Doctoral Dissertation

Hybrid Collaborative Filtering Approaches based on Deep Learning for Recommender System

Donghyun Kim (김 동 현)

Department of Computer Science and Engineering

Pohang University of Science and Technology

2017


POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
1986
포항공과대학교

추천 시스템을 위한 딥러닝 기반 하이브리드 협업 필터링 기법

Hybrid Collaborative Filtering Approaches based on Deep Learning for Recommender System

Hybrid Collaborative Filtering Approaches based on Deep Learning for Recommender System

by

Donghyun Kim
Department of Computer Science and Engineering
Pohang University of Science and Technology

A dissertation submitted to the faculty of the Pohang University of Science and Technology in partial fulfillment of the requirements for the degree of Doctor of philosophy in the Computer Science and Engineering

Pohang, Korea
06. 28. 2017
Approved by
Hwanjo Yu (Signature)
Academic advisor

Hybrid Collaborative Filtering Approaches based on Deep Learning for Recommender System

Donghyun Kim

The undersigned have examined this dissertation and hereby certify that it is worthy of acceptance for a doctoral degree from POSTECH

06. 28. 2017

Committee Chair	Hwanjo Yu	(Seal)
Member	Chi-Hyuck Jun	(Seal)
Member	Jong-Hyeok Lee	(Seal)
Member	Young Myoung Ko	(Seal)
Member	Minsu Cho	(Seal)

DCSE 김 동 현. Donghyun Kim
20120757
Hybrid Collaborative Filtering Approaches based on Deep Learning for Recommender System,
추천 시스템을 위한 딥러닝 기반 하이브리드 협업 필터링 기법
Department of Computer Science and Engineering , 2017, 59p, Advisor : Hwanjo Yu. Text in English.

ABSTRACT

Recommender system has received significant attention from academia and various industries, especially e-commerce service companies in the last decade. Recently, the exploding growth of the number of users and items in e-commerce service companies is causing an extremely high sparseness of relationships between users and items. As a result, the sparsity issue becomes one of the major obstacles to achieving a high performance in collaborative filtering (CF) based recommender system, which mostly relies on the relationships between users and items. To overcome the sparsity issue, several hybrid recommender systems have been proposed, which leverage auxiliary information related to users and items together with the ratings. However, they still fail to effectively exploit auxiliary information, and thus, new approaches are required for recommender systems.

In order to effectively exploit auxiliary information, we propose a two-step approach that adopts deep learning methods into CF-based recommender systems. First, to leverage both ratings and documents of items, we develop a novel

document context-aware hybrid recommendation model, which integrates Convolutional Neural Network (CNN) into Probabilistic Matrix Factorization (PMF) in order to effectively capture contextual information such as word order or surrounding words of a word in documents of items. Second, to boost the performance of the document context-aware hybrid recommender system, we introduce a new latent factor modeling method for items. The proposed method exploits the statistics of items to make our document context-aware hybrid recommendation model more robust to not only sparse but also skewed datasets. Our experiments will also show that our proposed recommendation models outperform the state-of-the-art recommendation models. The implementations of our models and related datasets will be available at `http://dm.postech.ac.kr/~cartopy`.


POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
1986
포항공과대학교

Contents

List of Tables III
List of Figures IV

I. Introduction and Motivation 1

II. Preliminary and Related Work 5
2.1 Matrix Factorization 5
2.2 Convolutional Neural Network 6
2.3 Deep Learning for Collaborative Filtering 6

III. Convolutional Neural Network meets Collaborative Filtering for Document Context-Aware Recommender Systems 9
3.1 Introduction 9
3.2 Method 10
3.2.1 Probabilistic Model of ConvMF 10
3.2.2 CNN Architecture of ConvMF 13
3.2.3 Optimization Methodology 16
3.3 Experiment 18
3.3.1 Experimental Setting 19
3.3.2 Experimental Results 23

IV. A Different Gaussian Noise Latent Factor Modeling Method for Document Context-Aware Recommender Systems 31
4.1 Introduction 31
4.2 Method 33
4.2.1 Modeling items for R-ConvMF 33
4.2.2 Optimization Methodology for R-ConvMF 35
4.2.3 Time Complexity Analysis 36
4.3 Experiment 38
4.3.1 Experimental Setting 38

4.3.2 Experimental Results 41

V. Conclusion 52

Summary (in Korean) 53

References 54

List of Tables

3.1 Data statistic on three real-world datasets 19

3.4 Case study on two shared weights of ConvMF 29

4.1 Relative improvements over PMF of four models with respect to various sparseness of the real world dataset – Movielens-1M 32

4.2 Data statistic on three real-world datasets 39

4.3 Comparison between 6 models: R-ConvMF, our two previous hybrid models and the three competitors 39

4.7 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 46

List of Figures

3.1 The graphical model of ConvMF: PMF part in left (dotted-blue); CNN part in right (dashed-red) 11

3.2 Our CNN architecture of R-ConvMF 13

3.3 Skewness of the number of ratings for items on each dataset 24

3.4 Ratio of items that have less than num. ratings to each entire dataset 24

3.5 Relative improvements of ConvMF+ over ConvMF 26

3.6 The effects of the dimension size of word embedding on Amazon dataset 27

3.7 Parameter analysis of λ_U and λ_V on three dataset 28

4.1 Skewness of the number of ratings for items on each dataset – Amazon dataset is the most skew. 42

4.2 Ratio of items that have less than num. ratings to each entire dataset – 50% of items in Amazon dataset have only one rating. . 42

4.3 Relative improvements over PMF of five models 45

4.4 RMSE and train time for $W^1$of R-ConvMF w.r.t the number of shared weights per window size on Amazon dataset 48

4.5 RMSE and train time for W of R-ConvMF w.r.t max length of documents on Amazon dataset 49

4.6 Parameter analysis of λ_U and λ_V on three dataset of R-ConvMF . 50

I. Introduction and Motivation

Triggered by Netflix Prize[1] in 2006, during the last decade, recommender system has gained considerable interest from academia and various industries, especially e-commerce service companies such as Amazon and Netflix that aim to provide appropriate items to users in a vast amount of items. Accordingly, several Collaborative Filtering (CF) based recommender systems [1, 2] which exploit the relationships between users and items have been actively developed and showed reasonable results. However, recently, the number of users and items in e-commerce service companies such as Amazon[2] and Netflix[3] has exploded, which leads to an extremely high sparseness of relationships between users and items from which user-to-item rating matrix is derived. Since users or items with few ratings are not well modeled by CF-based recommender system, the sparsity of a rating matrix degrades overall performance in CF-based recommender systems [3, 4].

To overcome the sparsity issue, a vast majority of recent research on recommender system has focused on leveraging auxiliary information related to users and items together with the ratings. As a result, several hybrid methods were actively proposed for recommender systems that leverage not only rating information but also auxiliary information such as demography of users [5], social networks [6], trust information of users [7, 8], description documents of items [9, 10, 6, 11, 12], images of items [13], reviews of users about items [14], quality of users and items [15, 16].

Particularly, since description documents of items such as reviews, abstracts

[1] http://www.netflixprize.com/

[2] `https://www.amazon.com`

[3] `https://www.netflix.com`

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 [9, 10, 11, 12]. Specifically, Wang and Blei developed *collaborative topic regression* (CTR) [11] model by probabilistically combining topic modeling method (LDA) and model-based CF recommender system. Variants of CTR were also proposed with different integration approaches [9, 10]. Most recently, Wang *et al.* developed *collaborative deep learning* (CDL) [12] model by integrating deep learning model (SDAE) into model-based CF recommender system, thereby estimating latent factors of items more accurately for better prediction of ratings.

Nevertheless, previous hybrid recommendation models failed to effectively exploit auxiliary information due to two following limitations on further improvements, which leaves us room for improvement in the performance of recommender systems. First, they do not fully exploit description documents, because 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 distinguish each occurrence of the 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 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. However, existing recommendation models do not reflect this tendency in modeling latent factors of items, and they simply model latent factors of items with the same noise although Gaussian noise is related to generate further accurate latent factors from probabilistic point of view. 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 this thesis, we propose a two-step approach that effectively incorporates description documents of items in model-based CF recommender systems for hybrid methods using deep learning methods. As the first step, in Chapter III, we integrate Convolutional Neural Network (CNN) into Probabilistic Matrix Factorization (PMF) [2] in order to effectively deal with both ratings and documents of items while exploiting contextual information. The fact that CNN effectively captures local features (or contextual information) of images (or documents) [17, 18] enables us to develop novel document context-aware hybrid recommender systems. We show that our proposed recommendation models significantly outperform the state-of-the-art recommendation models in terms of the rating prediction accuracy. In Chapter IV, as the second step, we exploit the statistics of items (i.e., the number of ratings given to items) to consider Gaussian noise differently with respect to modeling latent factors of items. This is the first approach

to hybrid recommendation models to best our knowledge. To demonstrate the effectiveness of this approach, we apply this approach to our proposed hybrid recommendation model, and we find that this approach significantly boosts the performance of hybrid recommendation models compared with those of models without this approach. Our implementation and datasets used in Chapter III and Chapter IV are available at `http://dm.postech.ac.kr/~cartopy`.

The remainder of this thesis is organized as follows. Chapter II briefly reviews preliminaries on the most representative collaborative filtering (CF) method and CNN, and summarizes recent methods using deep learning methods for CF. Chapter III explains our proposed method with an overview of ConvMF, and describes our CNN architecture and how to optimize R-ConvMF. Chapter IV introduces a different Gaussian noise based latent factor modeling method for boosting the performance of the hybrid recommendation models. Chapter V summarizes our contributions and gives future work.

II. Preliminary and Related Work

In this chapter, we briefly review matrix factorization (MF) and convolutional neural network (CNN). Also, we introduce recent CF techniques using deep learning methods.

2.1 Matrix Factorization

Traditional collaborative filtering techniques are categorized into two categories [3]: memory-based methods (e.g. nearest neighborhood) [19, 20, 1] and model-based methods (e.g. latent factor model) [1, 2]. In general, model-based methods are known to generate more accurate recommendation results [1]. Thus in this section, we describe MF, which is the most popular model-based method.

The goal of MF is to find latent vectors 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 [1]. 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, the latent vectors of user i and item j are represented as k-dimensional vectors, $u_i \in \mathbb{R}^k$ and $v_j \in \mathbb{R}^k$. The 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 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., typically features with the highest activation value) from the previous layer, which is usually a convolution layer.

Even though CNN has been originally developed for computer vision [17], the key idea of CNN has been actively applied to information retrieval and NLP such as search query retrieval [21, 22], sentence modeling and classification [23, 18], and other traditional NLP tasks [24], which have shown remarkable performance in dealing with content information. 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.

2.3 Deep Learning for Collaborative Filtering

Due to the remarkable performance of the deep learning methods in computer vision and NLP fields, several researchers recently adopted deep learning methods such as auto-encoder [5, 25, 12, 26], recurrent neural network (RNN) [27] or CNN [13, 28] to recommender systems.

Wu *et al.* [26] and Sedhain *et al.* [25] used the (de-noising) auto-encoder to build a user or an item-based collaborative filtering model that only considers rating information. However, they also suffer from the sparsity problem as traditional CF. To handle this problem, Wang *et al.* proposed CDL [12], which uses the auto-encoder to encode bag-of-words representations of description documents of items into compressed representations. These compressed representa-

tions are used to help generate more accurate item latent vectors, especially for items with few ratings. As a variant of CDL, DCF [5] was proposed to deal with both user and item related 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, they cannot effectively encode *contextual information* of documents due to the inherent limitation of the bag-of-word model as we've explained in Section I. Different from this work, our model reflects document contextual information into item latent vectors. Furthermore, in order to generate more accurate item latent vector even when datasets are extremely sparse and skewed, our model takes into account the statistic of datasets. We will discuss it later in Section 3.2.1 and Section 4.3.2.

Recently, Bansal *et al.* [27] 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 [13] used image features of items from a pre-trained CNN trained by massive external image datasets in order to improve top-N recommendation. However, different from our work, their model is not an integrated model of CNN and CF, and their pre-trained CNN designed for image classification is not suitable to our task. van den Oord *et al.* [28] 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 item latent vectors 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 collaborative information. In particular, the item latent vectors are mainly determined by the results of audio signal analysis via CNN rather than collaborative information. Thus, the overall recommendation performance falls short of that of weighted matrix factorization (WMF) [29], which is one of the conventional MF-based collaborative filtering techniques that deals with implicit feedback dataset.

III. Convolutional Neural Network meets Collaborative Filtering for Document Context-Aware Recommender Systems

3.1 Introduction

To address the former limitation of previous work, we exploit convolutional neural network (CNN), which is the state-of-the-art deep learning method that shows high performance in various domains such as computer vision [17], natural language processing (NLP) [24, 23, 18], and information retrieval [21, 22]. CNNs effectively capture local features (or contextual information) of images (or documents) through modeling components such as local receptive fields, shared weights, and sub-sampling [17]. Thus, we expect that the use of CNN facilitates deeper understanding of documents, and generates better latent factors of items than LDA and SDAE do, especially for items that should resort to their description documents due to the lack of ratings given to them. Moreover, CNNs are able to take advantage of exploiting pre-trained word embedding models such as Glove [30] whereas LDA and SDAE cannot exploit pre-trained word embedding models because they use bag-of-words model.

However, existing CNNs are not suitable for recommendation tasks, because their objectives are generally different from the objective of recommendation. Specifically, conventional CNNs mainly solve classification tasks whose goals are to predict labels of words, phrases, or documents. On the contrary, the objective of most recommender systems is regarded as a regression task aiming at accurately approximating numerical ratings of users given to items. In other words, existing CNNs cannot provide different ratings to the same item for different users. Thus,

existing CNNs cannot be directly applied to our task of recommendation.

To handle this technical issue, we propose a document context-aware hybrid method that integrates CNN into PMF for recommendation tasks. Then, we develop a document context-aware recommendation model, *convolutional matrix factorization* (ConvMF), which draws on the recommendation objective of PMF while effectively exploiting both rating information and contextual information. As a result, ConvMF predicts unknown ratings more accurately even when datasets are extremely sparse. Our extensive experiments showed that ConvMF significantly outperforms the state-of-the-art recommendation models. 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.

3.2 Method

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

3.2.1 Probabilistic Model of ConvMF

Figure 3.1 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

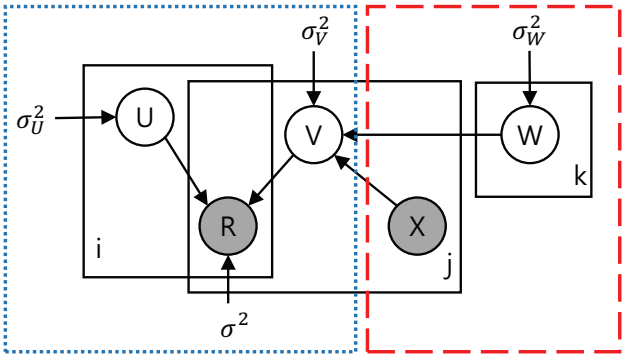


Figure 3.1: The graphical model of ConvMF: PMF part in left (dotted-blue); CNN part in right (dashed-red)

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).

Modeling ratings

In the probabilistic point of view, the conditional distribution of observed ratings as in [2] 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 i^{th} column vector of U, and v_j denotes a j^{th} 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.

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.

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

Modeling items

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, [1] 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.

$$v_j = CNN_W(X_j) + \epsilon_j, \quad \text{where } \epsilon_j \sim N(0, \sigma_V^2 I) \tag{3.1}$$

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

$$p(V|W, X, \sigma_V^2) = \prod_j^M N(v_j|CNN_W(X_j), \sigma_V^2 I), \tag{3.2}$$

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, Eqn.(3.2) plays an important role as a bridge between PMF and CNN that helps to analyze both description documents and ratings.

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

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

[1] Detail of W of CNN will be explained in Section 3.2.2

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

$$\begin{aligned} & p(U, V, W|R, X, \sigma^2, \sigma_U^2, \sigma_V^2, \sigma_W^2) \\ & \approx p(R|U, V, \sigma^2) p(U|\sigma_U^2) p(V|W, X, \sigma_V^2) p(W|\sigma_W^2) \end{aligned} \tag{3.3}$$

In the next subsection, we will explain how document latent vectors are obtained from our CNN model that will be used as the mean in Eqn.(3.2)

3.2.2 CNN Architecture of ConvMF

The objective of our CNN architecture is to generate document latent vectors from description documents of items, which are used to construct the columns of the item latent variable with epsilon variables. Figure 3.2 shows our CNN architecture that consists of four layers; 1) embedding layer, 2) convolution layer, 3) pooling layer, and 4) output layer.

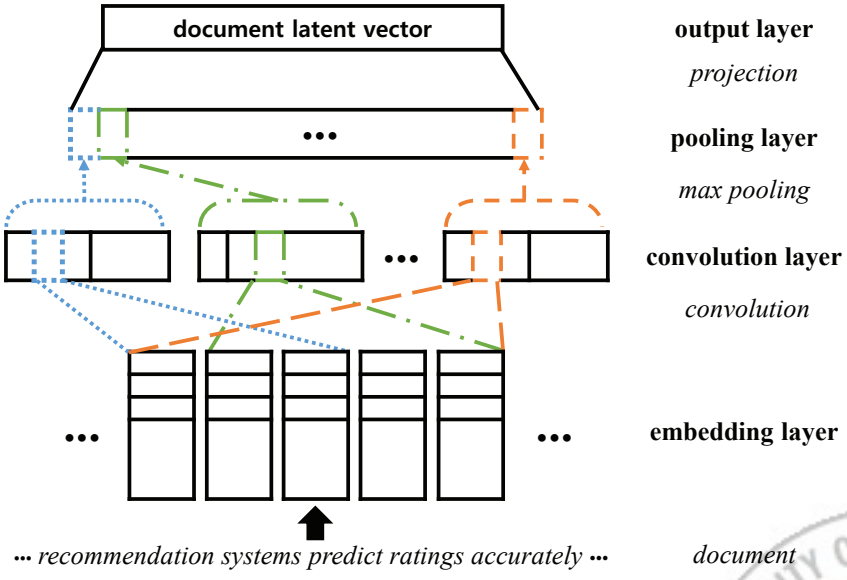


Figure 3.2: Our CNN architecture of R-ConvMF

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 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 [30]. 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.

Convolution layer

The purpose of the convolution layer is to extract 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 [24, 18] 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:

$$c_i^j = f(W_c^j * D_{(:,i:(i+ws-1))} + b_c^j) \tag{3.4}$$

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 [31, 32]. Then, a contextual feature vector $c^j \in \mathbb{R}^{l-ws+1}$ of a document with W_c^j is constructed by Eqn.(3.4):

$$c^j = [\ c_1^j,\ c_2^j,\ \ldots,\ c_i^j,\ \ldots,\ c_{l-ws+1}^j] \tag{3.5}$$

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 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 Eqn.(3.5).

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:

$$s = \tanh(W_{f_2}\{\tanh(W_{f_1}d_f + b_{f_1})\} + b_{f_2}) \tag{3.6}$$

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 raw document as an input, and returns a latent vector of each document as an output:

$$s_j = CNN_W(X_j) \tag{3.7}$$

where W denotes all the weight (W_c^j where $j = 1, 2, ..., n_c$, W_{f_1} and W_{f_2}) and bias (b_c^j where $j = 1, 2, ..., n_c$, b_{f_1} and b_{f_1}) variables in our CNN to prevent clutter, and X_j denotes a raw document of item j, and s_j denotes a document latent vector of item j.

3.2.3 Optimization Methodology

Given Eqn.(3.3) in Section 3.2.1 and our CNN architecture, we use maximum a posteriori (MAP) estimation to optimize the variables U, V and W as follows:

$$\begin{aligned} &\max_{U,V,W} p(U, V, W|R, X, \sigma^2, \sigma_U^2, \sigma_V^2, \sigma_W^2) \\ &= \max_{U,V,W} [p(R|U, V, \sigma^2)p(U|\sigma_U^2)p(V|W, X, \sigma_V^2)p(W|\sigma_W^2)] \end{aligned} \tag{3.8}$$

By taking negative logarithm on Eqn.(3.8), it is reformulated to minimize below loss function $\mathcal{L}$.

$$\begin{aligned} \mathcal{L} = &\sum_i^N \sum_j^M \frac{I_{ij}}{2}(r_{ij} - u_i^T v_j)^2 + \frac{\lambda_U}{2}\sum_i^N \|u_i\|^2 \\ &+ \frac{\lambda_V}{2}\sum_j^M \|v_j - CNN_W(X_j)\|^2 + \frac{\lambda_W}{2}\sum_k^{|w_k|} \|w_k\|^2, \end{aligned} \tag{3.9}$$

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.

To minimize $\mathcal{L}$, we adopt coordinate descent which iteratively updates a trainable variable while fixing the remaining variables. Specifically, Eqn.(3.9) 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 loss function $\mathcal{L}$ with respect to u_i (or v_j) as follows.

$$u_i \leftarrow (VI_iV^T + \lambda_U I_K)^{-1} VR_i \tag{3.10}$$

$$v_j \leftarrow (UI_jU^T + \lambda_V I_K)^{-1}(UR_j + \lambda_V CNN_W(X_j)) \tag{3.11}$$

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.(3.11) shows the effect of a document latent vector of CNN ($CNN_W(X_j)$) in updating v_j through λ_V as a balancing parameter that regulates how much document information is used as in [11].

However, W cannot be optimized 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 squared error function with L_2 regularized terms. Specifically, the function is regarded as a kind of neural networks where the input is a raw description document and the output is a column of the trained item latent variable in a iteration as follows:

$$\begin{aligned}\mathcal{E}(W) =& \frac{\lambda_V}{2}\sum_j^M \|(v_j - CNN_W(X_j))\|^2 \\ &+ \frac{\lambda_W}{2}\sum_k^{|w_k|}\|w_k\|^2 + \text{constant}\end{aligned} \tag{3.12}$$

Thus, we use the back-propagation algorithm typically used in neural networks

to obtain below gradient and optimize W until convergence of $\mathcal{E}$.

$$\nabla_{w_k}\mathcal{E}(W) = -\lambda_V \sum_{j}^{M}(v_j - \nabla_{w_k} CNN_W(X_j)) + \lambda_W w_k \tag{3.13}$$

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

3.3 Experiment

In this section, we evaluate the empirical performance of ConvMF on real-world datasets. Our extensive experiment results demonstrate four following claims.

1. ConvMF significantly outperforms other competitors when dataset is extremely sparse.

2. pre-trained word embedding model helps improve the performance of ConvMF when dataset is extremely sparse.

3. the best performing parameters verify that ConvMF well alleviates data sparsity.

4. ConvMF indeed captures subtle contextual differences.

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 3.1: Data statistic on three real-world datasets

3.3.1 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. Since MovieLens datasets does not include description documents of items, we obtained documents (i.e., plot summary) of corresponding items from IMDB [4]. Amazon dataset includes reviews on items and thus we used reviews as description documents of items.

Similar to [11] and [12], we preprocessed description documents for all datasets as follows:

1. We set maximum length of raw 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.

[2]http://grouplens.org/datasets/movielens/
[3]Preprocessed Amazon product data (http://jmcauley.ucsd.edu/data/amazon/)
[4]Plot summaries are available at http://www.imdb.com/

Model	ML-1m		ML-10m		AIV	
	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 3.2: Parameter Setting of λ_U and λ_V

6. We 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 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 3.1). Precisely, even though several users are removed by preprocessing, Amazon dataset is still extremely sparse compared with the others.

Competitors and parameter setting

We compared our models with the following competitors.

- PMF [2]: Probabilistic Matrix Factorization is a standard rating prediction model that only uses ratings for collaborative filtering.
- CTR [11]: 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 [12]: 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+: ConvMF+ is ConvMF with the pretraining word embedding model [30].

As reported in [12] (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 3.2 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 [12].[5]

Implementation detail

We implemented our models using Python 2.7 and Keras 0.3.3 [33] 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.

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

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

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 Eqn.(3.6) 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.

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 model 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 the means of the test errors.

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)
Improve	3.92%	2.79%	16.60%

Table 3.3: Overall test RMSE. "Improve" indicates the relative improvements of ConvMFover the best competitor, CDL.

3.3.2 Experimental Results

A1. Quantitative Results on MovieLens and Amazon 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 3.3 and 3.4, 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*

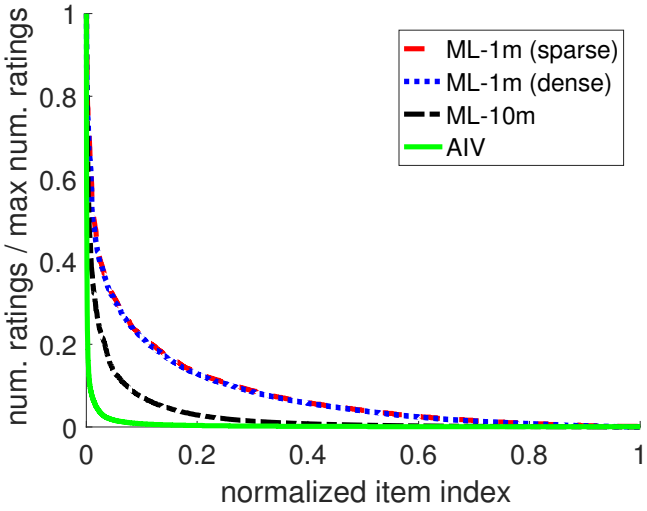


Figure 3.3: Skewness of the number of ratings for items on each dataset

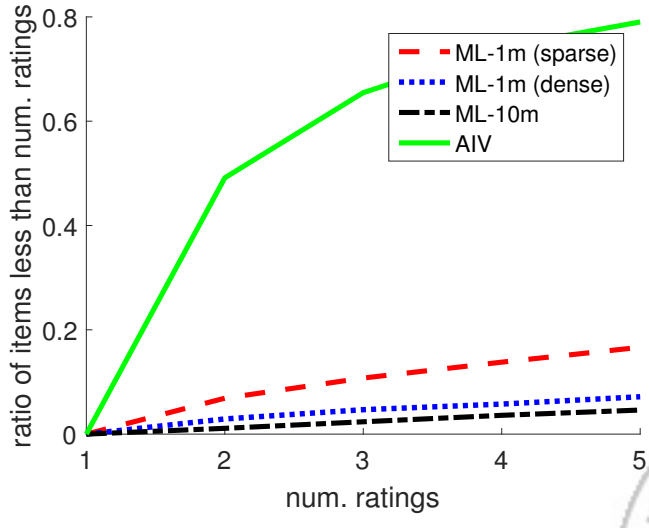


Figure 3.4: Ratio of items that have less than num. ratings to each entire dataset

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.

A2. Impact of the pre-trained word embedding model

Unlike CTR and CDL, ConvMF is basically able to take advantage of using pre-trained word embedding models such as Glove [30]. Thus, we investigate the impact of pre-trained word embedding models on our recommendation task by initializing the embedding layer of CNN of ConvMF using the pre-trained word embedding model of Glove [30].

Table 3.3 shows marginal changes of ConvMF+ over ConvMF on three datasets: -0.22%, 0.35% and 0.51% on the ML-1m, ML-10m and AIV datasets, respectively. Note that the sparseness of datasets is in increasing order of ML-1m, ML-10m and AIV. In spite of 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 the pre-trained word embedding model complement the shortage of rating information. We also observe that with given sufficient number of ratings to train latent variables, the 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 the pre-trained word embedding model obtained from external documents when a rating matrix is relatively dense.*

Moreover, we investigate relative improvements of ConvMF+ over ConvMF on various λ_V, a parameter that balances the usage of description documents

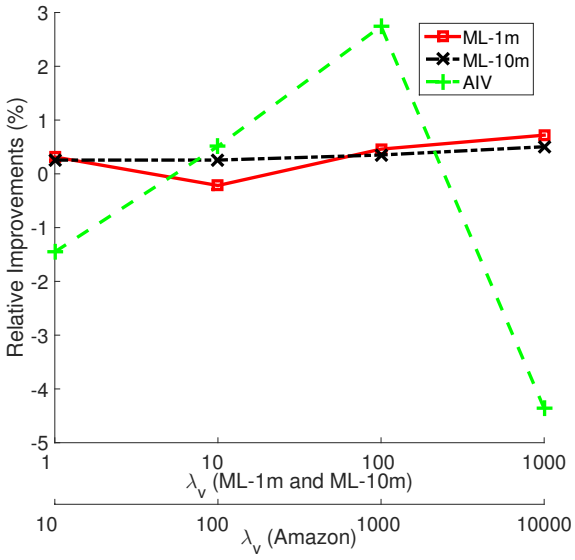


Figure 3.5: Relative improvements of ConvMF+ over ConvMF

during the training process. In Figure 3.5, for the MovieLens datasets, since the dataset has relatively large number of ratings, the pre-trained word embedding model does not have much impact on the performance of ConvMF on various λ_V. However, for the extremely sparse Amazon dataset, the pre-trained word embedding model indeed affects the performance of ConvMF. Specifically, when λ_V is 100 and 1000, the improvements of ConvMF+ over ConvMF reach almost 0.51% and 2.74% while applying the same parameter setting to ConvMF and ConvMF+ whereas the performance of ConvMF+ suddenly drops with other λ_V values. In other words, the relatively low or high λ_V with the pre-trained word embedding model fails to achieve high performance. This phenomenon implies that the latent variables can be under-fitted or over-fitted by the word embedding model when rating data is extremely sparse. Nevertheless, Figure 3.5 shows that *given a proper value of λ_V, adopting the pre-trained word embedding model increases the performance of ConvMF when the number of ratings is insufficient.*

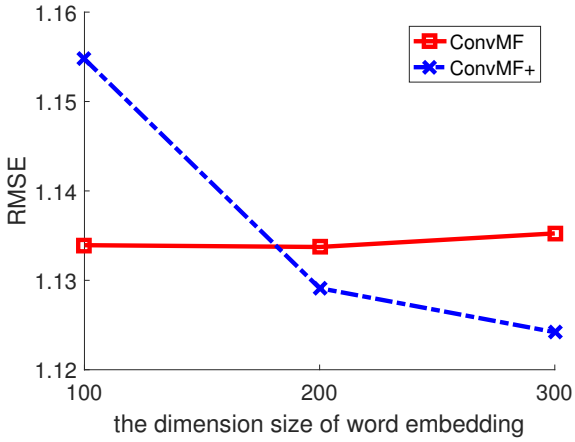


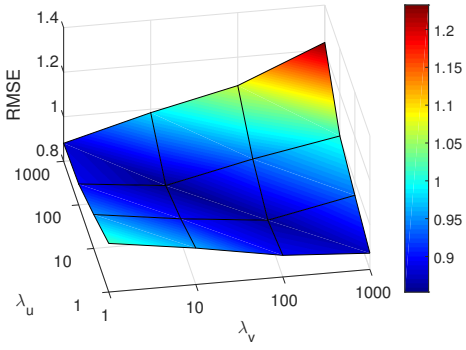
Figure 3.6: The effects of the dimension size of word embedding on Amazon dataset

A3. Parameter analysis

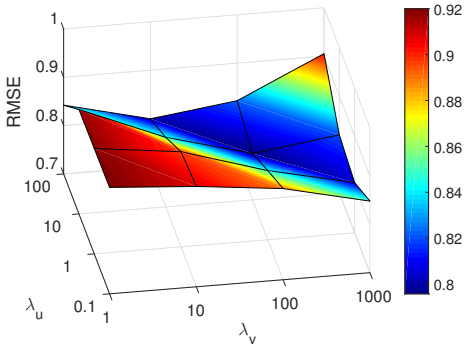
We investigate the impact of three parameters on the performance of ConvMF: p (the dimension size of word embedding), λ_u and λ_v.

Figure 3.6 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, and the information contained in the pre-trained word embedding model gets richer as p gets larger.*

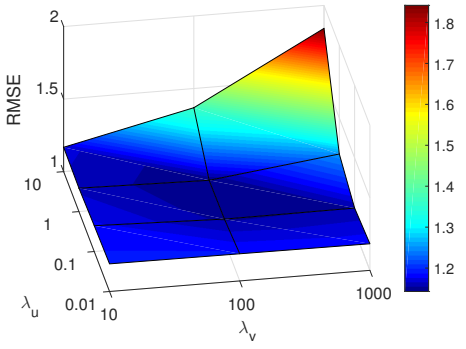
Figure 3.7 shows the impact of λ_u and λ_v on three real-word datasets. Regarding the changes of the best performing values of λ_u and λ_v from Figure 3.7(a) to (c), we observe that when the rating data becomes sparse, λ_u decreases while λ_v increases, to produce the best results. Precisely, the values of (λ_u, λ_v) of ConvMF are (100, 10), (10 and 100), and (1 and 100), on ML-1m, ML-10m, and



(a) MovieLens-1m



(b) MovieLens-10m



(c) Amazon Instant Video

Figure 3.7: Parameter analysis of λ_U and λ_V on three dataset

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 3.4: Case study on two shared weights of ConvMF

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 ConvMF and R-ConvMF well alleviates the data sparsity by balancing the usage of description documents.*

A4. 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 basic model, ConvMF trained on ML-10m dataset, and compare the contextual meaning of phrases captured by the shared weights (Table 3.4). The meanings 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 the 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 3.4. 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.*

IV. A Different Gaussian Noise Latent Factor Modeling Method for Document Context-Aware Recommender Systems

4.1 Introduction

Our proposed hybrid recommendation model for document context-aware recommender systems in Chapter III seems to effectively improve the rating prediction accuracy, however, our proposed integrated model still has the second limitation of previous work. Figure 4.1 experimentally shows the second limitation of existing recommendation models, because the improvements of hybrid recommendation models over basic CF model do not increase when sparseness of the dataset gets increased. As mentioned in Chapter I, this phenomenon is caused by considering Gaussian noise equally in modeling latent factors of items based on description documents. Specifically, inaccurate item latent factors obtained from the intermediate training process of hybrid recommendation models hinder further improvements in terms of the rating prediction.

In order to resolve the latter limitation, as the second step of our work, we propose a robust document context-aware hybrid method by introducing a different Gaussian noise latent factor modeling method for items. In order to explicitly consider Gaussian noise differently, our proposed latent factor modeling method uses the statistics of items (i.e. 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 hybrid recommendation model more robust to not only sparse but also skewed datasets by constructing latent factors of items even more accurately. To

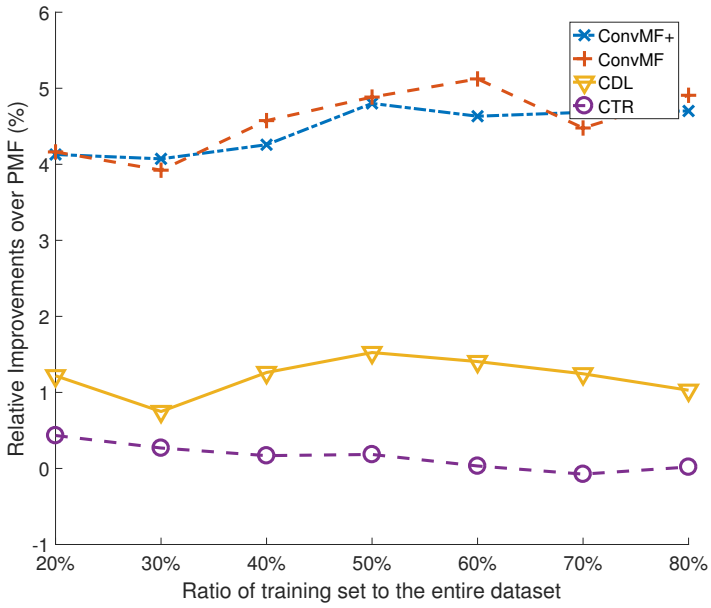


Table 4.1: Relative improvements over PMF of four models with respect to various sparseness of the real world dataset – Movielens-1M

demonstrate the effectiveness of R-ConvMF, we report extensive new experimental results of R-ConvMF using three different real-world dataset obtained from MovieLens[1] and Amazon[2].

In this chapter, our contributions are summarized as follows:

- We propose a robust document context-aware hybrid method to additionally exploit both contextual information and statistics of items together with ratings.
- We model latent factors of items from a probabilistic point of view to exploit not only contextual information but also statistics of items.
- We evaluate R-ConvMF to demonstrate the effectiveness of our proposed method by comparing the state-of-the-art models.

[1]http://grouplens.org/datasets/movielens/
[2]Preprocessed Amazon product data (http://jmcauley.ucsd.edu/data/amazon/)

4.2 Method

In this section, we introduce a different Gaussian noise latent factor modeling method for items based on our proposed document context-hybrid method in Chapter III. Accordingly, while omitting redundant explanations as much as possible for better readability, we describe our robust document context-aware hybrid method.

4.2.1 Modeling items for R-ConvMF

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.

$$v_j = CNN_W(X_j) + \epsilon_j, \tag{4.1}$$
$$\text{where } \epsilon_j \sim N(0, \sigma_V^2 I)$$

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

$$p(V|W, X, \sigma_V^2) = \prod_j^M N(v_j|CNN_W(X_j), \sigma_V^2 I), \tag{4.2}$$

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, Eqn.(4.2) 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 [34], 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 [35], 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 a different Gaussian noise based latent factor modeling method for items, which explicitly reflects the number of ratings given to each item into Eqn.(4.1) so as to consider Gaussian noise differently as follows:

$$v_j = CNN_W(X_j) + \epsilon_j, \qquad (4.3)$$
$$\text{where } \epsilon_j \sim N(0, \frac{\sigma_V^2}{f(n_j)} I)$$

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

$$p(V|W, X, \sigma_V^2) = \prod_j^M N(v_j|CNN_W(X_j), \frac{\sigma_V^2}{f(n_j)} I) \qquad (4.4)$$

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 4.3.2.

4.2.2 Optimization Methodology for R-ConvMF

Our proposed latent factor modeling methods for items reformulates the loss function of the previous hybrid document context-aware method as follows.

$$\begin{aligned}\mathcal{L} = &\sum_{i}^{N}\sum_{j}^{M}\frac{I_{ij}}{2}(r_{ij} - u_i^T v_j)^2 + \frac{\lambda_U}{2}\sum_{i}^{N}\|u_i\|^2 \\ &+ \frac{\lambda_V}{2}\sum_{j}^{M} f(n_j)\|v_j - CNN_W(X_j)\|^2 + \frac{\lambda_W}{2}\sum_{k}^{|w_k|}\|w_k\|^2,\end{aligned} \tag{4.5}$$

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 Eqn.(4.4) for the conditional distribution of the item latent variable instead of Eqn.(3.2), Eqn.(4.5) can be simply reduced to our previous model (ConvMF) using Eqn.(3.2) without using the number of ratings of each item (i.e. set $f(n_j) = 1$ for all items $(j = 1, ..., M)$).

To minimize $\mathcal{L}$, the update rules of U and V are changed as follows.

$$u_i \leftarrow (VI_iV^T + \lambda_U I_K)^{-1}VR_i \tag{4.6}$$

$$v_j \leftarrow (UI_jU^T + f(n_j)\lambda_V I_K)^{-1}(UR_j + f(n_j)\lambda_V CNN_W(X_j)) \tag{4.7}$$

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.

Similarly, the update rule of W is also changed as follows.

$$\begin{aligned}\mathcal{E}(W) = &\frac{\lambda_V}{2}\sum_{j}^{M} f(n_j)\|(v_j - CNN_W(X_j))\|^2 \\ &+ \frac{\lambda_W}{2}\sum_{k}^{|w_k|}\|w_k\|^2 + \text{constant}\end{aligned} \tag{4.8}$$

Eqn.(4.8) mathematically verifies that R-ConvMF outperforms ConvMF. Concretely, pairs of X_j and v_j are used to update W. Such v_j tends to be well-trained by the PMF part when the corresponding item has many ratings, while v_j tends to be ill-trained by the PMF part when the corresponding item

has few ratings. Intuitively, some pairs of X_j and v_j with few ratings on those items should have less effects on updating W than others. $f(n_j)$ in Eqn.(4.8) explicitly reflect this phenomenon with respects to the number of ratings on each item in order to optimize W of CNN effectively. Note that if we set $f(n_j)$ to 1 for all items to mimic ConvMF, then W would be trained by both well-trained v_j and ill-trained v_j with the same effect. Relatively less accurate W affects the optimization process of V directly, and U indirectly. This process would be repeated, and eventually, we cannot expect further improvements on the rating prediction accuracy.

We use the back-propagation algorithm which is typically used in neural networks in order to obtain Eqn.(4.9), and we optimize W until convergence of $\mathcal{E}$ or reaching the pre-defined number of iteration.

$$\nabla_{w_k}\mathcal{E}(W) = -\lambda_V \sum_{j}^{M} f(n_j)(v_j - \nabla_{w_k} CNN_W(X_j)) + \lambda_W w_k \tag{4.9}$$

Recall that W is the weights and biases of each layer, and even in gradients of W, we find out that $f(n_j)$ gives more weight to relatively accurate target v_j.

As shown in Algorithm 1 of R-ConvMF, 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.

4.2.3 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

Algorithm 1 R-ConvMF algorithm

Input: R: user-item rating matrix, X: description documents of items

Output: 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.(4.9)
        end for
    until convergence
until satisfying early stopping using a validation set

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.

4.3 Experiment

In this section, we evaluate the empirical performance of R-ConvMF on three real-world datasets. Compared with Chapter III, our new experiments are conducted to solve the following questions:

1. Does taking into account Gaussian noise differently for items improve the accuracy of our preliminary models?

2. How does the replacement of CNN to LSTM or GRU affect the performance (accuracy, training time) of R-ConvMF?

3. How does lemmatization affects the accuracy of R-ConvMF?

4. How hyper-parameters such as regularized parameters, the number of kernel, and max length of documents affect the accuracy of our models?

First question will demonstrate the effectiveness of our proposed latent factor modeling method, and the rest will verify how the degree of deeper understanding of description documents affects the performance of recommender systems via various experimental settings.

4.3.1 Experimental Setting

Datasets

Following the experimental setting of Chapter III, to demonstrate the effectiveness of our recommendation models in terms of rating prediction, we used three real-world datasets obtained from MovieLens and Amazon. As mentioned

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 4.2: Data statistic on three real-world datasets

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

Table 4.3: Comparison between 6 models: R-ConvMF, our two previous hybrid models and the three competitors

in Chapter III, 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 [3]. Amazon dataset includes reviews on items and thus we used reviews as description documents of items.

Competitors and parameter setting

We compared R-ConvMF with the following competitors. Table 4.3 shows their conceptual differences.

- PMF [2]: Probabilistic Matrix Factorization is a standard rating prediction model that only uses ratings for collaborative filtering.

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

- CTR [11]: 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 [12]: 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 previous hybrid recommendation model, which considers contextual information together with ratings.

- ConvMF+: ConvMF+ is ConvMF with the pretraining word embedding model [30].

- 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.

We follow the parameter settings of competitors in Chapter III. Table 4.4 shows the best performing values of common parameters (λ_U, λ_V) of each model including in R-ConvMF found by the grid search ($\lambda_U \in \{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$ and $\lambda_V \in \{0.01, 0.1, 1, 10, 100, 1000, 10000, 100000\}$).

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.

Model	ML-1m		ML-10m		AIV	
	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

Table 4.4: Parameter Setting of λ_U and λ_V

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.

4.3.2 Experimental Results

Table 4.5 shows the overall rating prediction errors of R-ConvMF, two previous proposed models, and the three competitors. Compared with the five competitors, R-ConvMF achieve significant improvements for all the datasets. Following subsections give detailed answers to the questions as we previously mentioned at the beginning of Section 4.3.

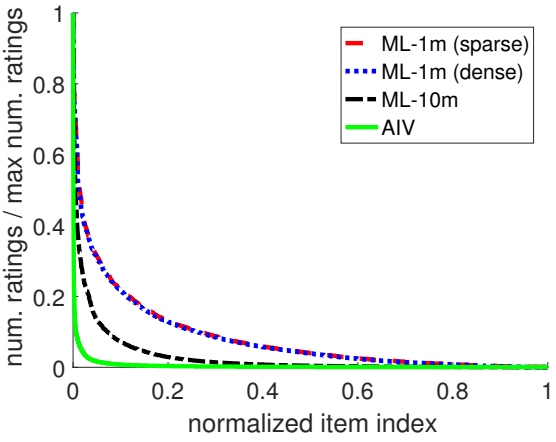


Figure 4.1: Skewness of the number of ratings for items on each dataset – Amazon dataset is the most skew.

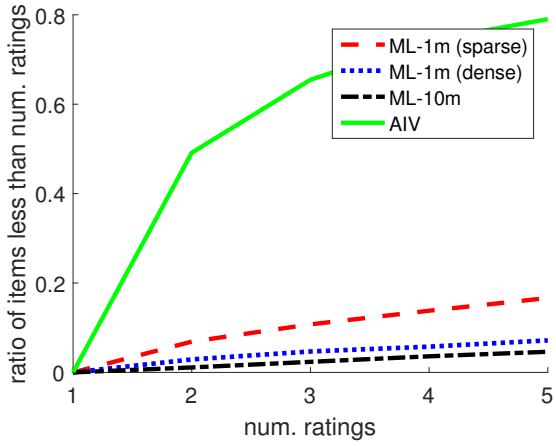


Figure 4.2: Ratio of items that have less than num. ratings to each entire dataset – 50% of items in Amazon dataset have only one rating.

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)

Table 4.5: Overall test RMSE

A1. Impact of taking into account Gaussian noise differently in modeling latent factors for items

Table 4.5 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 4.2.1, 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, Eqn.(4.9) in Section 4.2.2 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

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
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**

Table 4.6: Test RMSE over various sparseness of training data on ML-1m dataset

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 Eqn.(4.7). *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, Figure 4.1 and 4.2 show that while ML-1m (sparse) and ML-1m (dense) datasets whose 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 4.6 shows that three variant models based on our proposed method significantly outperform three competitors over all ranges of sparseness, and Figure 4.3 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

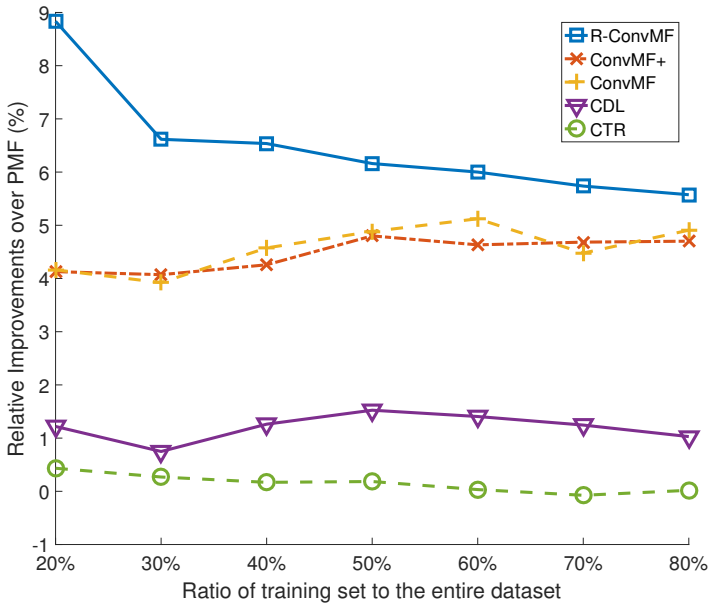


Figure 4.3: Relative improvements over PMF of five models

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.

A2. 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.

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

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.

Table 4.7: 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

dataset among three datasets with the same experimental setting in Section 4.3.1. 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 4.7 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 re-generate 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 4.7. 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.

A3. 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 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.

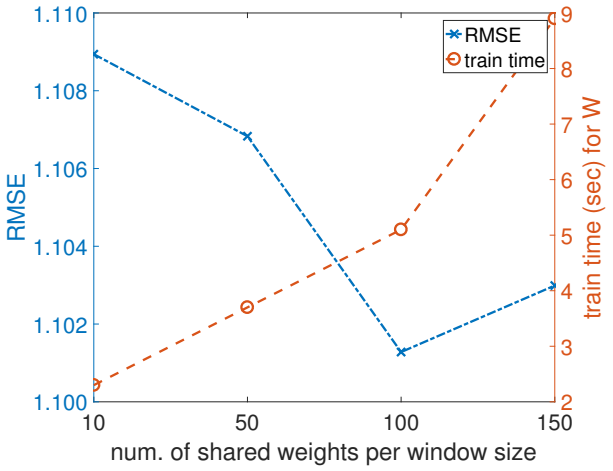


Figure 4.4: RMSE and train time for W[4]of R-ConvMF w.r.t the number of shared weights per window size on Amazon dataset

A4. 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 3.2.2, 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. Figure 4.4 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 win-

[4]train time for W denotes train time for W per inner-epoch as in Line 13–15 of Algorithm 1 in both Figure 4.4 and 4.5

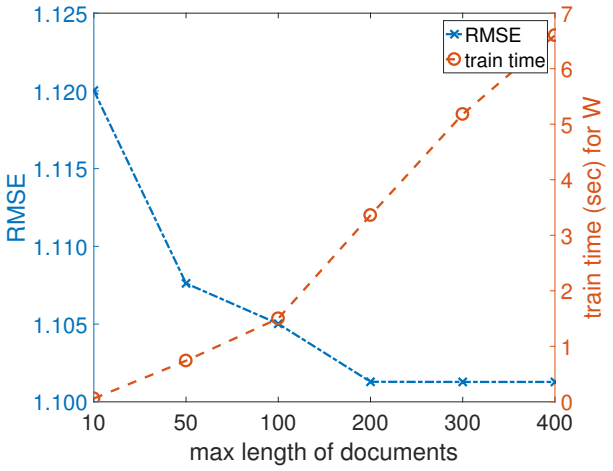
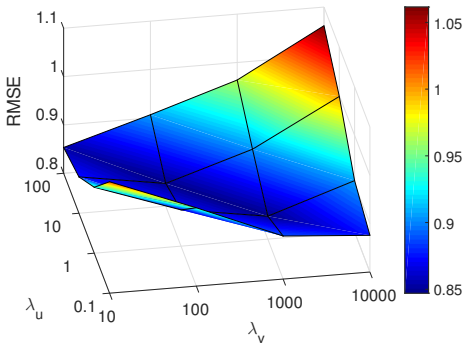


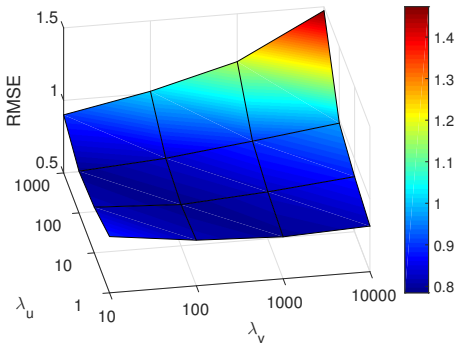
Figure 4.5: RMSE and train time for W of R-ConvMF w.r.t max length of documents on Amazon dataset

dow 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.*

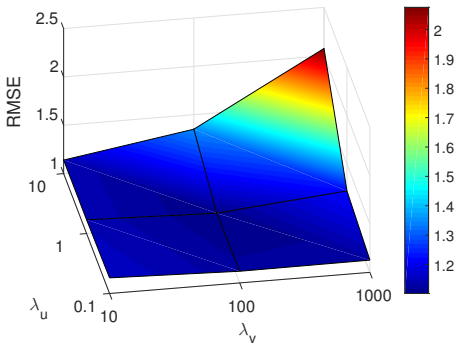
2) Impact of pre-determined maximum length of documents: In order to verify the impact of the amount of document information, we set the maximum length of documents from 10 to 400 in various ways, and evaluate R-ConvMFon Amazon dataset. Figure 4.5 shows that RMSE consistently is decreased until the maximum 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 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*



(a) MovieLens-1m



(b) MovieLens-10m



(c) Amazon Instant Video

Figure 4.6: Parameter analysis of λ_U and λ_V on three dataset of R-ConvMF

between maximum length of documents and train time.

3) Impact of two parameters λ_U and λ_V: Figure 4.6 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 4.6(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 R-ConvMFare (10, 100), (10, 100) and (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.*

V. Conclusion

In this thesis, we address two limitations of existing work: 1) they ignore contextual information. 2) they do not consider Gaussian noise differently. To solve the limitations, through the two-step approach, 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 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.

Moreover, since our proposed method adopts PMF, which is the standard MF-based recommendation model, our proposed method are able to be extended to combining other MF-based recommendation models such as SVD++ [1] that only consider ratings. 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.

요 약 문

본 학위 논문은 하이브리드 협업 추천 시스템에서의 두 가지 한계점을 다음과 같이 지적하였습니다. 먼저 기존의 방법론들은 문서의 문맥 정보를 고려하지 않았으며, 품목의 잠재 벡터를 모델링하는 과정에서 발생할 수 있는 가우시안 노이즈를 동일하게 고려하여 학습을 수행하는 한계가 있습니다. 이러한 두 가지 한계점은 품목의 문서 정보를 고려하는 추천 시스템의 성능을 저해하는 요소가 되며, 이에 따라 본 논문에서 우리는 문서의 문맥 정보를 고려함과 동시에 품목의 잠재 벡터 모델링 시 가우시안 노이즈를 품목의 통계 정보량에 따라 다르게 반영하여 왜곡성과 희소성이 극심한 데이터셋에 대해서도 효과적인 학습이 가능한 방법론을 제안하였습니다. 제안한 방법론의 효과성을 검증하기 위해 여러 종류의 다양한 실험을 수행하였으며 현존하는 방법론들과 비교해볼 때 추천 시스템에서의 평점 예측 정확도에서 상당한 성능 개선을 확인하였습니다.

뿐만 아니라, 제안한 방법론은 행렬분해 기술 중 가장 기본적인 행렬분해 기술인 PMF를 채택하였기 때문에 평점 정보를 활용하는 다른 종류의 행렬분해 추천 모델들에 적용될 수 있습니다. 게다가 본 방법론은 사용자-품목 평점 정보와 품목의 문서정보만을 활용하기 때문에 아마존 또는 넷플릭스와 같은 실제 전자 상거래 시스템에 쉽게 적용이 가능합니다. 특히 별도의 사용자 정보를 추가로 활용하지 않기 때문에 개인정보침해와 관련된 이슈들도 고려할 필요가 없습니다.

본 논문의 후속 연구로서, 딥러닝 연구에서 비지도 선학습 기술은 딥러닝 모델의 성능을 향상시키는데 도움이 된다고 알려져 있기 때문에, 문서를 더 정확히 파악하기 위해 나선형의 오토인코더를 개발하고자 합니다. 나선형의 오토인코더를 통하여 임베딩 계층의 잠재변수 이외의 다른 계층의 잠재변수들도 비지도 선학습이 가능해지며 이는 제안한 방법론의 성능을 개선시킬 여지가 있을 것이라 기대합니다.

References

[1] Yehuda 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.

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

[3] Gediminas Adomavicius and Alexander Tuzhilin. 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.

[4] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, January 2004.

[5] Sheng Li, Jaya Kawale, and Yun 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.

[6] Sanjay Purushotham, Yan Liu, and C.-C. Jay 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.

[7] C. Martinez-Cruz, C. Porcel, J. Bernabé-Moreno, and E. Herrera-Viedma. A model to represent users trust in recommender systems using ontologies and fuzzy linguistic modeling. *Inf. Sci.*, 311(C):102–118, August 2015.

[8] Chanyoung Park, Donghyun Kim, Jinoh Oh, and Hwanjo Yu. Improving top-k recommendation with truster and trustee relationship in user trust network. *Information Sciences*, 374:100–114, 2016.

[9] Guang Ling, Michael R. Lyu, and Irwin 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.

[10] Julian McAuley and Jure 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.

[11] Chong Wang and David 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.

[12] Hao Wang, Naiyan Wang, and Dit-Yan 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.

[13] R. He and J. McAuley. VBPR: visual bayesian personalized ranking from implicit feedback. In *AAAI Conference on Artificial Intelligence*, 2016.

[14] Xiaolin Zheng, Weifeng Ding, Zhen Lin, and Chaochao Chen. Topic tensor factorization for recommender system. *Inf. Sci.*, 372(C):276–293, December 2016.

[15] A Tejeda-Lorente, Carlos Porcel, Juan Bernabé-Moreno, and Enrique Herrera-Viedma. Refore: A recommender system for researchers based on bibliometrics. *Applied Soft Computing*, 30:778–791, 2015.

[16] Álvaro Tejeda-Lorente, Carlos Porcel, Eduardo Peis, Rosa Sanz, and Enrique Herrera-Viedma. A quality based recommender system to disseminate information in a university digital library. *Inf. Sci.*, 261:52–69, March 2014.

[17] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Intelligent Signal Processing*, pages 306–351. IEEE Press, 2001.

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

[19] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems*, 22(1):143–177, January 2004.

[20] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John 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.

[21] Jianfeng Gao, Patrick Pantel, Michael Gamon, Xiaodong He, and Li Deng. Modeling interestingness with deep neural networks. In *Proceedings of the*

2014 Empirical Methods in Natural Language Processing (EMNLP), pages 2–13, 2014.

[22] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire 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.

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

[24] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research (JMLR)*, 12:2493–2537, November 2011.

[25] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 111–112, New York, NY, USA, 2015. ACM.

[26] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, WSDM '16, pages 153–162, New York, NY, USA, 2016. ACM.

[27] Trapit Bansal, David Belanger, and Andrew McCallum. Ask the gru: Multi-task learning for deep text recommendations. In *Proceedings of the 10th ACM*

Conference on Recommender Systems, RecSys '16, pages 107–114, New York, NY, USA, 2016. ACM.

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

[29] Yifan Hu, Yehuda Koren, and Chris 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.

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

[31] Xavier Glorot, Antoine Bordes, and Yoshua 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.

[32] Andrew L. Maas, Awni Y. Hannun, and Andrew 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.

[33] François Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[34] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 880–887, New York, NY, USA, 2008. ACM.

[35] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 233–240, New York, NY, USA, 2016. ACM.

Acknowledgements

2012년 봄, 석박사 통합과정으로 포항공대 컴퓨터공학과 데이터 마이닝 연구실에 처음 발을 들였던 순간이 아직도 생생한데, 벌써 5년 반이 흘러 박사 학위를 받게 되니 이 감정을 뭐라고 표현할지 잘 모르겠습니다. 박사 학위를 받기 위한 기간을 6 7년을 생각했던 것과는 달리 여러 좋은 일들이 우연히 잘 겹쳐 조금 빨리 박사 학위를 받게 되었지만, 저 스스로 아직 더 많이 발전해야 한다는 생각과 이제 또 다른 시작이라는 생각이 드니 마냥 기쁘지만은 않은 것 같습니다. 그래도 드디어 끝나긴 했다는 마음에 조금은 후련하긴 합니다. 아마 하루 이틀 더 지나면 더 실감이 나겠지요.

박사학위를 받는 순간까지 부족한 저를 응원하고 격려해주셨던 모든 인연들께 감사드립니다. 먼저, 5년 반 동안 여러 주제의 연구 기회를 주신 저의 지도교수님이신 유환조 교수님 감사드립니다. 석박사 통합과정 동안 저를 좋게 봐주신 덕분에 그 기대에 부응하기 위해 열심히 노력하였고, 박사학위를 받을 수 있었습니다. 그리고 바쁘신 와중에도 박사 논문 심사위원 요청을 흔쾌히 받아주신 전치혁 교수님, 이종혁 교수님, 고영명 교수님, 조민수 교수님 감사드립니다. 박사 프로포절 발표 때 건설적인 코멘트를 잘 주셔서 박사 디펜스 준비 및 발표를 잘 마무리할 수 있었습니다. 앞으로도 좋은 연구자가 되도록 열심히 노력하겠습니다.

저의 두번째 지도교수님(?) 역할을 해주신 진오형, 일찍이 졸업하시고 지금은 연구실에 계시진 않지만 연구실 생활 생각하면 형이 제일 생각이 많이 나네요. MSRA에 가서 추천 시스템 연구 시작하면서 형과 디스커션 하던 때가 아직도 기억에 많이 남아요. 그 때는 형이 졸업 준비하느라 바쁘셨죠. 그 뒤에 한국으로 돌아와서 형과 논문 작업하면서 정말 많이 배운 것 같아요. 정말정말 감사드립니다. 어도비 취직 정말 축하드리고, 꽃길만 걸으시길 바랄게요. 얼른 결혼하세요. 저보다 2년 늦게 들어왔지만 누구보다 잘하고 있는 연구실 에이스인 동갑내기 친구 찬영이. 엄친아의 표본이면서 연구실에서 실제 갓역할을 정말로 하고 있다고 난 생각한다. 너와 디스

커션하고 논문 작업할 때는 정말 건설적이고 재밌는 것 같아. 4년차 밖에 안됐는데 실적 및 능력에서 나보다 더 뛰어나니 너도 금방 졸업할 수 있을거라 믿어 의심치 않아. 언제나 열심히 하니 정말 좋은 곳으로 갈 수 있을 거야. 나 열심히 할게 좀 써줘. 학부 때부터 같은 분반으로 같이 입학하여 대학원까지 같이 생활한 동기 예진이, 서로 연구실에 같이 있던 시간은 그렇게 길지는 않지만 우여곡절 끝에 같이 졸업하게 되서 기쁘고 정말 축하한다. UCSD에 포닥으로 빠르게 갈 수 있기를 바라고, 내가 1년 뒤에 지원하게 되면 잘 부탁할게. 지금은 학교에 없지만 좋은 연구자의 모습과 선배의 모습을 몸소 보여주신 정말 착한 진하형, 성철이형, 미국 오라클, 어도비에서도 여전히 멋진 모습으로 계실거라 생각합니다. 제가 만약에 미국에 갈 수 있게 된다면 꼭 봬요. 형들 덕분에 제가 한걸음 더 발전할 수 있었습니다.

지금은 연구실에 없지만, 연구실에 처음 들어왔을 때 계셨던 일환이형, 태훈이형, 재용이형, 선이누나, 승걸이형, 영철이형, 승호. 저를 많이 챙겨주시고 이뻐해주신 덕택에 저의 연구실 1, 2년차 생활이 즐거울 수 있었습니다. 감사합니다. 승호는 꼭 여자친구 생겨서 포항으로 안 오길 바랄게. 그리고 저희 연구실 비지터에서도 사라진 동기 규동이형 프로포절 잘 준비해서 성공적으로 마무리 할 수 있기를 바랄게요. 원래 통합과정으로 입학했지만 석사로 전환해서 졸업하여 지금은 행복하게 잘 살고 있는 동은이형, 유진이. 제가 좀 더 똑똑하고 잘 챙겨주었다면 진로가 바뀌었을까하는 아쉬움이 남아 늘 미안합니다. 둘 다 뭐든지 열심히 하니 각 회사에서 성공할 거라 생각합니다. 크리스마스의 선물(?)이었는지 기억이 정확하진 않지만 일찍 결혼하게 되어 석사 졸업한 진하, 네이버에서는 행복하고 저녁이 있는 삶을 지낼 수 있기를 바라. 그리고 기업과제로 엄청 많은 고생을 하고 있는 병주. 많이 도와주지 못해서 미안했고 어떻게든 해결책을 같이 잘 찾아보자. 박사생활 잘 마무리할 수 있을까 걱정을 많이 하는데, 너라면 잘 마칠 수 있을 거야. 찬영이를 이어 우리 연구실의 차세대 에이스인 랩장 동하, 여러 조교 일로 많이 바쁘고 연구에 대해 걱정이 많은 걸로 알지만 능력자니까 잘 헤쳐 나갈 거라 생각해. 내가 생각하기로는 우리 연구실에서 제일 멘탈이 튼튼한 동민이, 너의 그 특유의 안정감은 박사생활을 잘 마무리하는데 큰 도움이 될거라 생각해. 요새 연구실에서 가장 주목받는 진균이.

얼른 카톨릭대의 손에서 벗어나서 자유로워지길 바랄게. 말은 안해도 묵묵히 열심히 잘 하니까 연구실 생활 잘 마무리 할 수 있을 거야. 그 외에도 박학다식하고 연구실 정령 현준이, 연구실 회식할 때 옆에 앉으면 늘 잘 챙겨주는(?) 명섭이, 아무리 한국에 오래 지냈다고는 하지만 그래도 한국말이 어색할 텐데 영어로 의사소통을 잘 못해줘서 늘 미안한 파닛, 다들 능력 있고 멋진 사람들이니까 원하는 목표를 잘 이룰 수 있을 거라 생각해. 우리 연구실 신입생 4인방, 늘 겸손한 성보, 배시시 잘 웃는 준영이, 어딘가 천재 기질이 있는 경석이, 사람들에게 늘 먼저 다가가고 장난치는 준수. 다들 ML과 패턴수업 잘 듣는 거 보니 너희들이 진짜 갓이야. 내 학점은 비밀이란다. 박사생활이 앞이 안 보인다고 다들 걱정을 많이 하지만 나보다 뛰어나니 더 빨리 졸업하거나 더 좋은 곳으로 갈 거야. 지금처럼 4명이서 똘똘 뭉쳐서 서로 협력해서 힘들 수도 있는 연구실 생활 현명하게 잘 헤쳐 나가길 바랄게.

저의 대학생활 및 대학원생활에서 빠질 수 없는 컨디 08 동기들. 특히 준웅아 학부 때부터 대학원까지 참 많은 일들도 있었고 앞으로도 수없이 많을 테지만 박사 졸업 너도 잘 할 수 있기를 내가 응원할게. 그 동안은 복싱 같이 다니면서 서로 많이 치고 박고 하자. 앞으로 1년은 계속 보겠네. 그리고 서울에서 대학원 생활하고 있는 귀염둥이 막내 선기. 우리 중에 제일 똑똑하니까 너도 잘 졸업할 수 있을 거야. 골든나노파티클! 우리 중에 가장 잘나가는 이재훈 의사님. 방돌이 시절 너의 귀여움을 목격한 사람은 나밖에 없어 아쉽지만 너의 내재된 귀요미는 언젠가 또 다시 한번 드러나지 않을까 생각해. 너는 우리 중에 가장 열심히 하니까 정말 좋은 의사가 될 거라 믿어 의심치 않는다. 그리고 학교 다닐 때에는 너무나 착해서 연애는 잘할 수 있을까, 직장생활은 잘 할 수 있을까 늘 걱정이 되었었던 지승이. 보니까 회사에서 제일 잘나가는 것 같고, 오히려 회사 체질인 것 같아 참 다행이라고 생각해. 우리 중에 덕질이 가장 뚜렷하니 왠지 우리 중에 가장 성공할 것 같다는 생각도 가끔 들어. 그리고 지금은 힘든 시기를 겪고 있는 용승이. 능력은 출중하지만 멘탈이 약해서 정말 많이 걱정이 돼. 스스로를 저평가하지 말고 할 수 있다는 생각을 늘 가지길 바라. 너도 힘든 학부생활을 잘 마무리 했는데 다른 일들도 다 잘할 수 있을 거야. 많이 응원할게. 컨디 08 동기들 외에도, 제가 힘들 때 버팀목과 같은 존재였던 컨디

사람들. 진연이형, 은구형, 건우형, 영진이형, 꾸이누나, 인성, 인오, 동헌, 진성, 숙진, 성우, 예슬, 준혁, 호철 등등 모두 감사드립니다. 일일이 모두 언급하기에는 너무나도 많아서 다 언급하지 못하는 점 정말 죄송하게 생각합니다.

다른 집 아들딸들은 이미 취직하여 각자 부모님들께 효도하고 있는 것을 비교해보면, 많이 섭섭하실 수도 있음에도 불구하고 가방 끈 긴 아들인 저를 지금까지 응원해 주시고 아낌없는 사랑을 보여주신 부모님 감사합니다. 이제 드디어 저도 박사 학위를 받았으니 효도 많이 하도록 하겠습니다. 부모님과 함께 박사학위 수여식을 보낼 생각을 하니 벌써 기분이 좋네요. 그리고 멀리 돌아왔지만 묵묵히 자기 길을 가고 있는 동범아. 지금까지 오는 데 쉽지 않은 것을 누구보다 잘 알지만 그 동안 많이 응원해 주지 못해서, 좀 더 형 노릇을 잘 못해줘서 미안한 마음이 크다. 좀 더 잘 챙겨주도록 할게. 그리고 너가 하고자 하는 모든 일이 잘 되길 응원할게. 앞으로 서로 연락하고 볼 날들이 많으니 남부럽지 않은 형제가 되어 부모님께 효도하자.

마지막으로, 석박사 통합과정 중 가장 중요한 시기에 좋은 인연으로 많은 힘이 되어 준 지헌아. 회사 일에 바쁜 와중에도 너의 깊은 배려심과 이해심은 내가 박사 과정을 잘 마무리 하는데 큰 원동력이 된 것 같아. 앞으로 함께할 날이 많은데 서로 좋은 추억 많이 만들자. 고맙고 사랑해.

Curriculum Vitae

Name : Donghyun Kim

Education

2012. 3. - 2017. 8. Department of Computer Science and Engineering,
Pohang University of Science and Technology (M.S. & Ph.D.)

2008. 3. - 2012. 2. Department of Computer Science and Engineering,
Pohang University of Science and Technology (B.S.)

Experience

2013. 9. - 2014. 2. Research Intern (full time)
Microsoft Research Asia, Beijing, China

Publications

1. Chanyoung Park, **Donghyun Kim**, Jinoh Oh and Hwanjo Yu, "Do "Also-Viewed" Products Help User Rating Prediction?", WWW 2017 *(full paper, 17% accept, **oral presentation**)*.
2. **Donghyun Kim**, Chanyoung Park, Jinoh Oh, Sungyoung Lee and Hwanjo Yu, "Convolutional Matrix Factorization for Document Context-Aware Recommendation", ACM RecSys 2016 *(full paper, 18.2% accept, **oral presentation**)*.

3. Chanyoung Park, **Donghyun Kim**, Jinoh Oh and Hwanjo Yu, "TRecSo: Enhancing Top-k Recommendation with Social Information", WWW 2016 *(poster)*.

4. Yejin Kim, Yong Hyun Park, Ji Youl Lee, In Young Choi, **Donghyun Kim** and Hwanjo Yu, "Discovery of Prostate Specific Antigen Pattern to Predict Castration Resistant Prostate Cancer of Androgen Deprivation Therapy", ACM DTMBIO 2015.

5. Chanyoung Park, **Donghyun Kim**, Jinoh Oh and Hwanjo Yu, "Predicting User Purchase in E-commerce by Comprehensive Feature Engineering and Decision Boundary Focused Under-Sampling", ACM RecSys Challenge 2015 *(10th place, **top 1.7%**)*

6. Chanyoung Park, **Donghyun Kim**, Jinoh Oh and Hwanjo Yu, "Improving Top-K Recommendation with Truster and Trustee Relationship in User Trust Network" *Information Sciences* 2016 *(SCI, IF: 4.832)*.

7. **Donghyun Kim**, Chanyoung Park, Jinoh Oh and Hwanjo Yu, "Deep Hybrid Recommender Systems via Exploiting Document Context and Statistics of Items", *Information Sciences* 2017 *(SCI, IF: 4.832, To appear)*.

8. Chanyoung Park, **Donghyun Kim**, Min-Chul Yang, Jung-Tae Lee and Hwanjo Yu, "Your Click Knows It: Predicting User Purchase through Improved User-Item Pairwise Relationship", *Submitted to ACM CIKM 2017*

9. **Donghyun Kim**, Chanyoung Park and Hwanjo Yu, "Hybrid Autoencoder for Recommender Systems" *Preparation for AAAI 2018*.

10. **Donghyun Kim**, Chanyoung Park and Hwanjo Yu, "Enhancing Autoencoders for Recommender Systems via Gradient Unweighted Layer" *Preparation for WWW or ICML 2018*.


POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
1986
포항공과대학교