Convolutional Matrix Factorization for Document Context-Aware Recommendation

Donghyun Kim[1], Chanyoung Park[1], Jinoh Oh[1], Sungyoung Lee[2], Hwanjo Yu[*1]
[1]Pohang University of Science and Technology (POSTECH), Pohang, South Korea
[2]Kyung Hee University, Seoul, South Korea
{[1]kdh5377, [1]pcy1302, [1]kurin, [1]hwanjoyu}@postech.ac.kr, [2]sylee@oslab.khu.ac.kr

ABSTRACT

Sparseness of user-to-item rating data is one of the major factors that deteriorate the quality of recommender system. To handle the sparsity problem, several recommendation techniques have been proposed that additionally consider auxiliary information to improve rating prediction accuracy. In particular, when rating data is sparse, document modeling-based approaches have improved the accuracy by additionally utilizing textual data such as reviews, abstracts, or synopses. However, due to the inherent limitation of the bag-of-words model, they have difficulties in effectively utilizing contextual information of the documents, which leads to shallow understanding of the documents. This paper proposes a novel context-aware recommendation model, *convolutional matrix factorization* (ConvMF) that integrates convolutional neural network (CNN) into probabilistic matrix factorization (PMF). Consequently, ConvMF captures contextual information of documents and further enhances the rating prediction accuracy. Our extensive evaluations on three real-world datasets show that ConvMF significantly outperforms the state-of-the-art recommendation models even when the rating data is extremely sparse. We also demonstrate that ConvMF successfully captures subtle contextual difference of a word in a document. Our implementation and datasets are available at http://dm.postech.ac.kr/ConvMF.

Keywords

Collaborative Filtering; Document Modeling; Contexual Information

1. INTRODUCTION

The exploding growth of the number of users and items in e-commerce services increases the sparseness of user-to-item rating data. Eventually, this sparsity deteriorates the rating prediction accuracy of traditional collaborative filtering techniques [5, 8]. To enhance the accuracy, several recommendation techniques had been proposed that consider not only rating information but also auxiliary information such as demography of users, social networks, and item description documents [14, 15, 17, 19, 23, 24].

Recently, approaches based on document modeling methods such as Latent Dirichlet Allocation (LDA) and Stacked Denoising Auto-Encoder (SDAE) have been proposed to additionally utilize item description documents such as reviews, abstracts, or synopses [15, 17, 23, 24]. Specifically, Wang *et al.* proposed *collaborative topic regression* (CTR) that combines topic modeling (LDA) and collaborative filtering in a probabilistic approach [23]. Variants of CTR were proposed, which also integrates LDA into collaborative filtering to analyze item description documents with different integration approaches [15, 17]. Most recently, Wang *et al.* proposed *collaborative deep learning* (CDL) that integrates SDAE into probabilistic matrix factorization (PMF) [20], thereby generating more accurate latent model in terms of the rating prediction accuracy [24].

However, existing integrated models do not fully capture document information, as they assume the bag-of-words model that ignores *contextual information* of documents such as surrounding words and word orders. 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 consider the document as a bag of distinguished words, they cannot distinguish each occurrence of term "***trust***". Precisely, although each occurrence of "***trust***" seems to have almost the same meaning, there is a subtle syntactic difference between these words – a verb and a noun, respectively. Such subtle difference within a document is also a nontrivial factor for deeper understanding of the document, and further such understanding facilitates improvements in the rating prediction accuracy.

To address the aforementioned issue, we utilize convolutional neural network (CNN), which is the state-of-the-art machine learning methodology that shows high performance for various domains such as computer vision [13], natural language processing (NLP) [2, 10, 11], and information retrieval [4, 21]. CNN effectively captures local features of images or documents through modeling components such as local receptive fields, shared weights, and subsampling [13]. Thus, the use of CNN facilitates deeper understanding of documents, and generates better latent model than LDA and SDAE do, especially for items that resort to their description documents due to the lack of ratings. Moreover, CNN is able to take advantage of pre-trained word embedding models such as Glove [18] for deeper understanding of item description documents. Note that LDA and SDAE cannot exploit pre-trained word embedding models because they adopt bag-of-words model.

However, existing CNNs are not suitable for recommendation task, as their objectives are different from the objective of recommendation. Specifically, conventional CNNs mainly solve classifi-

*Corresponding author

RecSys'16, September 15-19, 2016, Boston, MA, USA

DOI: http://dx.doi.org/10.1145/2959100.2959165

cation task that is to predict labels of words, phrases, or documents. On the contrary, the objective of recommendation is regarded as a regression task aiming at accurately approximating ratings of users on items. Thus, the existing CNNs cannot be directly applied to our task of recommendation.

To handle the technical issue, we propose a document context-aware recommendation model, *convolutional matrix factorization* (ConvMF), which captures contextual information of item description documents by utilizing convolutional neural network (CNN) and further enhances the rating prediction accuracy. Precisely, ConvMF seamlessly integrates CNN into PMF, which is commonly used for recommendation tasks. Consequently, the integrated model follows the recommendation objective, and eventually effectively utilizes both collaborative information and contextual information. As a result, ConvMF accurately predicts unknown ratings even when the rating data is extremely sparse.

To demonstrate the effectiveness of ConvMF, we evaluate ConvMF on three different real-world datasets. Our extensive experiments over various sparsenesses of rating datasets demonstrate that ConvMF significantly outperforms the state-of-the-art models. The superiority of ConvMF verifies that ConvMF generates item latent models that effectively reflect contextual information of item description documents even when the rating data is extremely sparse. We also qualitatively demonstrate that ConvMF indeed captures subtle contextual differences of a word in a document. Furthermore, we investigate whether pre-trained word embedding model helps improve the rating prediction accuracy of ConvMF. Detailed experiment results are also available at http://dm.postech.ac.kr/ConvMF besides our implementation and datasets.

Our contributions are summarized as follows.

- We address limitations of the bag-of-words model based approaches and develop a novel document context-aware recommendation model (ConvMF).
- To exploit both ratings and item description documents, we seamlessly integrate CNN into PMF under probabilistic perspective.
- We extensively demonstrate the superiority of ConvMF over the state-of-the-art models on three real-world datasets with quantitative and qualitative results.

The remainder of the paper is organized as follows. Section 2 briefly reviews preliminaries on the most representative collaborative filtering technique and CNN. Section 3 introduces an overview of ConvMF, explains our CNN architecture of ConvMF, and describes how to optimize ConvMF. Section 4 experimentally evaluates ConvMF and discuss the evaluation results. Section 5 summarizes our contributions and gives future work.

2. PRELIMINARY

In this section, we briefly review matrix factorization (MF) (the most popular collaborative filtering technique) and convolutional neural network (CNN).

2.1 Matrix Factorization

Traditional collaborative filtering techniques are categorized into two categories [5]: memory-based methods (e.g. nearest neighborhood) [3, 7, 12] and model-based methods (e.g. latent factor model) [12, 20]. In general, model-based methods are known to generate more accurate recommendation results [12]. Thus in this section, we describe MF, which is the most popular model-based method.

The goal of MF is to find latent models of users and items on a shared latent space in which the strengths of user-item relationships (i.e., rating by a user on an item) are computed by inner products [12]. Suppose that we have N users, M items and a user-item rating matrix $R \in \mathbb{R}^{N \times M}$. In MF, the latent models of user i and item j are represented as k-dimensional models, $u_i \in \mathbb{R}^k$ and $v_j \in \mathbb{R}^k$. The rating r_{ij} of user i on item j is approximated by the inner-product of corresponding latent models 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 models 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.

2.2 Convolutional Neural Network

Convolutional neural network (CNN) is a variant of feed-forward neural networks with the following components: 1) convolution layer for generating *local features*, 2) pooling (or sub-sampling) layer for representing data as more concise representation by selecting only several representative local features (i.e., features having the highest score via the activation functions) from the previous layer, which is usually a convolution layer.

Even though CNN has been originally developed for computer vision [13], the key idea of CNN has been actively applied to information retrieval and NLP such as search query retrieval [4, 21], sentence modeling and classification [10, 11], and other traditional NLP tasks [2]. Although CNN for NLP tasks requires a significant amount of modification on the architecture of CNN, it eventually helps enhance the performance of various NLP tasks.

However, CNN has not yet been actively adopted to the field of recommender system. To the best of our knowledge, van den Oord *et al.* were first to apply CNN to music recommendation [22], where they analyzed songs in acoustic analysis point of view via CNN, and proposed a model that predicts the ratings based on the item latent model obtained by acoustic CNN. However, their CNN model, designed for acoustic signal processing, is not suitable for processing documents. Documents and acoustic signals have an inherent difference on the quality of surrounding features. A signal at a certain time is inherently similar to its surrounding signals, i.e., the signals that have slight time difference, while a word at a certain position in the document has a large semantical difference from the surrounding words. Such difference in the degree of similarity between surrounding features affects the quality of local features, and eventually requires different CNN architectures. Furthermore, the model does not fully reflect the collaborative information. In particular, the item latent models are mainly determined by the results of audio signal analysis via CNN rather than collaborative information. Thus, the performance of overall recommendation even does not achieve that of weighted matrix factorization (WMF) [9], which is one of the conventional MF-based collaborative filtering techniques dealing with implicit feedback dataset.

3. CONVOLUTIONAL MATRIX FACTORIZATION

In this section, we provide details of the proposed model, convolutional matrix factorization (ConvMF), through three steps: 1) We introduce the probabilistic model of ConvMF, and describe the

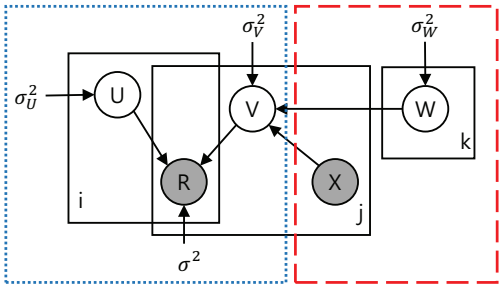


Figure 1: Graphical model of ConvMF model: PMF part in left (dotted-blue); CNN part in right (dashed-red)

key idea to bridge PMF and CNN in order to utilize both ratings and item description documents. 2) We explain the detailed architecture of our CNN, which generates document latent model by analyzing item description documents. 3) Finally, we describe how to optimize latent variables of ConvMF.

3.1 Probabilistic Model of ConvMF

Figure 1 shows the overview of the probabilistic model for ConvMF, which integrates CNN into PMF. 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 models ($U \in \mathbb{R}^{k \times N}$ and $V \in \mathbb{R}^{k \times M}$) whose product ($U^T V$) reconstructs the rating matrix R. In probabilistic point of view, the conditional distribution over observed ratings is given by

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

where $N(x|\mu, \sigma^2)$ is the probability density function of the Gaussian normal distribution with mean μ and variance σ^2, and I_{ij} is an indicator function as mentioned in Section 2.1.

As a generative model for user latent models, we place conventional priori, a zero-mean spherical Gaussian prior on user latent models with variance σ_U^2.

$$p(U|\sigma_U^2) = \prod_i^N N(u_i|0, \sigma_U^2 I)$$

However, unlike the probabilistic model for item latent models in conventional PMF, we assume that an item latent model is generated from three variables: 1) internal weights W in our CNN[1], 2) X_j representing the document of item j, and 3) epsilon variable as Gaussian noise, which enables us to further optimize the item latent model for the ratings. Thus, the final item latent model is obtained by the following equations.

$$v_j = cnn(W, X_j) + \epsilon_j$$
$$\epsilon_j \sim N(0, \sigma_V^2 I)$$

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

$$p(W|\sigma_W^2) = \prod_k N(w_k|0, \sigma_W^2)$$

Accordingly, the conditional distribution over item latent models is given by

$$p(V|W, X, \sigma_V^2) = \prod_j^M N(v_j|cnn(W, X_j), \sigma_V^2 I)$$

[1]Detail of W of CNN will be explained in Section 3.2

where X is the set of description documents of items. A document latent vector obtained from the CNN model is used as the mean of Gaussian distribution and Gaussian noise of the item is used as the variance of Gaussian distribution which plays an important role as a bridge between CNN and PMF that helps to fully analyze both description documents and ratings.

3.2 CNN Architecture of ConvMF

The objective of our CNN architecture is to generate document latent vectors from documents of items, which are used to compose the item latent models with epsilon variables. Figure 2 shows our CNN architecture that consists of four layers; 1) embedding layer, 2) convolution layer, 3) pooling layer, and 4) output layer.

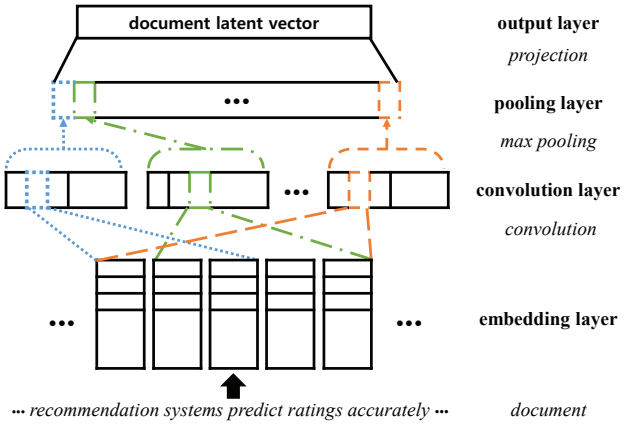


Figure 2: Our CNN architecture for ConvMF

Embedding Layer

The embedding layer transforms a raw document 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 vectors of words in the document. The word vectors are randomly initialized or initialized with pre-trained word embedding model such as Glove [18]. The word vectors are further trained through optimization process. 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.

Convolution Layer

The convolution layer extracts contextual features. As we've discussed in Section 2.2, documents are inherently different from signal processing or computer vision in the nature of contextual information. Thus, we use the convolution architecture in [2, 11] 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:

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

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 gradient, which causes slow optimization convergence and may lead to a poor local

minimum [16, 6]. Then, a contextual feature vector $c^j \in \mathbb{R}^{l-ws+1}$ of a document with W_c^j is constructed by Eqn.(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 $j = 1, 2, ..., n_c$).

Pooling Layer

The pooling layer extracts representative features from the convolution layer, and also deals with variable lengths of documents via pooling operation that constructs a fixed-length feature vector. After the convolution layer, a document is represented as n_c contextual feature vectors, where each contextual feature vector has 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 utilize 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.

Output Layer

Generally, at output layer, high-level features obtained from the previous layer should be converted for a specific task. Thus, we project d_f on a k-dimensional space of user and item latent models 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 architecture becomes a function that takes a raw document as input, and returns latent vectors of each documents as output:

$$s_j = cnn(W, X_j) \tag{4}$$

where W denotes all the weight and bias variables to prevent clutter and X_j denotes a raw document of item j, and s_j denotes a document latent vector of item j.

3.3 Optimization Methodology

To optimize the variables such as user latent models, item latent models, weight and bias variables of CNN, we use maximum a posteriori (MAP) estimation as follows.

$$\begin{aligned} &\max_{U,V,W} p(U, V, W | R, X, \sigma^2, \sigma_U^2, \sigma_V^2, \sigma_W^2) \\ &= \max_{U,V,W} [p(R|U, V, \sigma^2) p(U|\sigma_U^2) p(V|W, X, \sigma_V^2) p(W|\sigma_W^2)] \end{aligned} \tag{5}$$

By taking negative logarithm on Eqn.(5), it is reformulated as follows.

$$\begin{aligned} \mathcal{L}(U, V, W) = &\sum_i^N \sum_j^M \frac{I_{ij}}{2} (r_{ij} - u_i^T v_j)_2 + \frac{\lambda_U}{2} \sum_i^N \|u_i\|_2 \\ &+ \frac{\lambda_V}{2} \sum_j^M \|v_j - cnn(W, X_j)\|_2 + \frac{\lambda_W}{2} \sum_k^{|w_k|} \|w_k\|_2, \end{aligned} \tag{6}$$

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.

We adopt coordinate descent, which iteratively optimizes a latent variable while fixing the remaining variables. Specifically, Eqn.(6) becomes a quadratic function with respect to U (or V) while temporarily assuming W and V (or U) to be constant. Then, the optimal solution of U (or V) can be analytically computed in a closed form by simply differentiating the optimization function $\mathcal{L}$ with respect to u_i (or v_j) as follows.

$$u_i \leftarrow (V I_i V^T + \lambda_U I_K)^{-1} V R_i \tag{7}$$

$$v_j \leftarrow (U I_j U^T + \lambda_V I_K)^{-1} (U R_j + \lambda_V cnn(W, X_j)) \tag{8}$$

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. Eqn.(8) shows the effect of document latent vector of CNN in generating the item latent model v_j, where λ_V is a balancing parameter as in [23].

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

$$\begin{aligned} \mathcal{E}(W) = &\frac{\lambda_V}{2} \sum_j^M \|(v_j - cnn(W, X_j))\|^2 \\ &+ \frac{\lambda_W}{2} \sum_k^{|w_k|} \|w_k\|^2 + \text{constant} \end{aligned} \tag{9}$$

To optimize W, we use back propagation algorithm. (Recall that W is the weights and biases of each layer.)

The overall optimization process (U, V and W are alternatively updated) is repeated until convergence. 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 $v_j = cnn(W, X_j) + \epsilon_j$.

Time Complexity Analysis

For each epoch, all user and item latent models are updated in $O(k^2 n_R + k^3 N + k^3 M)$, where n_R is the number of observed ratings. Note that document latent vectors are computed while updating W. Time complexity for updating W is dominated by the computation of convolution layer, and thus all weight and bias variables of CNN are 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.

4. EXPERIMENT

In this section, we evaluate the empirical performance of ConvMF on real-world datasets. Our extensive experiment results demonstrate that 1) ConvMF significantly outperforms other competitors

when dataset is extremely sparse, 2) the improvements of ConvMF over the competitors increase further even when dataset becomes dense, 3) pre-trained word embedding model helps improve the performance of ConvMF when dataset is extremely sparse, 4) the best performing parameters verify that ConvMF well alleviates data sparsity, and 5) ConvMF indeed captures subtle contextual differences.

4.1 Experimental Setting

Datasets

To demonstrate the effectiveness of our models in terms of rating prediction, we used three real-world datasets obtained from MovieLens[2] and Amazon[3]. These datasets consist of users' explicit ratings on items on a scale of 1 to 5. Amazon dataset contains reviews on items as item description documents. Since MovieLens does not include item description documents, we obtained documents (i.e., plot summary) of corresponding items from IMDB [4].

Similar to [23] and [24], we preprocessed description documents for all datasets as follows: 1) set maximum length of raw documents to 300, 2) removed stop words, 3) calculated tf-idf score for each word, 4) removed corpus-specific stop words that have the document frequency higher than 0.5, 5) selected top 8000 distinct words as a vocabulary, 6) removed all non-vocabulary words from raw 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 specifically for the case of Amazon dataset, 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 to the others.

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 1: Data statistic on three real-world datasets

Competitors and Parameter Setting

We compared two versions of ConvMF with the following baselines.

- PMF [20]: Probabilistic Matrix Factorization is a standard rating prediction model that only uses ratings for collaborative filtering.
- CTR [23]: Collaborative Topic Regression is a state-of-the-art recommendation model, which combines collaborative filtering (PMF) and topic modeling (LDA) to use both ratings and documents.
- CDL [24]: 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 proposed model.
- ConvMF+: Convolutional Matrix Factorization with pre-trained word embedding model is another version of our proposed model, and we use Glove [18] for pre-trained word embedding model.

We set the size of latent dimension of U and V to 50 (as reported in [24]) and initialized U and V randomly from 0 to 1. Table 2 shows the best performing values of common parameters (λ_U, λ_V) of each model found by grid search. Since we use 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 [24].[5]

	ML-1m		ML-10m		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

Table 2: Parameter Setting of λ_U and λ_V

Implementation Detail

We implemented ConvMF using Python and Keras [1] library with NVidia Geforce TitanX 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. 2-1) ConvMF: we initialized word latent vectors randomly with dimension size of 200. These word latent vectors will be trained through the optimization process. 2-2) ConvMF+: we initialized word latent vectors by pre-trained word embedding models with dimension size of 200. These word latent vectors will be trained through the optimization process. 3) In 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. 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.

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 an objective function of conventional rating prediction model.

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

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

4.2 Experimental Results

1) Quantitative Results on MovieLens and Amazon Datasets

Table 3 shows the overall rating prediction error of ConvMF, ConvMF+, and three competitors. Note that "Improve" indicates the

[2]http://grouplens.org/datasets/movielens/
[3]http://jmcauley.ucsd.edu/data/amazon/
[4]Plot summaries are available at http://www.imdb.com/
[5]We tried different settings but the performance was almost the same.

Model	Dataset		
	ML-1m	ML-10m	AIV
PMF	0.8971	0.8311	1.4118
CTR	0.8969	0.8275	1.5496
CDL	0.8879	0.8186	1.3594
ConvMF	**0.8531**	**0.7958**	**1.1337**
ConvMF+	**0.8549**	**0.7930**	**1.1279**
Improve	3.92%	2.79%	16.60%

Table 3: Overall test RMSE

Model	Ratio of training set to the entire dataset (density)						
	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**
Improve	2.98%	3.20%	3.36%	3.41%	3.77%	3.27%	3.92%

Table 4: Test RMSE over various sparseness of training data on ML-1m dataset

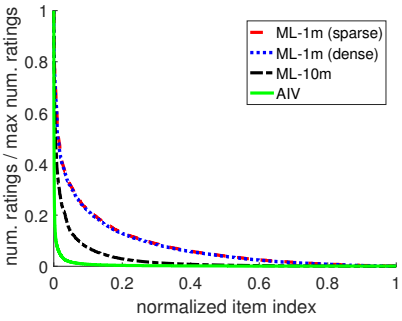


Figure 3: Skewness of the number of ratings for items on each dataset

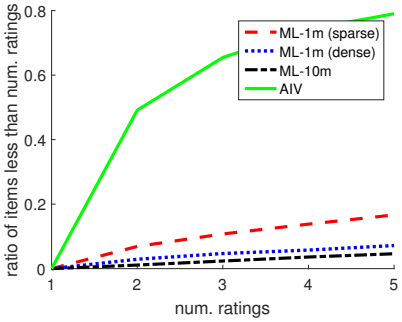


Figure 4: Ratio of items that have less than num. ratings (N) to each entire dataset

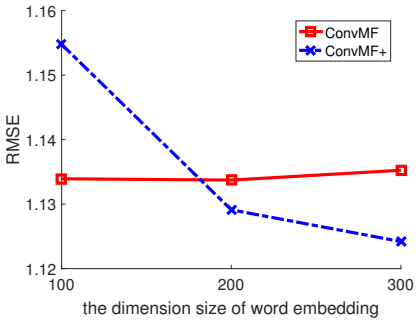


Figure 5: The effects of the dimension size of word embedding on Amazon dataset

relative improvements of "ConvMF" over the best competitor. Compared to three models, ConvMF and ConvMF+ achieve significant improvements on all the datasets.

For MovieLens datasets, which are relatively dense datasets, the improvements of ConvMF over the best competitor, CDL, are 3.92% on ML-1m dataset and 2.79% on ML-10m dataset. We also observe that the performance differences between PMF and the two competitors are marginal on ML-1m dataset. It implies that given enough ratings to generate user and item latent models, document analysis that fails to capture contextual information does not help generate more accurate latent models. However, *significant performance gap between ConvMF and PMF indicates that deeper understanding of documents helps adjust latent models more accurately even when enough ratings are given.*

For Amazon dataset, which is an extremely sparse and skewed dataset as shown in Figure 3 and 4, the improvement of ConvMF over the best competitor, CDL, is 16.60%. This improvement is more substantial compared to the improvements on relatively dense and balanced MovieLens datasets, which indicates that *ConvMF constructs accurate item latent models by effectively analyzing documents even with sparse and skewed data.* Note that the RMSE of CTR is higher than that of PMF although CTR additionally uses item documents. It implies that on such sparse and skewed dataset, it is likely that item latent models built by the LDA part of CTR and user latent models built by the PMF part of CTR do not reside in the same latent space.

2) Quantitative Results Over Various Sparseness on ML-1m

We generate seven additional datasets of different sparsenesses by randomly sampling from ML-1m dataset. As shown in Table 4, ConvMF significantly outperforms three competitors over all ranges of sparseness. Specifically, we observe that improvement of ConvMF over the best competitor (CDL) increases consistently from 2.98% to 3.92% when data density increases from 0.93% to 3.71%. It implies that ConvMF produces more accurate latent models by exploiting ratings when the number of ratings of each item evenly increases – the rating data becomes dense whereas the skewness of

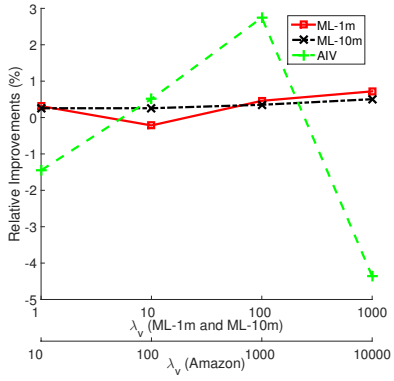


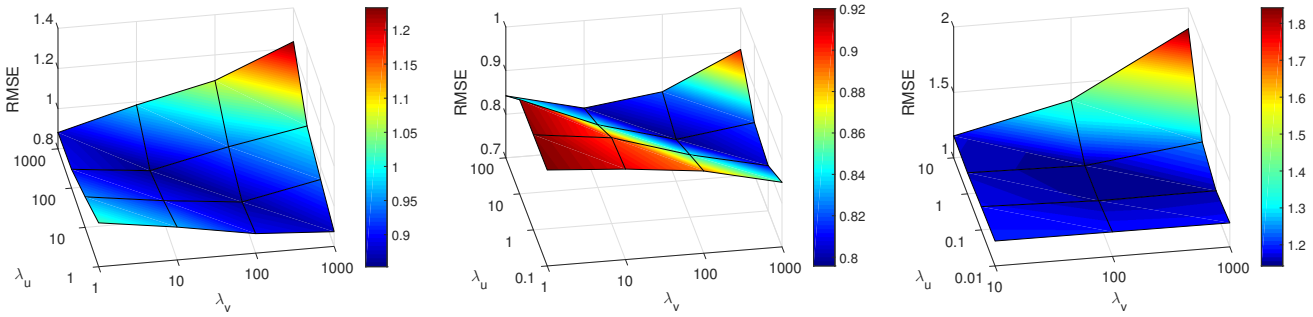
Figure 6: Relative improvements of ConvMF+ over ConvMF

the rating data is almost consistent, which is supported by Figure 3 and 4. To be precise, while ML-1m (sparse) and ML-1m (dense) dataset whose training set consists 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. *The performance of ConvMF consistently increases as the dataset gets denser, which indicates that CNN of ConvMF is well integrated into PMF for recommendation task.*

3) Impact of Pre-trained Word Embedding Model

Unlike CTR and CDL, ConvMF is able to take advantage of using pre-trained word embedding models such as Glove [18]. Thus, we investigate the impact of pre-trained word embedding model on our recommendation task by initializing the embedding layer of CNN of ConvMF using the pre-trained word embedding model of Glove [18].

Table 3 shows marginal changes of ConvMF+ over ConvMF on three datasets: -0.22%, 0.35% and 0.51% on ML-1m, ML-10m and AIV dataset, respectively. Note that the sparsity of datasets is in increasing order of ML-1m, ML-10m and AIV. In spite of



(a) MovieLens-1m (b) MovieLens-10m (c) Amazon Instant Video

Figure 7: Parameter analysis of λ_U and λ_V on three dataset

the marginal changes, we observe the consistent tendency that as the rating data gets sparser, the pre-trained word embedding model helps improve the performance of ConvMF. It is because the semantic and syntactic information of pre-trained word embedding model complement shortage of ratings. We also observe that given sufficient number of ratings to train latent models, pre-trained word embedding model rather deteriorates the performance of ConvMF. In other words, since ratings directly reflect the relationships between users and items, *it is better to fully leverage ratings rather than to additionally utilize pre-trained word embedding model obtained from external documents when rating data is relatively dense.*

Moreover, we investigate relative improvements of ConvMF+ over ConvMF on various λ_V, a parameter that balances between the importance of ratings and description documents in ConvMF. In Figure 6, for MovieLens datasets, since the dataset has relatively large number of ratings, pre-trained word embedding model does not have much impact on the performance of ConvMF on various λ_V. However, for extremely sparse Amazon dataset, pre-trained word embedding model indeed affects the performance of ConvMF. Specifically, when λ_V is 100 and 1000, the improvement of ConvMF+ over ConvMF reaches almost 0.51% and 2.74% with the same parameter setting whereas the performance of ConvMF+ suddenly drops with other λ_V values. In other words, relatively low or high λ_V with pre-trained word embedding model fails to achieve high performance. This phenomenon implies that latent models are under-fitted or over-fitted by the word embedding model when rating data is extremely sparse. Nevertheless, as shown in Figure 6, *given a proper value of λ_V, adopting pre-trained word embedding model increases the performance of ConvMF+ when the number of ratings is insufficient.*

4) Parameter Analysis

We investigate the impact of three parameters on the performance of ConvMF: p (dimension size of word latent vectors), λ_U and λ_V.

Figure 5 shows RMSE changes of ConvMF and ConvMF+ according to various p on Amazon dataset. Interestingly, in case of ConvMF, the increase of p from 100 to 300 does not boost the performance of ConvMF, but rather shows degradation. However, in case of ConvMF+, the increase of p reduces RMSE significantly. This demonstrates that when dataset is extremely sparse, *the performance of ConvMF+ is improved by adopting pre-trained word embedding model in which the information contained gets richer as p gets larger.*

Figure 7 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 Figure 7(a) to (c), we observe 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 are (100, 10), (10 and 100) and (1 and 100) on ML-1m, ML-10m and AIV, respectively. Note that when λ_U is high, user latent model is hardly updated. In other words, a high value of λ_U implies that item latent model tend to be projected to the latent space of user latent model (same applies to λ_V). Thus, when λ_U decreases and λ_V increases, user latent models are projected to the latent space of item latent model whose space is mainly built by description documents rather than by ratings. *These best performing values demonstrate that ConvMF well alleviates data sparsity by balancing the importance of ratings and description documents.*

Phrase captured by W_c^{11}	max(c^{11})	Phrase captured by W_c^{86}	max(c^{86})
people **trust** the man	***0.0704***	betray his **trust** finally	***0.1009***
Test phrases for W_c^{11}	**c_{test}^{11}**	**Test phrases for W_c^{86}**	**c_{test}^{86}**
people **believe** the man	***0.0391***	betray his **believe** finally	*0.0682*
people **faith** the man	*0.0374*	betray his **faith** finally	***0.0693***
people **tomas** the man	*0.0054*	betray his **tomas** finally	*0.0480*

Table 5: Case study on two shared weights of ConvMF

5) Qualitative Evaluation

The performance of ConvMF is affected by the contextual features extracted by shared weights W_c, and each contextual feature is associated with a phrase. In this section, we verify whether ConvMF is able to distinguish subtle contextual differences by comparing each contextual meaning of phrases captured by the shared weights.

Specifically, as a case study, we select W_c^{11} and W_c^{86} from the model trained on ML-10m dataset, and compare the contextual meaning of phrases captured by the shared weights (Table 5). The meaning of "*trust*" in the two phrases captured by the two shared weights seem to be similar to each other. However, there is a subtle difference on contextual meaning of the term "*trust*" in the two phrases. Indeed, the "*trust*" in the phrase captured by W_c^{11} is used as a verb whereas the "*trust*" in the phrase captured by W_c^{86} is used as a noun. To verify this, we slightly change the term "*trust*" in the phrases, and investigate the change of the feature values as shown in Table 5. When we replace "*trust*" with "*believe*" and "*faith*" in the phrases captured by W_c^{11}, the feature value of the former phrase is higher than that of the latter phrase. However, for the case of W_c^{86}, the feature value of the latter phrase is higher than that of the former phrase. This matches with our expectation that "*believe*" is syntactically more similar to the contextual meaning of "*trust*" as a verb captured by W_c^{11} whereas "*faith*" is syntactically more similar to the contextual meaning of "*trust*" as a noun captured by W_c^{86}. When we replace "*trust*" with "*tomas*", which has a

completely different meaning from "*trust*", "*tomas*" returns lower values for both W_c^{11} and W_c^{86} than "*believe*" and "*faith*".

This comparisons imply that ConvMF distinguishes a subtle contextual difference of the term "*trust*". As a result, we conclude that *ConvMF captures contextual meaning of words in documents and can even distinguish subtle contextual difference of the same word via different shared weights.*

5. CONCLUSION AND FUTURE WORK

In this paper, we address that considering contextual information such as surrounding words and word orders in description documents provides deeper understanding of description documents, and we develop a novel document context-aware recommendation model, ConvMF, that seamlessly integrates CNN into PMF in order to capture contextual information in description documents for the rating prediction. Extensive results demonstrate that ConvMF significantly outperforms the state-of-the-art competitors, which implies that ConvMF well deals with the sparsity problem with contextual information. Moreover, since ConvMF is based on PMF, which is the standard MF-based recommendation model, ConvMF is able to be extended to combining other MF-based recommendation models such as SVD++ [12] that only consider ratings.

As a next research direction, 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. By unsupervised way, it enables us to pre-train not only the weight variables of the embedding layer but also all the remaining weight variables of the CNN part of ConvMF. We expect that this unsupervised pre-training by the autoencoder significantly boosts the performance of recommendation when rating data is extremely sparse.

6. 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 ICT R&D program of MSIP/IITP [B0101-15-0307, Basic Software Research in Human-level Lifelong Machine Learning (Machine Learning Center)] and the Industrial Core Technology Development Program (10049079, Development of Mining core technology exploiting personal big data) funded by the Ministry of Trade, Industry and Energy (MOTIE, Korea)

7. REFERENCES

[1] F. Chollet. Keras. https://github.com/fchollet/keras, 2015.

[2] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research (JMLR)*, 12:2493–2537, Nov. 2011.

[3] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems*, 22(1):143–177, Jan. 2004.

[4] J. Gao, P. Pantel, M. Gamon, X. He, and L. Deng. Modeling interestingness with deep neural networks. In *Proceedings of the 2014 Empirical Methods in Natural Language Processing (EMNLP)*, pages 2–13, 2014.

[5] A. T. Gediminas Adomavicius. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 17(6):734–749, June 2005.

[6] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)*, volume 15, pages 315–323, 2011.

[7] J. L. Herlocker, J. A. Konstan, A. Borchers, and 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*, SIGIR '99, pages 230–237, New York, NY, USA, 1999. ACM.

[8] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, Jan. 2004.

[9] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining*, ICDM '08, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society.

[10] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, June 2014.

[11] Y. Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, 2014.

[12] 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*, KDD '08, pages 426–434, New York, NY, USA, 2008. ACM.

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Intelligent Signal Processing*, pages 306–351. IEEE Press, 2001.

[14] S. Li, J. Kawale, and Y. Fu. Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 811–820, New York, NY, USA, 2015. ACM.

[15] G. Ling, M. R. Lyu, and I. King. Ratings meet reviews, a combined approach to recommend. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 105–112, New York, NY, USA, 2014. ACM.

[16] A. L. Maas, A. Y. Hannun, and 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.

[17] J. McAuley and J. Leskovec. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 165–172, New York, NY, USA, 2013. ACM.

[18] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[19] S. Purushotham, Y. Liu, and 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)*, pages 759–766, 2012.

[20] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, volume 20, 2008.

[21] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 101–110, New York, NY, USA, 2014. ACM.

[22] A. van den Oord, S. Dieleman, and B. Schrauwen. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems 26*, pages 2643–2651. Curran Associates, Inc., 2013.

[23] C. Wang and 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*, KDD '11, pages 448–456. ACM Press, August 2011.

[24] H. Wang, N. Wang, and D.-Y. Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1235–1244, New York, NY, USA, 2015. ACM.