Collaborative Methods with Multiple Key Components and Domains for Recommender System

TOHOKU UNIVERSITY

DOCTORAL THESIS

Collaborative Methods with Multiple Key Components and Domains for Recommender System

---

Author:
Nguyen Thi Thuy Linh

Supervisor:
Prof. Tsukasa ISHIGAKI

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Data Science and Service Research
Graduate School of Economics and Management

July 13, 2020

Declaration of Authorship

I, Nguyen Thi Thuy Linh, declare that this thesis titled, "Collaborative Methods with Multiple Key Components and Domains for Recommender System" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

TOHOKU UNIVERSITY

Abstract

Graduate School of Economics and Management

Collaborative Methods with Multiple Key Components and Domains for Recommender System

by Nguyen Thi Thuy Linh

Along with the convenience offered by increased use of the internet, people have gradually changed their habits. For instance, they shop online using e-commerce sites instead of going to stores. They watch movies on Netflix and YouTube as alternatives to going to a cinema. However, because information has propagated expeditiously, users have difficulty finding the items they want. Often, only a few items are visible to users while others are buried in a long-tailed list. For that reason, many recommender systems (RS) exist. My research addresses their problems and provides solutions based on deep learning models.

The first challenge of an RS is suggesting interesting items to new users. To do so, an RS needs some interactions among users and items to occur. Hence, the system encounters serious obstacles with new or inactive users. To overcome this problem, modern RS tend to use as much information as possible. This trend was borne out of the increasing number of studies on hybrid methods that combine rating and auxiliary information. However, because of privacy concerns, in many cases, service providers can not require users to give their personal information. Therefore, numerous earlier reported methods only use item attributes for auxiliary information. To address these shortcomings, my manuscript provides a method to extract user profiles without using demographic data. My model learns user and item latent variables through two separate deep neural networks and also infers implicit relations between users and items using the information and their ratings.

To deal with the lack of interactions among users and items and improve accuracy, RS tend to combine numerous kinds of information. Nevertheless, many useful data, such as item descriptions, items' images or even transactions themselves, are unstructured, and traditional methods can not extract latent vectors effectively. Hence, how to obtain valuable information from unstructured data as well as how to integrate them into a single system has become the second challenge of RS. Recently, deep learning models have made a big step in extracting latent vectors of unstructured data and demonstrate their power in many applications from computer vision and natural language processing to RS and bioinformatics. My solutions are based on deep learning models to obtain better representation of user behavior and item description.

The third challenge is that RS are mainly based on user interaction history, sometimes, suggestions only involve domains where the user interacted, which make user be tedious. To address this problem, I propose a cross-domain model that can suggest items in the other domains where the user even does not have any interaction. My domain-to-domain translation model (D2D-TM), which is based on generative adversarial network (GAN) and variational autoencoder (VAE), uses the user interaction history. Domain cycle consistency (CC) constrains the inter-domain relations.

Acknowledgments

First, I would like to express my deepest gratitude to my supervisor, Prof. Ishigaki, who supported me a lot even before I entered Tohoku University. He taught me the methodology to carry out the research and how to present it as clear as possible. He was always ready with useful comments whenever I needed them not only with research, but also with other problems in studying. Without his guidance and persistent help, this dissertation would not have been possible.

Exceptional gratitude goes out to all down at Data Science Program (DSP) and Global Program of Economics and Management (GPEM) for giving me a chance to study at Tohoku University. Staff members in the two programs always supported me in both office works and student life.

Last but not least, I would like to thank my family and all of my friends who always encouraged me to go on.

Thanks for all your support!

Contents

List of Figures

List of Tables

List of Abbreviations

RS	**R**ecommender **S**ystem
AE	**A**uto**E**ncoder
VAE	**V**ariational **A**uto**E**ncoder
DAE	**D**enoising **A**uto**E**ncoder
GAN	**G**enerative **A**dversarial **N**etwork
CML	**C**ollaborative **M**ulti-key **L**earning
NeuCML	**Neu**ral **C**ollaborative **M**ulti-key **L**earning
D2D-TM	**D**omain-**T**o-**D**omain **T**ranslation **M**odel
MLP	**M**ultiple **L**ayer **P**erceptron
KL	**K**ullback leibler **L**eibler

List of Symbols

x	A scalar
$\mathbf{x}$	A vector
$\mathbf{X}$	A matrix
$a(.)$	an activation function

Chapter 1

Introduction about Recommender System

1.1 Recommender System

Along with the convenience offered by increased use of the internet, people have gradually changed their habits. For instance, they shop online using e-commerce sites instead of going to stores. They watch movies on Netflix and YouTube as alternatives to going to a cinema. However, because information has propagated expeditiously, users have difficulty finding items they want. Often, only a few items are visible to users while others are buried in a long-tailed list. For this reason, many recommender systems (RS) exist, that have become important in e-commerce or shared platforms. Everyone can see RS-based phenomenon easily when using the Internet. For example, YouTube automatically moves to videos related to the video that the user played when it ends or suggests videos that the user may like. Amazon suggests products you may concern and divides them into categories such as "Related to items you've viewed" or "People who bought this product also bought these items", and Facebook, Twitter or LinkedIn suggests friends, or posts.

Many big technology companies reported the importance of RS in their service systems. Amazon reported a 29% sales increase to $12.83 billion during its second fiscal quarter, up from $9.9 billion during the same time last year (Fortune.com, 2012)[1]. McKinsey estimated that 35% of Amazon.com's revenue is generated by its recommendation engine. They also estimated that 75% of what customers watch on Netflix comes from product recommendations[2]. Following Christopher Johnson – an machine learning engineer in Spotify – the new recommender system has helped Spotify increase its number of monthly users from 75 million to 100 million at a time, despite competition from rival streaming service Apple Music. According to YouTube, the implementation of an RS for more than a year, has led to successful results, with recommendations accounting for around 60% of video clicks on the homepage.

Traditionally, researchers and marketers have spent much effort in segmenting customers [23, 21]. Customers and products are divided into different groups so that a group of customers can be match to a suitable group of products to enhance purchase amounts. However, the relationship between customers and products is complicated, especially in an extensive system. Therefore, to provide better suggestions to the individual customer, both online and offline systems need to implement recommender systems. The advantages and disadvantages of traditional marketing

[1] https://fortune.com/2012/07/30/amazons-recommendation-secret/

[2] https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers.

	Marketing Segmentation	Recommender System
Data	• Customer demographics • Product information (price and category)	• Rating information (implicit and explicit feedback) • Content information (user and item heterogeneous information such as text, image and structural data)
Main Characteristics	• Grouping customers according to marketing segments • Grouping the products in categories that can be aligned with marketing segments • Encouraging customers indifferent segments to purchase products from categories selected by the marketer	• Interacting with individual user • Suggesting top-k items to the user • Helping the user find products they would like to purchase
Advantages	• Good at small data • Be possible to give explanation	• End-to-end automatic suggestion • Being able to combine many types of information to achieve high performance
Disadvantages	• Handling with only limited datasets and data types • Impossibility of extracting individual customer behavior	• Need much data

TABLE 1.1: Comparison between Marketing Segmentation and Recommender System

segmentation and recommender system are listed in Table 1.1. Information types used in RS are thoroughly explained in Section 1.1.1.

1.1.1 Primary Objects in Recommender System

In RS, there are two main objects: the user and the item. The user can be a customer or just a user who performed some actions in the system. An item is an object that receives users' actions. Items range from products in e-commerce systems and songs in online music to other users in social networks. Besides the two main objects, there are two more important types of information used in RS: rating and auxiliary information.

Rating information is the interaction history that a user gave to items, which is extremely important with an RS, as it supports RS to outperform traditional marketing segmentation. Based on rating information, systems can know what each user likes and how they feels, which allows for better learning of user behavior. Rating information can be obtained by two types of feedback: implicit or explicit. Explicit feedback is an assessment that users actively give to items in the form of rating scores or reviews. Reviews directly present how users feel about items. However, the number of reviews in systems is limited because it takes much time to write a review. Therefore, RS try to create their website so that users only need a click to give ratings. Besides the limitation of explicit feedback, RS can collect huge implicit feedback. With implicit feedback, rating between a user and an item will be 1 if this user had interactions with that item such as view, like or purchase. Otherwise, the rating will be 0. Implicit feedback may contain much information even more than explicit feedback. For example, before an user purchases an item, they will consider a bundle of related items. Based on this information, systems may know which elements the user considered the most such as price or quality. However, implicit feedback is massive and noisy, which makes obtaining useful information from it challenging.

Auxiliary information includes user and item information. Auxiliary information is mainly used in marketing segmentation, but current RS widely differ from marketing segmentation. Traditionally, only structural information of items such as genres and categories is used. However, thanks to new techniques such as deep learning, RS can extract latent features from unstructured information such as item's images or text descriptions to support the model. Concerning user information, marketing segmentation usually uses customer demographics, including sensitive information such as income. However, most users are unwilling to give their information except basic necessary ones such as age or address. Therefore, previously, most RS models ignored user information and only used item information. Recently, researchers have attempted to build user information based on user interactions.

There are some other types of information supported for improving performance of RS such as knowledge and geography. However, they depend on the task and purpose of RS.

1.1.2 Goals of Recommender System

According to [1], there are two primary models:

- *Rating prediction*: it predicts rating for a combination of user-item. The learning algorithm attempts to complete an incomplete $m \times n$ rating matrix that corresponds to rating scores that m users give to n items.

- *Ranking prediction*: it gives a list of top-k items in which user may be particularly interested. In reality, users may want to receive a list of interesting items, rather than predicted rating for a specific item.

Increasing product sales is the primary goal of an RS [1]. To achieve this, first of all, an RS needs to predict the most relevant items to individual users. However, to reach the broader business-centric goal of increasing revenue, the other common operational and technical goals of RS are the following:

- *Novelty*: users may know popular items without the system's support. Therefore, suggesting unpopular items is surely helpful in enhancing sales diversity as well as enriching users' interest.

- *Serendipity*: if a system can suggest items that truly surprise users, merchant can benefit from increasing sales diversity and discover new areas of users' interest.

- *Diversity*: if suggested items belong to different types or domains, there is a high probability that users are interested at least in one of them. The higher the diversity that system gives, the lower the chance that a user gets bored by repeated similar items.

1.1.3 Basic Models

The two main models are: collaborative filtering and content-based which based on two main information types: rating and content, respectively. There are many other types based on information such as knowledge-based and domain-based, but collaborative filtering and content-based are the most important.

Collaborative Filtering Models

Collaborative filtering (CF) models are mainly based on rating information. They include:

- Neighborhood-based: models that work with the assumption that if two users have similar history interaction, they have a high probability to have same taste, so that the user will like the items with which the other interacted.
- Model-based: models that attempt to construct user and item vectors from a rating matrix, and then the rating matrix is filled out by multiple user vectors to item vectors. In the first RS models, matrix factorization and singular value decomposition (SVD) are widely used. However, many deep learning models have recently been applied to obtain better representation vectors.

Content-Based Models

In content-based models, auxiliary information is used to extract user and item vectors. Content-based methods have some advantages in making recommendations for new items, when sufficient rating data are not available for that item. Traditionally, CF-based models can achieve higher performance than content-based methods and suggest surprisingly relevant items. However, recently, thanks to deep learning techniques, which are good at extracting latent vectors from unstructured data, content-based methods are necessary in many cases such as fashion or music recommendations.

Hybrid Models

Each model has its own advantages and disadvantages. Therefore, to achieve better performance, researchers tend to combine two or more methods. Based on how these methods are combined, hybrid methods are included:

- *Loosely hybrid methods*: component methods are optimized separately.
- *Tightly hybrid methods*: component methods are optimized together.

1.2 My Contribution

Many recommender system models focus on suggesting items to customers when they interacted with a bundle of items. However, when customers cannot find what they want in our system or interesting items are not suggested to them during their first visits, they may leave immediately. Hence, systems can lose many potential customers and incur in extra marketing expenses. My work focuses on giving better suggestions for new customers, including new systems.

If customers have interactions in a single domain only, and based on these interactions, system merely suggests items in this domain, customers may soon feel indifferent. Hence, besides the current domain, my work also recommends items in different domains that surprise customers. It is possible to not only keep customers stay longer in our system but also bring more profit.

In summary, my research draws attention to making new customers become frequent customers by suggesting items appropriate to current customer situation.

1.2.1 Cold Start and Data Privacy Problem

Cold Start Problem

In the winter, the extremely cold temperature makes cars' engine difficult to start up. Much engine is needed to warm them up and once they reach their optimal operating temperature, they will run smoothly. The cold start problem in recommender systems is similar. The more user and item information a system has, the easier it is for it to suggest relevant items. However, if a system gathers insufficient information, recommending become problematic, which is called the cold start problem.

In recommender systems, the most important information is rating; hence collaborative filtering methods are usually better than content-based methods only. However, collaborative filtering methods work well in the assumption that every user interacted with some items, and every items received some interactions from users. Therefore, cold start happens when users and items have scarce interaction in RS platforms. They can be new users, new items or inactive users and unpopular items.

With new or unpopular items, the standard solution is using a hybrid method that combines rating information and item information. However, with new and inactive users, because of privacy rules, RS usually ignore this problem and suggest the most popular items to them. However, these uninteresting suggestions can make new users leave our system or make users inactive. Therefore, solving the cold start problem for users is necessary to enhance both the number of users and profit for the platform.

Data Privacy Problem

To solve the cold start problem, a hybrid method, which uses both rating information and auxiliary information of both users and items, is helpful. When users register on an RS, they need to accept rules that allow the system to collect their history interactions such as click or purchase, which are needed for system services. However, users are unwilling to provide personal information that is unrelated directly to the services, such as income, age, or family members. Furthermore, RS have many difficulties in using user information gotten from a third party because privacy rules are strict. In auxiliary information, a user profile is a sensitive problem that demands careful utilization to avoid privacy violations. According to [2], privacy is regarded as "the right of a person to determine which personal information about himself/herself might be communicated to others". This right also is regulated in the privacy laws of many countries. For instance, Australia Privacy Laws[3] stipulate the following:

- Individuals must have the option of not identifying themselves, or of using a pseudonym when dealing with an Australian Privacy Principle (APP) entity in relation to a particular matter (Australian Privacy Principle 2.1).

[3]https://www.oaic.gov.au/individuals/privacy-fact-sheets/general/privacy-fact-sheet-17-australian-privacy-principles

- If an APP entity is an agency or organisation, then the entity must not collect personal information (other than sensitive information) unless the information is reasonably necessary for, or directly related to, one or more of the entity's functions or activities (Australian Privacy Principle 3.1, 3.2).

Following these rules, service providers can provide only anonymized data to a third party. Although the data are private, they are still desirable because they allow for aggregate analysis [9]. Examples are provided by manufacturers who want to know market-shares among their products and other competitors or researchers want to study marketing methods. These problems are readily solved by publishing raw data. However, such publication will violate privacy rules, as discussed above. Therefore, before publishing data, a provider must apply some privacy-preserving algorithms such as k-anonymity so that an entity in a dataset cannot be re-identified. K-anonymity is a grouped method by which every tuple in the private table being released is indistinguishably related to no fewer than k respondents [2]. However, even when using these algorithms, demographic data are still vulnerable if attackers make inferences from private information such as age, career, and zip code. If a company violates the privacy rules, it will become an important scandal that can blow out much of its values. For example, the recent scandal in which Facebook provided data of more than 50 millions users to Cambridge Analytica – a British political consulting firm – without their permission made the shares of this company drop by almost 40% [4]. Therefore, RS must avoid privacy violations.

My Solution

To solve the cold start problem while not violating privacy rules, my research provides an embedding method to extract user behavior from rating information without requiring any extra demographic data, which is called Collaborative Multi-key Learning (CML) [35]. Then, I two deep learning models based on variational autoencoder are suggested to capture user key vector and item key vector from user behavior and item description, respectively. Finally, using these two key component vectors, my suggested model is able to learn implicit relations between items and users concomitantly through a probabilistic generative model with neural networks. Experiments on real-world datasets demonstrate that my proposed model significantly outperforms the state-of-the-art baselines. Specially, my model provides high performance with a large margin in the cold start problem.

1.2.2 Matrix Factorization Problem

In previous work, I used matrix factorization for rating information. Matrix factorization breaks the rating matrix into two component matrices: user latent matrix and item latent matrix. However, the relationship among users and items are complicated; hence, matrix factorization will not work well in case of few interactions in the matrix.

My Solution

Instead of matrix factorization, I propose a neural network model in rating information because one of its advantages is that it can learn complex problems especially with unstructured data as rating information.

[4] https://www.cnbc.com/2018/11/20/facebooks-scandals-in-2018-effect-on-stock.html

There are nine main neural network structures, and I found that autoencoder (AE) is the most suitable for my purpose. AE approaches have recently become the most used methods to highlight latent vector. One advantage of AE models is that they learn the interest of users given to all items at the same time. Based on this, it is possible to highlight the relationship among items which makes it possible to archive high performance even for new users. However, it makes AE models hard to combine with content information. Therefore, my model provides a solution to combine both rating information and content information in AE approaches.

1.2.3 Tedious Suggestion Problem

Diversity and Serendipity

While the board business goals of RS include finding items that users will like most, suggesting them to users and enhancing profit, the core engine of RS is based on rating and content information to suggest the most related items to user. If user only focuses on a domain, there are a high chance that RS will pick the most similar items in the next suggestions. When all these recommended items are remarkably similar, the risk increases that the user might not like any of these items [1]. For example, if a user just bought a guitar, it may be impossible that they will buy another one from another shop. Tedious suggestions not only make users feel indifferent but also decrease the profits of providers in these platforms.

Therefore, to enhance profit and keep user to use system continuously, recommended items should belong to different types or different domains. Recommending items that are different types or out of the domain scope ensures that the user does not get bored by repeated recommendations of similar items and supports for cross-selling to raise the profit [1].

My Solution

My research aims to suggest interesting items that surprise users; then through surprise suggestions, my model can enhance cross-selling for providers. For example, if a user bought a protein supplement product in the health care product category, the system can suggest a sports outfit in the clothing category because when they wants to build their muscle, it is possible that they exercise frequently. To do that, I propose a cross-domain RS method.

A system contains a huge number of items across different categories. If a system makes a suggestion based on a user-item matrix of all items, computation costs will be high, and sometimes be impossible to sustain. Therefore, it is necessary to divide the whole dataset into smaller domains and to make suggestions for each single domain.

A domain is a particular field of thought, activity or interest [6]. Based on their different attributes, items can be divided into smaller domains following many levels:

- Attribute level: items are the same type and have different values in specific attributes. (i.e., drama and comedy movies, only different in genres).
- Type level: items are the same type but have differences in almost attributes (i.e., health care products and clothes in e-commerce system).
- Item level: items are distinct types (i.e., movies and products in E-commerce system).

- System level: items are almost the same but are collected in different ways or different operators (i.e., items in Netflix and Movielens are movies, but are collected in different platforms).

Therefore, if recommendation lists are included in different domains, tedious problem will be solved. In addition, cross-domain or multi-domain methods can solve other disadvantage of single-domain. For example, users usually only have interactions in some domains. Hence, with other domains, they do not have any interaction, which makes it difficult to give useful recommendations in such domains.

My model is called as domain-to-domain translation model (D2D-TM) [36], which based on variational autoencoder (VAE) and generative adversarial network (GAN) to extract homogeneous and divergent features from domains. Domain cycle consistency (CC) constrains the inter-domain relations. The experiments indicate that simply with a set of interaction history in a user's domain, D2D-TM not only boosts the prediction results of the domain, but also infers items in other domains with high performance. Therefore, it can solve both the tedious suggestion problem as well as the cold start problem.

Chapter 2

Deep Learning Techniques for Recommender System

A neural network is a model inspired by how brain works and enables a computer to learn from observation data as human. Along with the Digital Revolution which enriches data sources and the innovation of computer, deep learning has recently become a powerful set of techniques for learning in neural networks, and has widely demonstrated its powerful in many applications:

- Computer vision: object detection, face recognition, auto-driving, etc.
- Natural language processing: text analysis, speech recognition, translation, etc.
- Recommender system, bio-informatics, etc.

Deep learning allows not only for powerful performance but also the attractive learning feature representation from the scratch. In the next part, I demonstrate the basic concepts of deep learning, and a model frequently used in the present research: variational autoencoder (VAE). VAE are widely applied in RS to obtain latent vectors of both auxiliary and rating information. In the last part of this section, I introduce some recent studies that use VAE and achieve high performance.

2.1 Basic Concepts

Figure 2.1 represents the general structure of a neural network. In a neural network, numeric data points, called inputs, are fed into the neurons in the input layer. Each

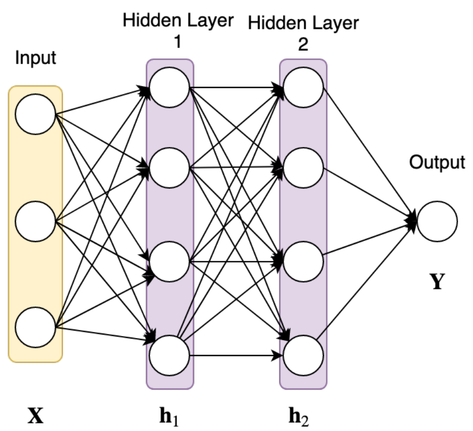


FIGURE 2.1: General Structure of Neural Network

neuron in a layer is multiplied with weights and then gives outputs of the neuron, which is transferred to the next layer. There are three types of general component layers: input layer, some hidden layers and output layer. To understand about neural network more in depth, first I start with the following linear regression:

$$f(\mathbf{x}, \mathbf{w}) = w_0 x_0 + w_1 x_1 + \cdots + w_m x_m = \begin{bmatrix} \mathbf{x} & 1 \end{bmatrix} \times \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} = \mathbf{xw}$$

Each neural can be considered as a linear regression model. Then a hidden layers with n neurons will be:

$$f(\mathbf{X}, \mathbf{W}) = \mathbf{XW} = \begin{bmatrix} f_0 \\ \vdots \\ f_n \end{bmatrix}$$

However, to create complex mappings between the network's inputs and outputs, each neural is wrapped by a non-linear activation functions, so that network can learn and model complex data, such as images, video, audio, and datasets which are non-linear or have high dimensions.

Therefore, the first hidden layer will be $\mathbf{h}^{(1)} = a^{(1)}(f(\mathbf{X}, \mathbf{W}))$ while $a(.)$ is activation function.

The i^{th} hidden layer will be $\mathbf{h}^{(i)} = a^{(i)}(f(\mathbf{h}^{(i-1)}, \mathbf{W}^{(i)}))$

Then the output will be $Y = a^{(\ell+1)}(f(\mathbf{h}^{(\ell)}, \mathbf{W}^{(\ell+1)}))$ where ℓ is the number of hidden layers.

The hidden layer thus calculated is called a fully connected layer. There are two other important layer types: convolutional layer and recurrent layer. While the convolutional layer is widely used for image processing, the recurrent layer outperforms in text or speech processing.

Deep learning model or deep neural network is a neural network with many hidden layers. Traditionally, neural networks can have one or two hidden layers. However, recent deep learning models can have more than 150 hidden layers.

2.1.1 Activation Function

The activation function is a mathematical "gate" between two layers. It can be considered as a transformation that converts values of neurons in current layers into needed range such as $[0, 1]$ or $[-1, 1]$. Furthermore, it can work as a switch to turn the neurons on or off.

In a deep learning network, there are four non-linear activation function which are used most frequently: sigmoid, tanh, relu and leaky relu. Function formulas as well as advantages and disadvantages of the four activation functions are presented in Table 2.1 and Figure 2.2

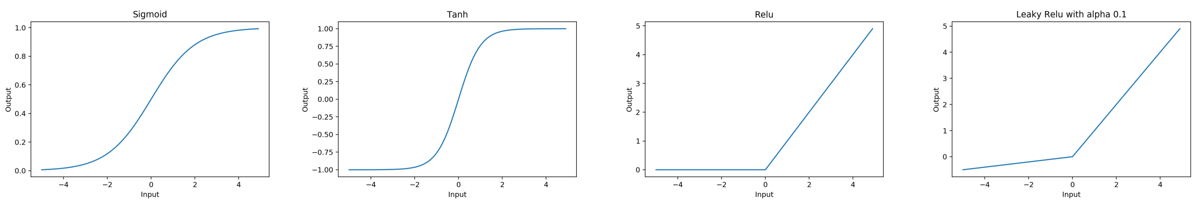
2.2 Variational Autoencoder (VAE)

VAE belongs to a family of AE models. AE aims to represent (code) for a set of data in an unsupervised manner by training the network to ignore signal "noise". Figure 2.3a represents the general structure of an AE model. AE usually includes two parts:

- Encoder: $\mathbf{h} = f(\mathbf{x})$ with $\mathbf{h}$ representing a set of input $\mathbf{x}$.

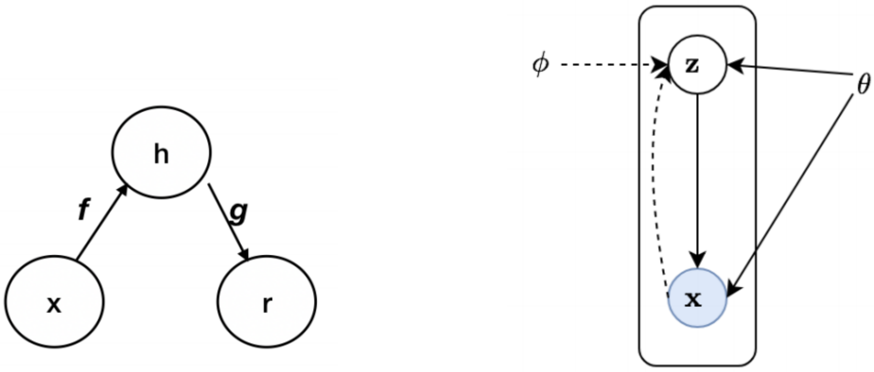
Activation Function	Advantages	Disadvantages
Sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$ Output: $[0, 1]$	• Smooth gradient. • The output of each neurons is normalized • Clear predictions	• Vanishing gradient: prediction is almost no change for very high or very low values of input. As a result, the network refuses to learn further and reach an accurate prediction slowly. • Computationally expensive • Outputs are not zero centered
Tanh $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ Output: $[-1, 1]$	• Smooth gradient • Normalized outputs and clear predictions following zero centered.	• Vanishing gradient • Computationally expensive
Relu $relu(x) = max(0, x)$	• Computationally efficient and non-linear: network can be quickly converged	• The dying ReLU problem: when inputs are not positive, the output of the relu function becomes zero; backpropagation thus cannot perform.
Leaky Relu $lrelu(x) = max(\alpha x, x)$	• Prevent the dying ReLU problem by keeping a small values for negative inputs which enables backpropagation. • Computationally efficient and non-linear	• Results are not consistent—leaky ReLU does not provide consistent predictions for negative input values.

TABLE 2.1: Advantages and Disadvantages of Activation Functions



(A) (B) (C) (D)

FIGURE 2.2: Activation Functions



(A) AE (B) VAE

FIGURE 2.3: General Structure of AE and VAE

- Decoder: Decoder: $\mathbf{r} = g(\mathbf{h}) = g(f(\mathbf{x}))$ with $\mathbf{r}$ is reconstruction of $\mathbf{x}$. AE tries to make $\mathbf{r}$ close as possible to $\mathbf{x}$ based on representation $\mathbf{h}$.

To obtain a representation vector $\mathbf{h}$, AE models need to minimize the loss function $\mathcal{L}(\mathbf{x}, \mathbf{r}) = \mathcal{L}(\mathbf{x}, g(f(\mathbf{x})))$. Dimension of $\mathbf{h}$ is usually much smaller than $\mathbf{x}$ to avoid becoming copy-paste function.

2.2.1 VAE Structure

Variational autoencoder (VAE) [24] is a probabilistic AE. The general structure of VAE is presented in Figure 2.3b. Unlike other AE models, the latent variable $\mathbf{z}$ is not generated directly by input, but is instead sampled from some prior distribution $p_\theta(\mathbf{z})$ with parameter set θ. The output is then generated from some conditional distribution $p_\theta(D|\mathbf{z})$, where D represents input data. Therefore, VAE can learn significant features and generate new instances that appear to have been sampled from the training set.

However, the true posterior $p_\theta(\mathbf{z}|D)$ is intractable, especially with continuous variables. Similarly to [24], we seek parameter set ϕ so that variational inference $q_\phi(\mathbf{z}|D)$ is approximate with the true posterior $p_\theta(D|\mathbf{z})$. To measure the quality of this approximation, we can use Kullback–Leibler divergence $\mathbb{KL}$ between the approximate and exact posteriors. Then, the problem becomes maximizing the lower bound $\mathcal{L}(\theta, \phi; D)$ as indicated below:

$$\mathcal{L}(\theta, \phi; D) = \mathbb{E}_{q_\phi(\mathbf{z}|D)}[\log p_\theta(D|\mathbf{z})] - \mathbb{KL}(q_\phi(\mathbf{z}|D)||p_\theta(\mathbf{z})) \tag{2.1}$$

2.2.2 VAE in Deep Neural Network

VAE in a deep neural network is called stacked variational autoencoder or simply SVAE. SVAE usually has a symmetric structure. As Figure 2.4 illustrates, hidden layer 1 has the same number of neurons as hidden layer 4, hidden layer 2 has the same number of neurons as hidden layer 3, and input has the same number of neurons as output.

In a deep learning network, to make training with back-propagation possible, a reparameterization trick [24] is applied to express a random variable $\mathbf{z}$ as a deterministic variable $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$, where $\boldsymbol{\mu}$ is a mean vector and $\boldsymbol{\sigma}$ is a vector that

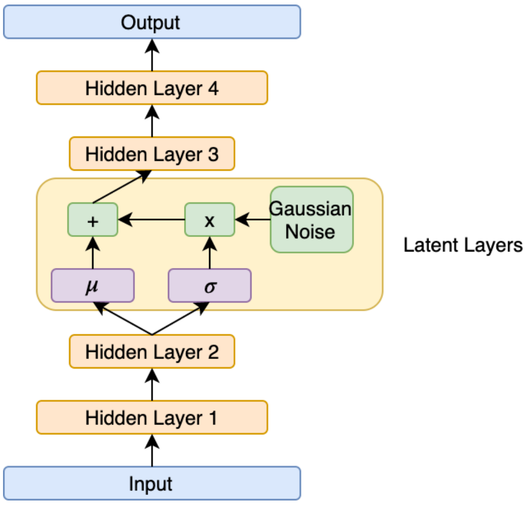


FIGURE 2.4: Network Structure of Stacked Variational Autoencoder

consists of a diagonal component of the covariance matrix. Both $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are outputs of the encoder network with input $\mathbf{x}$, denoted by $E(\mathbf{x})$. Furthermore, $\odot$ signifies an element-wise product; $\boldsymbol{\epsilon}$ is generated from a Gaussian distribution $\mathcal{N}(0, I)$ with I as the identity matrix. However, $\mathbf{x}_{rec}$ will be the output of the generator network with input $\mathbf{z}$ as $\mathbf{x}_{rec} = G(z)$.

It is noteworthy that VAE training is aimed at minimizing a variational upper bound, which is

$$\mathcal{L} = \mathbf{KL}(q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})) - \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z})] = \mathcal{L}_{KL} + \mathcal{L}_{rec}, \tag{2.2}$$

$$\begin{aligned} \text{with} \quad & \mathcal{L}_{KL} = \mathbf{KL}(q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})), \\ \text{and} \quad & \mathcal{L}_{rec} = -\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z})], \end{aligned}$$

where **KL** is the Kullback–Leibler divergence.

From now, to be simpler, I will call stacked variational autoencoder "variational autoencoder" or VAE.

2.3 VAE in Recommender System

As illustrated in Chapter 1, there are two main kinds of information in recommender systems: rating and content. VAE can extract important features of both information types, that provide give high performance, as proved by many pieces of research.

2.3.1 VAE for rating information

Muli-VAE [31] proposed a variant of VAE for recommendation with implicit data. The authors introduced a principled Bayesian inference approach for parameters estimation and demonstrated the advantages of multinomial likelihood function for click vectors compared with commonly used functions such as Gaussian or log likelihood. Multi-VAE only considers implicit user feedback – namely an input of Multi-VAE is a vector represented for a user. The length of the vector equals to the number of items. Each item is presented by a neuron. If the user has an interaction with an item, its neuron in the vector will be 1 and be 0 if vice versa. Multi-VAE structure

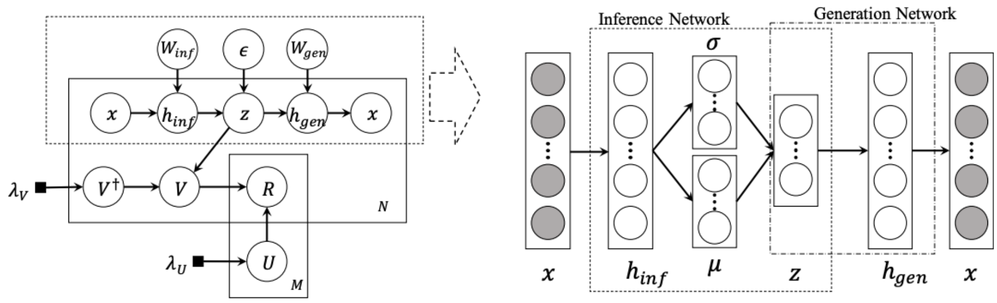


FIGURE 2.5: Collaborative Variational Autoencoder for Recommender System

is the same as Figure 2.4, in which output is the probabilistic that the user will have interactions in each neuron.

2.3.2 VAE for Content Information

Collaborative variational autoencoder (CVAE) [27] is a hierarchical Bayesian model which integrates stacked variational autoencoder (VAE) into probabilistic matrix factorization (PMF). While VAE focuses on extracting latent representation of item information, PMF concentrates on the relationship between users and items through interaction history. VAE and PMF are tightly combined, which enables CVAE to balance the influences of side information and interaction history. Figure 2.5 illustrates the graphical model of CVAE and its generative process is as follows:

- For each layer l of the generation network
  - For each column n of the weight matrix W_l, draw: $W_{l,*n} \sim \mathcal{N}(0, \lambda_w^{-1} I_{K_l})$
  - Draw the bias vector $b_l \sim \mathcal{N}(0, \lambda_w^{-1} I_{K_l})$
  - For each row j of h_l, draw $h_{l,j*} \sim \mathcal{N}(\sigma(h_{l-1,j*} W_l + b_l), \lambda_s^{-1} I_K)$
- For each user i, draw the latent variable $u_i \sim \mathcal{N}(0, \lambda_u^{-1} I_K)$
- For each item j
  - Draw prior distribution of the content variable, chosen to be a unit Normal distribution: $z_j \sim \mathcal{N}(0, I_K)$
  - Draw a latent offset $v'_j \sim \mathcal{N}(0, I_K)$
  - Draw latent variable of item as $v_j = v'_j + z_j$
- Draw a rating r_{ui} for each user-item pair (u,i), $r_{ui} \sim \mathcal{N}(U_u^T V_i, C_{ui}^{-1})$

where W_l and b_l are the weight matrix and biases vector for layer l, X_l represents layer l. λ_w , λ_s, λ_n, λ_v, λ_u are hyperparameters, C_{ui} is a confidence parameter for determining the confidence to observations.

Chapter 3

Collaborative Multi-Key Learning with an Anonymization Dataset for a Recommender System

3.1 Introduction

Existing RS methods can be categorized roughly into three classes [3]: content-based methods, collaborative iltering (CF) based methods and hybrid methods. Content-based methods [44, 37, 50] use auxiliary information such as user profiles or item descriptions to identify and recommend relevant items to users. Alternatively, CF-based methods [17, 31, 53] use a history view or buying patterns of users, so-called rating information, to calculate similarity among users and users or among items and items. They then suggest similar items to a user or suggest items that a similar user has sought or bought. Generally, CF-based methods can achieve higher performance than content-based methods and can suggest surprisingly relevant items. Nevertheless, their performance is low in cases of sparse data or a cold start [40], whereas content-based methods can accommodate users. Therefore, recently, hybrid methods [30, 58, 8], which are a combination of collaborative and content information, have gained popularity.

Rating information is extremely important with an RS. As I mentioned before, rating information can be feedback of two types: implicit or explicit. In typical explicit feedback, a user will provide ratings for items on a Likert scale [22], with or without a review. Although explicit feedback can be negative or positive, implicit feedback is only positive. With implicit feedback, rating between a user and an item will be 1 if this user had interactions to that item such as view, like or purchase. Otherwise, the rating will be 0. Therefore, explicit feedback might represent user behavior better than implicit feedback. In attempting to improve performance, a recommendation system will try to collect feedback that is as explicit as possible. However, with explicit feedback, it is easier to require a user to assign a rating score than to write a review because the review costs much time to write. For that reason, to obtain a high-performance recommendation system, but to enable its deployment with many recommendation systems, I propose a method that combines a rating score with implicit feedback.

To achieve high performance while remaining suitable with many situations in which demographic data are unavailable or too sensitive to use, my research presents a collaborative multi-key learning (CML) method that takes advantage of an average rating score with implicit feedback in a deep learning model. Two keys of my model, user categorical and item textual information, are generated from public

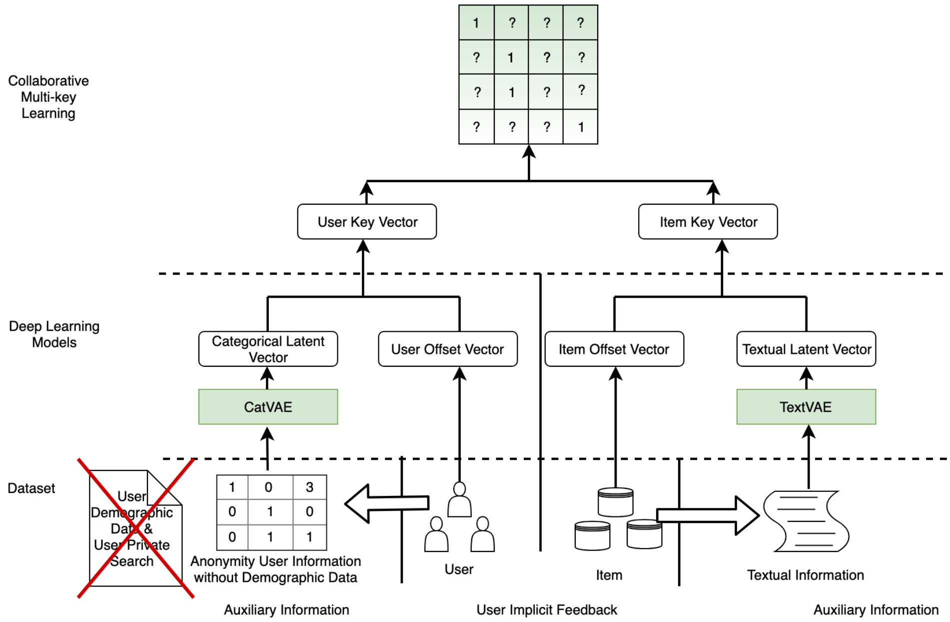


FIGURE 3.1: Flowchart of CML for a recommender system.

sources such as an average rating score and an item description, followed by optimization in multi-key learning. Therefore, CML can not only cooperate with user and item information to enhance performance; it can also perform appropriately with many information systems.

Figure 3.1 portrays a flowchart of my proposed framework for a recommender system that uses information while alleviating privacy concerns. The user information is created by the user's view and purchase history, whereas the textual information is created by the title and description of products.

The main contributions of this section are summarized as presented below.

- Achieve high performance without demographic data
- Exploit the combination of average rating score and implicit feedback in a deep learning model
- Propose deep learning models based on variational autoencoder to capture latent representation of auxiliary information from many sources: variational autoencoder for textual information (TextVAE) from products and variational autoencoder for categorical information (CatVAE) from users.
- Provide a user key vector and an item key vector for recommendation tasks by learning effective latent representations for content and implicit relations between items and users concomitantly through a probabilistic generative model with neural networks.
- Experiments on real-world datasets to demonstrate that my proposed model significantly outperforms state-of-the-art baselines.

The remainder of this paper is organized as follows: Section 3.2 presents a brief review of related works. Section 3.3 introduces my proposed model. Then Section 3.4 presents my experiment and a comparison of my results to those obtained using other methods, followed by a conclusion in Section 3.5.

3.2 Related Work

Numerous reports describe recommender systems. I only review methods that are most related to my research.

Regarding auxiliary information, collaborative topic modeling (CTR) [46] presents a model that uses latent Dirichlet allocation (LDA) to learn latent variables. Yet, these latent variables are often insufficiently effective, especially when the auxiliary information is very sparse. To avoid heavy feature engineering processes, researchers have recently emphasized applications of deep learning models that show great potential in computer science areas, to extract features. Collaborative deep learning (CDL) [47], collaborative knowledge base embedding (CKE) [56], and collaborative variational autoencoder [27] have been proposed. They show promising performance. CDL uses stacked denoising autoencoder (DAE) to extract features from textual information and combines it with rating information through joint learning. Collaborative Variational Autoencoder is the same as CDL, except it uses variational autoencoder (VAE) [24] instead of denoising autoencoder. VAE seems better than DAE for cases in which corruption of the input in observation space requires data specific corruption schemes, whereas, if given a fixed noise level, then it will degrade the robustness of representation learning [27]. Nevertheless, these methods completely ignore user information.

Regarding the use of user profiles, deep collaborative filtering [26] presents a method that combines demographic and product information. To avoid using demographic data, Multi-VAE [31] and deep matrix factorization (DMF) [54] use a user-item rating vector as the user profile input. Multi-VAE attempts to reconstruct a user profile through VAE whereas DMF uses matrix factorization to learn latent features of both users and item through neural networks, which allows a user-item rating vector and item-user rating vector as input. However, both models are CF methods. For that reason, they use no content information.

3.3 Proposed Collaborative Multi-Key Learning

This section presents the CML method, which not only learns feature vectors of user and item information through two separated deep learning models. It also presents how to combine latent vectors obtained from two models in a collaborative filtering system. My model is divisible into three parts: categorical user information, textual item information, and collaborative multi-key learning information.

Here, I designate a user index i $(i = 1, \cdots, I)$ and item index j $(j = 1, \cdots, J)$. For this study, I use datasets of two types: user information data without privacy concerns and textual information data. I denote user i data as a vector $\mathbf{s}_i$, which is a stack vector of one-hot-encoding feature content vector. Textual data (title and description of items) are represented by a bag-of-words matrix $\mathbf{X}$, which is a J-by-M matrix, where J is the number of items and M is the vocabulary size. In addition, $\mathbf{x}_j$ is a vector which row j in $\mathbf{X}$ is transposed.

My goal is production of a good predictor r_{ij} of interaction of user i for item j using dataset $\mathbf{S} = [\mathbf{s}_1, \cdots, \mathbf{s}_I]$ and $\mathbf{X}$.

3.3.1 Variational Autoencoder

Variational autoencoder (VAE) [24] is a probabilistic AE. Different from other AE models, latent variable $\mathbf{z}$ is not generated directly by input, but is instead sampled from some prior distribution $p_\theta(\mathbf{z})$ with parameter set θ. Then output is generated

TABLE 3.1: Summary of key notation used in this work. All vectors are denoted as bold lowercase

$\mathbf{s}_i$, $\mathbf{x}_j$	User information of user i and textual information of item j
r_{ij}	Interaction between user u and item v
$\mathbf{u}_i$, $\mathbf{v}_j$	Representation vector of user i or item j
$\mathbf{u}_i^{\dagger}$, $\mathbf{v}_j^{\dagger}$	Offset vector of user i and item j
$\mathbf{z}_s$, $\mathbf{z}_x$	Latent vector of user information and textual latent vector of item
$\mathbf{e}_s$, $\mathbf{d}_s$, $\mathbf{e}_x$, $\mathbf{d}_x$	encoded and decoded layers of user and textual information
$\mathbf{Q}$, $\mathbf{c}$, $\mathbf{W}$, $\mathbf{b}$	Set of weight and bias parameters connected among user or textual information and encoded layers, latent layers, and decoded layers

from some conditional distribution $p_\theta(D|\mathbf{z})$, where D represents input data. Therefore, VAE can learn significant features and generate new instances that appear to have been sampled from the training set.

However, the true posterior $p_\theta(\mathbf{z}|D)$ is intractable, especially with continuous variables. Similarly to [24], I seek parameter set ϕ so that variational inference $q_\phi(\mathbf{z}|D)$ is approximate with the true posterior $p_\theta(D|\mathbf{z})$. To measure the quality of this approximation, I can use Kullback–Leibler divergence $\mathbb{KL}$ between the approximate and exact posteriors. Then my problem becomes maximization of the lower bound $\mathcal{L}(\theta, \phi; D)$ as shown below.

$$\mathcal{L}(\theta, \phi; D) = \mathbb{E}_{q_\phi(\mathbf{z}|D)}[\log p_\theta(D|\mathbf{z})] - \mathbb{KL}(q_\phi(\mathbf{z}|D)||p_\theta(\mathbf{z})) \tag{3.1}$$

3.3.2 Variational Autoencoder for Categorical Embedding (CatVAE)

In this subsection, I investigate an unsupervised deep learning model called CatVAE to learn latent representations of categorical information.

Actually, CatVAE, a multiple hidden layer VAE for categorical information, comprises three parts: an encoder, a learning latent vector by probability, and a decoder. As described in Section 3.3.1, latent variable $\mathbf{z}_{s,i}$ of user i's information is generated by some posterior distribution $p_{\theta_s}(\mathbf{z}_{s,i})$ with parameter set θ_s for user information. I denote a dimension of latent vectors $\{\mathbf{z}_{s,i}\}$ as K_s. Here, $\{\mathbf{z}_{s,i}\}$ represents a set of $\mathbf{z}_{s,i}$ $(i, = 1, \cdots, I)$. I use the notation hereinafter for other variables or data. Then output $\mathbf{s}_i$ is generated by some conditional distribution $p_{\theta_s}(\mathbf{s}_i|\mathbf{z}_{s,i})$. I strive to find ϕ_s such that Kullback–Leibler divergence between $q_{\theta_s}(\{\mathbf{z}_{s,i}\}|\mathbf{S})$ and $p_{\theta_s}(\{\mathbf{z}_{s,i}\}|\mathbf{S})$ is minimized.

In CatVAE, with a user, I designate $\mathbf{e}_s$ and $\mathbf{d}_s$ respectively as encoder layers and decoder layers. The output of encoder layer l of user i information is represented as $\mathbf{e}_{s,l,i}$, whereas the decoder layer output n is represented as $\mathbf{d}_{s,n,i}$. Latent vector $\mathbf{z}_{s,i}$ is generated from multivariate normal distribution $N(\boldsymbol{\mu}_{s,i}, diag(\boldsymbol{\sigma}_{s,i}))$, where $\boldsymbol{\mu}_{s,i}$ is the mean vector and $\boldsymbol{\sigma}_{s,i}$ is a vector which consists of diagonal component of covariance matrix. In addition, $\boldsymbol{\mu}_{s,i}$, $\boldsymbol{\sigma}_{s,i}$ are generated by the encoder network. Here, $\{\mathbf{Q}_{e,l}$, $\mathbf{Q}_{d,n}$, $\mathbf{c}_{e,l}$, $\mathbf{c}_{d,n}\}$ $(l = 1, \cdots, L_s, n = 1, \cdots, N_s)$ respectively stand for weight matrices and bias vectors of encoder layer l and decoder layer n. $\mathbf{Q}_\mu$, $\mathbf{Q}_\sigma$, $\mathbf{c}_\mu$, $\mathbf{c}_\sigma$ are weight matrices and bias vectors from the last layer encoder to latent variables. For convenience, I use $\mathbf{Q}$, and $\mathbf{c}$ to denote the collection of all layers of weight matrices and biases in categorical embedding. Also, L_s-N_s CatVAE corresponds to an L_s layer encoder and N_s layer decoder.

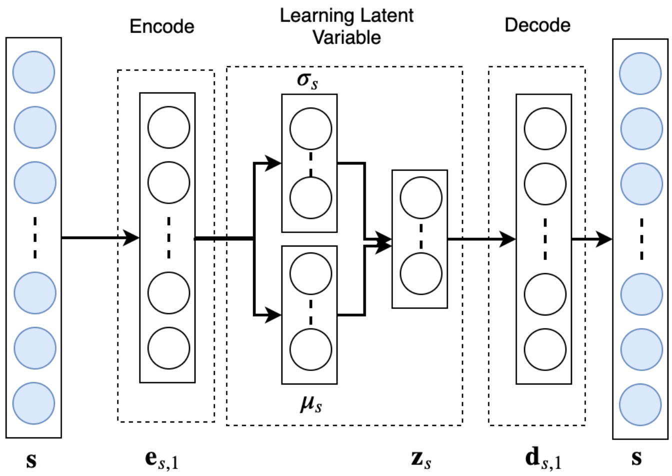


FIGURE 3.2: Illustration of a 1-1 CatVAE.

Figure 3.2 presents my 1-1 CatVAE, which has one layer encoder and one layer decoder. First, the input of user information is encoded by some hidden layers $\mathbf{e}_{s,l,i}$. Then, latent variable $\mathbf{z}_{s,i}$ is generated from $N(\boldsymbol{\mu}_{s,i}, diag(\boldsymbol{\sigma}_{s,i}))$ produced from a dense function of the last encoded layer. The generative process of CatVAE is explained below.

1. **Encode process**: For each layer l in encoded layers $\mathbf{e}_s$
   (a) For each column k weight matrix $\mathbf{Q}_{e,l}$, draw $\mathbf{Q}_{e,l,k} \sim \mathcal{N}(\mathbf{0}, \lambda_q^{-1}\mathbf{I})$
   (b) For bias parameter, draw $\mathbf{c}_{e,l} \sim \mathcal{N}(\mathbf{0}, \lambda_q^{-1}\mathbf{I})$
   (c) For the output of layer, draw $\mathbf{e}_{s,l,i} \sim \mathcal{N}(f(\mathbf{Q}_{e,l}\mathbf{e}_{s,l-1,i} + \mathbf{c}_{e,l}), \lambda_s^{-1}\mathbf{I})$
2. **Generate latent variable**: For each user, perform the following.
   (a) For a mean variable, draw $\boldsymbol{\mu}_{s,i} \sim \mathcal{N}(f(\mathbf{Q}_\mu \mathbf{e}_{s,L_s,i} + \mathbf{c}_\mu), \lambda_s^{-1}\mathbf{I})$
   (b) For the standard deviation, draw $\sigma^2_{s,i} \sim \mathcal{N}(f(\mathbf{Q}_\sigma \mathbf{e}_{s,L_s,i} + \mathbf{c}_\sigma), \lambda_s^{-1}\mathbf{I})$
   (c) For a latent variable, draw $\mathbf{z}_{s,i} = \boldsymbol{\mu}_{s,i} + \boldsymbol{\sigma}_{s,i} \odot \epsilon$
3. **Decode process**: For each layer n in $\mathbf{d}_s$
   (a) For each column k weight matrix $\mathbf{Q}_{d,n}$, draw $\mathbf{Q}_{d,n,k} \sim \mathcal{N}(\mathbf{0}, \lambda_q^{-1}\mathbf{I})$
   (b) For a bias parameter, draw $\mathbf{b}_{d,n} \sim \mathcal{N}(\mathbf{0}, \lambda_w^{-1}\mathbf{I})$
   (c) For the output of a layer, draw $\mathbf{d}_{s,n,i} \sim \mathcal{N}(f(\mathbf{Q}_{d,n}\mathbf{d}_{s,n-1,i} + \mathbf{c}_{d,n}), \lambda_s^{-1}\mathbf{I})$

where λ_w and λ_x are hyperparameters, $\mathbf{e}_{x,0,j} = \mathbf{x}_j$ and $\mathbf{d}_{x,0,j} = \mathbf{z}_{x,j}$.

- λ_q and λ_s are hyperparameters

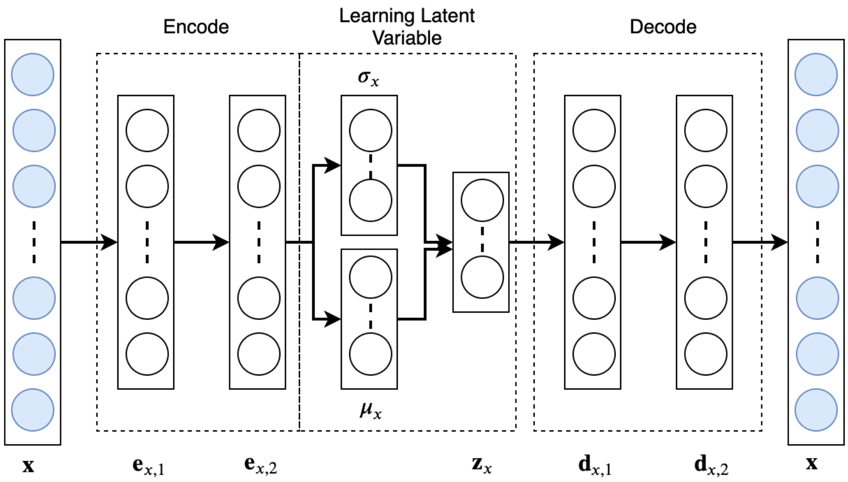


FIGURE 3.3: Illustration of a 2-2 TextVAE.

- $\mathbf{I}$ is the unit matrix.
- $f(.)$ is the activation function, which can be reLU, tanh, or sigmoid.
- $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
- operation $\odot$ means $\mathbf{A} = \mathbf{B} \odot \mathbf{C}$ if $\mathbf{A}_{ij} = \mathbf{B}_{ij} \times \mathbf{C}_{ij}$
- $\mathbf{e}_{s,0,i} = \mathbf{s}_i$ and $\mathbf{d}_{s,0,i} = \mathbf{z}_{s,i}$

3.3.3 Variational Autoencoder for Texual Embedding (TextVAE)

In this subsection, similarly to the previous categorical embedding part, TextVAE is multiple hidden layers of VAE for textual information. Similarly to user information, I have $\mathbf{e}_x$, $\mathbf{z}_x$, and $\mathbf{d}_x$ respectively as encoder layers, latent vector and decoder layers. I designate $\mathbf{e}_{x,l,j}$, $\mathbf{z}_{x,l,j}$ and $\mathbf{d}_{x,l,j}$ for item j in the same manner as user information. $\mathbf{z}_{x,l,i}$ has dimension K_x. It is generated from $N(\boldsymbol{\mu}_{x,j}, diag(\boldsymbol{\sigma}_{x,j})$. In addition, $\mathbf{W}$, $\mathbf{b}$ are weight matrices and biases of all layers, whereas $\{\mathbf{W}_{e,l}, \mathbf{W}_{d,n}, \mathbf{b}_{e,l}, \mathbf{b}_{d,n}\}$ $(l = 1, \cdots, L_x, n = 1, \cdots, N_x)$, $\mathbf{W}_\mu$, $\mathbf{W}_\sigma$, $\mathbf{b}_\mu$, $\mathbf{b}_\sigma$ are defined similarly to CatVAE. I must also find parameter set ϕ_x for textual information such that $q_{\phi_x}(\mathbf{z}_{x,j}|\mathbf{x}_j)$ is approximate with $p_{\theta_x}(\mathbf{z}_{x,j}|\mathbf{x}_j)$.

Figure 3.3 presents my illustrations for 2-2 TextVAE. The generative process of latent variables is shown below:

1. **Encode process**: For each layer l in encoded layers $\mathbf{e}_x$
   (a) For each column k weight matrix $\mathbf{W}_{e,l}$, draw $\mathbf{W}_{e,l,k} \sim \mathcal{N}(\mathbf{0}, \lambda_w^{-1}\mathbf{I})$
   (b) For the bias parameter, draw $\mathbf{b}_{e,l} \sim \mathcal{N}(\mathbf{0}, \lambda_w^{-1}\mathbf{I})$
   (c) For the output of a layer, draw $\mathbf{e}_{x,l,j} \sim \mathcal{N}(f(\mathbf{W}_{e,l}\mathbf{e}_{x,l-1,j} + \mathbf{b}_l), \lambda_x^{-1}\mathbf{I})$
2. **Generate latent variable**: For each item,
   (a) For a mean variable, draw $\boldsymbol{\mu}_{x,j} \sim \mathcal{N}(f(\mathbf{W}_\mu \mathbf{e}_{x,L_x,j} + \mathbf{b}_\mu), \lambda_x^{-1}\mathbf{I})$

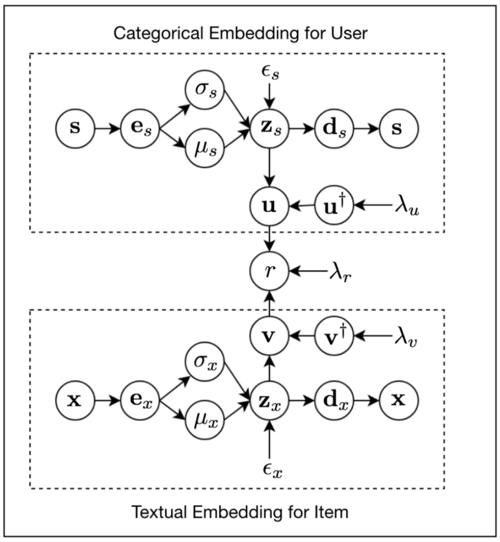


FIGURE 3.4: Collaborative Multi-key Learning Model.

(b) For the standard deviation, draw
$\sigma^2_{x,j} \sim \mathcal{N}(f(\mathbf{W}_\sigma \mathbf{e}_{s,L_x,j} + \mathbf{b}_\sigma), \lambda_x^{-1}\mathbf{I})$

(c) For a latent variable, draw
$\mathbf{z}_{x,j} = \boldsymbol{\mu}_{x,j} + \boldsymbol{\sigma}_{x,j} \odot \epsilon$

3. **Decode process**: For each layer n in $\mathbf{d}_{x,j}$

   (a) For each column k weight matrix $\mathbf{W}_{d,n}$, draw
   $\mathbf{W}_{d,n,k} \sim \mathcal{N}(\mathbf{0}, \lambda_w^{-1}\mathbf{I})$

   (b) For a bias parameter, draw $\mathbf{b}_{d,n} \sim \mathcal{N}(\mathbf{0}, \lambda_w^{-1}\mathbf{I})$

   (c) For the output of a layer, draw
   $\mathbf{d}_{x,n,j} \sim \mathcal{N}(f(\mathbf{W}_{d,n}\mathbf{d}_{x,n-1} + \mathbf{b}_{d,n}), \lambda_x^{-1}\mathbf{I})$

where λ_w and λ_x are hyperparameters, $\mathbf{e}_{x,0,j} = \mathbf{x}_j$ and $\mathbf{d}_{x,0,j} = \mathbf{z}_{x,j}$.

3.3.4 Collaborative Multi-key Learning

Using CatVAE and TextVAE, I obtained two feature variables: $\mathbf{z}_s$ and $\mathbf{z}_x$. Considering these feature variables as the "key" components, I propose a CML model as shown in Figure 3.4. I designated r_{ij} as the interaction of user i to item j. The formula is presented below.

1. For categorical embedding: Get latent variable $\{\mathbf{z}_{s,i}\}$ for all users as 3.3.2

2. For textual embedding: Get latent variable $\{\mathbf{z}_{x,j}\}$ for all items as 3.3.3

3. For each user i:

   (a) Draw a latent user offset vector $\mathbf{u}_i^\dagger \sim \mathcal{N}(\mathbf{0}, \lambda_u^{-1}\mathbf{I})$.

   (b) Set user key vector to be $\mathbf{u}_i = \mathbf{u}_i^\dagger + \mathbf{z}_{s,i}$.

4. For each item j:

   (a) Draw a latent item offset vector $\mathbf{v}_j^\dagger \sim \mathcal{N}(\mathbf{0}, \lambda_v^{-1}\mathbf{I})$.

(b) Set item key vector as $\mathbf{v}_j = \mathbf{v}_j^\dagger + \mathbf{z}_{x,j}$.

5. Draw a rating r_{ij} for each user–item pair (i, j):
$r_{ij} \sim \mathcal{N}(\mathbf{u}_i^T \mathbf{v}_j, C_{ij}^{-1})$

Here C_{ij} is a confidence parameter similar to that for CTR [46] ($C_{ij} = a$ if $r_{ij} = 1$ and $C_{ij} = b$ otherwise)

Learning the parameters: As in [27], I seek parameters ϕ_s and ϕ_x such that $\mathbb{KL}(q_{\phi_s}(\{\mathbf{z}_{s,i}\}|\{\mathbf{s}_i\})||p(\{\mathbf{z}_{s,i}\}))$ and $\mathbb{KL}(q_{\phi_x}(\{\mathbf{z}_{x,j}\}|\{\mathbf{x}_j\})||p(\{\mathbf{z}_{x,j})\})$ are minimized. Then, maximizing the posterior probability of $\{\mathbf{u}_i\}$, $\{\mathbf{v}_j\}$, $\{r_{ji}\}$, $\mathbf{W}$, $\mathbf{b}$, $\mathbf{Q}$, and $\mathbf{c}$ is equivalent to maximizing the Evidence Lower Bound as shown below.

$$\begin{aligned}
\mathcal{L}^{MAP} = &- \sum_{i,j} \frac{C_{ij}}{2}(r_{ij} - \mathbf{u}_i^T \mathbf{v}_j)^2 - \frac{\lambda_u}{2} \sum_i (\mathbb{E}_{q_{\phi_s}(\{\mathbf{z}_{s,i}\}|\mathbf{S})}) \|\mathbf{u}_i - \mathbf{z}_{s,i}\|_2^2 \\
&- \frac{\lambda_v}{2} \sum_j (\mathbb{E}_{q_{\phi_x}(\{\mathbf{z}_{x,j}\}|\mathbf{X})}) \|\mathbf{v}_j - \mathbf{z}_{x,j}\|_2^2 + \mathbb{E}_{q_{\phi_s}(\{\mathbf{z}_{s,i}\}|\mathbf{S})} \log p(\mathbf{S}|\{\mathbf{z}_{s,i}\}) \\
&+ \mathbb{E}_{q_{\phi_x}(\{\mathbf{z}_{x,j}\}|\mathbf{X})} \log p(\mathbf{X}|\{\mathbf{z}_{x,j}\}) - \mathbb{KL}(q_{\phi_s}(\{\mathbf{z}_{s,i}\}|\mathbf{S}) \| p(\{\mathbf{z}_{s,i}\})) \\
&- \mathbb{KL}(q_{\phi_x}(\{\mathbf{z}_{x,j}\}|\mathbf{X}) \| p(\{\mathbf{z}_{x,j}\})) - \frac{\lambda_q}{2} \sum_t (\|\mathbf{Q}_t\|_F^2 + \|\mathbf{c}_t\|_2^2) \\
&- \frac{\lambda_w}{2} \sum_t (\|\mathbf{W}_t\|_F^2 + \|\mathbf{b}_t\|_2^2)
\end{aligned} \tag{3.2}$$

To maximize the objective in Eq. 3.2, I use an EM model as presented below.

1. Pre-train two unsupervised models, CatVAE and TextVAE, to get latent variables for initialization.

2. **E step:** Employ a stochastic gradient descent (SGD) algorithm to optimize $\{\boldsymbol{\mu}_{s,i}\}$, $\{\boldsymbol{\sigma}_{s,i}\}$, $\{\boldsymbol{\mu}_{x,j}\}$ and $\{\boldsymbol{\sigma}_{x,j}\}$. The gradient of $\mathcal{L}$ is obtainable.

$$\nabla_{\boldsymbol{\mu}_{s,i}} \mathcal{L}(\theta_s, \phi_s; \mathbf{s}_i) \simeq - \boldsymbol{\mu}_{s,i} + \frac{1}{L} \sum_{l=1}^{L} (\Lambda_{\mathbf{u}_i}(\mathbb{E}_{q\theta_U}[\mathbf{u}_i] - \mathbf{z}_{s,i}^{(l)}) + \nabla_{\mathbf{z}_{s,i}^{(l)}} \log p_{\theta_s}(\mathbf{s}_i|\mathbf{z}_{s,i}^{(l)}))$$

$$\begin{aligned}
\nabla_{\boldsymbol{\sigma}_{s,i}} \mathcal{L}(\theta_s, \phi_s; \mathbf{s}_i) \simeq &\frac{1}{\boldsymbol{\sigma}_{s,i}} - \boldsymbol{\sigma}_{s,i} + \frac{1}{L} \sum_{l=1}^{L} [\Lambda_{\mathbf{u}_i}(\mathbb{E}_{q\theta_U}[\mathbf{u}_i] - \mathbf{z}_{s,i}^{(l)}) \\
&+ \nabla_{\mathbf{z}_{s,i}^{(l)}} \log p_{\theta_s}(\mathbf{s}_i|\mathbf{z}_{s,i}^{(l)})] \odot \epsilon^{(l)}
\end{aligned}$$

$$\nabla_{\boldsymbol{\mu}_{x,j}} \mathcal{L}(\theta_x, \phi_x; \mathbf{x}_j) \simeq - \boldsymbol{\mu}_{x,j} + \frac{1}{L} \sum_{l=1}^{L} (\Lambda_{\mathbf{v}_j}(\mathbb{E}_{q\theta_V}[\mathbf{v}_j] - \mathbf{z}_{x,j}^{(l)}) + \nabla_{\mathbf{z}_{x,j}^{(l)}} \log p_{\theta_x}(\mathbf{x}_j|\mathbf{z}_{x,j}^{(l)}))$$

$$\begin{aligned}
\nabla_{\boldsymbol{\sigma}_{x,j}} \mathcal{L}(\theta_x, \phi_x; \mathbf{x}_j) \simeq &\frac{1}{\boldsymbol{\sigma}_{x,j}} - \boldsymbol{\sigma}_{x,j} + \frac{1}{L} \sum_{l=1}^{L} [\Lambda_{\mathbf{v}_j}(\mathbb{E}_{q\theta_V}[\mathbf{v}_j] - \mathbf{z}_{x,j}^{(l)}) \\
&+ \nabla_{\mathbf{z}_{x,j}^{(l)}} \log p_{\theta_x}(\mathbf{x}_j|\mathbf{z}_{x,j}^{(l)})] \odot \epsilon^{(l)}
\end{aligned}$$

Therein,

- L represents the number of samples in a datapoint,
- $\epsilon^{(l)} \sim \mathcal{N}(0, I)$, and $\mathbf{z}^{(l)} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \epsilon^{(l)}$, and

- $\Lambda_{\mathbf{u}_i} \leftarrow (\mathbb{E}_{q\theta_V}[\mathbf{V}\mathbf{C}_j\mathbf{V}^T] + \lambda_u\mathbf{I})$,
  where $\mathbb{E}_{q\theta_V}[\mathbf{V}\mathbf{C}_j\mathbf{V}^T] = \mathbb{E}_{q\theta_V}[\mathbf{V}]\mathbf{C}_j\mathbb{E}_{q\theta_V}[\mathbf{V}]^T + \sum_j C_{ij}\Lambda_{v_j}^{-1}$, and
- $\Lambda_{\mathbf{v}_j} \leftarrow (\mathbb{E}_{q\theta_U}[\mathbf{U}\mathbf{C}_i\mathbf{U}^T] + \lambda_u\mathbf{I})$
  where $\mathbb{E}_{q\theta_U}[\mathbf{U}\mathbf{C}_j\mathbf{U}^T] = \mathbb{E}_{q\theta_U}[\mathbf{U}]\mathbf{C}_i\mathbb{E}_{q\theta_U}[\mathbf{U}]^T + \sum_i C_{ij}\Lambda_{u_i}^{-1}$.

3. **M step**: Update **U** and **V** as shown below.

$$\mathbf{u}_i \leftarrow (\mathbf{V}\mathbf{C}_i\mathbf{V}^T + \lambda_u\mathbf{I}_K)^{-1}(\mathbf{V}\mathbf{C}_i\mathbf{R}_i + \lambda_u(\mathbb{E}_{q\theta_{\mathbf{z}_s}}[\mathbf{z}_{s,i}]))$$

$$\mathbf{v}_j \leftarrow (\mathbf{U}\mathbf{C}_j\mathbf{U}^T + \lambda_v\mathbf{I}_K)^{-1}(\mathbf{U}\mathbf{C}_i\mathbf{R}_i + \lambda_v(\mathbb{E}_{q\theta_{\mathbf{z}_x}}[\mathbf{z}_{x,j}]))$$

   Then calculate $\mathcal{L}^{MAP}$ as 3.2 and repeat until convergence.

4. return to step 2 until convergence

3.3.5 Predict

I set D as representing the observed data: $D = \{\mathbf{S}, \mathbf{X}\}$. After all parameters, U, V, and the weights of the inference network and generation network are learned, the predictions can be made as presented below.

$$\mathbb{E}[r_{ij}|D] = (\mathbb{E}[\mathbf{u}_i^\dagger|D] + \mathbb{E}[\mathbf{z}_{s,i}|D])^T(\mathbb{E}[\mathbf{v}_j^\dagger|D] + \mathbb{E}[\mathbf{z}_{x,j}|D])$$

For point estimation, the prediction can be simplified as

$$r_{ij}^* = (\mathbf{u}_i + \boldsymbol{\mu}_{s,i})^T(\mathbf{v}_j + \boldsymbol{\mu}_{x,j}).$$

An item that has never been seen before will have no $\mathbf{v}$ term, but the $\boldsymbol{\mu}_x$ can be inferred through the content. As a result, both sparsity and cold start difficulties are alleviated, leading to robust recommendation performance.

3.4 Experiments

This section explains evaluation of my proposed method for use with real-world datasets from Amazon. Subsequently, I present a comparison with other state-of-the-art methods. The experimentally obtained results constitute evidence of significant improvement over competitive baselines.

3.4.1 Dataset Description

To demonstrate the effectiveness of my proposed method, I use four real datasets of Amazon [1] from different domains for empirical studies: Tools and Home Improvement, Sports and Outdoor, Health and Personal Care, and Home and Kitchen. With each of the datasets, I took two parts: metadata and 5-core.

Metadata include item information such as id, title, description, categories, brand, imageUrl, and price. I combined the title and description and followed the same procedure as that explained in another report of the relevant literature [46] to preprocess the text information. After removing stop words, the top S discriminative words according to the tf-idf [43] values are chosen to form the vocabulary. I chose S equal to 8000 in each dataset.

[1] http://jmcauley.ucsd.edu/data/amazon/

TABLE 3.2: Structure of categorical user information

Feature column	Feature content	Comments
0–6	Weekday	Weekday when the user gave rating score: 0 – Monday, 1 – Tuesday, etc.
7–11	Rating score	Rating which the user gave to items appeared in training.
12–	Categories	List categories of a dataset

TABLE 3.3: Attributes of datasets after preprocessing: #user, #item, and #feature respectively denote the number of users, number of items, and number of user categorical features. Dense rate refers to the density percentage of the rating matrix

Dataset	#user	#item	#feature	dense rate (%)
Tools and Home Improvement	2118	7780	830	0.2
Sports and Outdoor	4062	11560	994	0.13
Health and Personal Care	5584	13790	786	0.13
Home and Kitchen	7981	19184	896	0.08

Here, 5-core includes rating information such as user id, item id, rating (1–5 stars), review, and time. Each user and item has at least five interactions. I only keep users that have more than 10 interactions. I treated ratings as implicit feedback, which leads to the following.

$$r_{ij} = \begin{cases} 1 & \text{if user } i \text{ rated for item } j \\ 0 & \text{otherwise} \end{cases}$$

I took time, the rating score from 5-core reviews, and the item category from metadata to create categorical user information. I create three one hot encoding vectors corresponding to three feature contents which consist of weekday, rating score, and category in Table 3.2. Assuming that user i rates an item j belonging to category 1 as "very good (five star)" on Monday, the hot encoding vector is

$$\mathbf{s}_{i,j} = [\underbrace{1,0,0,0,0,0,0}_{weekday},\underbrace{0,0,0,0,1}_{rating},\underbrace{1,0,\cdots,0}_{category}]^T.$$

Then I create input vector $\mathbf{s}_i$ for user i as

$$\mathbf{s}_i = |J_i|^{-1} \sum_{j \in J_i} \mathbf{s}_{i,j},$$

where J_i is an item set which user i rated in the dataset.

After preprocessing, I have details of four datasets as in Table 3.3.

3.4.2 Evaluation Scheme

For recommendation tasks, to simulate reality, I sorted user-rated items following time. To prove that my model can work well in many cases, with each dataset, I have two settings: sparse and dense. With each user, I took one (for sparse setting)

TABLE 3.4: Hyperparameter settings of PMF, CDL, CVAE, and my model for sparse and dense settings of respective datasets

Methods	Tools and Home Improvement		Sports and Outdoor	
	sparse	dense	sparse	dense
PMF	$\epsilon = 1$, $\lambda_u = 0.01$, $\lambda_v = 0.01$	$\epsilon = 3$, $\lambda_u = 0.01$, $\lambda_v = 0.01$	$\epsilon = 3$, $\lambda_u = 0.01$, $\lambda_v = 0.01$	$\epsilon = 3$, $\lambda_u = 0.01$, $\lambda_v = 0.01$
CDL	$\lambda_u = 10$, $\lambda_v = 1$, $\lambda_r = 0.1$	$\lambda_u = 10$, $\lambda_v = 1$, $\lambda_r = 1$	$\lambda_u = 10$, $\lambda_v = 1$, $\lambda_r = 0.1$	$\lambda_u = 10$, $\lambda_v = 1$, $\lambda_r = 10$
CVAE	$\lambda_u = 10$, $\lambda_v = 1$, $\lambda_r = 10$	$\lambda_u = 10$, $\lambda_v = 1$, $\lambda_r = 1$	$\lambda_u = 1$, $\lambda_v = 100$, $\lambda_r = 0.1$	$\lambda_u = 10$, $\lambda_v = 1$, $\lambda_r = 1$
CML	$\lambda_u = 10$, $\lambda_v = 10$, $\lambda_r = 1$	$\lambda_u = 10$, $\lambda_v = 1$, $\lambda_r = 1$	$\lambda_u = 10$, $\lambda_v = 10$, $\lambda_r = 1$	$\lambda_u = 1$, $\lambda_v = 10$, $\lambda_r = 10$

Methods	Health and Personal Care		Home and Kitchen	
	sparse	dense	sparse	dense
PMF	$\epsilon = 3$, $\lambda_u = 0.01$, $\lambda_v = 0.01$	$\epsilon = 3$, $\lambda_u = 0.01$, $\lambda_v = 0.01$	$\epsilon = 3$, $\lambda_u = 0.01$, $\lambda_v = 0.01$	$\epsilon = 3$, $\lambda_u = 0.01$, $\lambda_v = 0.01$
CDL	$\lambda_u = 1$, $\lambda_v = 10$, $\lambda_r = 0.1$	$\lambda_u = 0.1$, $\lambda_v = 1$, $\lambda_r = 0.1$	$\lambda_u = 0.01$, $\lambda_v = 100$, $\lambda_r = 0.1$	$\lambda_u = 1$, $\lambda_v = 1$, $\lambda_r = 1$
CVAE	$\lambda_u = 1$, $\lambda_v = 100$, $\lambda_r = 0.1$	$\lambda_u = 0.1$, $\lambda_v = 100$, $\lambda_r = 10$	$\lambda_u = 1$, $\lambda_v = 100$, $\lambda_r = 10$	$\lambda_u = 0.1$, $\lambda_v = 100$, $\lambda_r = 10$
CML	$\lambda_u = 1$, $\lambda_v = 100$, $\lambda_r = 1$	$\lambda_u = 1$, $\lambda_v = 10$, $\lambda_r = 1$	$\lambda_u = 1$, $\lambda_v = 10$, $\lambda_r = 0.1$	$\lambda_u = 1$, $\lambda_v = 10$, $\lambda_r = 10$

or eight (for dense setting) first items for training, with one item for validation and the rest for testing.

For evaluation, I adopt the following three representative top-N recommendation measures:

- Recall: Percentage of purchase items that are in the recommended list

$$\text{recall@M} = \frac{\text{number of items a user likes among top M}}{\text{total number of items that the user likes}}$$

- Hit: stands for the hit ratio, or the percentage of users that have at least one correctly recommended item in their list

$$\text{Hit} = \begin{cases} 1 & \text{if there is at least a recommended item in user list} \\ 0 & \text{otherwise} \end{cases}$$

- NDCG [51]: The most frequently used list evaluation measure that incorporates the positions of correctly recommended items.

The final reported result is the average recall over all users.

3.4.3 Baselines

The models included in my comparison are listed below.

- **PMF**: probabilistic matrix factorization [39] models latent factors of users and items by a gaussian distribution. PMF is a collaborative filtering method. It uses no user or item information.

TABLE 3.5: Recall@10 of four datasets in both sparse and dense settings (%)

Methods	Tools and Home Improvement		Sports and Outdoor	
	sparse	dense	sparse	dense
PMF	0.168	0.28	0.31	0.49
Multi-VAE	0.6	1.1	1.2	2.45
CDL	0.94	2.62	1.34	3.87
CVAE	1.28	3.08	1.6	4.47
CML	**1.39**	**3.11**	**1.95**	**4.53**
Improvement	8.6%	1%	20.6%	1.3%
Methods	Health and Personal Care		Home and Kitchen	
	sparse	dense	sparse	dense
PMF	0.15	0.18	0.2	0.33
Multi-VAE	0.21	0.44	0.46	0.86
CDL	0.47	2.06	0.41	1.3
CVAE	1.02	2.55	0.87	1.57
CML	**1.24**	**2.62**	**0.93**	**1.59**
Improvement	21.9%	2.7%	6.7%	1.3%

TABLE 3.6: Hit@10 of four datasets in both sparse and dense settings

Methods	Tools and Home Improvement		Sports and Outdoor	
	sparse	dense	sparse	dense
PMF	0.023	0.019	0.042	0.035
Multi-VAE	0.08	0.074	0.15	0.14
CDL	0.11	0.13	0.16	0.2
CVAE	0.15	0.16	0.19	0.22
CML	**0.17**	**0.17**	**0.22**	**0.23**
Improvement	13%	6.25%	15.8%	4.5%
Methods	Tools and Home Improvement		Sports and Outdoor	
	sparse	dense	sparse	dense
PMF	0.027	0.03	0.028	0.025
Multi-VAE	0.046	0.057	0.063	0.065
CDL	0.075	0.12	0.059	0.079
CVAE	0.14	0.147	0.117	0.098
CML	**0.164**	**0.153**	**0.126**	**0.1**
Improvement	17%	4.08%	7.69%	2.04%

TABLE 3.7: NDCG@10 of four datasets in both sparse and dense settings

Methods	Tools and Home Improvement		Sports and Outdoor	
	sparse	dense	sparse	dense
PMF	0.009	0.0093	0.02	0.017
Multi-VAE	0.0423	0.038	0.078	0.073
CDL	0.0592	0.067	0.088	0.107
CVAE	0.0729	0.087	0.112	**0.12**
CML	**0.077**	**0.088**	**0.124**	**0.12**
Improvement	5.62%	1.15%	10.7%	0%

Methods	Tools and Home Improvement		Sports and Outdoor	
	sparse	dense	sparse	dense
PMF	0.012	0.015	0.013	0.012
Multi-VAE	0.022	0.029	0.032	0.033
CDL	0.04	0.061	0.028	0.041
CVAE	0.083	0.076	0.061	0.05
CML	**0.092**	**0.079**	**0.063**	**0.052**
Improvement	10.84%	3.95%	3.28%	4%

- **Multi-VAE**[31]: is a collaborative filtering method that uses VAE to reconstruct a user-item matrix. With each user, Multi-VAE creates a user profile by one hot user-item vector
- **CDL**: collaborative deep learning [47] is a probabilistic feedforward model for joint learning of stacked denoising autoencoder (SDAE) and collaborative filtering.
- **CVAE**: collaborative variational autoencoder [27] is a generative latent variable model that jointly models the generation of content and rating and uses variational bayes with inference network for variational inference.
- **CML**: collaborative multi-key learning is my proposed model, which learns and optimizes both textual representation for items and categorical representation for user simultaneously through two separate variational autoencoder models. It then combines them in a multi-key learning process.

3.4.4 Experimental Settings

In the experiments, I first use grid search and a validation set to ascertain the optimal hyperparameters for PMF, Multi-VAE, CDL, CVAE, as well as CML. Table 3.4 shows the best hyperparameters for each algorithm along with each dataset in both sparse and dense settings. With PMF, I chose ϵ in $[0.1, 10]$, λ_v and λ_u in $[0.01, 10]$. With CDL, CVAE, and CML, I chose λ_v in $[1, 100]$, λ_u and λ_r in $[0.1, 10]$.

After grid search, I realized that dimensions of latent variables $K_s = K_x = 50$ gave the best performance in addition to high speed. Therefore, I kept $K_s = K_x = 50$ in every algorithm with every datasets. Furthermore, I set $C_{ij} = 1$ if user i has interaction to item j, and $C_{ij} = 0.01$ if not.

With Multi-VAE, I used a three-layer model (L=6), which is '#product-600-200-50-200-600-#product'. With each layer, I applied an activation function tanh as in the original paper.

With CDL and CVAE, I used a two-layer model (L=4), which has detailed architecture as '8000-200-50-200-8000'.

With my model, I used 1-1 CatVAE model '$|\mathbf{s}_i|$-100-50-100-$|\mathbf{s}_i|$' for user information, where $|\mathbf{s}_i|$ is the user information dimension, and using 2-2 TextVAE model '8000-200-50-200-8000' for item information.

3.4.5 Performance Comparison

Tables 3.5, 3.6, and 3.7 respectively portray Recall, Hit, and NDCG results obtained using PMF, CDL, CVAE, and CML in four datasets with both sparse and dense settings. From these results, I obtained the following observations.

- Learning features through a deep learning network improves the performance to a considerable degree. This observation derives from the result that PMF, the only model which uses no content information, performs worse than other models. The best model, CML, outperforms PMF 727% and 639% in term Recall@10 and Hit@10, respectively on sparse setting of Tools and Home Improvement dataset. Results indicate that hybrid methods are much better than collaborative filtering methods.

- Extracting features from user categorical and item textual information can boost the results. From the user profile, I took the summary not only of the user–item matrix, but also time and categories. Therefore, my model can outperform Multi-VAE by 184% to 590%

- Learning features through VAE also improves the performance. CDL (which is using stacked denoising autoencoder to learning features) also performs worse than CVAE, as does my model. On Tools and Home Improvement, CML can excel 19% in dense setting, but 48% in a sparse setting. A possible reason is that DAE only works well when I have more data.

- Learning features through both user categorical information and item textual information can enhance the performance. Compared with CVAE, which only uses item textual information, my model outperforms in all datasets with both sparse and dense settings. That result is reasonable because user information is important. However, because of privacy concerns, it is difficult to implement in recommender systems, whereas CML can use user information with alleviation of privacy concerns.

Overall, it is readily apparent that CVAE is a strong baseline that beats CDL and PMF in all datasets. Therefore, I specifically examine comparison of my model and CVAE, which reveals that CML outperforms CVAE in all four datasets with margins from 1% to 21.9% in Recall@10, and from 2.04% to 17%, especially with sparse setting (margin 6.7%–21.9%) in Hit@10 and to 10% in NDCG@10. In addition, CML works well in when a dataset is sparse. In a sparse setting of Sports and Outdoor (dense rate 0.0087%) and Health and Personal Care (dense rate 0.0073%), CML can boost results respectively by 20.6% and 21.9% for Recall@10.

To elucidate the effects of different hyperparameters on my model, I tested 1) an activation function, 2) dimensions of latent variable $\mathbf{z}$, and 3) the number of hidden layers on Tool and Home Improvements with a sparse setting.

TABLE 3.8: Effects of different hyperparameters on CML

Activation function	$f(.)$	**reLU**	**tanh**	**sigmoid**
	Recall@10	0.8	0.79	1.39
Dimension of latent variable	$K_s = K_x$	**20**	**50**	**100**
	Recall@10	1.15	1.39	1.26
Number of hidden layers	#layers	**0**	**1**	**2**
	Recall@10	1.12	1.39	1.14

Activation Function: *relu* and *tand* are usually more suitable for deep models than *sigmoid*. However, in Table 3.8, *sigmoid* is apparently outstanding against *relu* and *tand*. A possible reason is that CML is a probabilistic model so that *sigmoid*, for which the output is in $(0, 1)$ is better.

Dimensions of latent variable: When I increase the number of dimensions of a latent variable, I might obtain more feature information. However, as Table 3.8 presents, more features does not mean greater effectiveness. In AE, if K_s or K_x is too large, then the model will become a copy-paste model, and might not learn anything.

Number of hidden layers: I only change the number of layers of categorical information. Increasing the number of layers will deepen the network, but such deepening is not always better. In Table 3.8, one might note that if the layers are too few (0 layer), the model seems underfitting, but if the layers are too numerous (2 layers), the model is apparently overfitting.

3.5 Conclusion

This section presents a proposal of the CML model that can extract user information and product content as well as learn implicit relations between items and users. This model can learn user information without using demographic data. Moreover, it can combine content information of items through stochastic deep learning models. Consequently, it can be adopted easily by many e-commerce companies. Experiments have demonstrated that my proposed model can outperform state-of-the-art methods for recommendation to a significant degree and with more robust performance. Experiments have also demonstrated that my model works well when the recommender system is a new one that has only one rating for each user. Therefore, start-up e-commerce companies can also apply my model without difficulty.

Chapter 4

Neural Collaborative Multi-key Learning for Recommendation System

4.1 Introduction

In the previous chapter, I used matrix factorization in rating information step. Matrix factorization breaks rating information into two component matrices: user latent matrix and item latent matrix. Although matrix factorization is famous and widely used in RS, the relationship among users and items is complicated; hence in case of few interactions in the matrix, it is difficult for matrix factorization to achieve high performance. Recently, neural networks, especially deep learning models broke down numerous previous barriers and received close attention from both the academy and the industry. For RS, plenty of researches presented substitute methods based on neural networks for and proved that neural networks possibly outperform traditional models such as matrix factorization and singular value decomposition (SVD) [17, 41, 31].

To optimize recommender systems in rating information steps by neural networks, there are nine main approaches. As Zhange et al. argue, they are based on nine main structures of neural networks [57]. Four of them are used more frequently: multiple layer perceptron (MLP), convolutional neural network (CNN), recurrent neural network (RNN) and autoencoder (AE). MLP is a feed-forward network with one or more hidden layers between input layer and output layer [57]. MLP approaches such as NeuCF [17] learn user latent vector and item latent vector separately, and then concatenate them into a single vector. The output of this model is the score of the user and the item. One advantage of this method is that it can combine with content information easily, so there are many studies based on MLP. However, because each input is an combination of a user and an item, MLP-based models may have difficulty in extracting the relationship among users and among items, which may lead to low performance, especially with new users or new items. CNN is a special feed-forward network that focuses on extracting the relationship of local features by using convolutional layers and pooling operations [13]. RNN is used for sequential data in which input of current data includes the output of previous data [13]. Most papers that used CNN [18, 45] and RNN [52, 19] models are considered interaction items of a user as a sequence. CNN-based models divide input sequence into blocks with fixed window size, while RNN-based models sort items following time order. The output of the previous item will be added to the current item. Therefore, these methods are not suitable for inferring items for new users, as they have few interactions. Recently, some studies have used AE models

TABLE 4.1: Technique comparisons of related papers with NeuCML. In user information, -, +, ++ represent for not using user information, using user information with demographic data and constructing user information without demographic data. -, + in item information columns represent for not using and using item information respectively.

Model	User information	Item information	Prediction method
PMF [39]	-	-	Matrix Factorization
NeuCF [17]	-	-	Matrix Factorization + Multiple Layer Perceptron
Multi-VAE [31]	-	-	Autoencoder
BINN [28]	++	-	Recurrent Neural Network
CVAE [27]	-	+	Matrix Factorization
JVAE-CF [25]	+	-	Autoencoder
CML	++	+	Matrix Factorization
NeuCML	++	+	Autoencoder

for rating information such as AutoRec and Multi-VAE [41, 31] and performed high accuracy. AE approaches have recently become the most used methods to highlight latent vector. One advantage of AE models is that they learn users' interest which given to all items at the same time. Based on this, they can highlight the relationship among items that can achieve high performance even for new users. However, the input of these models needs to include the information relative to all items, which is hard to combine with content information due to the high input dimension. In my research, I propose an AE model which combine both rating information and content information. The main features of my work and other pieces of research are summarized at Table 4.1

The main contributions of this section are outlined below.

- Propose Denoising Unbalanced Autoencoder for rating information step which can boost model especially in sparse cases.
- Propose a method to combine content information and rating information into AE model
- Experiments on real-world datasets to demonstrate that my proposed model significantly outperforms state-of-the-art baselines.

The remainder of this section is organized as follows: Section 4.2 presents my proposed model; Section 4.3 illustrates my experiments and Section 4.4 compares of my results to those obtained using other methods. Finally, the conclusion is provided in Section 4.5

4.2 Neural Collaborative Multi-Key Learning Model

This section presents the NeuCML method, which is an end-to-end deep learning model. My model is divisible into three parts: user information, item information, and collaborative multi-key learning information. Collaborative multi-key learning information is an combination of content information and rating information. The general structure of the proposed NeuCML is presented in Figure 4.1.

TABLE 4.2: List of denotation

Denote	Size	Meaning
I, J		number of user and item
V_u, V_i		size of user information and item information vector
$\mathbf{M}_u$	$I \times V_u$	one-hot-encoding feature content matrix for user information
$\mathbf{M}_i$	$J \times V_i$	a bag-of-words matrix for textual data of item information
$\mathbf{X_r}$	$I \times J$	user-item rating matrix
$\mathbf{Z}_u, \mathbf{Z}_i$		stack of latent vector gotten from CatVAE and TextVAE as CML
$\mathbf{X}_c$	$I \times J$	user-item content matrix
$\mathbf{x}_c, \mathbf{x}_r$	I	content vector and rating vector of a user u

Different AE structures are applied to all three parts. Content latent vectors are extracted from auxiliary information by two variational autoencoder (VAE) models. Then, a denoising unbalanced autoencoder network reconstructs the combination of content latent vectors and rating information to predict users' interest for all items. General structures of two models are represented in Figure 4.2 c and d respectively.

Here, I designate a user index $i(i = 1, \cdots, I)$ and an item index $j(j = 1, \cdots, J)$. Both content information and rating information are used to enhance system knowledge about users and items. In content information, similar to CML, user information of an user i, $\mathbf{s}_i$ is constructed from time and sub-categories in q transaction, and item information of an item j, $\mathbf{t}_j$ is the output of tf-idf for item description and title. I denote $\mathbf{M}_u$ and $\mathbf{M}_i$ as user and item information respectively, where $\mathbf{M}_u = [\mathbf{s}_0, \mathbf{s}_1, \cdots, \mathbf{s}_I]$ and $\mathbf{M}_i = [\mathbf{t}_0, \mathbf{t}_1, \cdots, \mathbf{t}_I]$.

Rating information is denoted as X_r for all users, and $\mathbf{x}_r$ for a single user. My goal is to generate a probability vector $\mathbf{y}$ for each user, where $\mathbf{y}_i = [y_{i,0}, y_{i,1}, \cdots, y_{i,J}]$ and $y_{i,j}$ measures interest probability that user i will give to item j. The details of the denotation are presented in Table 4.2.

4.2.1 User-Item Content Matrix

$\mathbf{M}_u$ and $\mathbf{M}_i$ matrices which are represented for user information and item information, are then learned by two separated VAE networks to collect user and item content latent matrices $\mathbf{Z}_u$ and Z_i respectively.

As all AE models do, VAE includes into two steps: encoder and decoder. Assuming that I use L hidden layers for VAE encoder and decoder, with each input $\mathbf{s}$ of user information, for encoder I have :

$$\mathbf{h}^u_{e,i} = a(\mathbf{W}^u_{e,i-1}\mathbf{h}^u_{e,i-1}),$$

where a is activation function and $\mathbf{h}_{e,0} = \mathbf{s}$. Then, a latent vector is extracted by:

$$\begin{aligned} \boldsymbol{\mu}^u &= a(\mathbf{W}^u_{e,L,\mu}\mathbf{h}^u_{e,L}), \text{ and } \boldsymbol{\sigma}^u = a(\mathbf{W}^u_{e,L,\sigma}\mathbf{h}^u_{e,L}) \\ \mathbf{z}^u &= \boldsymbol{\mu}^u + \boldsymbol{\sigma}^u \odot \boldsymbol{\epsilon}, \end{aligned}$$

where $\boldsymbol{\epsilon} = [\epsilon_0, \epsilon_1, \cdots, \epsilon_{||\mathbf{z}||}]$ and $\epsilon_i \sim \mathcal{N}(0, 1)$. Similar to encoder, I have decoder as:

$$\mathbf{h}^u_{d,i} = a(\mathbf{W}^u_{d,i-1}\mathbf{h}^u_{d,i-1}),$$

where $\mathbf{h}^u_{d,0} = \mathbf{z}^u$ and $\mathbf{h}^u_{d,L} = \tilde{\mathbf{s}}$ which is the generative vector of input vector $\mathbf{s}$. User content latent matrix $\mathbf{Z}_u$ is constructed by $[\mathbf{z}^u_1, \mathbf{z}^u_2 \cdots \mathbf{z}^u_I]$.

I use a similar VAE structure for item information to get item content latent matrix $\mathbf{Z}_i$, which is constructed by $[\mathbf{z}^i_1, \mathbf{z}^i_2 \cdots \mathbf{z}^i_J]$. Then, the user-item content matrix is constructed as:

$$\mathbf{X}_c = \mathbf{Z}_u \times \mathbf{Z}_i^T \tag{4.1}$$

I also denote $\mathbf{x}_c = \mathbf{z}^u \times \mathbf{Z}_i^T$ as user content vector.

4.2.2 Denoising Unbalanced Autoencoder for Rating Information

Unlike traditional denoising autoencoder (DAE) which corrupted the initial input $\mathbf{x}$ to get a partially destroyed version $\tilde{\mathbf{x}}$ using stochastic mapping $\tilde{\mathbf{x}} \sim q_{\mathcal{D}}(\tilde{\mathbf{x}}|\mathbf{x})$, I propose another strategy to corrupt input vector $\mathbf{x}_r$ as $\tilde{\mathbf{x}} \sim q_{\mathcal{D}}(\tilde{\mathbf{x}}|\mathbf{x}_r, \mathbf{x}_c))$, which is parameterized by the desired proportion λ. A fixed number λ is added into $\mathbf{x}_r$ to "fill out" "blanks" of rating information with content information.

$$\tilde{\mathbf{x}} = (\mathbf{x}_r * (1 - \lambda) + \lambda) \odot \mathbf{x}_c \tag{4.2}$$

The AE is trained to "understand" the probability of interest from the user to each item. The corrupted input $\tilde{\mathbf{x}}$ is then mapped, as with the basic AE, to a hidden representation $\mathbf{z} = f_\theta(\tilde{\mathbf{x}}))$. The hidden representation can be learned by a n-layers to highlight important information from a complex corrupted input $\tilde{\mathbf{x}}$ as follows:

$$\begin{aligned}
\mathbf{h}_i &= a(\mathbf{W}_{i-1}\mathbf{h}_{i-1}) \text{ where } a \text{ is an activation function} \\
\mathbf{h}_0 &= \tilde{\mathbf{x}} \\
\mathbf{z} &= a(\mathbf{W}_n \mathbf{h}_n)
\end{aligned}$$

From the hidden representation, I reconstruct an $\tilde{\mathbf{x}}_r = s(\mathbf{W}'\mathbf{z})$ where s is the activation function. Different from other AE models in which encoder and decoder are symmetric, I use a single layer to reconstruct a simple rating information matrix from hidden representation, although a deeper network is used in the encoder part. That is why I name it denoising balanced autoencoder.

4.2.3 Multinomial Likelihood Loss Function

Multinomial likelihood is proved in CCF and Multi-VAE [55, 31] as the most suitable loss function for click vector. In my model,the multinomial likelihood loss function for each user u is represented as:

$$\mathcal{L}(\tilde{\mathbf{x}}_r, \mathbf{x}) = \sum \mathbf{x} \odot \log \sigma(\tilde{\mathbf{x}}) \tag{4.3}$$

where σ is softmax function and $\odot$ represents an element-wise multiplication.

4.3 Experiments

This section explains the evaluation of my proposed method for use with real-world datasets from Amazon. Subsequently, I present a comparison with other state-of-the-art methods. The experimentally obtained results constitute evidence of significant improvement over competitive baselines.

TABLE 4.3: Attributes of datasets after preprocessing: #user, #item, and #cat respectively denote the number of users, number of items, and number of user categorical features. Dense rate refers to the density percentage of the rating matrix

Dataset	#user	#item	#cat	dense (%)
Home and Kitchen	7981	19184	896	0.08
Movies and TV	11543	13604	546	0.2
Kindle Store	7188	17545	990	0.12
Toys and Games	3192	10074	406	0.17
Beauty	3796	9889	252	0.18
Office Products	1775	2381	381	0.78
Patio, Lawn and Garden	323	945	326	1.6

4.3.1 Dataset Description

To demonstrate the effectiveness of my proposed method, I used seven categories of Amazon datasets with various data sizes as well as dense rates. I used the same structure to build user information as CML.

With item information, I combined the title and description and followed the same procedure as that explained in another report [46] to preprocess the text information. After removing stop words, the top S discriminative words according to the tf-idf values are chosen to form the vocabulary. I chose a maximum of S of 8000 in each dataset.

After preprocessing, I gathered the details of the datasets as indicated in Table 4.3. Next, I divided each dataset into two settings: sparse and dense. In the sparse setting, I took only one item for training and the remaining for testing, whereas, I took eight items for training in dense setting.

4.3.2 Evaluation Scheme

For the recommendation tasks, to simulate reality, I sorted user-rated items following time. To prove that my model can work well in many cases, with each dataset, I had two settings: sparse and dense. With each user, I took one (for the sparse setting) or eight (for the dense setting) first items for training, with one item for validation and the rest for testing.

For the evaluation, I adopted the following three representative top-N recommendation measures:

- Recall: percentage of purchase items that are in the recommended list

$$\text{recall@M} = \frac{\text{number of items a user likes among top M}}{\text{total number of items that the user likes}}$$

- NDCG [51]: the most frequently used list evaluation measure that incorporates the positions of correctly recommended items.

- mAP: mean average precision. It not only measures whether items the user like are in suggestion list, but also computes order of these items.

The final reported result is the average of all users.

TABLE 4.4: mAP@50, NDCG@50 and Recall@50 of 8 Amazon datasets in sparse setting

Dataset	Models	mAP	NDCG	Recall
Home and Kitchen	NeuMF	0.04	0.011	0.03
	Multi-VAE	0.074	0.114	0.033
	CML	0.069	0.107	0.031
	NeuCML	0.09	0.127	0.037
	Improve(%)	**21.62**	**11.4**	**12.12**
Movies and TV	NeuMF	0.115	0.1	0.022
	Multi-VAE	0.276	0.18	0.051
	CML	0.257	0.172	0.045
	NeuCML	0.302	0.186	0.053
	Improve(%)	**9.42**	**3.33**	**3.92**
Beauty	NeuMF	0.041	0.067	0.022
	Multi-VAE	0.068	0.101	0.039
	CML	0.081	0.097	0.029
	NeuCML	0.088	0.113	0.043
	Improve(%)	**8.64**	**11.88**	**10.26**
Toys and Games	NeuMF	0.029	0.053	0.014
	Multi-VAE	0.057	0.083	0.024
	CML	0.055	0.082	0.022
	NeuCML	0.064	0.089	0.025
	Improve(%)	**12.28**	**7.23**	**4.17**
Patio, Lawn and Garden	NeuMF	0.391	0.291	0.122
	Multi-VAE	0.491	0.343	0.122
	CML	0.361	0.307	0.101
	NeuCML	0.675	0.41	0.13
	Improve(%)	**37.47**	**19.53**	**6.56**
Office Products	NeuMF	0.271	0.195	0.086
	Multi-VAE	0.399	0.219	0.109
	CML	0.284	0.211	0.079
	NeuCML	0.417	0.229	0.113
	Improve(%)	**4.51**	**4.57**	**3.67**
Kindle Store	NeuMF	0.026	0.044	0.01
	Multi-VAE	0.072	0.091	0.0231
	CML	0.05	0.075	0.018
	NeuCML	0.079	0.095	0.0244
	Improve(%)	**9.72**	**4.4**	**4.17**

TABLE 4.5: mAP@50, NDCG@50 and Recall@50 of 8 Amazon datasets in dense setting

Dataset	Models	mAP	NDCG	Recall
Home and Kitchen	NeuMF	0.002	0.006	0.003
	Multi-VAE	0.054	0.078	0.041
	CML	0.057	0.08	0.044
	NeuCML	0.062	0.087	0.048
	Improve(%)	**8.77**	**8.75**	**9.1**
Movies and TV	NeuMF	0.048	0.115	0.048
	Multi-VAE	0.252	0.136	0.056
	CML	0.535	0.223	0.12
	NeuCML	0.54	0.227	0.124
	Improve(%)	**0.93**	**1.79**	**3.33**
Beauty	NeuMF	0.165	0.096	0.06
	Multi-VAE	0.054	0.07	0.038
	CML	0.265	0.157	0.105
	NeuCML	0.271	0.158	0.107
	Improve(%)	**8.64**	**11.88**	**1.9**
Toys and Games	NeuMF	0.036	0.058	0.033
	Multi-VAE	0.027	0.047	0.024
	CML	0.115	0.105	0.072
	NeuCML	0.125	0.108	0.074
	Improve(%)	**8.7**	**2.86**	**2.78**
Patio, Lawn and Garden	NeuMF	0.298	0.195	0.137
	Multi-VAE	0.306	0.188	0.144
	CML	0.204	0.182	0.142
	NeuCML	0.348	0.21	0.15
	Improve(%)	**13.73**	**11.7**	**4.17**
Office Products	NeuMF	0.086	0.106	0.056
	Multi-VAE	0.08	0.106	0.052
	CML	0.091	0.106	0.059
	NeuCML	0.11	0.13	0.076
	Improve(%)	**20.88**	**22.64**	**28.81**
Kindle Store	NeuMF	0.064	0.084	0.037
	Multi-VAE	0.059	0.065	0.026
	CML	0.327	0.205	0.127
	NeuCML	0.378	0.22	0.132
	Improve(%)	**15.6**	**7.32**	**3.94**

4.3.3 Baselines

The models included in my comparison are listed below.

- **NeuMF** [17]: neural collaborative filtering which is combined of general matrix factorization (GMF) and multiple layer perception (MLP). GMF and MLP use the same input and item vectors, but are separately optimized. The final output is based on the output of them.
- **Multi-DAE** [31]: is a collaborative filtering method that uses denoising autoencoder to reconstruct a user-item matrix. With each user, Multi-DAE creates a user profile by one hot user-item vector
- **CML**: Collaborative multi-key learning learns and optimizes both textual representation for items and categorical representations for users simultaneously through two separate variational autoencoder models. It then combines them in a multi-key learning process.
- **NeuCML**: neural collaborative multi-key learning uses denoising unbalance autoencoder to infer interest probability of all items for each user by the combination of rating information and latent content information. Latent content information is constructed based on user and item latent information which obtained through two separate VAE models.

4.3.4 Experiment Settings

In the experiments, I first used q grid search and a validation set to ascertain the optimal hyperparameters. I used the code provided by authors for all baselines. With NeuMF, I found that a deeper network has higher performance in the Amazon dataset. As grid search result, I picked number of factors as 50 and I used a network as $[1000 - 200 - 50]$ for multiple layer perceptron part.

With my model, I used two VAE networks that have q number of neurons in each layer as $[100 - 50 - 100]$ and $[400 - 200 - 50 - 200 - 400]$ to learn the latent vector of user information and item information respectively. For rating information, in the sparse setting, I found that a shallow network with one or two hidden layers as $[200]$ or $[200 - 100]$ achieved the best performance whereas broad, deep structures such as $[4000 - 2000 - 1000]$ or $[5000 - 3000 - 2000]$ performed better in dense setting. Furthermore, I chose λ which balances rating information and content information as 0.01

4.3.5 Performance Comparison

Tables 4.4 and 4.5 portray mAP@50, NDCG@50 and Recall@50 results obtained using NeuMF, CML, Multi-VAE and NeuCML in seven datasets with sparse and dense settings respectively. From these results, I obtained the following observations.

- Denoising unbalanced autoEncoder is better than matrix factorization, especially with the sparse setting. This observation derives from NeuCML outperforming than CML by 2.27% – 10.26% in dense settings, and 13.64% – 48.28% in Recall@50.
- VAE model is better than the combination of GMF and MLP. This observation derives from NeuCML outperforming than NeuMF by 53.9% – 203.8%; Multi-VAE also outperforms better than NeuMF by 25.57% – 176.9% in mAP@50 of all sparse settings

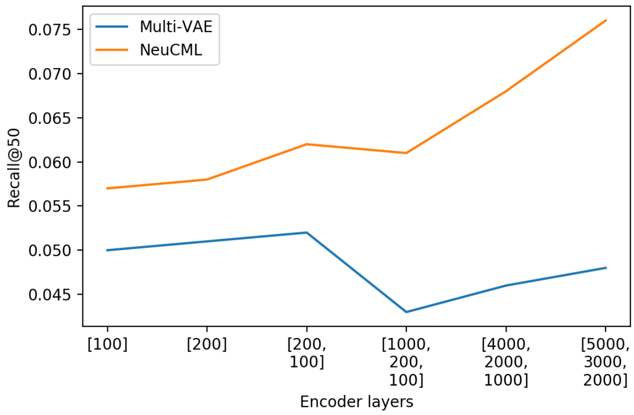


FIGURE 4.1: Recall@50 of Office Products dataset with different layers and number of neurons settings

- Adding auxiliary information can boost systems not only in the accuracy but also in the order of item lists. That mAP@50 of my model is better than Multi-VAE by 4.51% – 38.46% in sparse settings and 13.72% – 540.7% in dense settings proved this observations. User information in my model is built based on transactions. Therefore, the denser of a transaction, the better a user information system can get, which is why my model outperforms Multi-VAE in dense settings with large margin.

To elucidate the effects of different numbers of layers as well as number of neurons, I tested 6 combination of layers and neurons for rating information in the Office Products category with Multi-VAE and NeuCML. The result is illustrated in Figure 4.3. In Multi-VAE, the click input vector is so sparse that a deeper and wider network cannot learn more information. However, in NeuCML, the click input vector is combined with content information to become denser. The rich source of information needs a deeper and wider network to highlight more important features.

4.4 Conclusion

This section presented a proposal of the NeuCML model that can extract user information and product content as well as learn implicit relations between items and users. User information can be extracted from transactions without using demographic data. This model proposes a method to combine content information and AE collaborative filtering approach that can achieve high performance in both sparse and dense settings. In addition, this model also proposes a DUAE model, that appropriately learns latent features of dense vectors as click vectors and content information combination, and generates sparse click vectors.

Chapter 5

D2D-TM: A Cycle VAE-GAN for Multi-Domain Collaborative Filtering

5.1 Introduction

In a recommender system, there are possible to have plenty items such as hundreds of millions products in Amazon and billions users in Facebook. Meanwhile, each users interest only couple of items in some certain categories. It leads interaction matrix among users and item becoming extremely sparse. To solve this problem, the recommender system tends to divide items into small domains in which the items have similar attributes [11]. Each domain will have specific characteristics. For example, Amazon divides its items into categories based on their uses such as clothes or health care products. With clothes, the most important features possibly are color and material while customers conceivaly choose Health Care Products from prestige producers. Netflix separated movies according to their genres like action or comedy. Therefore, to ascertain these characteristics, each domain must be considered separately. For that reason, many studies have specifically examined a single domain [12, 8, 7]. Nevertheless, single domains still present numerous difficulties [20]. For example, they can not work well when a user has no interaction in the considered domain or when companies want to cross-sell their products. That these problems are solvable using items from multi-domains [5] has spurred my interest in proposing multi-domain recommender systems.

Algorithms that specifically address a single domain can process items from multiple domains easily by aggregating all items into a single domain. However, because all items are learned by a sole network or function, difficulties arise in capturing the specific characteristics of respective domains. For instance, a user enjoyed to watch action comedies or sci-fi dramas movies may be supposed to be attractive by action movies if comedies and dramas categories are consolidated because number of action movies are overabundant compared to other genres. With this misunderstanding, system tends to suggest action dramas movies which are different from user's type. Conversely, some algorithms specifically addressing multiple domains extract latent features of the respective domains by a separated network [29, 34]. Although they can highlight the particular features of each domain, they have less chance to obtain similar features among domains. If there is no information about drama class, the user above is perhaps endorsed romantic dramas or horror dramas movies since they are more popular. Nevertheless, not only specific characteristics (differences) of each domain such as action comedies or sci-fi dramas are requisite to obtain, but also mapping their similarities as "user who like this kind of movies in action comedies will also like that kind of movies in sci-fi dramas" is imperative. For that reason, multi-domain systems must capture both to achieve good performance.

Some other multi-domain studies have specifically examined the transfer of knowledge from a source domain that is much denser to a target domain, or from specific sources such as user search query or social network information [42, 38, 10]. Nevertheless, many companies are unable to implement such methods because it is sometimes impossible to get much denser data or to collect data from these external sources.

To address these difficulties, I propose a multi-domain network structure that can capture both similar and different features among domains and which can treat every domain equally by taking only implicit feedback inside the system as input. My model is extended from unsupervised image-to-image translation networks (UNIT) [32] for the recommender systems, called a domain-to-domain translation model (D2D-TM). It is based on generative adversarial networks (GANs) and variational autoencoders (VAEs). D2D-TM uses the user interaction history of each domain as input and extract its features through a VAE-GAN network as well as restrains domains by domain cycle consistency (CC). In a UNIT network, two VAE networks extract highlights of the respective domains, then map them to create a fake image which GAN then attempts to clarify with a real image. In my model, GAN has the same purpose. D2D-TM generates an interaction list that a user might like in domain B based on the user interaction history in domain A. Subsequently, GAN works to classify a generated vector and a real vector in domain B, so that the generated network is improved. Interaction vector for domain A is generated in similar way if system has interaction history in domain B. However, layers of VAE of two kinds with two purposes exist in my network: distinct layers and shared layers. First, in each domain, distinct layers serve to classify user behaviors. With example above, distinct layers in comedy and drama classes are required to point up that user is interested in action Comedy and sci-fi drama respectively. Following, share layers map specific behaviors of a domain to another domain. In addition, I improve CC in UNIT network to domain cycle consistency so that it is more appropriate to recommendation task. Different from UNIT, D2D-TM requires a set of mutual users to train. Other users can be inferred directly, without training, by using information of only one domain.

In summary, the main contributions of this section are the following.

- Propose a multi-domain recommender system that can extract both homogeneous and divergent features among domains through the VAE-GAN-CC network.
- Propose an end-to-end deep learning approach for a collaborative filtering recommender system that only uses the user interaction history as input
- Infer cross-domain and single-domain in a solely network
- Conduct rigorous experiments using two real-world datasets with four couple domains. Results of those experiments underscore the effectiveness of the proposed system over state-of-the-art methods by a large margin.

The remainder of this section is organized as explained in the following. First, Section 5.2 reviews related approaches and techniques for recommender systems including VAEs, GANs, and a cross-domain recommender system. Section 5.3 presents an explanation of details of my method with subsequent description of experiments in Section 5.4. I also present conclusions in Section 5.5.

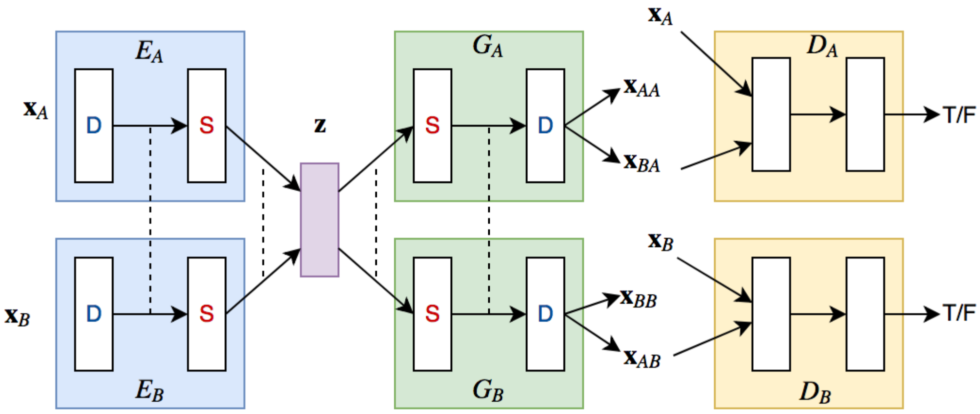


FIGURE 5.1: The general structure of D2D-TM comprises six subnetworks including two encoders E_A, E_B, two generators G_A, G_B, and two discriminators D_A, D_B. Click vectors of a user in two domains $\mathbf{x}_A$ and $\mathbf{x}_B$ are reconstructed respectively through two encoder–generator pairs: $\{E_A, G_A\}$ and $\{E_B, G_B\}$. Encoder and generator networks have some distinct layers, denoted as D, to extract and generate divergent features of two domains and some share layers, denoted as S, to map these divergent features. Click vector $\mathbf{x}_A$ is translated to vector $\mathbf{x}_{AB}$ of domain B through $\{E_A, G_B\}$. Also, click vector $\mathbf{x}_B$ is translated to vector $\mathbf{x}_{BA}$ of domain A through $\{E_B, G_A\}$. The translated vectors, along with click vectors, then, are given into two discriminator networks D_A, D_B to evaluate the actual conditions.

5.2 Related Work

Extensive studies have been conducted of RS, with reports presented in a myriad of publications. This section is aimed at reviewing a representative set of approaches that are closely related to my research.

5.2.1 Autoencoder

Autoencoder (AE) uses unsupervised learning, which has been shown to be effective for learning latent variables in many deep-learning problems. Collaborative deep learning (CDL) [47] and collaborative variational autoencoder (CVAE) [27] are two well known papers that respectively describe the application of denoising autoencoder and variational autoencoder in hybrid methods. Two studies have used AE to extract latent features from item description text, with related reports proposing joint learning between these latent features and collaborative filtering. The recent method, Multi-VAE [31] uses VAE for Collaborative Filtering to reconstruct a user–item matrix. It achieves good results despite using only rating information.

5.2.2 Generative Adversarial Network (GAN)

As new unsupervised learning network, GAN can achieve promising results, especially in the realm of computer vision. Nevertheless, few GAN applications have been reported for use with recommender systems. Actually, IRGAN [48] was the first model to apply a GAN not only to an information retrieval area but also to a RS. IRGAN extends discriminator and generator processes in traditional GANs to discriminative retrieval and generative retrieval. Whereas discriminative retrieval learns to predict relevant score r given labeled relevant query–document pairs, generative retrieval is designed to generate a fake document to deceive discriminative retrieval.

Recently, adversarial personal ranking (APR) [16], which enhances the Bayesian personal ranking with adversarial network has arisen as a new approach of GAN to recommender systems, along with GAN-HBNR [4], which proposes a GAN-based representation learning approach for heterogeneous bibliographic network.

5.2.3 Cross-Domain Recommender System

Today, companies are striving to provide diverse products and services to users. For example, Amazon is not only e-commerce platform, but also an online movie and music platform. Therefore, cross-domain recommender systems are necessary for them. Moreover, cross-domain RSs can solve data sparsity and the cold start problem, which are important issues related to single-domain RSs. Several works exploring cross-domain RSs have included multiview deep neural network (MV-DNN) [10], neural social collaborative ranking (NSCR) [49], and cross-domain content-boosted collaborative filtering neural network (CCCFNET) [29]. Actually, MV-DNN extracts rich features from the user's browsing and search histories to model user interests, whereas item features are extracted from three sources including the title, categories, and contents with news or description with Apps. Then it calculates a relevant score using a cosine function. Another method, NSCR, attempts to learn embedding of bridge users with user–user connections taken from social networks and user–item interaction. Alternatively, CCCFNET aims to learn content-based embedding so that the model can transfer both content-based and collaborative filtering across different domains simultaneously. A point of similarity among these methods is that they require external information from other sources. For example, MV-DNN requires a user search query, NSCR combines with user social network account, whereas CCCFNET takes content information. Sometimes, it is impossible to get this knowledge. Therefore, I propose a cross-domain model that uses only implicit feedback inside the system.

5.3 Method

I use $u \in \{1, \cdots, U\}$ to index users, $i_A \in \{1, \cdots, I_A\}$ to index items belonging to domain A, and $i_B \in \{1, \cdots, I_B\}$ to index items belonging to domain B. In this work, I consider learning implicit feedback. The user-by-item interaction matrix is the click [1] matrix $\mathbf{X} \in \mathbb{N}^{U \times I}$. The lower case $\mathbf{x}_u = [x_{u1}, x_{u2}, \cdots, x_{uI}]^T \in \mathbb{N}^I$ is a bag-of-words vector, which is the number of clicks for each item from user u. With two domains, I have matrix $\mathbf{X}_A \in \mathbb{N}^{U \times I_A}$ with $\mathbf{x}_A = [x_{A1}, x_{A2}, \cdots, x_{AI_A}]^T \in \mathbb{N}^{I_A}$ for domain A, and $\mathbf{X}_B \in \mathbb{N}^{U \times I_B}$ with $\mathbf{x}_B = [x_{B1}, x_{B2}, \cdots, x_{BI_B}]^T \in \mathbb{N}^{I_B}$ for domain B. For simplicity, I binarize the click matrix, meaning that $x_{ui} = 1$ if user u has click on item i and $x_{ui} = 0$ otherwise. Also, 0 can be regarded as missing values in $\mathbf{X}$, and can be generated through my framework. It is straightforward to extend its use to general count data.

[1] I use the verb "click" for concreteness. In fact, this can be any type of interaction such as "watch", "view," or "rating."

5.3.1 Framework

My framework, as presented in Figure 5.1, is based on variational autoencoder (VAE) [24] and generative adversarial network (GAN) [14]. In my model, VAE models have the main responsibility of extracting a latent feature of input, whereas GAN specifically examines classification of a user real interaction vector and a generated vector which supports the VAE networks. GAN is applied exclusively for training phrase. D2D-TM comprises six subnetworks including two domain click vector encoders E_A and E_B, two domain click vector generators G_A and G_B, and two domain adversarial discriminators D_A and D_B. I maintain the framework structure as explained in a report of an earlier study[32]. In addition, I share weights of the last few layers in E_A and E_B, so that my model not only extracts different characteristics of two model in the first layers; it also learns their similarities. In parallel, I also share weights of the few first layers in G_A and G_B to make my model able to generate both similar and divergent features. In Figure 5.1, share layers are denoted as S, whereas distinct layers are denoted as D.

In training, the user interaction vectors for domain A and B are extracted highlight representations by D layers in the encoder; then these features are shared weight in S layers in assumption that user has some consistency behaviors among domains. Furthermore, I obtain latent vector $\mathbf{z}_A$ and $\mathbf{z}_B$, which are used for not only reconstructing interaction vectors, but also generating interactions for opposite domains. To generate, latent vector $\mathbf{z}_A$ for domain A is reconstructed by S layers, then masked by D layers in G_B. Finally, the GAN discriminator is used to detect which vector was generated from the other source.

5.3.2 VAE

VAE includes two processes: an encoder that maps input x to a latent representation z and a generator that re-maps z to x_{rec}: $z \sim q(z|x)$ and $x_{rec} \sim p(x|z)$, with $q(z|x)$ and $p(x|z)$ are two conditional distributions.

In a deep learning network, to make training with back-propagation possible, a reparameterization trick [24] is applied to express a random variable $\mathbf{z}$ as a deterministic variable $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$, where $\boldsymbol{\mu}$ is a mean vector and $\boldsymbol{\sigma}$ is a vector that consists of a diagonal component of the covariance matrix. Both $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are outputs of the encoder network with input $\mathbf{x}$, denoted by $E(\mathbf{x})$. Also, $\odot$ signifies an element-wise product; $\boldsymbol{\epsilon}$ is generated from a Gaussian distribution $\mathcal{N}(0, I)$ with I as the identity matrix. However, $\mathbf{x}_{rec}$ will be the output of generator network with input $\mathbf{z}$ as $\mathbf{x}_{rec} = G(z)$.

It is noteworthy that VAE training is aimed at minimizing a variational upper bound, which is

$$\mathcal{L} = \mathbf{KL}(q(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})) - \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z})] = \mathcal{L}_{KL} + \mathcal{L}_{rec}, \tag{5.1}$$

$$\begin{aligned} \text{with} \quad \mathcal{L}_{KL} &= \mathbf{KL}(q(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})), \\ \text{and} \quad \mathcal{L}_{rec} &= -\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z})], \end{aligned}$$

where **KL** is the Kullback–Leibler divergence.

In my model, the encoder–generator pair $\{E_A, G_A\}$ constitutes a VAE for domain A, term VAE_A. As explained above, the distribution of the latent code $\mathbf{z}_A$, which is generated from $q_A(\mathbf{z}_A|\mathbf{x}_A)$, is given as $\boldsymbol{\mu}_A + \boldsymbol{\sigma}_A \odot \boldsymbol{\epsilon}$ with $\boldsymbol{\mu}_A$ and $\boldsymbol{\sigma}_A$ as the output of encoder network E_A. In this case, both $q_A(\mathbf{z}_A|\mathbf{x}_A)$ and $p(\mathbf{z}_A)$ are Gaussian distributions. Therefore,

$$\begin{aligned}\mathcal{L}_{KL_A} &= \mathbf{KL}(q_A(\mathbf{z}_A|\mathbf{x}_A)\|p(\mathbf{z}_A)) \\ &= \frac{1}{2}\sum_{k=1}^{K}(1+\log(\sigma_{Ak}^2)-\mu_{Ak}^2-\sigma_{Ak}^2),\end{aligned}$$

with K as the dimension of $\mathbf{z}$ and where σ_{Ak}, μ_{Ak} respectively represent elements of vector $\boldsymbol{\sigma}_A$ and $\boldsymbol{\mu}_A$. Then, I try to generate vector $\mathbf{x}_{AA}$ by a conditional distribution $p_{G_A}(\mathbf{x}_A|\mathbf{z}_A)$, which means that $\mathbf{x}_{AA}$ is a reconstruction of the input click vector $\mathbf{x}_A$ through generator network G_A with input $\mathbf{z}_A$:

$$\mathbf{x}_{AA} \sim p_{G_A}(\mathbf{x}_A|\mathbf{z}_A).$$

Assume that the click vector of user u for domain A is $\mathbf{x}_A = [x_{A1},\cdots,x_{AI_A}]^T$, and that the number of clicks is N_A, then $\sum_i^{I_A} x_{Ai} = N_A$. However, let $\boldsymbol{\pi}_A = f(G_A(\mathbf{z}_A))$ with $f(.)$ is softmax function, so $\sum_i^{I_A}\pi_{Ai} = 1$. Therefore, reconstruction vector $\mathbf{x}_{AA}$ of this user can be a sample from a multinomial distribution $\mathrm{Mult}(N_A,\boldsymbol{\pi}_A)$ or $p_{G_A}(\mathbf{x}_A|\mathbf{z}_A) = \mathrm{Mult}(N_A,\boldsymbol{\pi}_A)$. Therefore, the reconstruction loss for $\mathbf{x}_{AA}$ is

$$\begin{aligned}\mathcal{L}_{rec_A} &= -\mathbb{E}_{\mathbf{z}_A\sim q_A(\mathbf{z}_A|\mathbf{x}_A)}[\log p_{G_A}(\mathbf{x}_A|\mathbf{z}_A)] \\ &= -\mathbb{E}_{\mathbf{z}_A\sim q_A(\mathbf{z}_A|\mathbf{x}_A)}[\sum_i^{I_A} x_{Ai}\log\pi_{Ai}].\end{aligned}$$

Hereinafter, I also use a multinomial distribution for p_{G_B}.

$$\mathcal{L}_{VAE_A} = \lambda_1\mathcal{L}_{KL_A} + \lambda_2\mathcal{L}_{rec_A}. \tag{5.2}$$

The hyperparameters λ_1 and λ_2 control the weights of the reconstruction term. The **KL** divergence terms penalize deviation of the latent code from the prior distribution.

Similarly, $\{E_B, G_B\}$ constitutes a VAE for domain B: The distribution of latent code $\mathbf{z}_B$, which is generated from $q_B(\mathbf{z}_B|\mathbf{x}_B)$, is given as $\boldsymbol{\mu}_B + \boldsymbol{\sigma}_B \odot \boldsymbol{\epsilon}$. The reconstructed click vector is $\mathbf{x}_{BB} \sim p_{G_B}(\mathbf{x}_B|\mathbf{z}_B)$. In addition,

$$\begin{aligned}\mathcal{L}_{VAE_B} =& \lambda_1\mathcal{L}_{KL_B} + \lambda_2\mathcal{L}_{rec_B} \\ =& \lambda_1\mathbf{KL}(q_B(\mathbf{z}_B|\mathbf{x}_B)\|p(\mathbf{z}_B)) - \lambda_2\mathbb{E}_{\mathbf{z}_B\sim q_A(\mathbf{z}_B|\mathbf{x}_B)}[\log p_{G_B}(\mathbf{x}_B|\mathbf{z}_B)].\end{aligned} \tag{5.3}$$

5.3.3 Domain Cycle-Consistency (CC) and Weight-Sharing

I can translate a click vector $\mathbf{x}_A$ in domain A to a click vector in domain B through applying $p_{G_B}(\mathbf{x}_B|\mathbf{z}_A)$, terms $\mathbf{x}_{AB}$. Similarly, click vector $\mathbf{x}_{BA}$ from domain B to domain A is generated as $p_{G_A}(\mathbf{x}_A|\mathbf{z}_B)$.

To ensure that $\mathbf{x}_{AB} \approx \mathbf{x}_B$ and $\mathbf{x}_{BA} \approx \mathbf{x}_A$, first, I enforce a weight-sharing constraint relating two VAEs. Specifically, I share the weights of the last few layers of E_A and E_B that are responsible for extracting high-level representations of the input click vectors in the two domains. In parallel, I share the weights of the first few layers of G_A and G_B responsible for decoding high-level representations for reconstructing the input click vector. Weight-sharing usually is used in parallel architectures which two networks are trained simultaneously. In my case, weight-sharing not only helps my model converge better, but also supports encoders to extract common features between two domains. Moreover, because neurons corresponding to same features are triggered in various scenarios, weight-sharing can improve generating ability of my model.

However, weight sharing alone does not guarantee that two domain are matched. I propose a domain cycle consistency with two cycles to constrain representations between two domains. Cycle consistency is a way of using transitivity to supervise CNN training, which is applied in many image-to-image translation papers [59, 32]. This loss pushes encoder and decoder to be consistent into each others. In detail, with domain A, first, I constrain $\mathbf{x}_{AB}$, which is generated from $\mathbf{x}_A$, closes to $\mathbf{x}_B$.

$$\mathbf{x}_{AB} \sim p_{G_B}(\mathbf{x}_B|\mathbf{z}_A).$$

Then, I re-map $\mathbf{x}_{AB}$ to domain A and compel it to close to $\mathbf{x}_A$.

$$\mathbf{x}_{ABA} \sim p_{G_A}(\mathbf{x}_A|\mathbf{z}_{AB}) \text{ where } \mathbf{z}_{AB} \sim q_B(\mathbf{z}_{AB}|\mathbf{x}_{AB}).$$

With same encoder and decoder network as VAE, I apply VAE loss function to domain cycle consistency as:

$$\begin{aligned}\mathcal{L}_{CC_A} =& \mathcal{L}_{rec_{AB}} + \mathcal{L}_{KL_{AB}} + \mathcal{L}_{rec_{ABA}} \\ =& -\lambda_3 \mathbb{E}_{\mathbf{z}_A \sim q_A(\mathbf{z}_A|\mathbf{x}_A)}[\log p_{G_B}(\mathbf{x}_B|\mathbf{z}_A)] + \lambda_4 \mathbf{KL}(q_B(\mathbf{z}_{AB}|\mathbf{x}_{AB}) \| p(\mathbf{z}_{AB})) \\ & - \lambda_3 \mathbb{E}_{\mathbf{z}_{AB} \sim q_B(\mathbf{z}_{AB}|\mathbf{x}_{AB})}[\log p_{G_A}(\mathbf{x}_A|\mathbf{z}_{AB})]. \end{aligned} \tag{5.4}$$

As VAE, I also have hyperparameter λ_3 and λ_4 to control weights among two terms. As domain A, with domain B, I have:

$$\begin{aligned}\mathbf{x}_{BA} &\sim p_{G_A}(\mathbf{x}_A|\mathbf{z}_B) \\ \mathbf{x}_{BAB} &\sim p_{G_B}(\mathbf{x}_B|\mathbf{z}_{BA}) \text{ where } \mathbf{z}_{BA} \sim q_A(\mathbf{z}_{BA}|\mathbf{x}_{BA}).\end{aligned}$$

And, loss cycle consistency of domain B is:

$$\begin{aligned}\mathcal{L}_{CC_B} =& \mathcal{L}_{rec_{BA}} + \mathcal{L}_{KL_{BA}} + \mathcal{L}_{rec_{BAB}} \\ =& -\lambda_3 \mathbb{E}_{\mathbf{z}_B \sim q_A(\mathbf{z}_B|\mathbf{x}_B)}[\log p_{G_A}(\mathbf{x}_A|\mathbf{z}_B)] + \lambda_4 \mathbf{KL}(q_A(\mathbf{z}_{BA}|\mathbf{x}_{BA}) \| p(\mathbf{z}_{BA})) \\ & - \lambda_3 \mathbb{E}_{\mathbf{z}_{BA} \sim q_A(\mathbf{z}_{BA}|\mathbf{x}_{BA})}[\log p_{G_B}(\mathbf{x}_B|\mathbf{z}_{BA})]. \end{aligned} \tag{5.5}$$

5.3.4 Generative Adversarial Network (GAN)

GAN generally includes two processes: a generator and discriminator. Whereas the discriminator functions to recognize real and generated data, the generator is designed to generate fake ones that resemble real ones. This competition drives both processes to improve their network until the counterfeits are indistinguishable from the genuine articles [14]. In my model, VAE with cycle consistency works as a generator process. I have two GANs: $GAN_A = \{VAE_A, D_A\}$ and $GAN_B = \{VAE_B, D_B\}$.

With domain A, there are three outputs of VAE:

$$\begin{aligned}\mathbf{x}_{AA} &\sim p_{G_A}(\mathbf{x}_A|\mathbf{z}_A) \\ \mathbf{x}_{BA} &\sim p_{G_A}(\mathbf{x}_A|\mathbf{z}_B) \\ \mathbf{x}_{ABA} &\sim p_{G_A}(\mathbf{x}_A|\mathbf{z}_{AB}).\end{aligned}$$

However, I mainly emphasize resampling of a click vector from domain B to domain A. My discriminator process will be used to detect the generated click vector $\mathbf{x}_{BA}$ and real click vector $\mathbf{x}_A$. Then, optimizing GAN for domain A will yield

$$\mathcal{L}_{\mathrm{GAN}_A} = \lambda_0 \mathbb{E}_{\mathbf{x}_A \sim P_A [\log D_A(\mathbf{x}_A)]} + \lambda_0 \mathbb{E}_{\mathbf{z}_B \sim q_B(\mathbf{z}_B|\mathbf{x}_B)} [\log(1 - D_A(\mathbf{x}_{BA})]. \tag{5.6}$$

Like domain A, I try to detect generated click vector $\mathbf{x}_{AB}$ and real vector $\mathbf{x}_B$. Then the loss discriminator of domain B will be

$$\mathcal{L}_{\mathrm{GAN}_B}(E_A, G_B, D_B) = \lambda_0 \mathbb{E}_{\mathbf{x}_B \sim P_B [\log D_B(\mathbf{x}_B)]} + \lambda_0 \mathbb{E}_{\mathbf{z}_A \sim q_A(\mathbf{z}_A|\mathbf{x}_A)} [\log(1 - D_B(\mathbf{x}_{AB})]. \tag{5.7}$$

5.3.5 Learning

I solve the learning problems of VAE$_A$, VAE$_B$, CC$_A$, and CC$_B$, GAN$_A$, and GAN$_B$ jointly as

$$\begin{aligned}\min_{E_A,E_B,G_A,G_B} \max_{D_A,D_B} [&\mathcal{L}_{VAE_A}(E_A, G_A) + \mathcal{L}_{GAN_A}(E_B, G_A, D_A) + \mathcal{L}_{CC_A}(E_A, G_A, E_B, G_B) \\ &+ \mathcal{L}_{VAE_B}(E_B, G_B) + \mathcal{L}_{GAN_B}(E_A, G_B, D_B) + \mathcal{L}_{CC_B}(E_B, G_B, E_A, G_A)],\end{aligned} \tag{5.8}$$

where $\mathcal{L}_{VAE_A}$, $\mathcal{L}_{VAE_B}$, $\mathcal{L}_{GAN_A}$, and $\mathcal{L}_{GAN_B}$ are defined respectively in 5.2, 5.3, 5.6 and 5.7.

First, I pre-train VAE_A and VAE_B separately to extract the representations of two domains. Then, because GAN works as a competition among generator and discriminator while the generator tries to make a generated vector resemble a real vector, the discriminator attempts to classify them. I will optimize the generator and discriminator process sequentially. I summarize the training process as

1. Minimize generator

   $$\mathcal{L}_{gen} = \mathcal{L}_{VAE_A} + \mathcal{L}_{VAE_B} + \mathcal{L}_{CC_A} + \mathcal{L}_{CC_B} + \log(1 - D_A(\mathbf{x}_{BA})) + \log(1 - D_B(\mathbf{x}_{AB})).$$

2. Maximize $\mathcal{L}_{GAN_A}$ and $\mathcal{L}_{GAN_B}$ separately

3. Repeat Steps 1 and 2 until convergence.

5.3.6 Predict

For Cross-Domain

- From domain A to domain B: Here I assume that user u only clicked some items in domain A, and has no interaction with any item in domain B. I have a history click vector $\mathbf{x}_A$. Then I want to recommend items in domain B to him by generating vector $\mathbf{x}_{AB}$ in which the higher probability means greater interesting items to this user. First, encoder E_A extracts highlight features of $\mathbf{x}_A$ with $\mathbf{z}_A \sim q_A(\mathbf{z}_A|\mathbf{x}_A)$. Then $\mathbf{z}_A$ is masked with weight features of domain B through $\mathbf{x}_{AB} \sim p_{G_B}(\mathbf{x}_B|\mathbf{z}_A)$

- From domain B to domain A: Similarly, with a history click vector $\mathbf{x}_B$ of user u in domain B, I predict click vector $\mathbf{x}_{BA}$ in domain A as $\mathbf{x}_{BA} \sim p_{G_A}(\mathbf{x}_A|\mathbf{z}_B)$ with $\mathbf{z}_B \sim q_B(\mathbf{z}_B|\mathbf{x}_B)$.

For Single Domain

- For domain A: First, I get $\mathbf{z}_A \sim q_A(\mathbf{z}_A|\mathbf{x}_A)$ as before. To infer next items user may like in domain A, I predict click vector x_{AA} as $\mathbf{x}_{AA} \sim p_{G_A}(\mathbf{x}_A|\mathbf{z}_A)$

- For domain B: Similar with domain A, I infer x_{BB} as $\mathbf{x}_{BB} \sim p_{G_B}(\mathbf{x}_B|\mathbf{z}_B)$

5.4 Experiments

Because no public dataset exists for multi-domain RS, this section presents the experimental setup as well as empirical evaluation of my proposed method through my own datasets based on real-world datasets such as those of Amazon[2] [15, 33] and Movielens[3]. My experiments are designed to answer the following research questions:

1. Does my proposed method outperform single domain state-of-the-art models such as Multi-VAE and CDL as well as multi-domain SOTA models such as CCCFNET? And how great is the relative improvement?

2. What are the effects of the respective components such as domain CC or GAN in my method?

3. What are the effects of multinomial reconstruction loss function with the Lecky rectified linear unit (ReLu) activation function, which are key hyperparameters of my model, on model performance?

4. Can my proposed model extract specific features in each domain and map them to another domain?

5.4.1 Dataset Description

Amazon

I create two datasets from four Amazon review subsets: Health_Clothes from Health and Personal Care and Clothing, Shoes and Jewelry; Video_TV from Video Games and Movies and TV. In each dataset, I maintain a list of users who gave reviews in both subsets as well as the products which the users reviewed. I treat the rating as implicit feedback.

$$r_{ij} = \begin{cases} 1 & \text{if user } i \text{ rated for item } j \\ 0 & \text{otherwise} \end{cases}$$

Movielens

From dataset Movielens 1M, I create two subsets: Drama_Comedy and Romance_Thriller. The Drama_Comedy dataset includes users who rated both Drama and Comedy movies as well as the rated movies. I prepare the Romance_Thriller dataset similarly and consider rating scores as implicit feedback of the Amazon dataset.

I name datasets following an A_B structure. For instance, the dataset designates as Health_Clothing means domain A is Health and Personal Care products; domain B is Clothing, Shoes and Jewelry products. After preprocessing, I have details of

[2]http://jmcauley.ucsd.edu/data/amazon/
[3]https://grouplens.org/datasets/movielens/

TABLE 5.1: Information of datasets after preprocessing, #user, #item_A, and #item_B respectively represent the number of users, the number of items in domain A, and the number of items in domain B. Dense_A and dense_B respectively refer to the density percentages of rating matrixes from domain A and domain B

.

Dataset	#user	#item_A	#item_B	dense_A	dense_B
Health_Clothing	6557	16069	18226	0.08	0.05
Video_TV	5459	10072	28578	0.14	0.1
Drama_Comedy	6023	1490	1081	3.3	3.3
Romance_Thriller	5891	455	475	5.27	6.4

four datasets as shown in Table 5.1. From this, it is apparent that Movielens is much denser than Amazon. Therefore, it can be regarded as having tested my model in both the sparse and dense case.

5.4.2 Evaluation Scheme

I use two ranking-based metrics: Recall@K and normalized discounted cumulative gain (NDCG@K) [51]. With each user, I sort the predicted list and take the K highest score items. Then I compare the results with ground truth items.

Recall@K is defined as the percentage of purchase items that are of the recommended list:

$$\text{Recall@K} = \frac{\text{Number of items that a user likes in the top K}}{\text{Total number of items that a user likes}}$$

However, NDCG@K is defined as the most frequently used list evaluation measure that incorporates consideration of the position of correctly recommend items. First, the discounted cumulative gain (DCG) of a user is regarded as

$$\text{DCG@K} = \sum_{i=1}^{K} \frac{2^{hit_i} - 1}{\log_2(i+1)}$$

where

$$hit_i = \begin{cases} 1 & \text{if item } i^{th} \text{ in groud truth list} \\ 0 & \text{otherwise} \end{cases}$$

Because DCG is unequal among users, it is normalized as

$$\text{NDCG@K} = \frac{DCG@K}{IDCG@K},$$

where the ideal discounted cumulative gain is represented as IDCG.

$$\text{IDCG@K} = \sum_{i=1}^{|HIT|} \frac{2^{hit_i} - 1}{\log_2(i+1)}$$

Therein, $|HIT|$ is a list of ground truth up to position K.

The final result represents the average over all users.

5.4.3 Experimental Settings

I divide all users in each dataset randomly following 70% for training, 5% for validating to optimize hyperparameters, and 25% for testing. I train models using the entire click history of training users. In validation and test processes, I use a click vector of domain A to predict the click vector of domain B and vice versa.

The overall structure for Drama_Comedy and Romance_Thriller dataset is [I-200-100-50-100-200-I]: the first [100] is the shared layer in the encoder; the second [100] is the shared layer in the generator; [50] represents the latent vector dimension; and I stands for the number of products in domain A or B.

For the Amazon dataset, because the number of products in each domain is much greater than in the Movielens dataset, the overall structure for Health_Clothing and Video_TV dataset is [I-600-200-50-200-600-I], whereas the first [200] is share-layer in encoder, the second [200] is the share-layer in the generator, [50] is latent vector dimension, and I is the number of products in domain A or B. I also found that with a sparse dataset such as Amazon, adding a dropout layer to the input layer will yield a better result.

With each hidden layer in the encoder and generator, I apply a leaky ReLU activation function with a scale of 0.5. With the discriminator network, I use structure [100-1] for all datasets and apply tanh function for each hidden layer, except for the last layer.

All hyper-parameters demanded above are chosen based on Recall@50 in validation sets.

5.5 Performance Comparison

5.5.1 Baselines

The models included in my comparison are listed below:

- **CDL**: collaborative deep learning [47] is a probabilistic feedforward model for the joint learning of stacked denoising autoencoder (SDAE) and for collaborative filtering. For item contents, I combine the title and descriptions in Health_Clothing and Video_TV datasets and use movie descriptions from the IMBD website [4] for Drama_Comedy and Romance_Thriller datasets. Then I merge products of the two domains into one set. Subsequently, I follow the same procedure as that explained in [47] to preprocess the text information. After removing stop words, the top discriminate words according to the tf-idf values were chosen to form the vocabulary. I chose 8000 words for each dataset. Next, I use grid search and the validation set to ascertain the optimal hyperparameters. I search λ_u in [0.1,1,10], λ_v in [1, 10, 100] and λ_r in [0.1, 1, 10]. Results demonstrate that the two-layer model with detailed architecture as '8000-200-50-200-8000' yielded the best results in validation sets.

- **Multi-VAE**: Multi-VAE [31] is a collaborative filtering method that uses VAE to reconstruct a user–item rating matrix. I concatenate two user–item matrixes from two domains so that the click vector of user u is $[x_{1A}, x_{2A}, \cdots, x_{IA}, x_{1B}, \cdots, x_{IB}]$. Results indicate that structure '#products-600-200-50-200-600-#products' with a dimension of latent vector 50 yielded superior results in validation sets.

[4] https://www.imdb.com/

- **CCCFNET**: content-boosted collaborative filtering neural network [29] is a state-of-the-art hybrid method for cross-domain recommender systems. With a user, it utilizes a one-hot-encoding vector that extracts from a user–item rating matrix, but with the item, it combines both one-hot-encoding vector from the user–item matrix and item attributes. Then, after learning, user hidden representation will include collaborative filtering (CF) factors and content preference, whereas item hidden representation includes CF factors and item content representation. I combine text attributes as in CDL with a user–item matrix, so that with each domain, the item input vector is $[x_{u1}, x_{u2}, \cdots, x_{uN}, x_{w1}, x_{w2}, \cdots, x_{wS}]$ for which N is the number of users and S is 8000. The best neural network structure is '200-50'.

- **APR**: adversarial personal ranking [16] enhances Bayesian personal ranking with an adversarial network. I use publicly available source code provided by authors, but it cannot achieve competitive performance for the datasets used for this study. Therefore, I do not plot the results of APR in Figure 5.2

5.5.2 Cross-Domain Performance

Figure 5.2 presents Recall and NDCG results of Multi-VAE, CDL, CCCFNET, and D2D-TM for each domain in four datasets. In light of these results, I have made the following observations.

- With **Multi-VAE**, it has some similar characteristics with my model such as utilization of user interaction vectors as input and learning features through VAE. A salient difference is that my model can learn differences of two domains in low-levels of encoder and generate them in high-levels of generator. Results demonstrate that if two domains differ in a certain attribute (Romance_Thriller and Drama_Comedy dataset), my model is only 2.9%–7.8% higher than Multi-VAE in Recall@50. However, with two domains that differ in many attributes such as Health, Personal Care, and Clothing, Jewelry in the Health_Clothing dataset, my model outperforms Multi-VAE by 44.8% in Recall@50. Another reason is that only VAE might let the system overfit while extracting features by VAE. In such cases, discriminating by GAN helps the system avoid overfitting. Therefore, it can learn latent features better. The result demonstrates that learning specific features of each domain and integrating VAE-GAN can enhance performance. I present more details about VAE-GAN in Section 4.5.1 as well as specific features of domains in Section 4.6.

- With **CDL**, although it is a hybrid method combined with text information, my model still can achieve superior performance to that of CDL by 17.9% (Thriller) to 129% (Health) in Recall@50. The first reason is similar to that for Multi-VAE: single-domain methods do not work well in multiple domains. Moreover, different from CDL, my model must only train some users who have many interactions in both domains, but it can conduct inference for all users. It not only reduces sparsity problems; it is also appropriate with real systems in cases for which no retraining is necessary when a new user comes.

- With **APR**, I are unable to obtain competitive performance. In addition to the same reasons given for Multi-VAE and CDL, another possible reason is that GAN might work well for generating problems but not for extracting features as VAE. In my model, VAE is the main model to learn features. The purpose of

GAN is supporting VAE in obtaining good features of two domains by trying to distinguish generations between them.

- Comparison with **CCCFNET**, a hybrid cross-domain method, demonstrates that my model can outperform it by 52.7% (Health) to 88.8% (Thriller) in Recall@50. A possible reason is that the VAE-GAN model can learn latent features better than the simple Multilayer perceptron model can.

All four algorithms in baselines worked with the assumption that a user's behavior does not change. Even with CCCFNET, the user behavior is modeled as a sole network. However, based on special characteristics of each domain, user behavior presents some differences among domains. For example, a user who is a saver has only bought inexpensive clothes, but for health care products, the user must follow a doctor's advice and might make purchases based on perceived effectiveness, not on price. My model can capture both similar and different features of user behavior. Therefore, it is reasonable that my model can outperform the baselines.

Figure 5.4 and Figure 5.5 respectively present the effectiveness of each component in my model as well as results of multinomial likelihood.

5.5.3 Single Domain Performance

My model outperforms not only in cross-domain problem but also single domain tasks. Figure 5.3 showed my results in Health_Clothing and Drama_Comedy datasets compared with Multi-VAE and CDL. Opposite with cross-domain, in single domain task, my model exceeded other models with high margin in case two domains are quite similar such as Drama movies and Comedy movies. That my NDCG@10 surpassed about 30% and 31% in Drama dataset as well as 12% and 28% in Comedy dataset compared with Multi-VAE and CDL respectively showed that my model pushed true positive items into higher position. In my model, addition knowledge learned from other domain provided a re-ranking sort which boosted homogeneous user behavior among domains. Inferred user behavior is determined based on not only similar users in current domain, but only similar others in additional domain.

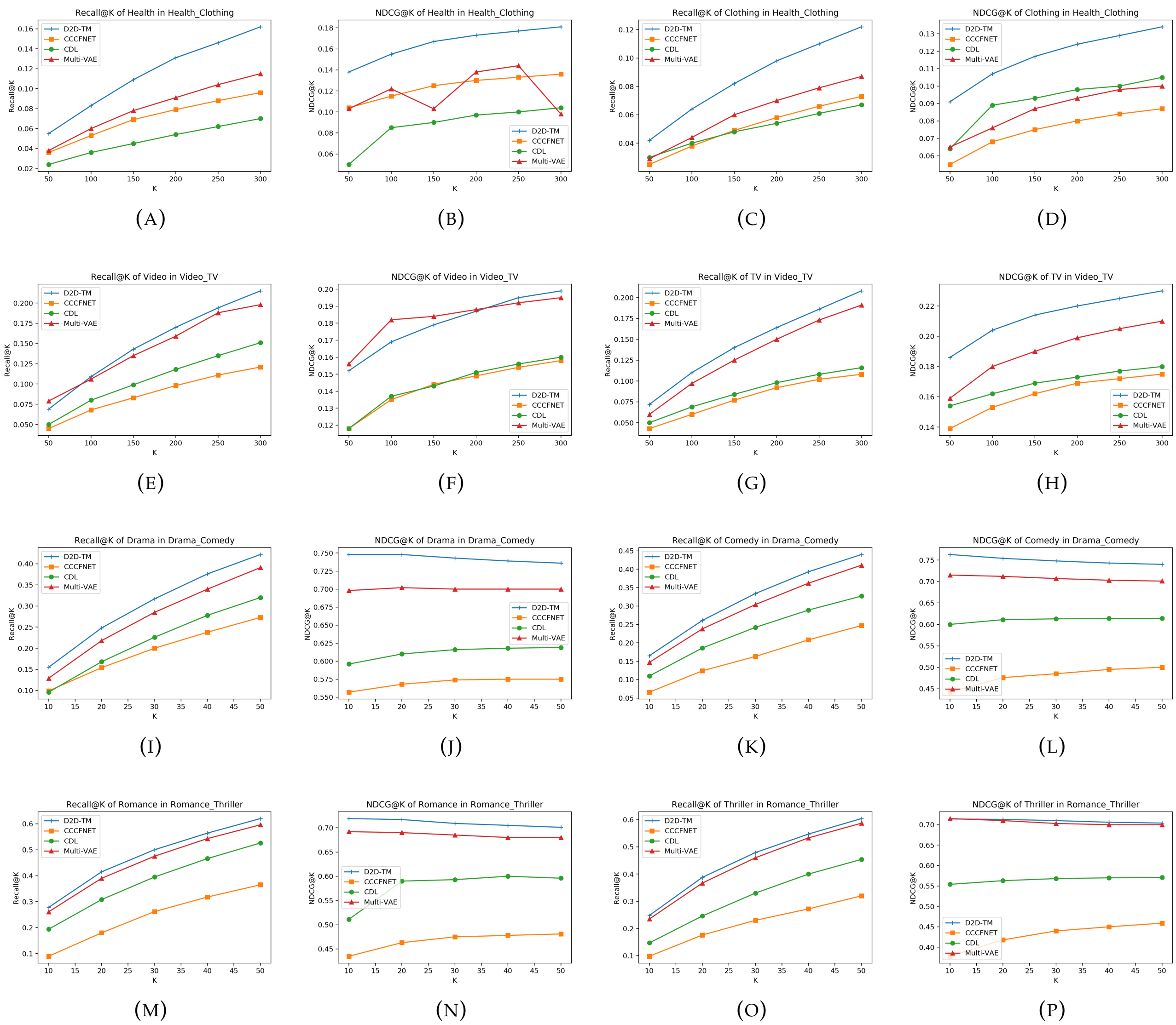


FIGURE 5.2: Recall and NDCG for cross-domain

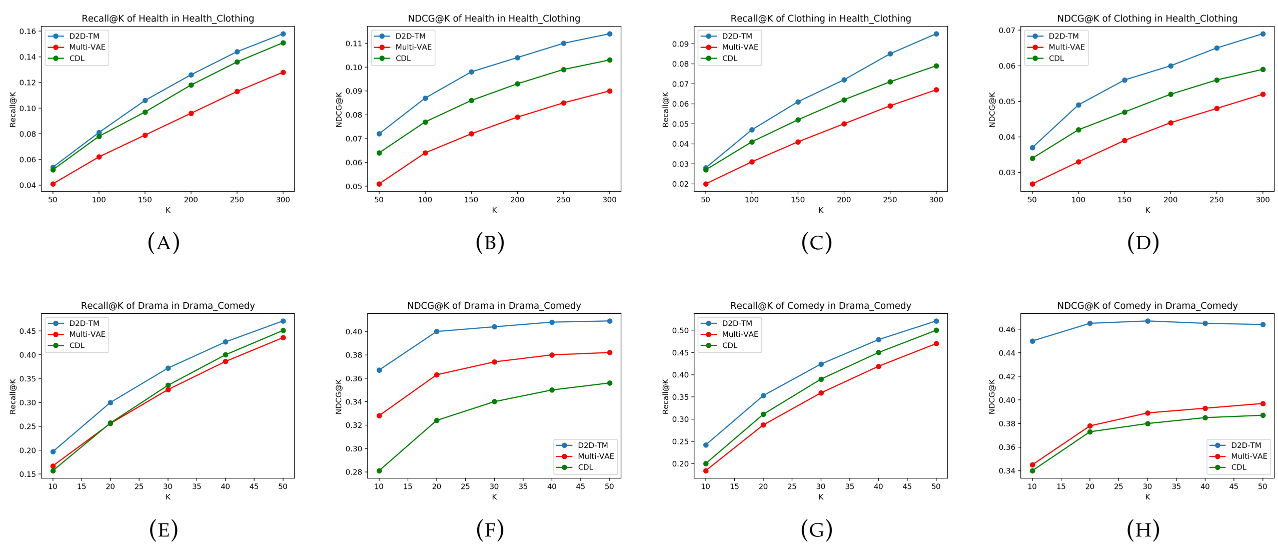
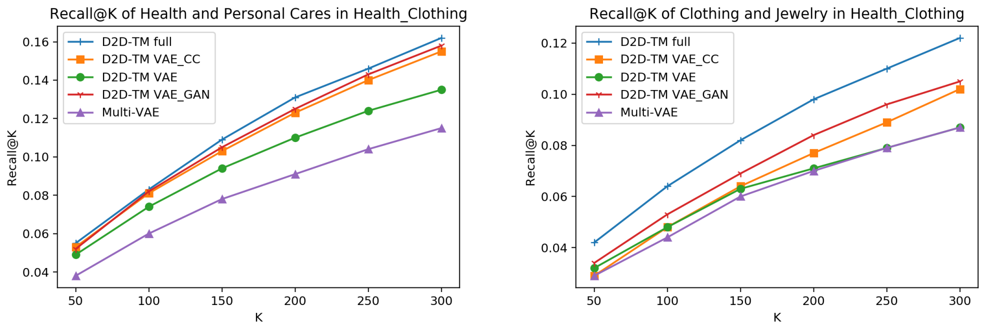
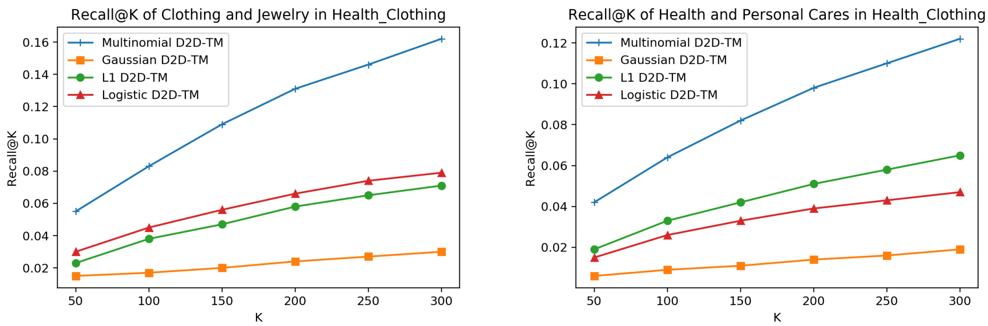


FIGURE 5.3: Recall and NDCG in same domain



(A) (B)

FIGURE 5.4: Comparing recall of model components in the Health_Clothing dataset.



(A) (B)

FIGURE 5.5: Comparing the recall of reconstruction loss functions for the Health_Clothing dataset.

5.5.4 Component

Because VAE is key model to learn latent features, I keep VAE and try to ignore CC, GAN, or both. I designate D2D-TM full, D2D-TM VAE_CC, D2D-TM VAE_GAN, and D2D-TM VAE respectively as my original model, model ignoring CC, ignoring GAN and ignoring both CC and GAN. Experiments presented in Figure 5.4 demonstrate that both CC and GAN are important to achieve high performance. However, results obtained for D2D-TM VAE_GAN are slightly better than those obtained for D2D-TM VAE_CC. A possible result is that GAN creates a strong constraint to distinct features of two domains so that VAE can avoid overfitting and extract latent features better.

Weight-sharing and CC are important parts by which similarity can be learned between two domains, shown as D2D-TM VAE_CC is higher than D2D-TM VAE 8.1% in Health and Personal Care.

The result that D2D-TM VAE is slightly better than Multi-VAE also demonstrates that learning different domains separately can improve performance.

5.5.5 Reconstruction Loss Function

In the UNIT framework, they use L1 loss for reconstruction. That is suitable with image data, but with click data, Multinomial log loss is more appropriate. Otherwise,

TABLE 5.2: List of Comedy movies the user watched

Input Comedy Movies	Genres
Do not be a Menace to South Central While Drinking Your Juice in the Hood (1996)	Comedy
Cocoon (1985)	Comedy, Sci-Fi
Galaxy Quest (1999)	Adventure, Comedy, Sci-Fi
Men in Black (1997)	Action, Adventure, Comedy, Sci-Fi
The Cable Guy (1996)	Comedy
Sleeper	Comedy, Sci-Fi
Back to the Future (1985)	Comedy, Sci-Fi
Beverly Hills Ninja (1997)	Action, Comedy
Back to the Future Part II (1989)	Comedy, Sci-Fi
10. The Adventures of Buckaroo Banzai Across the Eighth Dimension (1984)	Adventure, Comedy, Sci-Fi

many studies of RS used log likelihood (log loss) or Gaussian likelihood (square loss). Therefore, I experimented with loss of four types. With L1 loss, log loss, and square loss, activation function tanh can achieve superior results.

Figure 5.5 shows that the Multinomial log likelihood can outperform other types. A possible reason is that with the click dataset, each element in the input vector is 0 or 1. Therefore, the square loss and L1 loss are unsuitable. Otherwise, the click input is assumed to be generated from a multinomial distribution. Demonstrably, it is better than log likelihood.

5.6 Qualitative Comparison

To gain deeper insight into how D2D-TM is better than algorithms and examine single-domain algorithms such as Multi-VAE and multi-domain algorithms such as CCCFNET, I first examine an exemplary user in the Drama_Comedy dataset. I have a list Comedy movies this user watched in Table 5.2. Using it, I attempt to predict the top 10 Drama movies by D2D-TM, Multi-VAE, and CCCFNET. Multi-VAE gave a list of both Drama and Comedy movies, but I only take the top 10 Drama movies. Table 5.3 presents these predicted lists of movies obtained using the three algorithms.

In the list of Comedy movies this user watched, if I do not see the specific domain, it is reasonable to infer that this user likes to watch Action, Adventure, Comedy and Sci-Fi movies. For that reason, Multi-VAE gave suggestions that combine as many favorite genres as possible. Moreover, when looking deeper into the dataset, one can observe that Action movies are more popular than Adventure or Sci-Fi movies. If a movie belongs to many genres, I count one for each genre. Consequently, I have 155 Action movies, whereas Adventure and Sci-Fi movies are 76 and 53 respectively. That might be a reason most of the suggestions of Multi-VAE are Action movies (7/10).

However, CCCFNET suggested popular movies in the Drama domain that are not combined with other genres or combined with Thriller or Romantic. The reason is that CCCFNET can not obtain the mutual features of this user type in two domains such as attraction to mixed-genre movies or interesting movies combined with Sci-Fi. The reason is that CCCFNET optimizes user and item representation by summarizing the rating loss of two domains, which creates a weak constraint among

TABLE 5.3: Qualitative Comparison between D2D-TM, Multi-VAE, and CCCFNET to highlight the effectiveness of algorithms used specifically for multi-domain and algorithms specifically for a single domain. *Italic typeface* is used to denote correctly predicted movies.

Top 10 predicted drama movies (D2D-TM)	**Genres**
1. Star Wars: Episode V – The Empire Strikes Back (1980)	Action, Adventure, Drama, Sci-Fi, War
2. 2001: A Space Odyssey (1968)	Drama, Mystery, Sci-Fi, Thriller
3. E.T. the Extra-Terrestrial (1982)	Children's, Drama, Fantasy, Sci-Fi
4. Close Encounters of the Third Kind (1977)	Drama, Sci-Fi
5. The Day the Earth Stood Still (1951)	Drama, Sci-Fi
6. Contact (1997)	Drama, Sci-Fi
7. Starman (1984)	Adventure, Drama, Romance, Sci-Fi
8. Twelve Monkeys (1995)	Drama, Sci-Fi
9. Gattaca (1997)	Drama, Sci-Fi, Thriller
10. Deep Impact (1998)	Action, Drama, Sci-Fi, Thriller
Top 10 predicted Drama movies (Multi-VAE)	**Genres**
1. Braveheart (1995)	Action, Drama, War
2. Saving Private Ryan (1998)	Action, Drama, War
3. Star Wars: Episode V – The Empire Strikes Back (1980)	Action, Adventure, Drama, Sci-Fi, War
4. The Godfather (1972)	Action, Crime, Drama
5. Gladiator (2000)	Action, Drama
6. E.T. the Extra-Terrestrial (1982)	Children's, Drama, Fantasy, Sci-Fi
7. Stand by Me (1986)	Adventure, Comedy, Drama
8. The Patriot (2000)	Action, Drama, War
9. The Silence of the Lambs (1991)	Drama, Thriller
10. The Godfather: Part II (1974)	Action, Crime, Drama
Top 10 predicted Drama movies (CCCFNET)	**Genres**
1. A Civil Action (1998)	Drama
2. Gone with the Wind (1939)	Drama, Romance, War
3. Rules of Engagement (2000)	Drama, Thriller
4. Bringing Out the Dead (1999)	Drama, Horror
5. The General's Daughter (1999)	Drama, Thriller
6. Return to Me (2000)	Drama, Romance
7. Erin Brockovich (2000)	Drama
8. Frequent (2000)	Drama, Thriller
9. 2001: A Space Odyssey (1968)	Drama, Mystery, Sci-Fi, Thriller
10. The Man in the Iron Mask (1998)	Action, Drama, Romance

domains, and which presents difficulty capturing similar features of two domains. Moreover, if a user has few interactions in a domain, then the result will not be good.

Different from Multi-VAE and CCCFNET, most of the Drama movies D2D-TM suggested also belong to Sci-Fi (10/10). Checking the training dataset carefully revealed that there are 11 users who have similar behavior to that of the considered user. They rated about 20 Comedy movies, which are mostly combined Action, Adventure and Sci-Fi genres. They have more than five mutual movies selected with the considered user. When I examine Drama movies that they watched, all were interested only in Drama and Sci-Fi movies. They did not watch movies that are combined with Action. This result illustrated that D2D-TM can highlight the similarities and differences of user behavior in two domains based on the history of other users. It can also map these characteristics together.

5.7 Conclusion

This section presented a proposal of the D2D-TM network structure that is able to extract both homogeneous and divergent features among domains merely by using the user interaction history. This model is the first ever reported to apply VAE-GAN-CC to multi-domain RS. Results of the experiments described herein have demonstrated that my proposed model can strongly outperform state-of-the-art methods for recommendation while simultaneously providing more robust performance. My model outperforms single domain models because these models join items in two domains, then only can extract homogeneous features. In addition, my model transcends cross-domain models such as CCCFNET, which learns the domains separately, because they only can obtain divergent features. Thanks to being able to extract efficiently both homogeneous and divergent features, if two domains are different in many characteristics such as health care products and clothing products, D2D-TM is capable to outperform with high margin. Moreover, because my network uses only implicit feedback, it can be adopted easily for use by many companies. However, D2D-TM learns and infers with two domains only. In the future, I will improve D2D-TM into multi-domain models so that domains are not chosen by hand as current version, but all domains are learned, then system suggests not only interesting items but also interesting domains to users.

Chapter 6

Conclusion

6.1 Conclusion

In this dissertation, I introduced about recommender system sas well as deep learning models. A strength of deep learning models is they can extract latent representations from heterogeneous information. These latent representations assist recommender systems in achieving high performance as well as overcome the cold start problem.

I also addressed three problems of recommender system. The first one is the user cold start problem. How to give good recommendations to new users or users who have few interactions is important question concerning many recommender systems. My research provided collaborative multi-key learning (CML) model – an effective way to extract user behavior from implicit feedback without requiring user demographic data. My model can thus contribute to providing a rich user information source to achieve high performance even in cold start situations. I used two variational autoencoder networks to obtain user key vectors and item key vectors from auxiliary information. Then I proposed a probabilistic collaboration model with neural network to combine the key components with rating information. Experiments on real world datasets indicated that my collaboration model significantly outperforms other baselines.

In addition, I contributed with an update version of CML which is called as neural collaborative multi-key learning (NeuCML). In NeuCML, I proposed a denoising unbalanced autoencoder (DUAE) network instead of probabilistic matrix factorization (PMF) in CML to solve the low accuracy problem of PMF for new users. Both theory and experiments illustrate the advantage of DUAE in learning complex relationships among items when compared with PMF, especially with new users. Furthermore, I presented a method to combine DUAE with auxiliary information which possibly overcomes the problem of AE models for rating information.

The last problem concerns tedious suggestions. Tedious suggestions may not only lead users to leave the system, but also decrease the profit of providers. To solve the issue, I proposed a domain-to-domain translation model (D2D-TM) for cross-domain recommender system. With my model, RS can recommend items in domains in which the user does not have any interaction. My model is based on variational autoencoder (VAE) and generative adversarial network (GAN) to extract homogeneous and divergent features from domains. Domain cycle consistency (CC) constrains the inter-domain relations. The experiments demonstrated that only with a set of interaction history in a domain of a user, D2D-TM not only boosts the prediction results of the domain, but also infers items in other domains with high performance.

Through the use of deep learning models, I proposed collaboration models that cooperated many components information such as auxiliary information and rating of different domains to achieve high performance as well as to solve the existing problems of recommender systems.

6.2 Future Plan

Future work may delve deep into how to include other components into end-to-end networks. Some suggestions to improve cross-domain recommender systems are the following:

- Checking whether the model works with multi-domain simultaneously. This model allows systems to know not only which items a user may like, but also which categories the user may be interested next.
- Investigating whether D2D-TM has a higher performance if content information is used. Current D2D-TM can solve the cold start problem if a user is new in one domain and has some interactions in another domain. With cooperating content information, D2D-TM may solve cold start problem if a user is new in the system.

Computational costs should be investigated when implementing model into real recommender system. My current model uses one-hot-encoding for items with which a user has interactions; hence the computational cost is high if there are millions or billions of products. An embedding method for items may be suitable in the future.

Bibliography

[1] Charu C. Aggarwal. *Recommender Systems: The Textbook*. 1st. Springer Publishing Company, Incorporated, 2016. ISBN: 3319296574.

[2] Charu C. Aggarwal and Philip S. Yu. *Privacy-Preserving Data Mining: Models and Algorithms*. 1st ed. Springer Publishing Company, Incorporated, 2008. ISBN: 0387709916, 9780387709918.

[3] J. Bobadilla et al. "Recommender Systems Survey". In: *Know.-Based Syst.* 46 (July 2013), pp. 109–132. ISSN: 0950-7051.

[4] Xiaoyan Cai, Junwei Han, and Libin Yang. "Generative Adversarial Network Based Heterogeneous Bibliographic Network Representation for Personalized Citation Recommendation". In: *AAAI*. 2018.

[5] Iván Cantador et al. "Cross-domain recommender systems". In: *Recommender Systems Handbook*. Springer, 2015, pp. 919–959.

[6] Berkovsky Shlomo Cantador Iván Fernández-Tobías Ignacio and Cremonesi Paolo. *Cross-Domain Recommender Systems*. Boston, MA: Springer US, 2015, pp. 919–959. ISBN: 978-1-4899-7637-6. DOI: 10.1007/978-1-4899-7637-6_27. URL: https://doi.org/10.1007/978-1-4899-7637-6_27.

[7] Jingyuan Chen et al. "Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention". In: *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM. 2017, pp. 335–344.

[8] Wei-Ta Chu and Ya-Lun Tsai. "A Hybrid Recommendation System Considering Visual Information for Predicting Favorite Restaurants". In: *World Wide Web* 20.6 (Nov. 2017), pp. 1313–1331. ISSN: 1386-145X. DOI: 10.1007/s11280-017-0437-1. URL: https://doi.org/10.1007/s11280-017-0437-1.

[9] Graham Cormode et al. "Anonymizing Bipartite Graph Data Using Safe Groupings". In: *Proc. VLDB Endow.* 1.1 (Aug. 2008), pp. 833–844. ISSN: 2150-8097. DOI: 10.14778/1453856.1453947. URL: http://dx.doi.org/10.14778/1453856.1453947.

[10] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. "A Multi-View Deep Learning Approach for Cross Domain User Modeling in Recommendation Systems". In: *Proceedings of the 24th International Conference on World Wide Web*. WWW '15. Florence, Italy: International World Wide Web Conferences Steering Committee, 2015, pp. 278–288. ISBN: 978-1-4503-3469-3. DOI: 10.1145/2736277.2741667.

[11] Ignacio Fernández-Tobías et al. "Cross-domain recommender systems: A survey of the state of the art". In: *Spanish Conference on Information Retrieval*. sn. 2012, p. 24.

[12] Yuyun Gong and Qi Zhang. "Hashtag Recommendation Using Attention-based Convolutional Neural Network". In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. IJCAI'16. New York, New York, USA: AAAI Press, 2016, pp. 2782–2788. ISBN: 978-1-57735-770-4.

[13] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016. ISBN: 0262035618.

[14] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[15] Ruining He and Julian McAuley. "Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering". In: *Proceedings of the 25th International Conference on World Wide Web*. WWW '16. Montréal, Québec, Canada: International World Wide Web Conferences Steering Committee, 2016, pp. 507–517. ISBN: 978-1-4503-4143-1. DOI: 10.1145/2872427.2883037. URL: https://doi.org/10.1145/2872427.2883037.

[16] Xiangnan He et al. "Adversarial Personalized Ranking for Recommendation". In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR '18. Ann Arbor, MI, USA: ACM, 2018, pp. 355–364. ISBN: 978-1-4503-5657-2. DOI: 10.1145/3209978.3209981.

[17] Xiangnan He et al. "Neural Collaborative Filtering". In: *Proceedings of the 26th International Conference on World Wide Web*. WWW '17. Perth, Australia: International World Wide Web Conferences Steering Committee, 2017, pp. 173–182. ISBN: 978-1-4503-4913-0. DOI: 10.1145/3038912.3052569. URL: https://doi.org/10.1145/3038912.3052569.

[18] Xiangnan He et al. "Outer Product-Based Neural Collaborative Filtering". In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. IJCAI'18. Stockholm, Sweden: AAAI Press, 2018, 2227–2233. ISBN: 9780999241127.

[19] Balázs Hidasi et al. "Session-based recommendations with recurrent neural networks". In: *arXiv preprint arXiv:1511.06939* (2015).

[20] Liang Hu et al. "Personalized recommendation via cross-domain triadic factorization". In: *Proceedings of the 22nd international conference on World Wide Web*. ACM. 2013, pp. 595–606.

[21] Hyunseok Hwang, Taesoo Jung, and Euiho Suh. "An LTV model and customer segmentation based on customer value: a case study on the wireless telecommunication industry". In: *Expert Syst. Appl.* 26 (2004), pp. 181–188.

[22] Gawesh Jawaheer, Martin Szomszor, and Patty Kostkova. "Comparison of Implicit and Explicit Feedback from an Online Music Recommendation Service". In: *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems*. HetRec '10. Barcelona, Spain: ACM, 2010, pp. 47–51. ISBN: 978-1-4503-0407-8.

[23] Su-Yeon Kim et al. "Customer segmentation and strategy development based on customer lifetime value: A case study". In: *Expert Systems with Applications* 31.1 (2006), pp. 101 –107. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2005.09.004. URL: http://www.sciencedirect.com/science/article/pii/S0957417405001934.

[24] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *CoRR* abs/1312.6114 (2013).

[25] Wonsung Lee, Kyungwoo Song, and Il-Chul Moon. "Augmented Variational Autoencoders for Collaborative Filtering with Auxiliary Information". In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. CIKM '17. Singapore, Singapore: Association for Computing Machinery, 2017, 1139–1148. ISBN: 9781450349185. DOI: 10.1145/3132847.3132972. URL: https://doi.org/10.1145/3132847.3132972.

[26] 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. Melbourne, Australia: ACM, 2015, pp. 811–820. ISBN: 978-1-4503-3794-6. DOI: 10.1145/2806416.2806527. URL: http://doi.acm.org/10.1145/2806416.2806527.

[27] Xiaopeng Li and James She. "Collaborative Variational Autoencoder for Recommender Systems". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '17. Halifax, NS, Canada: ACM, 2017, pp. 305–314. ISBN: 978-1-4503-4887-4.

[28] Zhi Li et al. "Learning from History and Present: Next-Item Recommendation via Discriminatively Exploiting User Behaviors". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '18. London, United Kingdom: Association for Computing Machinery, 2018, 1734–1743. ISBN: 9781450355520. DOI: 10.1145/3219819.3220014. URL: https://doi.org/10.1145/3219819.3220014.

[29] Jianxun Lian et al. "CCCFNet: A Content-Boosted Collaborative Filtering Neural Network for Cross Domain Recommender Systems". In: *Proceedings of the 26th International Conference on World Wide Web Companion*. WWW '17 Companion. Perth, Australia: International World Wide Web Conferences Steering Committee, 2017, pp. 817–818. ISBN: 978-1-4503-4914-7. DOI: 10.1145/3041021.3054207.

[30] Jianxun Lian et al. "xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '18. London, United Kingdom: ACM, 2018, pp. 1754–1763. ISBN: 978-1-4503-5552-0. DOI: 10.1145/3219819.3220023. URL: http://doi.acm.org/10.1145/3219819.3220023.

[31] Dawen Liang et al. "Variational Autoencoders for Collaborative Filtering". In: *Proceedings of the 2018 World Wide Web Conference*. WWW '18. Lyon, France: International World Wide Web Conferences Steering Committee, 2018, pp. 689–698. ISBN: 978-1-4503-5639-8. DOI: 10.1145/3178876.3186150. URL: https://doi.org/10.1145/3178876.3186150.

[32] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. "Unsupervised Image-to-Image Translation Networks". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 700–708.

[33] Julian McAuley et al. "Image-Based Recommendations on Styles and Substitutes". In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '15. Santiago, Chile: ACM, 2015, pp. 43–52. ISBN: 978-1-4503-3621-5. DOI: 10.1145/2766462.2767755. URL: http://doi.acm.org/10.1145/2766462.2767755.

[34] Weiqing Min et al. "Cross-Platform Multi-Modal Topic Modeling for Personalized Inter-Platform Recommendation." In: *IEEE Trans. Multimedia* 17.10 (2015), pp. 1787–1801.

[35] L. Nguyen and T. Ishigaki. "Collaborative Multi-key Learning with an Anonymization Dataset for a Recommender System". In: *2019 International Joint Conference on Neural Networks (IJCNN)*. 2019, pp. 1–9.

[36] L. Nguyen and T. Ishigaki. "D2D-TM: A Cycle VAE-GAN for Multi-Domain Collaborative Filtering". In: *2019 IEEE International Conference on Big Data (Big Data)*. 2019, pp. 1175–1180.

[37] Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. "Deep Content-based Music Recommendation". In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'13. Lake Tahoe, Nevada: Curran Associates Inc., 2013, pp. 2643–2651. URL: http://dl.acm.org/citation.cfm?id=2999792.2999907.

[38] Weike Pan et al. "Mixed factorization for collaborative recommendation with heterogeneous explicit feedbacks". In: *Information Sciences* 332 (2016), pp. 84–93.

[39] Ruslan Salakhutdinov and Andriy Mnih. "Probabilistic Matrix Factorization". In: *Proceedings of the 20th International Conference on Neural Information Processing Systems*. NIPS'07. Vancouver, British Columbia, Canada: Curran Associates Inc., 2007, pp. 1257–1264. ISBN: 978-1-60560-352-0.

[40] Andrew I. Schein et al. "Methods and Metrics for Cold-start Recommendations". In: *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '02. Tampere, Finland: ACM, 2002, pp. 253–260. ISBN: 1-58113-561-0.

[41] Suvash Sedhain et al. "AutoRec: Autoencoders Meet Collaborative Filtering". In: *Proceedings of the 24th International Conference on World Wide Web*. WWW '15 Companion. Florence, Italy: Association for Computing Machinery, 2015, 111–112. ISBN: 9781450334730. DOI: 10.1145/2740908.2742726. URL: https://doi.org/10.1145/2740908.2742726.

[42] Bracha Shapira, Lior Rokach, and Shirley Freilikhman. "Facebook single and cross domain data for recommendation systems". In: *User Modeling and User-Adapted Interaction* 23.2-3 (2013), pp. 211–247.

[43] Karen Sparck Jones. "A Statistical Interpretation of Term Specificity and Its Application in Retrieval". In: *Document Retrieval Systems*. GBR: Taylor Graham Publishing, 1988, 132–142. ISBN: 0947568212.

[44] Alessandro Suglia et al. "A Deep Architecture for Content-based Recommendations Exploiting Recurrent Neural Networks". In: *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*. UMAP '17. Bratislava, Slovakia: ACM, 2017, pp. 202–211. ISBN: 978-1-4503-4635-1. DOI: 10.1145/3079628.3079684. URL: http://doi.acm.org/10.1145/3079628.3079684.

[45] Jiaxi Tang and Ke Wang. *Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding*. 2018. arXiv: 1809.07426 [cs.IR].

[46] 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. San Diego, California, USA: ACM, 2011, pp. 448–456. ISBN: 978-1-4503-0813-7.

[47] 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. Sydney, NSW, Australia: ACM, 2015, pp. 1235–1244. ISBN: 978-1-4503-3664-2.

[48] Jun Wang et al. "IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models". In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '17. Shinjuku, Tokyo, Japan: ACM, 2017, pp. 515–524. ISBN: 978-1-4503-5022-8. DOI: 10.1145/3077136.3080786.

[49] Xiang Wang et al. "Item Silk Road: Recommending Items from Information Domains to Social Users". In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '17. Shinjuku, Tokyo, Japan: ACM, 2017, pp. 185–194. ISBN: 978-1-4503-5022-8. DOI: 10.1145/3077136.3080771.

[50] Xinxi Wang and Ye Wang. "Improving Content-based and Hybrid Music Recommendation Using Deep Learning". In: *Proceedings of the 22Nd ACM International Conference on Multimedia*. MM '14. Orlando, Florida, USA: ACM, 2014, pp. 627–636. ISBN: 978-1-4503-3063-3. DOI: 10.1145/2647868.2654940. URL: http://doi.acm.org/10.1145/2647868.2654940.

[51] Yining Wang et al. "A Theoretical Analysis of NDCG Type Ranking Measures". In: *COLT*. 2013.

[52] Chao-Yuan Wu et al. "Recurrent Recommender Networks". In: *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. WSDM '17. Cambridge, United Kingdom: Association for Computing Machinery, 2017, 495–503. ISBN: 9781450346757. DOI: 10.1145/3018661.3018689. URL: https://doi.org/10.1145/3018661.3018689.

[53] Yao Wu et al. "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. San Francisco, California, USA: ACM, 2016, pp. 153–162. ISBN: 978-1-4503-3716-8. DOI: 10.1145/2835776.2835837. URL: http://doi.acm.org/10.1145/2835776.2835837.

[54] Hong-Jian Xue et al. "Deep Matrix Factorization Models for Recommender Systems". In: *IJCAI*. 2017.

[55] Shuang-Hong Yang et al. "Collaborative Competitive Filtering: Learning Recommender Using Context of User Choice". In: *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '11. Beijing, China: Association for Computing Machinery, 2011, 295–304. ISBN: 9781450307574. DOI: 10.1145/2009916.2009959. URL: https://doi.org/10.1145/2009916.2009959.

[56] Fuzheng Zhang et al. "Collaborative Knowledge Base Embedding for Recommender Systems". In: *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: ACM, 2016, pp. 353–362. ISBN: 978-1-4503-4232-2.

[57] Shuai Zhang, Lina Yao, and Aixin Sun. "Deep learning based recommender system: A survey and new perspectives". In: *arXiv preprint arXiv:1707.07435* (2017).

[58] Yongfeng Zhang et al. "Joint Representation Learning for Top-N Recommendation with Heterogeneous Information Sources". In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. CIKM '17. Singapore, Singapore: ACM, 2017, pp. 1449–1458. ISBN: 978-1-4503-4918-5. DOI: 10.1145/3132847.3132892. URL: http://doi.acm.org/10.1145/3132847.3132892.

[59] Jun-Yan Zhu et al. "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks". In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. 2017, pp. 2242–2251. DOI: 10.1109/ICCV.2017.244.