tf-idf score for each word; (4) We removed corpus-specific stop words that have the document frequency higher than 0.5; (5) We selected top 8000 distinct words as a vocabulary as in [27]; (6) We removed all non-vocabulary words from input documents. As a result, average numbers of words per document are 97.09 on MovieLens-1m (ML-1m), 92.05 on MovieLens-10m (ML-10m) and 91.50 on Amazon Instant Video (AIV), respectively.

We removed items that do not have their description documents in each dataset, and for the case of Amazon dataset, due to the extreme sparseness, we removed users that have less than 3 ratings. As a result, statistics of each data show that three datasets have different characteristics (Table 1). Precisely, even though several users are removed by preprocessing, Amazon dataset is still extremely sparse compared with the others.

4.1.2. Competitors and parameter setting

We compared our models with the following competitors. Table 2 shows their conceptual differences.

- PMF [20]: Probabilistic Matrix Factorization is a standard rating prediction model that only uses ratings for collaborative filtering.

[6] Plot summaries are available at http://www.imdb.com/.

Table 1
Data statistic on three real-world datasets .

Dataset	# users	# items	# ratings	density
ML-1m	6,040	3,544	993,482	4.641%
ML-10m	69,878	10,073	9,945,875	1.413%
AIV	29,757	15,149	135,188	0.030%

Table 2
Comparison between 6 models: our variant models and the three competitors.

Model	The usage of				
	ratings	doc.	context. info.	pre-train word emb.	statistics of items
PMF	✓	–	–	–	–
CTR	✓	✓	–	–	–
CDL	✓	✓	–	–	–
ConvMF	✓	✓	✓	–	–
ConvMF+	✓	✓	✓	✓	–
R-ConvMF	✓	✓	✓	✓	✓

Table 3
Parameter Setting of λ_U and λ_V.

	ML-1 m		ML-10 m		AIV	
Model	λ_U	λ_V	λ_U	λ_V	λ_U	λ_V
PMF	0.01	10000	10	100	0.1	0.1
CTR	100	1	10	100	10	0.1
CDL	10	100	100	10	0.1	100
ConvMF	100	10	10	100	1	100
ConvMF+	100	10	10	100	1	100
R-ConvMF	10	100	10	100	1	100

- CTR [26]: Collaborative Topic Regression is a state-of-the-art recommendation model, which uses both ratings and documents by combining collaborative filtering (PMF) and topic modeling (LDA).
- CDL [27]: Collaborative Deep Learning is another state-of-the-art recommendation model, which enhances rating prediction accuracy by analyzing documents using SDAE.
- ConvMF: Convolutional Matrix Factorization is our preliminary model, which considers contextual information together with ratings.
- ConvMF+: ConvMF+ is ConvMF with the pretraining word embedding model [18].
- R-ConvMF: Robust Convolutional Matrix Factorization is an enhanced model of ConvMF+ in which Gaussian noise are explicitly taken into account differently. We use the square root with normalization by mean of the numbers of ratings given to items for the transformation function.

As reported in [27] (the best competitor), we set the size of latent dimension of U and V to 50, and initialized U and V randomly from 0 to 1. Table 3 shows the best performing values of common parameters (λ_U, λ_V) of each model found by the grid search ($\lambda_U \in \{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$ and $\lambda_V \in \{0.01, 0.1, 1, 10, 100, 1000, 10000, 100000\}$). Since we used explicit datasets, we set the precision parameter of CTR and CDL to 1 if r_{ij} is observed and 0 otherwise. The rest of the CDL parameters were set as reported in [27].[7]

4.1.3. Implementation detail

We implemented our models using Python 2.7 and Keras 0.3.3 [3] with NVidia Geforce Titan X GPU. To train the weights of CNN, we used mini-batch based RMSprop, and each mini-batch consists of 128 training items. As for the detailed CNN architecture, we used the following settings:

1. We set the maximum length of documents to 300.
   (a) ConvMF: we initialized word latent vectors randomly with the dimension size of 200. These word latent vectors will be trained through the optimization process.

[7] We tried different settings but the performance was almost the same.

Table 4
Overall test RMSE.

Model	Dataset		
	ML-1m	ML-10m	AIV
PMF	0.8971 (0.0020)	0.8311 (0.0010)	1.4118 (0.0105)
CTR	0.8969 (0.0027)	0.8275 (0.0004)	1.5496 (0.0104)
CDL	0.8879 (0.0015)	0.8186 (0.0005)	1.3594 (0.0139)
ConvMF	**0.8531** (0.0018)	**0.7958** (0.0006)	**1.1337** (0.0043)
ConvMF+	**0.8549** (0.0018)	**0.7930** (0.0006)	**1.1279** (0.0043)
R-ConvMF	**0.8470** (0.0010)	**0.7844** (0.0007)	**1.1012** (0.0055)

(b) ConvMF+ and R-ConvMF: we initialized word latent vectors by the pre-trained word embedding model with the dimension size of 200. These word latent vectors will be trained through the optimization process.

2. For the convolution layer, we used various window sizes (3, 4, and 5) for shared weights to consider various length of surrounding words, and we used 100 shared weights per window size.
3. For the output layer, we set the dimension size of the intermediate projection vector in Eq. (3) as 200.
4. Instead of the L_2 regularizer related to weights of CNN, we used dropout and set dropout rate to 0.2 to prevent CNN from over-fitting.

4.1.4. Evaluation protocol

To evaluate the overall performance of each model on the real world datasets, we randomly split each dataset into a training set (80%), a validation set (10%) and a test set (10%). The training set contains at least a rating on every user and item so that PMF deals with all users and items.

As the evaluation measure, we used root mean squared error (RMSE), which is directly related to the objective functions of conventional rating prediction models. Specifically, since our models and competitors use the squared error based objective function between a true rating and a predicted rating, undoubtedly RMSE is the most reasonable metric in our evaluation.

$$\text{RMSE} = \sqrt{\frac{\sum_{i,j}^{N,M}(r_{ij} - \hat{r}_{ij})^2}{\#\text{ of ratings}}}$$

We reported test errors of each model, which gives the lowest validation errors within 200 iterations with early-stopping. For the reliability of the results, we repeated this evaluation procedure 5 times starting from the data split process, and finally we reported means and standard deviations of test errors.

4.2. Experimental results

Table 4 shows the overall rating prediction errors of our variant models, and the three competitors. Compared with the three competitors, our variant models achieve significant improvements for all the datasets. Following subsections give detailed answers to the questions as we previously mentioned at the beginning of Section 4.

4.2.1. Impact of taking into account Gaussian noise differently in modeling latent factors for items

Table 4 shows that R-ConvMF, which takes into account Gaussian noise differently performs better than ConvMF and ConvMF+ for three real-world datasets. Specifically, the improvements over ConvMF (or ConvMF+) are 0.71% (0.93%), 1.43% (1.08%) and 2.87% (2.37%) for ML-1m, ML-10m and AIV dataset, respectively. Note that *the more skewed the dataset, the more significant the improvement of R-ConvMF*.

Here, we discuss why these results are reported in detail. As we explained in Section 3.2.3, we reflect the number of ratings given to each item into the variance of the probability distribution to deal with Gaussian noise differently for items. As a result of the explicit consideration of Gaussian noise differently, Eq. (15) in Section 3.3 implies that the trainable variables in CNN of R-ConvMF are dominantly learned by columns of the item variable where corresponding items have a relatively high number of ratings (i.e., the items tend to be modeled so as to have less Gaussian noise). Besides, it is well-known that the higher the number of ratings given to each item, the more likely the corresponding columns of the item latent variable are trained more accurately. Thus, we infer that the CNN of R-ConvMF can be trained more accurately by accurately trained columns of the item latent variable. Moreover, the accurately trained CNN of R-ConvMF generates more accurate document latent vectors, which help update the item latent variable more accurately through Eq. (13). *Therefore, we conclude that R-ConvMF makes a series of these update processes work in synergy with one another, which boosts the rating prediction accuracy.*

For further investigation of R-ConvMF, we generate seven additional datasets with different sparseness by randomly sampling from ML-1m dataset. As a result, Figs. 3 and 4 show that while ML-1m (sparse) and ML-1m (dense) datasets whose

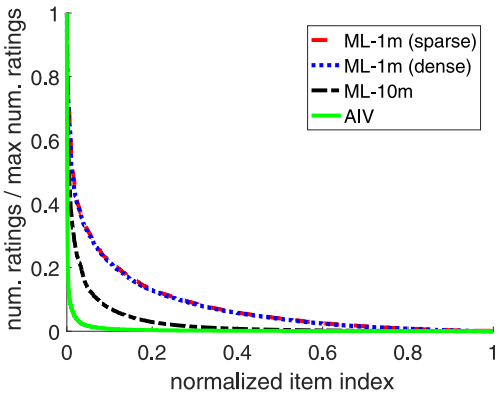


Fig. 3. Skewness of the number of ratings for items on each dataset – Amazon dataset is the most skew.

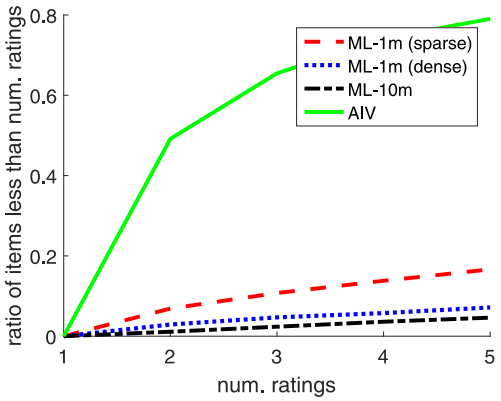


Fig. 4. Ratio of items that have less than num. ratings to each entire dataset – 50% of items in Amazon dataset have only one rating.

Table 5
Test RMSE over various sparseness of training data on ML-1m dataset.

	Ratio of training set to the entire dataset (density)						
Model	20% (0.93%)	30% (1.39%)	40% (1.86%)	50% (2.32%)	60% (2.78%)	70% (3.25%)	80% (3.71%)
PMF	1.0168	0.9711	0.9497	0.9354	0.9197	0.9083	0.8971
CTR	1.0124	0.9685	0.9481	0.9337	0.9194	0.9089	0.8969
CDL	1.0044	0.9639	0.9377	0.9211	0.9068	0.8970	0.8879
ConvMF	**0.9745**	**0.9330**	**0.9063**	**0.8897**	**0.8726**	**0.8676**	**0.8531**
ConvMF+	**0.9748**	**0.9316**	**0.9093**	**0.8905**	**0.8771**	**0.8657**	**0.8549**
R-ConvMF	**0.9270**	**0.9069**	**0.8876**	**0.8778**	**0.8645**	**0.8652**	**0.8471**

training sets consist of 20% and 80% of the entire dataset, respectively, have almost the same skewness over items, the former has relatively smaller number of ratings on items than the latter one. Table 5 shows that three variant models based on our proposed method significantly outperform three competitors over all ranges of sparseness, and Fig. 5 shows that the improvements of ConvMF and ConvMF+ over PMF are almost consistent between 4% and 5%. Interestingly, compared with ConvMF and ConvMF+, when the dataset becomes more extremely sparse, the improvement of R-ConvMF over PMF consistently increases from 5.57% to 8.83%. The results demonstrate that *in order to optimize the item latent variable effectively, CNN of R-ConvMF is more seamlessly integrated with PMF for the rating prediction task by considering Gaussian noise differently through the statistics of items, which eventually makes R-ConvMF more robust to extremely sparse datasets.*

4.2.2. Performance comparison between usages of CNN, LSTM and GRU

Based on our proposed method, we additionally develop two variant models by replacing CNN with standard LSTM and GRU to evaluate the effectiveness of our final model, R-ConvMF, in terms of understanding documents for recommendation tasks. LSTM and GRU as variants of RNN among deep learning methods are originally designed for sequence processing in NLP and IR fields, and thus the two variant models also capture contextual information.

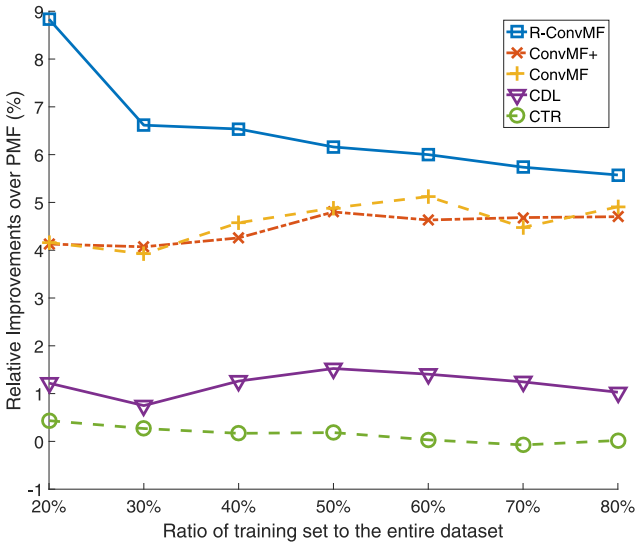


Fig. 5. Relative improvements over PMF of five models.

Table 6
Performance comparison between usages of CNN, LSTM and GRU: *a* – RMSE, *b* – STD, *c* – train time for *U, V* and *W* per epoch and *d* – train time for *W* per inner-epoch as in Line 13–15 of Algorithm 1 .

Module	RMSE *a(b)	train time *c(d)
CNN (R-ConvMF)	1.1012 (0.0055)	40.5 (5) sec.
LSTM (variant)	1.1494 (0.0166)	278 (52) sec.
GRU (variant)	1.1304 (0.0051)	212 (39) sec.

In order to clearly show the differences of the performance, we evaluate R-ConvMF and the two variant models on Amazon dataset as the most sparse dataset among three datasets with the same experimental setting in Section 4.1. Accordingly, we easily compare the impact of understanding documents among each model. For the two variant models, we use the same parameter settings; for λ_U and λ_V, we conducted the grid search in a similar way to our models, but the two variant models shows the best results when using the same parameters as in R-ConvMF. Table 6 shows RMSE and train time of each model on Amazon dataset, and R-ConvMF gives the best results in terms of both effectiveness and efficiency.

Let us discuss the results in terms of effectiveness. Compared with LSTM and GRU, CNN tries to focus on extracting representative local features from a sequence such as keywords in a document. These features are used to construct a latent vector for a document of an item. However, LSTM and GRU try to construct a latent vector by accumulating equally and sequentially extracted features from a sequence. Moreover, this characteristic makes two variant models more likely to be sensitive to noises such as typos in description documents for items. Even with clean data, they are more suitable to regenerate the entire sequence from general features in tasks such as machine translation task or language modeling task. Therefore, since our recommendation task is to find latent factors for items, which best represent items, we infer that CNN is more appropriate than LSTM and GRU.

In terms of efficiency, train time of two variant models is about five to seven times slower than that of R-ConvMF. This is caused by the differences in train speed of *W* between CNN, LSTM and GRU as shown in Table 6. To be precise, in order to train *W*, LSTM and GRU use backpropagation through time while CNN uses backpropagation. Solving gradients by backpropagation through time have long-term dependencies, which has limitations in parallel computing in GPU when compared with solving gradients by backpropagation. If two variants analyze very long documents, then train speed of *W* becomes extremely slower. However, since CNN does not have any long-term dependencies for solving gradients, CNN is more preferred to LSTM and GRU for our recommendation task.

4.2.3. Impact of considering lemmatization

We also investigate how lemmatization affects the rating prediction accuracy of R-ConvMF. We use NLTK's WordNet lemmatization and apply it to Amazon dataset. After lemmatization, RMSE of R-ConvMF is 1.1057, and note that before lemmatization, RMSE of R-ConvMF is 1.1012. The performance is slightly decreased by 0.4%, but it seems to be the result of loss of syntactic information of words by lemmatization. Lemmatization is useful when documents is represented as bag-of-word models, because it converts syntactically different words into the same word. However, it is well-known that word

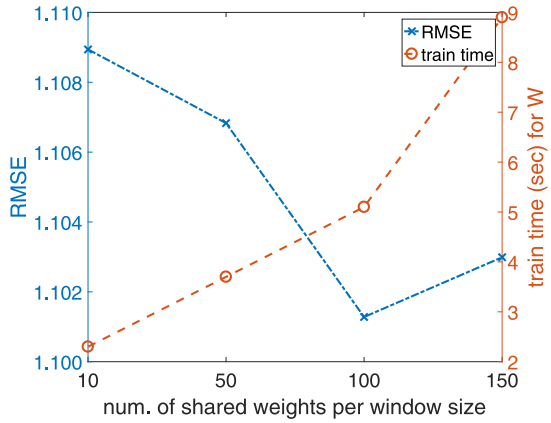


Fig. 6. RMSE and train time for W(train time for W denotes train time for W per inner-epoch as in Line 13–15 of Algorithm 1 in both Figs. 6 and 7) of R-ConvMF w.r.t the number of shared weights per window size on Amazon dataset.

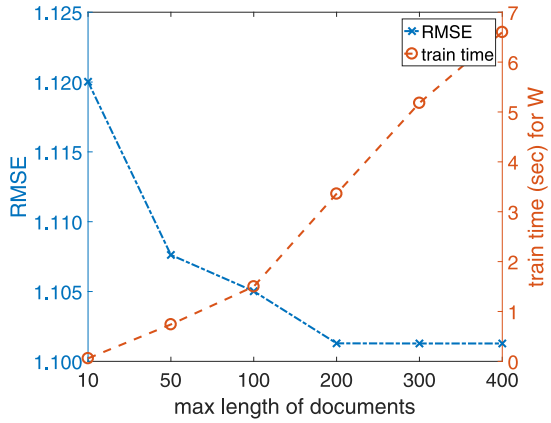


Fig. 7. RMSE and train time for W of R-ConvMF w.r.t max length of documents on Amazon dataset.

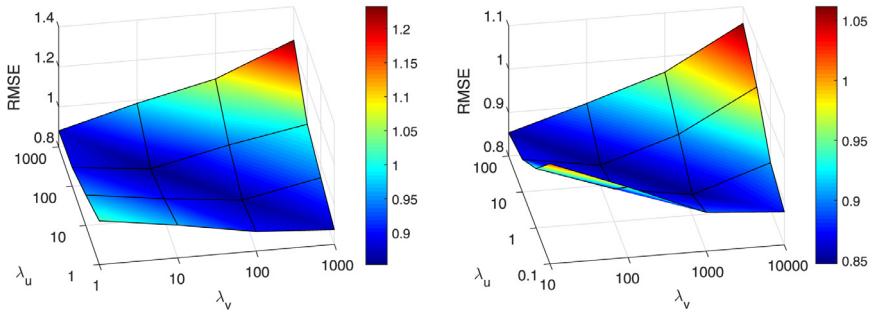
embedding models well capture syntactic and semantic similarity of words through considering contextual information and they represent words as latent vectors, which also include syntactic and semantic information. Thus, it would be more effective to keep words to include syntactical information when we use word embedding models. This is why deep learning methods generally work well from scratch.

4.2.4. Parameter analysis

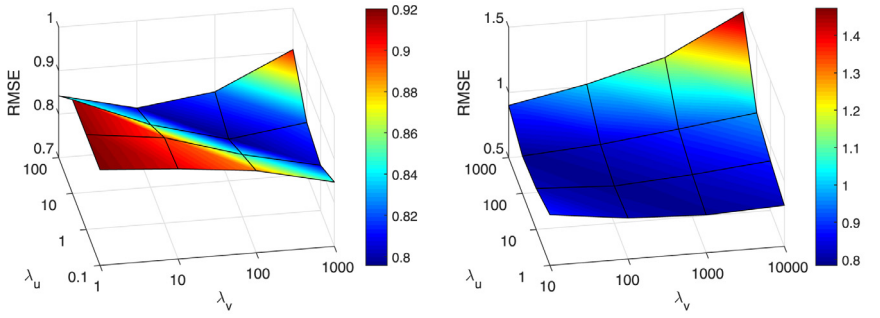
We investigate the impact of following parameters on the performance of our models: (1) the number of shared weights per window size, (2) pre-determined maximum length of documents and (3) two parameters λ_U and λ_V.

1) Impact of the number of shared weight per window size: As we explained in Section 3.1.2, one shared weight extracts one type of contextual features analogous to a topic in documents. Thus, in order to extract multiple types of contextual features for deeper understanding of documents, the usage a sufficient number of shared weights can be preferred. However, too many parameters by an excessive number of shared weights cause the over-fitting problem or the degradation of train speed. To demonstrate this observation, we evaluate R-ConvMF with various number of shared weights per window size on Amazon dataset. Fig. 6 shows a trade-off between the accuracy and train time until 100 shared weights per window size. However, when the number of shared weights per window size increases more than 100, we observe that the accuracy also gets worse due to the over-fitting problem. *As a result, it is important to use an appropriate number of shared weights while considering the trade-off.*

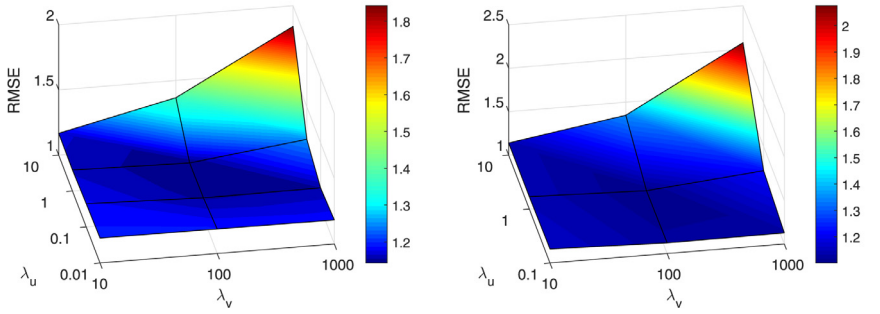
2) Impact of pre-determined maximum length of documents: In order to verify the impact of the amount of document information, we set the maximum length of documents mentioned in Section 4.1.3 from 10 to 400 in various ways, and evaluate R-ConvMFon Amazon dataset. Fig. 7 shows that RMSE consistently is decreased until the maximum length of documents reaches 200, and then converged. This is because the mean of lengths of documents for items on Amazon dataset is 91.50. Thus, we infer that additional document information can be consistently obtained to construct document latent vectors accurately until the maximum length of documents reaches 200, and further document information can not be obtained even



(a) MovieLens-1m



(b) MovieLens-10m



(c) Amazon Instant Video

Fig. 8. Parameter analysis of λ_U and λ_V on three dataset; ConvMF(left) and R-ConvMF(right).

if maximum length of documents is increased more than 200. Meanwhile, since the amount of document information to be processed increases, R-ConvMF requires more train time. *As a result, when we utilize document information using CNN, it is required to carefully analyze statistics of documents and consider trade-off between maximum length of documents and train time*.

3) Impact of two parameters λ_U and λ_V: Fig. 8 shows the impact of λ_U and λ_V on three real-word datasets. Regarding the changes of the best performing values of λ_U and λ_V from Fig. 8(a) to (c), we found that when the rating data becomes sparse, λ_U decreases while λ_V increases to produce the best results. Precisely, the values of (λ_U, λ_V) of ConvMF (or R-ConvMF) are (100, 10), (10 and 100) and (1 and 100) (or (10, 100), (10, 100), (1, 100)) on ML-1m, ML-10m and AIV, respectively. This pattern is natural because a relatively high value of λ_V allows the item latent variable to be updated by resorting to description documents when the ratings are insufficient. Note that a relatively high value of λ_U is hardly updated so that the item latent variable tends to be projected to the latent space of the user latent variable in the training process. Therefore, when λ_U decreases and λ_V increases, the user latent variable tends to be projected to the latent space of the item latent variable whose space is mainly built by description documents. *As a result, these best performing values demonstrate that our models well alleviate the data sparsity by balancing the usage of description documents.*

5. Conclusion and future work

In this paper, we address two limitations of existing work: (1) they ignore contextual information. (2) they do not consider Gaussian noise differently. To solve the limitations, we propose a robust document context-aware hybrid method that seamlessly integrates CNN into PMF in order to capture contextual information in description documents for the rating prediction while considering Gaussian noise differently through using the statistics of items. Extensive new experimental results demonstrate that our final recommendation model, R-ConvMF, significantly outperform the state-of-the-art competitors, which implies that the model effectively deal with the sparsity problem. Besides, our proposed method only uses user-to-item rating data and description documents of items, and thus the method can be easily applied into recommender systems of real-world e-commerce companies such as Amazon and Netflix. Note that the companies usually have user-to-item rating data and description documents of items. As future work, since it is widely known that unsupervised pre-training on deep neural network has much impact on performance, we try to develop convolutional autoencoder for description documents. Through the unsupervised approach, we can pre-train not only the weight variables of the embedding layer but also all the remaining weight variables of the CNN part of R-ConvMF. We expect that this unsupervised pre-training by the autoencoder significantly boosts the performance of recommendation when rating data is extremely sparse.

Acknowledgments

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (No. 2012M3C4A7033344) and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2016R1E1A1A01942642).

References

[1] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions, IEEE Trans. Knowl. Data Eng.(TKDE) 17 (6) (2005) 734–749.
[2] T. Bansal, D. Belanger, A. McCallum, Ask the GRU: Multi-task learning for deep text recommendations, in: Proceedings of the 10th ACM Conference on Recommender Systems, in: RecSys '16, ACM, New York, NY, USA, 2016, pp. 107–114, doi:10.1145/2959100.2959180.
[3] F. Chollet, Keras, 2015, (https://github.com/fchollet/keras).
[4] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, Natural language processing (almost) from scratch, J. Mach. Learn. Res. (JMLR) 12 (2011) 2493–2537.
[5] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11), 15, 2011, pp. 315–323.
[6] R. He, J. McAuley, VBPR: visual bayesian personalized ranking from implicit feedback, in: AAAI Conference on Artificial Intelligence, 2016.
[7] J.L. Herlocker, J.A. Konstan, A. Borchers, J. Riedl, An algorithmic framework for performing collaborative filtering, in: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, in: SIGIR '99, ACM, New York, NY, USA, 1999, pp. 230–237, doi:10.1145/312624.312682.
[8] Y. Hu, Y. Koren, C. Volinsky, Collaborative filtering for implicit feedback datasets, in: Proceedings of the 8th IEEE International Conference on Data Mining, in: ICDM '08, IEEE Computer Society, Washington, DC, USA, 2008, pp. 263–272, doi:10.1109/ICDM.2008.22.
[9] D. Kim, C. Park, J. Oh, S. Lee, H. Yu, Convolutional matrix factorization for document context-aware recommendation, in: Proceedings of the 10th ACM Conference on Recommender Systems, in: RecSys '16, ACM, New York, NY, USA, 2016, pp. 233–240, doi:10.1145/2959100.2959165.
[10] Y. Kim, Convolutional neural networks for sentence classification, in: Proceedings of the 2014 Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1746–1751.
[11] Y. Koren, Factorization meets the neighborhood: A multifaceted collaborative filtering model, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD '08, ACM, New York, NY, USA, 2008, pp. 426–434, doi:10.1145/1401890.1401944.
[12] S. Li, J. Kawale, Y. Fu, Deep collaborative filtering via marginalized denoising auto-encoder, in: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, in: CIKM '15, ACM, New York, NY, USA, 2015, pp. 811–820, doi:10.1145/2806416.2806527.
[13] G. Ling, M.R. Lyu, I. King, Ratings meet reviews, a combined approach to recommend, in: Proceedings of the 8th ACM Conference on Recommender Systems, in: RecSys '14, ACM, New York, NY, USA, 2014, pp. 105–112, doi:10.1145/2645710.2645728.
[14] A.L. Maas, A.Y. Hannun, A.Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in: International Conference on Machine Learning Workshop on Deep Learning for Audio, Speech, and Language Processing, 2013.
[15] C. Martinez-Cruz, C. Porcel, J. Bernabé-Moreno, E. Herrera-Viedma, A model to represent users trust in recommender systems using ontologies and fuzzy linguistic modeling, Inf. Sci. 311 (C) (2015) 102–118, doi:10.1016/j.ins.2015.03.013.
[16] J. McAuley, J. Leskovec, Hidden factors and hidden topics: understanding rating dimensions with review text, in: Proceedings of the 7th ACM Conference on Recommender Systems, in: RecSys '13, ACM, New York, NY, USA, 2013, pp. 165–172, doi:10.1145/2507157.2507163.
[17] C. Park, D. Kim, J. Oh, H. Yu, Improving top-k recommendation with truster and trustee relationship in user trust network, Inf. Sci. 374 (2016) 100–114.
[18] J. Pennington, R. Socher, C.D. Manning, Glove: global vectors for word representation, in: Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1532–1543.
[19] S. Purushotham, Y. Liu, C.-C.J. Kuo, Collaborative topic regression with social matrix factorization for recommendation systems, in: Proceedings of the 29th International Conference on Machine Learning (ICML), 2012, pp. 759–766.
[20] R. Salakhutdinov, A. Mnih, Probabilistic matrix factorization, Advances in Neural Information Processing Systems, 20, 2008.
[21] R. Salakhutdinov, A. Mnih, Bayesian probabilistic matrix factorization using markov chain monte carlo, in: Proceedings of the 25th International Conference on Machine Learning, in: ICML '08, ACM, New York, NY, USA, 2008, pp. 880–887, doi:10.1145/1390156.1390267.
[22] S. Sedhain, A.K. Menon, S. Sanner, L. Xie, Autorec: autoencoders meet collaborative filtering, in: Proceedings of the 24th International Conference on World Wide Web, in: WWW '15 Companion, ACM, New York, NY, USA, 2015, pp. 111–112, doi:10.1145/2740908.2742726.
[23] A. Tejeda-Lorente, C. Porcel, J. Bernabé-Moreno, E. Herrera-Viedma, Refore: a recommender system for researchers based on bibliometrics, Appl. Soft Comput. 30 (2015) 778–791.
[24] A. Tejeda-Lorente, C. Porcel, E. Peis, R. Sanz, E. Herrera-Viedma, A quality based recommender system to disseminate information in a university digital library, Inf. Sci. 261 (2014) 52–69, doi:10.1016/j.ins.2013.10.036.
[25] A. van den Oord, S. Dieleman, B. Schrauwen, Deep content-based music recommendation, in: Advances in Neural Information Processing Systems 26, Curran Associates, Inc., 2013, pp. 2643–2651.

[26] C. Wang, D.M. Blei, Collaborative topic modeling for recommending scientific articles, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD '11, ACM Press, 2011, pp. 448–456.
[27] H. Wang, N. Wang, D.-Y. Yeung, Collaborative deep learning for recommender systems, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD '15, ACM, New York, NY, USA, 2015, pp. 1235–1244, doi:10.1145/2783258.2783273.
[28] Y. Wu, C. DuBois, A.X. Zheng, M. Ester, Collaborative denoising auto-encoders for top-n recommender systems, in: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, in: WSDM '16, ACM, New York, NY, USA, 2016, pp. 153–162, doi:10.1145/2835776.2835837.
[29] X. Zheng, W. Ding, Z. Lin, C. Chen, Topic tensor factorization for recommender system, Inf. Sci. 372 (C) (2016) 276–293, doi:10.1016/j.ins.2016.08.042.