



โครงการ

การเรียนการสอนเพื่อเสริมประสบการณ์

ชื่อโครงการ การกรองร่วมกันโดยยึดไอเทมและผู้ใช้เป็นหลักโดยใช้ตัวเข้ารหัส
อัตโนมัติร่วมกับโครงข่ายประสาทเทียมลึก
Both User-based and Item-based Collaborative Filtering
based on Autoencoders with Deep Neural Networks

ชื่อนิสิต นายณัฐพล นภาศัย เลขประจำตัว 593 36237 23
นายพัชรพล พรหมณี เลขประจำตัว 593 36432 23

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์
สาขาวิชา วิทยาการคอมพิวเตอร์

ปีการศึกษา 2562

คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

การกรองร่วมกันโดยยึดไอเทมและผู้ใช้เป็นหลักโดยใช้ตัวเข้ารหัส
อัตโนมัติร่วมกับโครงข่ายประสาทเทียมลึก

นายณัฐพล นภาสัย
นายพัชรพล พรหมณี

โครงการนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์ ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2562
ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

Both User-based and Item-based Collaborative Filtering based
on Autoencoders with Deep Neural Networks

Nattapon Napasai
Patcharapol Promanee

A Project Submitted in Partial Fulfillment of the Requirements
for the Degree of Bachelor of Science Program in Computer Science
Department of Mathematics and Computer Science
Faculty of Science
Chulalongkorn University
Academic Year 2019
Copyright of Chulalongkorn University

นายณัฐพล นภาศัย, นายพัชรพล พรหมณี : การกรองร่วมกันโดยยึดไอเทมและผู้ใช้เป็นหลัก โดยใช้ตัวเข้ารหัสอัตโนมัติร่วมกับโครงข่ายประสาทเทียมลึก. (Both User-based and Item-based Collaborative Filtering based on Autoencoders with Deep Neural Networks) อ.ที่ปรึกษาโครงการหลัก : รศ. ดร.ศรันญา มณีโรจน์, อ.ที่ปรึกษาโครงการร่วม : ผศ. ดร.มนนัทธ์ พงษ์พานิช, 58 หน้า.

ระบบแนะนำเป็นส่วนสำคัญที่ช่วยให้ผู้ใช้สามารถใช้ข้อมูลได้อย่างมีประสิทธิภาพมากขึ้น ซึ่งระบบนี้พยายามที่จะแนะนำไอเทมให้กับผู้ใช้โดยใช้ข้อมูลจากการกระทำที่ผ่านมาของตัวผู้ใช้อเอง การกรองร่วมกันคือหนึ่งในวิธีการของระบบแนะนำที่ใช้ข้อมูลประเภทเรตติงของผู้ใช้หรือจำนวนการคลิกในการซื้อสินค้าของผู้ใช้ ซึ่งจะนำข้อมูลเหล่านั้นมาเปรียบเทียบระหว่างผู้ใช้หนึ่งคนกับผู้ใช้อื่น ๆ ที่มีลักษณะของข้อมูลที่คล้ายคลึงกัน การกรองร่วมกันสามารถแบ่งออกเป็น 2 ประเภทหลัก ๆ คือ การกรองร่วมกันโดยยึดไอเทมเป็นหลักและการกรองร่วมกันโดยยึดผู้ใช้เป็นหลัก โดยการกรองร่วมกันโดยยึดผู้ใช้เป็นหลักจะใช้ตรรกะที่ว่าคนที่มีความสนใจเหมือนกัน จะชอบอะไรคล้ายกัน ซึ่งการกรองร่วมกันโดยยึดผู้ใช้เป็นหลักจะใช้หลักการนี้ในการคำนวณหาเรตติงจากการใช้เรตติงของคนอื่นในการคำนวณร่วมด้วย สำหรับการกรองร่วมกันโดยยึดไอเทมเป็นหลักก็ใช้หลักการคล้ายกับที่กล่าวมาเช่นกันแต่เปลี่ยนจากผู้ใช้เป็นไอเทม ปัจจุบันงานวิจัยต่าง ๆ พยายามที่จะประยุกต์ใช้โครงข่ายประสาทเทียมกับการกรองร่วมกันเข้าด้วยกัน เพราะเมื่อประยุกต์เข้าด้วยกันแล้วจะช่วยให้สามารถทำลายขีดจำกัดที่ว่ากรองร่วมกันสามารถเรียนรู้ได้เฉพาะข้อมูลที่มีโครงสร้างข้อมูลเป็นเชิงเส้นเท่านั้นเนื่องจากโครงข่ายประสาทเทียมจะช่วยให้สามารถเรียนรู้ข้อมูลที่มีโครงสร้างข้อมูลแบบไม่เชิงเส้น ตัวเข้ารหัสอัตโนมัติมีเป้าหมายในการสร้างข้อมูลใหม่จากข้อมูลเดิมให้มีลักษณะคล้ายกันมากที่สุดโดยการนำข้อมูลนำเข้ามามีมิติสูงซึ่งข้อมูลที่ถูกลดมิติลงจะอยู่ที่ชั้นกลางและเราจะเรียกชั้นกลางนี้ว่าชั้นซ่อนเร้นซึ่งชั้นนี้เป็นส่วนที่สามารถใช้เป็นตัวแทนของมุลนำเข้าได้ จากนั้นขยายมิติของข้อมูลในชั้นกลางซึ่งทำให้ได้ข้อมูลส่งออกที่มีลักษณะคล้ายกับข้อมูลเดิม จากงานวิจัยที่ผ่านมา เราพบว่า ยังไม่มีการพิจารณาความสัมพันธ์ของผู้ใช้เป้าหมายกับผู้ใช้คนอื่น ๆ หรือไอเทมเป้าหมายกับไอเทมคนอื่น ๆ ซึ่งความสัมพันธ์เหล่านี้เป็นข้อดีของเทคนิคการกรองร่วมกัน ดังนั้น เราต้องการนำเสนอโมเดลระบบแนะนำจากตัวเข้ารหัสอัตโนมัติเพื่อเรียนรู้ตัวแทนที่แสดงลักษณะความคล้ายระหว่างผู้ใช้เป้าหมายกับผู้ใช้คนอื่น ๆ และไอเทมเป้าหมายกับไอเทมอื่น ๆ สุดท้ายนี้เราได้ทำการทดลองและพบว่าโมเดลที่เราเสนอมีประสิทธิภาพมากกว่าโมเดลอื่น ๆ ของงานวิจัยที่ผ่านมา

ภาควิชา.....คณิตศาสตร์และวิทยาการคอมพิวเตอร์.....ลายมือชื่อ นิสิต..... *ณัฐพล นภาศัย*
 ลายมือชื่อ นิสิต..... *พัชรพล พรหมณี*
 สาขาวิชา.....วิทยาการคอมพิวเตอร์.....ลายมือชื่อ อ.ที่ปรึกษาโครงการหลัก..... *ศรันญา มณีโรจน์*
 ปีการศึกษา.....2562.....ลายมือชื่อ อ.ที่ปรึกษาโครงการร่วม..... *มนนัทธ์ พงษ์พานิช*

5933623723, 5933643223 : MAJOR COMPUTER SCIENCE

KEYWORDS : RECOMMENDER SYSTEM, USER-BASED COLLABORATIVE FILTERING, ITEM-BASED COLLABORATIVE FILTERING, IMPLICIT FEEDBACK, NEURAL COLLABORATIVE FILTERING

NATTAPON NAPASAI, PATCHARAPOL PROMANEE : BOTH USER-BASED AND ITEM-BASED COLLABORATIVE FILTERING BASED ON AUTOENCODERS WITH DEEP NEURAL NETWORKS.

ADVISOR : ASSOC. PROF. SARANYA MANEEROJ, Ph.D., CO-

ADVISOR : ASST. PROF. MONNAT PONGPANICH, Ph.D., 58 pp.

Recommender systems have a major contribution, that is, it allows users to interact with content efficiently. A recommender system advises users by filtering items based on users' previous actions. Collaborative filtering (CF) is one of the recommender system algorithms which is built on explicit feedback (e.g., user ratings) and implicit feedback (e.g., number of clicks and purchases). It compares a target user with others who have similar preferences. Further, there are two well known types of CF: user-based CF and item-based CF. User-based CF assumes that people who have similar tastes tend to react to items similarly. For item-based CF, it tries to find look-alike items instead of look-alike users. Nowadays, many research attempt to apply the neural network into CF because there is a limitation in CF that CF can learn only linear representation, but the neural network can learn both linear and non-linear representation. Autoencoder reconstructs the input data in the output layer by encoding the input data into a low dimensional middle layer called the hidden layer to form latent representation, and then the output from the hidden layer is decoded by the output layer to reconstruct the data. From previous works, we have noticed that relations between a target user (item) and other users (other items) were not utilized and this relationship is a plus point for collaborative filtering technique. Therefore, we have proposed the autoencoder recommender system model that learns a representation of similarity between a target user (item) and other users (items). Finally, the experimental results have shown that the proposed model performs better than state of the art methods.

Department : Mathematics and Computer Science Student's Signature Nattapon Napsai
Student's Signature Parichamol pramee
Field of Study : Computer Advisor's Signature Chanya Mueoj
Academic Year : 2019 Co-advisor's Signature Monnat Pongpanich

ACKNOWLEDGEMENTS

This "Both user-based and item-based collaborative filtering based on autoencoders with deep neural networks" project has been completed. We would like to thank the individuals and groups for giving us advice and the best assistance. Both in academic and research operations, which are

My project supervisors, Assoc. Prof. Dr. Saranya Maneeroj and Asst. Prof. Dr. Monnat Pongpanich for providing advice and guidance for the research. As well as helping to review this report for completeness and encouragement for us always.

Asst. Prof. Dr. Dittaya Wanvarie for collaboration with AWS, which has a great impact on our project's success.

The project Examination Committee, Asst. Prof. Dr. Jaruloj Chongstitvatana and Asst. Prof. Dr. Krung Sinapiromsaran for the suggestions that made this project development even more complete.

Finally, we would like to express my gratitude towards my parents for supporting, advising and encouraging us throughout the research process.

CONTENTS

	Page
ABSTRACT IN THAI	iv
ABSTRACT IN ENGLISH.....	v
ACKNOWLEDGEMENTS	viii
CONTENTS	viii
LIST OF TABLES	x
LIST OF FIGURES.....	xi
CHAPTER I INTRODUCTION	1
1.1 Background and rationale.....	1
1.2 Objectives.....	2
1.3 Scope	2
1.4 Project activities	3
1.5 Benefits.....	4
1.6 Report outlines	5
CHAPTER II RELATED WORKS.....	6
2.1 Recommender systems	6
2.2 Content based methods.....	7
2.3 Cosine similarity	7
2.4 Collaborative filtering methods.....	8
2.5 Deep learning in the recommender system	10
2.6 Neural collaborative filtering	11
2.7 Autoencoder	12
2.8 Recommender system evaluation.....	15

	Page
CHAPTER III METHODOLOGY	17
3.1 Model	17
CHAPTER IV EXPERIMENTAL EVALUATION	22
4.1 Dataset	22
4.2 Evaluation metrics	23
4.3 Experimental results	24
CHAPTER V CONCLUSION	29
5.1 Conclusion	29
5.2 Suggestion	29
REFERENCES	30
APPENDIX A The Project Proposal of Course 2301399 Project Proposal Academic Year 2019	34
BIOGRAPHY	46

LIST OF TABLES

	Page
Table 1.1 The gantt chart that explains the processes of methods in the timeline.....	3
Table 4.1 The comparison between DHA-RS and UICF-AE	28
Table A.1 The gantt chart that explains the processes of methods in the timeline	42

LIST OF FIGURES

	Page
Figure 2.1 Example of company that use a recommender systems.....	6
Figure 2.2 Types of recommender systems algorithms.....	7
Figure 2.3 Content-based filter.....	7
Figure 2.4 Cosine similarity	8
Figure 2.5 Measuring the similarity between two users.....	8
Figure 2.6 Explicit feedback form.....	9
Figure 2.7 Implicit feedback form.....	9
Figure 2.8 User-based collaborative filtering.....	10
Figure 2.9 Item-based collaborative filtering	10
Figure 2.10 Neural collaborative filtering model.....	11
Figure 2.11 Multi-layer perceptron formula.....	12
Figure 2.12 Autoencoder	13
Figure 2.13 The autorec model.....	13
Figure 2.14 Autorec equation.....	14
Figure 2.15 A novel deep hybrid recommender system framework based on autoencoders.....	15
Figure 2.16 Normalized discounted cumulative gain formula.....	16
Figure 3.1 Both user-based and item-based collaborative filtering based on autoencoders with deep neural networks model.....	18
Figure 3.2 The user/item representation learning.....	19
Figure 3.3 The user/item similarity representation learning	20
Figure 3.4 User profile from a concatenation user representation and user similarity representation	20
Figure 3.5 Item profile from a concatenation item representation and item similarity representation	21
Figure 3.6 NCF framework	21
Figure 4.1 The movielens dataset 1M	22
Figure 4.2 The number of hidden layers effect on HR@10.....	25
Figure 4.3 The number of hidden layers effect on NDCG@10	26

	Page
Figure 4.4 The number of the output layer dimensions effect on HR@10	27
Figure 4.5 The number of the output layer dimensions effect on NDCG@10	27
Figure A.1 User-based CF	35
Figure A.2 Item-based CF	35
Figure A.3 Two approaches of autorec: user-based autorec (left) which takes user-item rating vector as the inputs and item-based autorec (right) which takes item-user rating vector as the input.....	37
Figure A.4 Both user-based and item-based collaborative filtering based on autoencoders with deep neural networks.....	39

CHAPTER I

INTRODUCTION

1.1 Background and rationale

With the growing size of information, the recommender system (RS) plays an important role, that is, allowing users to interact with content efficiently [1]. The recommender system advises users by filtering items based on users' previous actions. It has been employed by many sites (e.g., Netflix, Amazon, and Spotify) to suggest their products or services for customers to raise their incomes. There are three main types of recommender systems: content-based filtering (CBF), collaborative filtering (CF), and hybrid methods. CBF aims to recommend other similar items to users based on item features (e.g., item descriptions) from users' previous actions. However, sometimes item features cannot be gathered, so it is not possible to use CBF.

CF came to fix this limitation [2]. CF is built on explicit feedback (e.g., user ratings) and implicit feedback (e.g., number of clicks and purchases). It predicts user's ratings on items based on their past activities. Further, there are two well known types of CF: user-based CF and item-based CF. User-based CF (UBCF) assumes that people who have similar tastes tend to react to items similarly. This approach calculates the target user rating by using the user rating who has the highest similarity among users. Item-based CF (IBCF) tries to find items that look alike instead of look-alike users.

Nowadays, many research attempt to apply the neural network with CF. He et al. [9] proposed neural collaborative filtering (NCF) that can learn relations between users and items for rating prediction. Because there is a limitation in CF that CF can learn only data with the linear pattern, but the neural network can learn both data with the linear and non-linear pattern. A neural network can learn input features in both supervised and unsupervised ways. Besides CF, a neural network has been showing its capability in many fields, such as computer vision and natural language processing.

Meanwhile, an autoencoder is one of the most popular neural network models in the recommender system. Sedhain et al. [4] proposed autorec which took advantage of autoencoders for the CF framework. There are two well known types of autoencoders: (1) denoising autoencoder (DAE); (2) stacked denoising autoencoder (SDAE). Wu et al. [5] proposed collaborative denoising autoencoder (CDAE) which

took the corrupted input data and this allowed the model to perform better. Strub et al. proposed a hybrid recommender system based on autoencoders (CFN) which fed the corrupted input with side information to the autoencoder resulting in reducing cold-start problems. SDAE is the extended version of DAE by adding layers symmetrically. Encoding layers of SDAE are corrupted to make the model have better performance than DAE. Dong et al. [7] used SDAE for user representation and item representation with integrating side information for matrix factorization. Liu et al. [8] used two SDAE for user representation and item representation from auxiliary information to form user and item latent vectors then fed to the NCF framework for the rating prediction.

According to previous methods, there are two ways of using an autoencoder in the recommender system: predicting rating from the user-item rating matrix and learning either user representation or item representation at the bottleneck layer. The uses of autoencoders in the second approach have shown that there was no information of other users or other items considered because of the scope of mini-batch training, which could not get the information of all users and items at the same time. This means they did not consider the relations among users or among items which are the UBCF and the IBCF advantages, respectively.

In this work, we chose to extend the use of autoencoders in the second approach because autoencoders have a powerful representation learning. We included the advantages of UBCF and IBCF which incorporate user and item relations in order to increase the accuracy of the recommendation.

1.2 Objectives

1. To propose two more autoencoders that aim to learn user similarity representation and item similarity representation from user-user similarity matrix and item-item similarity matrix, respectively.

2. To use the NCF framework: Multi-Layer Perceptron (MLP) to learn user-item relations and predict the rating.

1.3 Scope

1. Using a dataset named MovieLens-1M, which contains 6040 users, 3706 items, and a million ratings. The max rating is 5. The min rating is 1. Zero means there is no rating for the movie.

2. Evaluation metrics are the hit rate (HR) and the normalized discounted cumulative gain (NDCG).

3. The dataset has to contain users, items, and ratings.

1.4 Project activities

A. project Plan

1. Do a literature review on the recommender system.
2. Identify the problems and limitations of previous works.
3. Design and analyze the improved method of the recommender system.
4. Implement the proposed system.
5. Evaluate the accuracy of the proposed system.
6. Analyze and discuss the experimental results.
7. Do the documentation.

B. Schedule

Table 1.1 :The gantt chart that explains the processes of methods in the timeline.

Project Activities	Year 2019						Year 2020			
	Jul.	Aug.	Sep.	Oct.	Nov.	Dec.	Jan.	Feb.	Mar.	Apr.
1. Read research papers and academic articles about the recommender system.										
2. Identify the problems and limitations of previous works.										
3. Design and analyze the improvement method of the										

recommender system.										
4. Develop and test the efficiency of the proposed system.										
5. Experiment to evaluate the accuracy of the proposed system.										
6. Analyze and discuss the experiment result.										
7. Provide the report documentation.										

1.5 Benefits

1.5.1 Benefits for users

1. Users have a new recommender system, which has better accuracy than the previous work.
2. Possibly increase the product sales.
3. Possibly increase the income of the user's company.
4. Users can possibly access the content efficiently.

1.5.2 Benefits for the system developers

1. Achieved both theoretical and practical knowledge in the recommender system field.
2. Learn new tools and programs, which are significant to develop the system.
3. Practice how to plan the work properly.
4. Improve problem-solving skills.

1.6 Report outlines

This report consists of five chapters as follows:

In chapter I, motivation, objective and scope of this project is described.

In chapter II, we review the technical knowledge and related work.

In chapter III, we present the methodology including how we preprocess the dataset and train the model.

In chapter IV, we present results and evaluate the performance.

In chapter V, we conclude and discuss our work.

CHAPTER II

LITERATURE REVIEW

2.1 Recommender systems

Recommender systems [10] are algorithms that aim to suggest relevant items to users and allow users to interact with content efficiently (items could be anything depending on industries, for example, items can be movies to watch, text to read, products to buy etc.) by filtering items based on users' previous actions. Many companies use recommender systems to suggest their products or services for customers to raise their incomes. Examples of companies that use the recommender systems are Amazon, Netflix, Facebook, and Google (Figure 2.1).

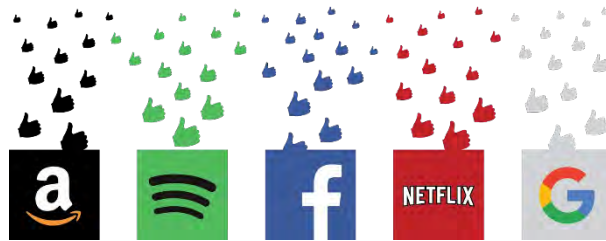


Figure 2.1 : Examples of companies that use recommender systems

Recommender systems can be divided into three main types: content based methods, collaborative filtering methods and hybrid methods (Figure 2.2). Collaborative filtering methods can be divided into two main types: user-based and item-based. Hybrid method is a combination of both methods.

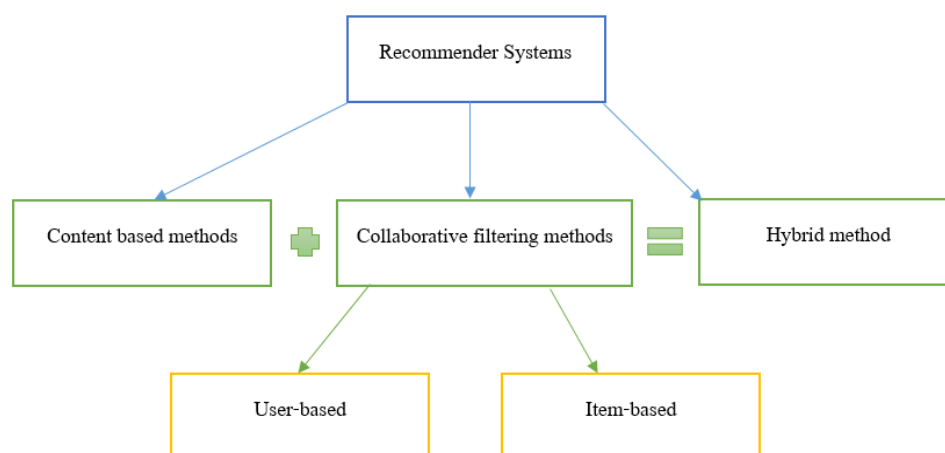


Figure 2.2 : Types of RS algorithms

2.2 Content based methods

Content based methods [11] are algorithms that aim to recommend other similar items to users based on item features (e.g., item descriptions) from users' previous actions. From figure 2.3, this user has watched some movies, the system will find a movie which is most similar to the previously watched movies from all movies in the system for recommending to the user.

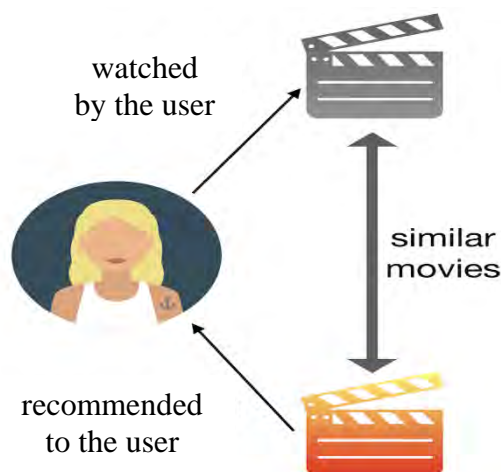


Figure 2.3 : Content-based filtering

2.3 Cosine similarity


Cosine similarity [12] measures the similarity between two vectors from an inner product space. It is often used to measure movies or user similarity in recommender system. From Figure 2.4, this is a formula of cosine similarity where A

and B are vectors with equal dimensions. From Figure 2.5, this is an example of finding cosine similarity between user 1 and user 2 based on a user rating matrix (left) to get a user-user similarity matrix (right).

$$\cos \theta = \frac{A \cdot B}{\|A\|_2 \|B\|_2} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

Figure 2.4 : Cosine similarity

Movie	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5
User 1	0	5	4	1	0
User 2	5	5	3	2	0



User	User 1	User 2
User 1	1	0.7776
User 2	0.7776	1

$$\text{CosineSim (user1,user2)} = \frac{(0*5) + (5*5) + (4*3) + (1*2) + (0*0)}{\sqrt{0^2+5^2+4^2+1^2+0^2} \sqrt{5^2+5^2+3^2+2^2+0^2}} = 0.77761579136$$

Figure 2.5 : Measuring the similarity between two users

2.4 Collaborative filtering methods

Collaborative filtering methods [13] are built on explicit feedback e.g., user ratings (Figure 2.6) and implicit feedback e.g., number of clicks and purchases (Figure 2.7). It compares the target user with others who have similar preferences. It is based on the logic that if people similar to person A (judging from the past) like this, so should the person A. There are two well known types of CF: UBCF and IBCF.



				
John 	5	1	3	5
Tom 	?	?	?	2
Alice 	4	?	3	?

Figure 2.6 : Explicit feedback form



				
John 	1	1	1	1
Tom 	?	?	?	1
Alice 	1	?	1	1

Figure 2.7 : Implicit feedback form

2.4.1 User-based collaborative filtering

UBCF assumes that people who have similar tastes tend to react to items similarly. For example, in Figure 2.8, when we want to recommend a movie to a target user, we need to search for the user who has the highest similarity (blue square on the right of figure 2.8). This can be done by calculating cosine similarity between the target user and other users. After we get the user who has the highest similarity, UBCF recommends a target item which is the movie that the target user did not watch before to the target user.

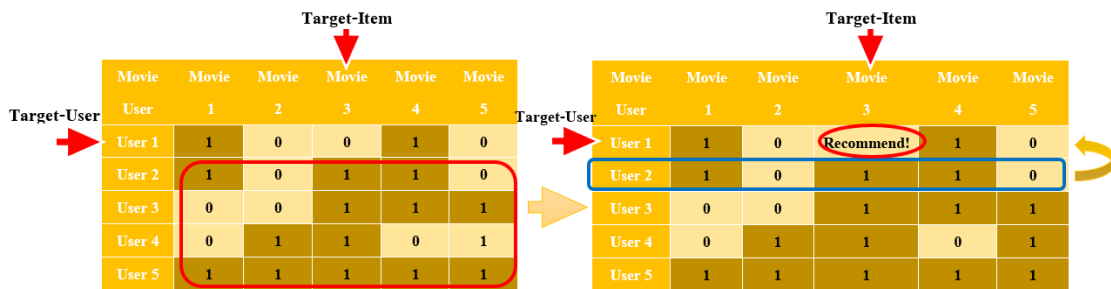


Figure 2.8 : User-based collaborative filtering

2.4.2 Item-based collaborative filtering

For IBCF, it tries to find items that look alike instead of look-alike users to recommend to the target user. In Figure 2.9, the approach focuses on using the reacted items which are items rated by the target user (the red square of Figure 2.9) to calculate cosine similarity between the item that the target user reacted with other items to search the item which has the highest similarity (blue column on the right of figure 2.9). The unreacted item with highest similarity to items found in the previous step (red ellipse in Figure 2.9) of the target user will be used to recommend to the target user.

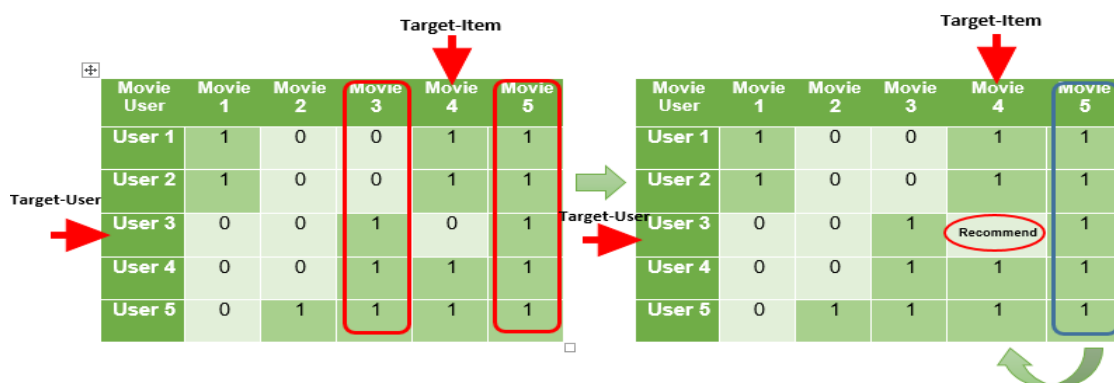


Figure 2.9 : Item-based collaborative filtering

2.5 Deep learning in the recommender system

The three types of the mentioned recommender systems have a limitation, namely only being able to learn data with the linear pattern but deep learning can learn both data with the linear and non-linear pattern. Therefore, deep learning plays an important role in the recommender system because it helps the model to learn more on non-linearly separable data. There are two main models that are commonly used in the

recommender system to predict ratings are neural collaborative filtering [9] and autoencoder.

2.6 Neural collaborative filtering

Neural collaborative filtering [9] came to fix the data with the non-linear pattern inseparable limitation of the collaborative filtering by adding part of the neural network for learning data with the non-linear pattern. Figure 2.10 shows the general framework of NCF which uses two types of input: (1) a user identity vector (one-hot encoding of user ID which is a binary vector whose dimension is equal to the number of users, where a value of one appears only one time and the rest of the values are zero) from the user identity matrix and (2) an item identity vector (one-hot encoding of item ID which is a binary vector whose dimension is equal to the number of items, where a value of one appears only one time and the rest of the values are zero) from the item identity matrix. Then, a user identity vector and an item identity vector are embedded to form a user latent vector and an item latent vector, respectively. A user latent vector is a vector which has meaning in terms of interaction among users. On the other hand, an item latent vector is a vector which has meaning in terms of interaction among items.

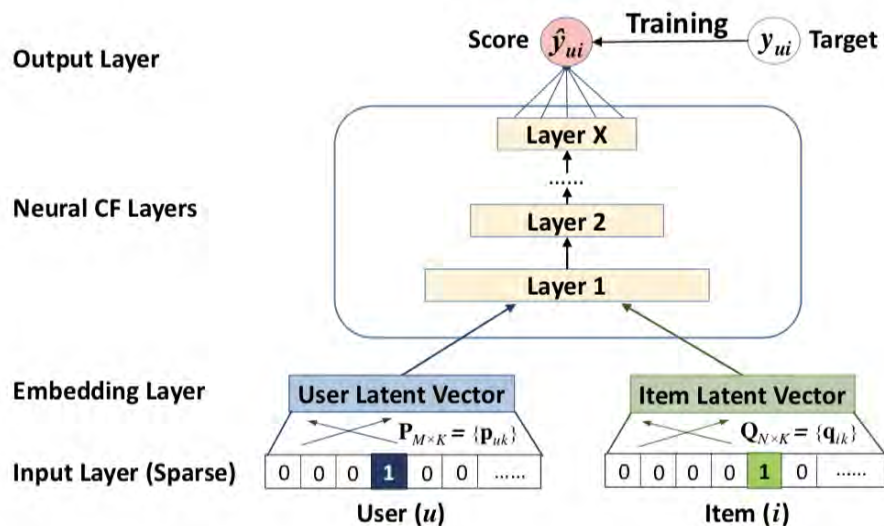


Figure 2.10 : Neural collaborative filtering model

After that, a concatenated vector between a target user latent vector and an target item latent vector is fed into a Multi-Layer Perceptron, which is called Neural CF Layers (Figure 2.10) in this work, which can extract the interaction between a target user and a target item from the concatenated vector. Finally, a rating prediction can be obtained from the last layer of MLP as in equations shown in Figure 2.11

$$\begin{aligned}
 \mathbf{z}_1 &= \phi_1(\mathbf{p}_u, \mathbf{q}_i) = \begin{bmatrix} \mathbf{p}_u \\ \mathbf{q}_i \end{bmatrix}, \\
 \phi_2(\mathbf{z}_1) &= a_2(\mathbf{W}_2^T \mathbf{z}_1 + \mathbf{b}_2), \\
 &\dots\dots \\
 \phi_L(\mathbf{z}_{L-1}) &= a_L(\mathbf{W}_L^T \mathbf{z}_{L-1} + \mathbf{b}_L), \\
 \hat{y}_{ui} &= \sigma(\mathbf{h}^T \phi_L(\mathbf{z}_{L-1})),
 \end{aligned}$$

Figure 2.11 : Multi-layer perceptron formula

where \mathbf{z}_1 is a concatenated vector from a user latent vector and an item latent vector

ϕ_1 is a concatenation between two vectors,

p_u or p_{uk} (Figure 2.10) is a user latent vector,

q_i or q_{ik} (Figure 2.10) is an item latent vector,

W_L is a weight of layer L,

b_L is a bias of layer L,

a_L is an activation function of layer L,

σ is a sigmoid function,

h is a weight of an output layer,

\hat{y}_{ui} is a rating prediction,

y_{ui} (Figure 2.10) is a target rating which is the real rating to use for comparison with the rating prediction in the training part.

2.7 Autoencoder

Nowadays, an autoencoder [14] plays a major role in the recommender system. The autoencoder is an unsupervised model of the neural network. It tries to reduce the dimension of input data by encoding the input data into a low dimensional middle layer called the hidden layer (bottleneck) to form a latent representation, and then the output from the hidden layer is decoded by reconstructing the data from the hidden layer as in

Figure 2.12. There are two main ways of using the autoencoder for recommender systems: using the autoencoder to predict rating and using the autoencoder for learning either user representation or item representation.

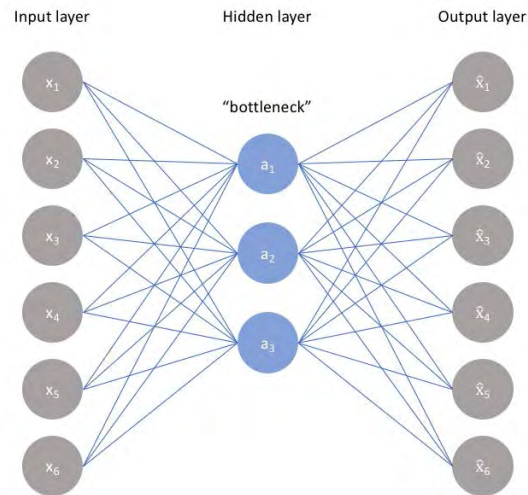


Figure 2.12 : Autoencoder

2.7.1 Using autoencoder to predict

Many works attempt to apply the autoencoder into recommender systems by using it to predict a rating for a target user and a target item. From Figure 2.13, autorec [4] is a model that uses an autoencoder that takes $r^{(i)}$ which is a user-item rating vector constructed from the user-item rating matrix as an input.

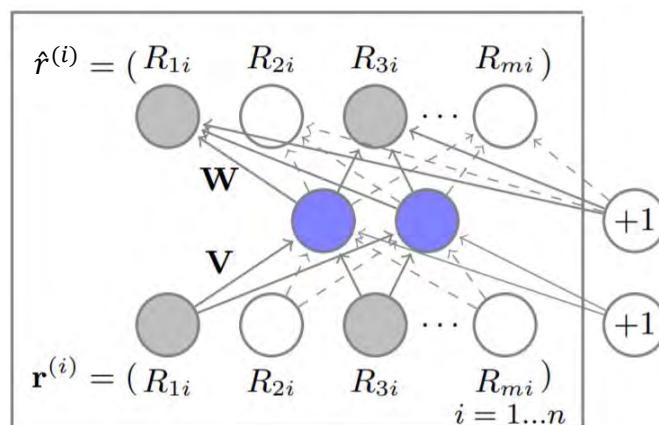


Figure 2.13 : The autorec model

A user-item rating vector is fed into an autoencoder and the output from the reconstruction of the autoencoder is the new rating from the rating prediction as the equation in Figure 2.14.

$$h(\mathbf{r}; \theta) = f(\mathbf{W} \cdot g(\mathbf{V}\mathbf{r} + \boldsymbol{\mu}) + \mathbf{b})$$

Figure 2.14 : Autorec equation

where $h(r; \theta)$ or $\hat{r}^{(i)}$ (Figure 2.13) is the new rating reconstructed from the autoencoder,

$r^{(i)}$ is a user-item rating vector,

V is a weight of encoding part,

W is a weight of decoding part,

μ is a bias of encoding part,

b is a bias of decoding part,

g is an activation function of encoding part,

f is an activation function of decoding part.

2.7.2 Using autoencoder to predict

The second way for using the autoencoder in the recommender system is using the autoencoder for learning either user representation or item representation at the bottleneck layer. A novel deep hybrid recommender system framework based on autoencoders (DHA-RS) [8] uses SDAE that takes a target user and a target item side information vector (a vector showing attributes associated with users or items such as gender, age, etc.) from a given user-item side-information matrix as an input for features-extraction to get user representation and item representation by minimizing errors between the output and the original user-item features. Then, a concatenation between the embedding of the user ID one-hot vector and the user representation forms the user profile and a concatenation between the embedding of the item ID one-hot vector and the item representation forms the item profile as shown in Figure 2.15 in part (i) feature extraction and ID embedding module. Then, they fed both the user profile and the item profile into DHA-RS framework to learn input to get user-item relation and

predict the rating from the last layer of Neural CF layers as Figure 2.15 in part (ii) neural collaborative filtering module.

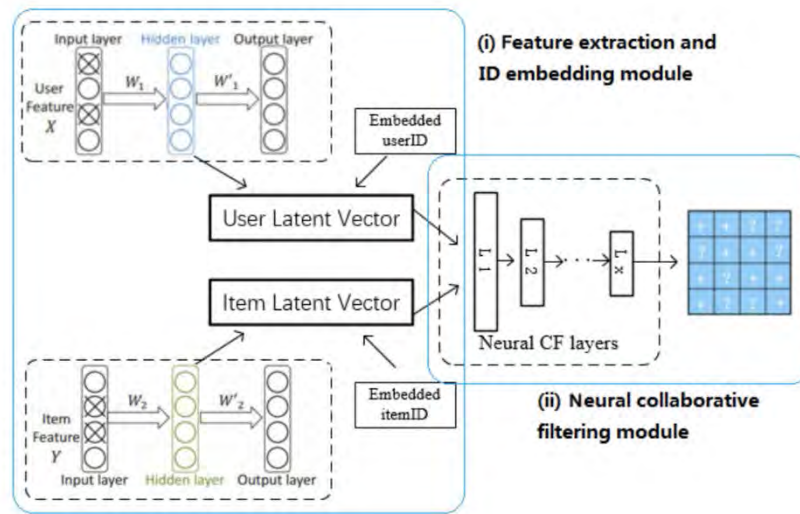


Figure 2.15 : A novel deep hybrid recommender system framework based on autoencoders

2.8 Recommender system evaluation

In the evaluation of the recommender system, there are several evaluation methods such as loss, accuracy. There are the two main evaluation measures that most researchers use to evaluate results of the recommender system: hit rate and normalized discounted cumulative gain.

2.8.1 Hit rate

The hit rate [15] is an evaluation of the recommender system that generates a top k recommendation list by sorted rating from given data. Similarly to prediction, a rating of all items associated with a target user is predicted. Then, a top k prediction list is generated by sorted rating prediction. After that, if each item in the top k prediction list is found in the top k recommendation list, HR will be increased by one. Finally, we repeat the process with all users to generate an average HR as a final HR score which is called HR@ k score because the number of top ranking used is k .

2.8.2 Normalized discounted cumulative gain

The normalized discounted cumulative gain [16] is an evaluation of the recommender system that generates a top k recommendation list by sorted rating from given similarly with HR but the difference is HR does not consider positions in top k that is HR increased when a predicted item is found in the top k recommendation list, but the NDCG considers positions in top k as follows (Figure 2.16).

$$\text{nDCG}_p = \frac{\text{DCG}_p}{\text{IDCG}_p}$$

$$\text{DCG}_p = \sum_{i=1}^p \frac{rel_i}{\log_2(i+1)}$$

$$\text{IDCG}_p = \sum_{i=1}^{|REL_p|} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

Figure 2.16 : Normalized discounted cumulative gain formula

where rel_i is a value of the rating,

i is an index,

p is the number of ranks,

REL_p is the number of ranks in descending order by the values.

As equation above, NDCG takes an index of a prediction item which is found in the top k recommendation list as a divisor. After that, we repeat the process with all users to generate an average NDCG as a final NDCG score which is called NDCG@k score, similar to HR.

According to previous research, the use of an autoencoder in the second approach (learning to get user representation or item representation), has shown that information from other users or other items is not considered because of the scope of mini-batch training, which could not get the information from all users and items at the same time. This means it did not consider the relations among users or among items which are the advantage of UBCF and IBCF, respectively. In the next chapter, we will explain the models that we propose to solve these problems.

CHAPTER III

METHODOLOGY

We propose a new autoencoder recommendation model that applies two more autoencoders which include advantages of UBCF and IBCF. The added autoencoders aim to learn user similarity representation from the user-user similarity matrix and item similarity representation from the item-item similarity matrix to form the user profile and the item profile. Consequently, the model should yield a more accurate predicted rating.

3.1 Model

We incorporated the user similarity representation and the user representation which form the UBCF latent-vector or user profile, and incorporated item similarity representation and the item representation which form the IBCF latent-vector or item profile. From Figure 3.1, we proposed both user-based and item-based collaborative filtering based on autoencoders with deep neural networks model (UICF-AE) which divided into two parts: (1) an autoencoder for learning the representation of the user/item and the user/item similarity and (2) a deep neural network for rating prediction part.

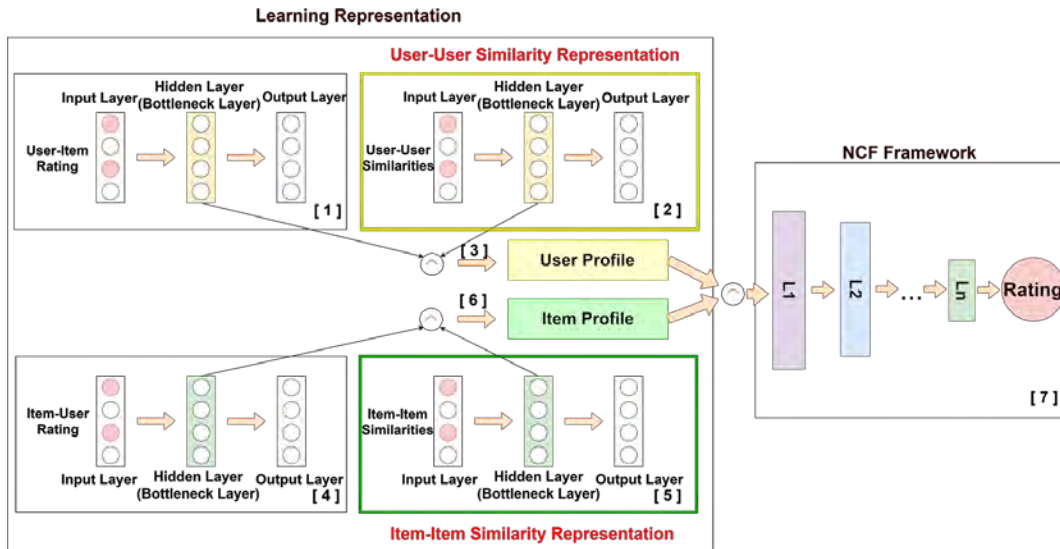


Figure 3.1 : Both user-based and item-based collaborative filtering based on autoencoders with deep neural networks model

First, we illustrate the overview of UICF-AE which integrates the latent feature representation of a user and an item with the deep neural network. From Figure 3.1, as previously mentioned, the proposed model is composed of the learning representation part and the deep neural network part. The first part is composed of the user/item representation learning part and the user/item similarity learning part. The first subpart of the learning representation part are the user/item representation learning which consists of two autoencoders. From number 1 in Figure 3.1, the first autoencoder takes a user-item rating vector constructed from the user-item rating matrix as the input. From number 4, Figure 3.1, the second autoencoder takes an item-user vector constructed from the item-user rating matrix as the input. The second sub-parts of the learning representation part is the user/item similarity representation learning which consists of two autoencoders. From number 2 in Figure 3.1, the first autoencoder takes a user-user similarity vector constructed from the user-user similarity matrix as the input. From number 5 in Figure 3.1, the second autoencoder takes an item-item similarity vector constructed from the item-item similarity matrix as the input. We incorporated two sub-parts of the learning representation part to form the user latent vector (Number 3 in Figure 3.1) and the item latent vector respectively (Number 4 in Figure 3.1). From number 7 in Figure 3.1, the second part is the rating predict part which is the deep neural network that takes the user latent vector and the item latent vector to predict rating.

3.1.1 The representation learning by autoencoder

We now explain each subpart in more detail. Each part has different objectives as follows:

1) The user/item representation learning

The learning representation consists of two autoencoders (Figure 3.2). From Figure 3.2 number 1, as previously mentioned, the first autoencoder takes a user-item rating vector from the user-item rating matrix as the input. From Figure 3.2 number 4, the second autoencoder takes an item-user vector from the item-user rating matrix as the input. For each autoencoder, once the input is fed into the autoencoder, representation of the input is learned in the hidden layer by minimizing the errors between the output and the input (Loss). If loss is very low, the representation is very good because it can reconstruct the original input even using a smaller feature dimension.

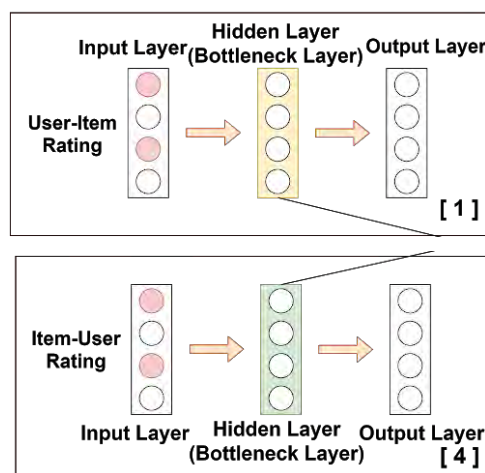


Figure 3.2 : The user/item representation learning

2) The user/item similarity representation learning

The learning representation consists of two autoencoders. From Figure 3.3 number 2, as previously mentioned, the first autoencoder takes a user-user similarity vector from the user-user similarity matrix (calculated by cosine similarity between user-item rating matrix) as the input. From Figure 3.3 number 5, The second autoencoder takes an item-item similarity vector from the item-item similarity matrix (calculated by cosine similarity between item-user rating matrix) as the input. The user-user similarity matrix and the item-item similarity matrix are matrices based on the

cosine similarity formula as discussed in chapter two. For each autoencoder, once the input is fed into the autoencoder, the representation of the input is learned in the hidden layer by minimizing the errors between the output and the input (Loss). If loss is very low, the representation is very good because it can reconstruct the original input using a smaller feature dimension.

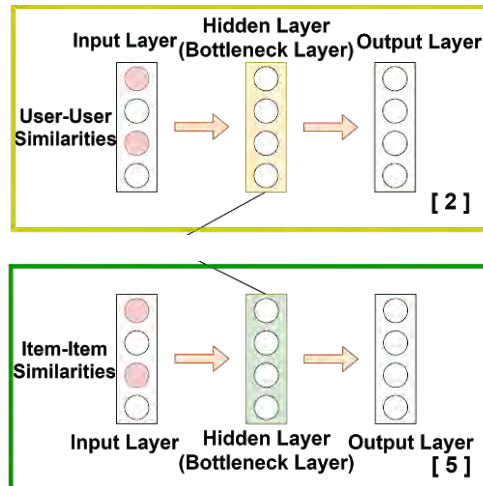


Figure 3.3 : The user/item similarity representation learning

3.1.2 Rating prediction with the deep neural network

After we get the user representation, the item representation, the item similarity representation, and the user similarity representation, we concatenate the user representation and the user similarity representation defined as the user profile (Figure 3.4) and concatenate the item representation and the item similarity representation defined as the item profile (Figure 3.5).

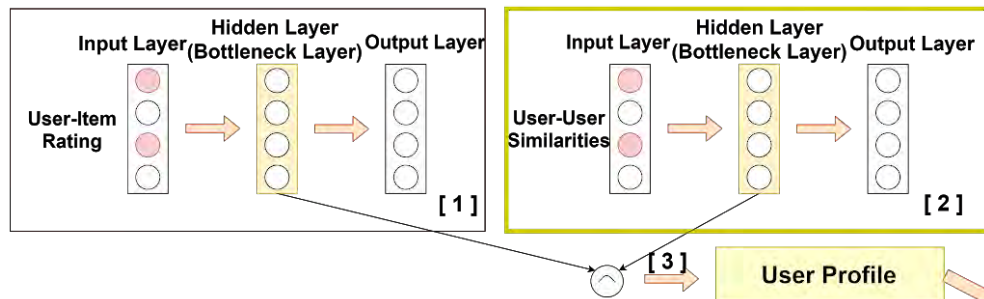


Figure 3.4 : User profile from a concatenation user representation and user similarity representation

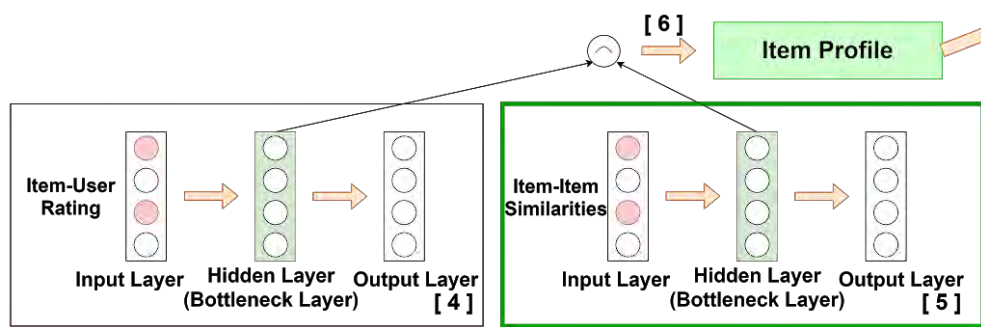


Figure 3.5 : Item profile from a concatenation item representation and item similarity representation

NCF is applied to learn user-item relations and predict the rating by feeding the user profile and the item profile into the NCF framework (Figure 3.6) based on the Multi-Layer Perceptron formula as discussed in chapter two. Weights and bias are used in calculations in each layer to reconstruct the input.

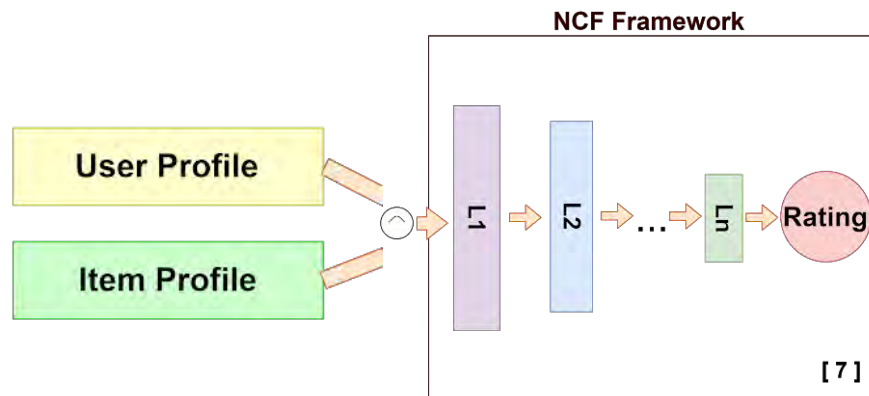


Figure 3.6 : NCF framework

CHAPTER IV

EXPERIMENTAL EVALUATION

We want to evaluate the proposed model by comparing it with the previous research which does not have the advantages of UBCF and IBCF. This chapter shows the details of the dataset, evaluation metrics and experimental results.

4.1 Dataset

We trained the model with MovieLens dataset 1M which is the benchmark dataset used in the previous work and retrieved from the Department of Computer Science and Engineering at the University of Minnesota. This dataset contains 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 MovieLens users who joined MovieLens in 2000. Each record consists of a user ID, a movie ID and its rated value (Figure 4.1).

	userId	movieId	rating	timestamp
0	1	1193	5	978300760
1	1	661	3	978302109
2	1	914	3	978301968
3	1	3408	4	978300275
4	1	2355	5	978824291
5	1	1197	3	978302268
6	1	1287	5	978302039
7	1	2804	5	978300719
8	1	594	4	978302268
9	1	919	4	978301368

Figure 4.1 : The movielens dataset 1M

We chose to use implicit feedback data because the implicit feedback data is much easier to collect in real-world applications [9]. We preprocessed ratings to implicit feedback by the condition that for each movie, if a movie has been rated by a user, then we set the rating to one, else we set the rating to zero. After that, we split the dataset

into 994169 rows for the train set and 6040 rows for the test set. The test set is constructed using a leave-one-out evaluation method since we want to recommend a rank-list as much as possible for each user. We used HR and NDCG which are rank-list evaluation metrics to evaluate the proposed model [15, 16].

4.2 Evaluation metrics

The recommender system that applied the neural network is difficult to evaluate accuracy of the predicted rating. Therefore, we evaluated recommendation rank lists which are a measure of ranking quality instead of evaluating accuracy. We chose the normalized discounted cumulative gain [16] and the hit rate [15] because they were the rank list metrics which are used in real life scenarios and also previous research. In our test set, there are 6040 records, one record per user with a rated movie value. We use negative sampling, that is, from the test set, we obtained a record from one user at a time (a movie in this record has been rated) and random sampling 99 movies that haven't been rated by this user.

4.2.1 The hit rate method

For the hit rate method [15], we fed the negative sampling data into the model and obtained predicted ratings from the model. Next, we sorted the predicted ratings and obtained the hit rate. If the movie ID that has been rated shows up in the top ten, we incremented the hit rate value by one. We repeat this process until all users have been sampled to get HR@10 which is an HR score from using the top ten rankings. For example, in a test set, we use one record which has user ID 0 and rated movie ID 47. We random 99 movie ID which is not rated to list as [1, 0, 0, 0, 0, 0,..., 0]. After that, the model will predict the rating from the negative sampling data, then get the new rating as [0.955, 0.456, 0.432, 0.0123,..., 0.998]. We sort the list by predicted rating. Next, we checked if the top ten movie IDs in the sorted predicted rating list match with the rated movie ID in the negative sampling data. If a match occurs, we incremented the hit rate value by one. Finally, we repeat this process until all users have been sampled and divided by the number of tests set to get HR@10.

4.2.2 The normalized discounted cumulative gain method

For the normalized discounted cumulative gain (NDCG) [16], we fed negative sampling data into the proposed model to get a result similar to the hit rate. We sorted predicted ratings and calculated DCG. If the movie has not been rated before, a term contributing to DCG value will be zero. If the movie has been rated before, a term is calculated according to the formula. As previously mentioned in chapter two, from the formula of IDCG in NDCG, our data set has an implicit feedback type that is binary data; therefore, we do not need to calculate IDCG because it will always be one. Therefore, it calculates only DCG to get NDCG@10 which is an NDCG score from using the top ten rankings. For example, we construct a test set from a record with user ID 0 and rated movie ID 47 then we random 99 movie ID which has not been rated to form a list as [1, 0, 0, 0, 0, 0, ..., 0]. After that, the model will predict the rating for the previous list and predicted ratings are obtained which might look like e.g., [0.955, 0.456, 0.432, 0.0123, ..., 0.998]. We sort the list by predicted ratings. Next, we checked if the top ten movie IDs in the sorted predicted rating list match with the rated movie ID in the negative sampling data. If a match occurs, we incremented the NDCG value by one and divide the NDCG value by the position of the movie ID 47 in the top ten list. Finally, we repeat this process until all users have been sampled and divided by the number of tests set to get NDCG@10.

4.3 Experimental results

We want to compare results between the proposed model (UICF-AE) which uses 4 autoencoders to extract the representation of user, item, user similarity (which shows the relation between users and takes advantage of UBCF) and item similarity (which shows the relation between items and takes advantage of IBCF). The four representations are fed to MLP to predict rating with a novel deep hybrid recommender system framework based on autoencoders model (DHA-RS) [8] which uses two autoencoders to extract representation of user and item and uses embedding of the user ID and the item ID to show the relation between users and between items but their method did not fully utilize the strength of CF. Therefore, we use the same evaluation methods as DHA-RS that are HR and NDCG to compare the performance of models. Further, we began by setting the hyperparameters based on DHA-RS [8] best hyperparameters. We set an adam optimizer to optimize the model and set the learning

rate to 0.0001, set the number of hidden layers of each autoencoder to 128 dimensions since autorec [4] experimental results have shown that the loss of autoencoder dramatically reduces when the number of hidden layers is more than 100, set the regularization to 100 and the loss parameter to 100 because DHA-RS [8] experimental results have shown that the model is not sensitive to this values, and set the batch size to 512. We refer to these sets of parameters as the base parameter model. For activation functions, we used relu activation function to all hidden layers in the proposed model because relu can reduce the vanishing gradient problem unlike sigmoid, relu is appropriate with sparse data, and relu allows the model to be less likely overfitting [9]. Except for the output layer of MLP we used sigmoid activation function to predict the implicit rating.

4.3.1 The multi-layer perceptron layers

We want to test how the number of MLP layers affects the model's performance. We set the output layer to 64 dimensions and tested on the number of MLP layers with one, two, three and four layers. For example, if the number of hidden layers is one, the MLP dimensions of each layer are 64 and 1, respectively. If the number of hidden layers is two, the MLP dimensions of each layer are 128, 64 and 1, respectively. We use the base parameter model to test and compare with DHA-RS to get the result shown in Figure 4.2 and Figure 4.3.

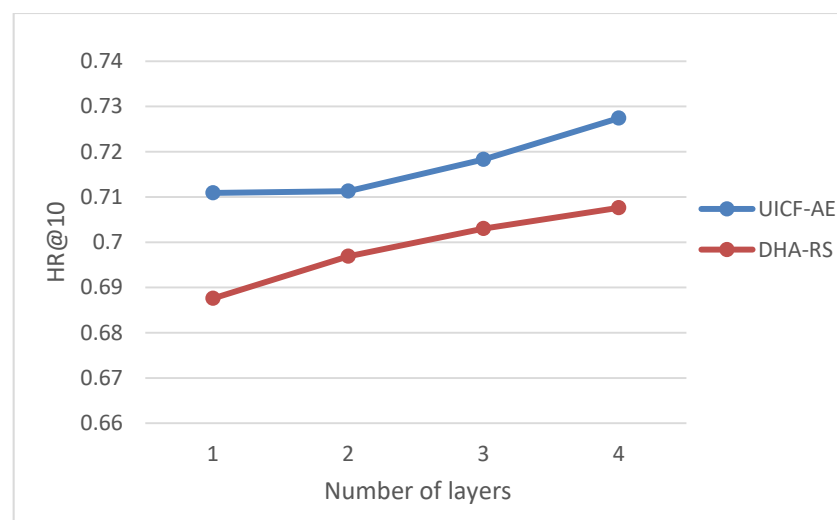


Figure 4.2 : The number of hidden layers effect on HR@10

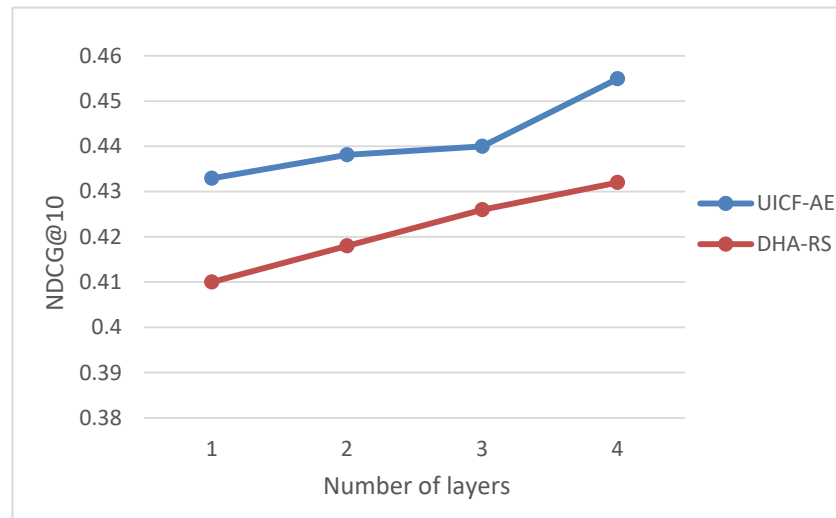


Figure 4.3 : The number of hidden layers effect on NDCG@10

From Figure 4.2, the best result of HR@10 of the proposed model on test set is MLP with four layers which has HR@10 of 0.7274 while DHA-RS has the highest HR@10 value less than 0.71 followed by MLP with three layers and MLP with two layers that have HR@10 values 0.7183 and 0.7113, respectively.

From Figure 4.3, the best result of NDCG@10 of the proposed model on the test set is MLP with four layers which has NDCG@10 of 0.4549 while DHA-RS has the highest NDCG@10 value less than 0.44 followed by MLP with three layers and MLP with two layers that have the NDCG@10 values 0.4400 and 0.4381, respectively.

4.3.2 The output layer dimensions of multi-layer perceptron

After we get the number of layers of MLP, we want to test how the number of the output dimensions of MLP layers affects the model's performance. We set the layer of MLP to 4 layers from the previous results (Section 4.2.1) and tested for the number of output layer dimensions of MLP with 64, 32, 16 and 8. For example, if the output layer dimensions of MLP has 64 dimensions, each layer of the MLP has dimensions with 512, 256, 128 and 64 respectively. If the output layer dimensions of MLP have 32 dimensions, each layer of the MLP has dimensions with 256, 128, 64 and 32 respectively. We use the base parameter model to test and compare with DHA-RS to get the result shown in Figure 4.4 and Figure 4.5.

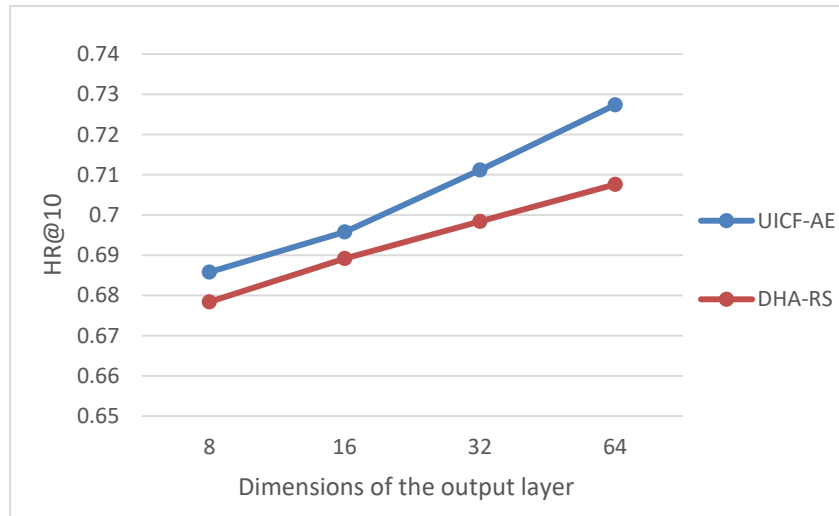


Figure 4.4 : The number of the output layer dimensions effect on HR@10

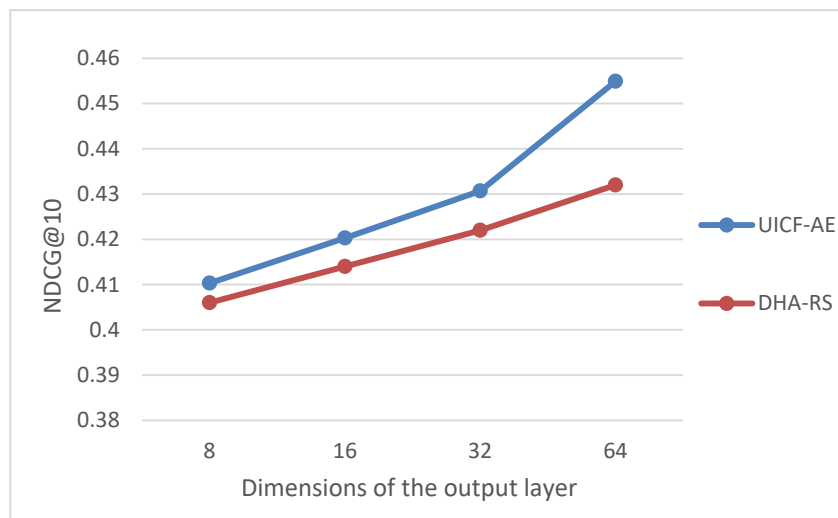


Figure 4.5 : The number of the output layer dimensions effect on NDCG@10

From Figure 4.4, the best result of HR@10 of the proposed model on the test set is the output layer dimensions with 64 dimensions which has HR@10 of 0.7274 while DHA-RS has the highest HR@10 value less than 0.71 followed by the dimension of the output layers with 32 dimensions and the dimension of the output layers with 16 dimensions that have HR@10 values 0.7112 and 0.6958, respectively.

From Figure 4.5, the best result of NDCG@10 of the proposed model the on test set is the output layer dimensions with 64 dimensions which has HR@10 of 0.4549 while DHA-RS has the highest HR@10 value less than 0.44 followed by the dimension of the output layers with 32 dimensions and the dimension of the output layers with 16 dimensions that have HR@10 values 0.4307 and 0.4203, respectively.

Table 4.1 : The comparison between DHA-RS and UICF-AE

Model	Evaluation method	
	HR@10	NDCG@10
DHA-RS	0.7076	0.4320
UICF-AE	0.7274	0.4549

From Table 4.1, the results show that when we switch from embedding user ID and item ID to using the representation similarity of items and users by adding autoencoders, it makes our model outperforms DHA-RS because DHA-RS did not consider the relations among users or among items but they used only embedding of the user ID and the item ID for showing the relations among users or among items. It is a relationship among users and among items in terms of positions which can not extract representation of other users that are similar with target users but, our model uses autoencoder to find the representation of similarity which utilizes the advantages of UBCF and IBCF. For this reason, the efficiency of our model is higher than DHA-RS.

CHAPTER V

CONCLUSION

5.1 Conclusion

In this work, we proposed UICF-AE which utilize the advantages of UBCF and IBCF by applying two more autoencoders to learn representations of similarity between a target user (Item) and other users (Items). UICF-AE consists of four autoencoders: (1) a user representation learning autoencoder, (2) an item representation learning autoencoder, (3) a user similarity representation learning autoencoder and (4) an item similarity representation learning autoencoder. Then, we concatenated the user representation and the user similarity representation to form a user profile and the item representation and the item similarity representation to form an item profile. After that, we fed a concatenated vector of the user profile and the item profile to MLP for rating prediction. Finally, the experimental results have shown that UICF-AE outperforms previous work for both metrics; HR is 2.8% higher and NDCG is 5.3% higher.

5.2 Suggestion

1. We find the similarities of each user and each item using only the cosine similarity method. There are many ways to find similarities, which may affect model performance.

2. From results 4.1.1, the performance of the model depends on the number of layers which tends to increase continuously with the number of layers.

3. From results 4.1.2, the performance of the model depends on the number of dimensions of the output layer which tends to increase continuously with the number of dimensions of the output layer.

4. In the NCF framework, we use only the multi-layer perceptrons. DHA-RS has shown that using generalized matrix factorization (GMF) is more efficient than MLP. However, our method outperformed DHA-RS when using MLP. This suggested that if GMF alone or both GMF and MLP are employed, a higher performance should be achieved.

REFERENCES

- [1] Dawen Liang; Rahul G. Krishnan; Matthew D. Hoffman; and Tony Jebara, Los Gatos, Cambridge, San Francisco, Variational Autoencoders for Collaborative Filtering. Proceedings of the 2018 World Wide Web Conference, 23–27 April, 2018, Lyon, France. Copyright 2018 ACM ISBN 978-1-4503-5639-8/18/04.
- [2] Hao Wang; Naiyan Wang; and Dit-Yan Yeung, Hong Kong University of Science and Technology, Collaborative Deep Learning for Recommender Systems. Proceedings of the 21th ACM SIGKDD International, 10-13 August, 2015, Sydney, NSW, Australia. Copyright 2015 ACM ISBN 978-1-4503-3664-2/15/08.
- [3] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2018. Deep Learning based Recommender System: A Survey and New Perspectives. Journal of ACM Computing Survey. 1, 1, Article 1 (July 2018), 35 pages.
- [4] Suvash Sedhain; Aditya Krishna Menon; Scott Sanner; and Lexing Xie, Australian National University/NICTA, AutoRec: Autoencoders Meet Collaborative Filtering. Proceedings of the 24th International Conference on World Wide Web, 18-22 May, 2015, Florence, Italy. Copyright 2015 ACM 978-1-4503-3473-0/15/05.
- [5] Yao Wu; Christopher DuBois; Alice X. Zheng; and Martin Ester, Simon Fraser University, Dato Inc., Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, 22-25 February, 2016, San Francisco, California, USA. Copyright 2016 ACM ISBN 978-1-4503-3716-8/16/02.
- [6] Florian Strub; Jer´emie Mary; and Romaric Gaudel, Hybrid Recommender System based on Autoencoders. Proceedings of the 1st Workshop on DLRS 2016, 15-15 September, 2016, Boston, MA, USA. Copyright 2016 ACM ISBN 978-1-4503-4795-2/16/11.

[7] Xin Dong; Lei Yu; Zhonghuo Wu; Yuxia Sun; Lingfeng Yuan; and Fangxi Zhang, A Hybrid Collaborative Filtering Model with Deep Structure for Recommender Systems. Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, 04-09 February, 2017, San Francisco, California, USA. Copyright 2017, Association for the Advancement of Artificial Intelligence (www.aaai.org).

[8] Yu Liu, Shuai Wang, M. Shahrukh Khan, and Jieyu He. 2018. A novel deep hybrid recommender system based on auto-encoder with neural collaborative filtering. Journal of Big Data Mining and Analytics. 1, 3, Article 1 (September 2018), 211-221 pages.

[9] Xiangnan He; Lizi Liao; and Hanwang Zhang, Neural Collaborative Filtering. Proceedings of the 26th International Conference on WWW '17, 03-07 April, 2017, Perth, Australia. Copyright 2017 ACM 978-1-4503-4913-0/17/04.

[10] Hemang Vyas. (2018). Code Your Own Popularity Based Recommendation System WITHOUT a Library in Python. Retrieve from <https://hackernoon.com/popularity-based-song-recommendation-system-without-any-library-in-python-12a4fbfd825e> [20 January2019]

[11] Emma Grimaldi. (2018). How to build a content-based movie recommender system with Natural Language Processing. Retrieve from <https://towardsdatascience.com/how-to-build-from-scratch-a-content-based-movie-recommender-with-natural-language-processing-25ad400eb243> [22 January2019]

[12] Selva Prabhakaran. (2018). Cosine Similarity – Understanding the math and how it works (with python codes). Retrieve from <https://www.machinelearningplus.com/nlp/cosine-similarity/> [25 January2019]

[13] Manish Barnwal. (2018). Types of data in recommender systems. Retrieve from <https://medium.com/theboredhuman/types-of-data-in-recommender-systems-7a1d76969137> [29 January2019]

[14] Artem Oppermann. (2018). Deep Autoencoders For Collaborative Filtering. Retrieve from <https://towardsdatascience.com/deep-autoencoders-for-collaborative-filtering-6cf8d25bbf1d> [30 January2019]

[15] Susan Li. (2019). Evaluating A Real-Life Recommender System, Error-Based and Ranking-Based. Retrieve from <https://towardsdatascience.com/evaluating-a-real-life-recommender-system-error-based-and-ranking-based-84708e3285b> [30 January2019]

[16] Pranay Chandekar. (2013). Evaluate your Recommendation Engine using NDCG. Retrieve from <https://towardsdatascience.com/evaluate-your-recommendation-engine-using-ndcg-759a851452d1> [30 January2019]

APPENDICES

APPENDIX A

The Project Proposal of Course 2301399 Project Proposal Academic Year 2019

Project Title (Thai)	การกรองร่วมกันโดยยึดไอเทมและผู้ใช้เป็นหลักโดยใช้ตัวเข้ารหัส อัตโนมัติร่วมกับโครงข่ายประสาทเทียมลึก
Project Title (English)	Both User-based and Item-based Collaborative Filtering based On Autoencoder with Deep Neural Networks
Project Advisor	1. Assoc. Prof. Dr. Saranya Maneeroj 2. Asst. Prof. Dr. Monnat Pongpanich
By	1. Patcharapol Promanee 5933643223 2. Nattapon Napasai 5933623723 Computer Science Program, Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University

Background and Rationale

With the growing size of information, the recommender system plays an important role, that is, allowing users to interact with content efficiently [1]. The recommender system tries to advise users by filtering items based on users' previous actions. It has been employed by many sites (e.g., Netflix, Amazon, and Spotify) to provide their products or services for customers to raise their incomes. There are three main types of recommender systems: content-based filtering (CBF), collaborative filtering (CF), and hybrid methods. CBF aims to recommend other similar items to users based on item features (e.g., item descriptions) from users' previous actions. However, item features sometimes can not be gathered, so it is not possible to use CBF. CF came to fix this limitation [2]. CF are built on explicit feedback (e.g., user ratings) and implicit feedback (e.g., number of clicks and purchases). It compares the target user with others who have similar preferences. It is based on the logic that people like you (judging from the past) like this and so should you. Further, they are two well known types of CF:

user-based and item-based. User-based CF assumes that people who have similar tastes tend to react to items similarly. This approach calculates the target user rating by using other users' rating (Figure A.1). In this figure, among other users (red square on the left of figure A.1), we used users who have the highest similarity (blue square on the right of figure A.1) to the target user to calculate the rating. For item-based CF, it tries to find items that look alike instead of users look alike (Figure A.2). In this figure, the approach focuses on using the reacted items (the red square on the left of figure A.2) of the target user. The highest similarity items (the blue square on the right of figure A.2) to the unreacted item of the target user will be used to calculate the rating of the unreacted item of the target user. The hybrid method is a combination of CBF and CF.

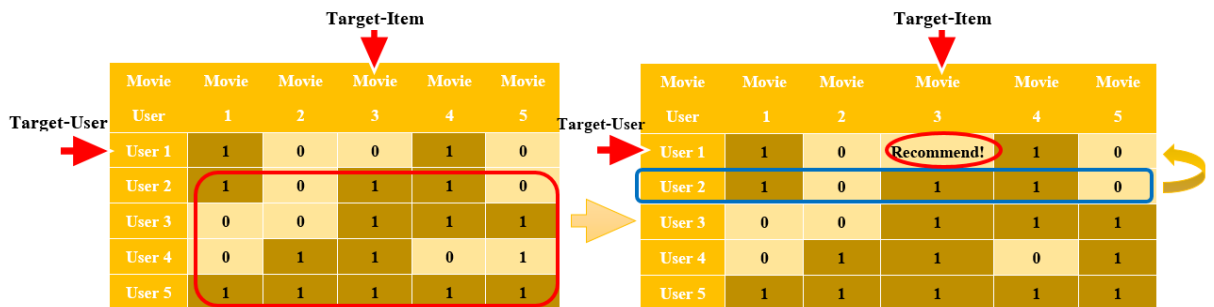


Figure A.1 : User-based CF

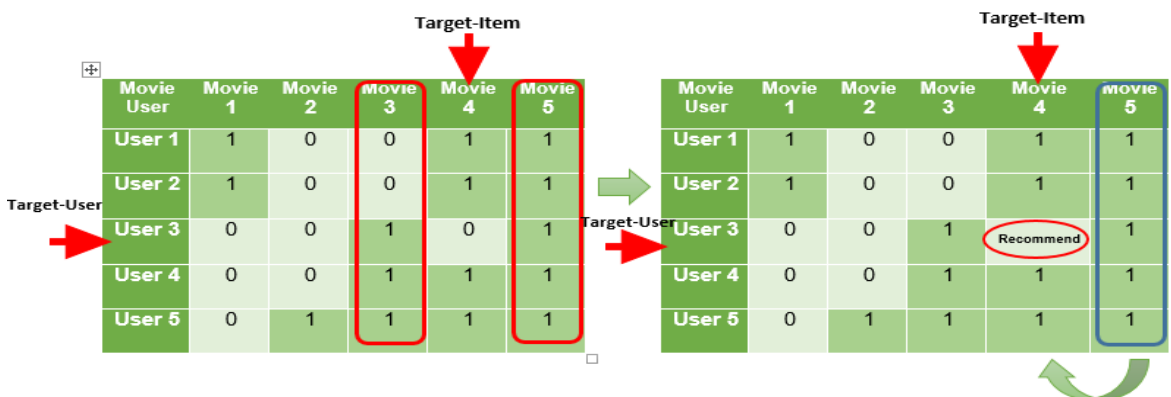


Figure A.2 : Item-based CF

Nowadays, many pieces of research attempt to apply the neural network into CF (neural network based CF) because there is a limitation in CF that CF can learn only

linear representation, but the neural network can learn both linear and non-linear representation [1]. A neural network can learn input features in both supervised and unsupervised ways. Besides CF, a neural network has been showing its capability in many fields, such as computer vision and natural language processing [9].

Autoencoder is one of the most popular neural network models in the recommender system. Autoencoder is an unsupervised model, which aims to reconstruct the input data in the output layer by encoding the input data into the middle layer called the hidden layer to form latent representation, and then the output from the hidden layer is decoded by the output layer to reconstruct the data. There are two well known types of autoencoder: (1) Denoising Autoencoder (DAE); (2) Stacked Denoising Autoencoder (SDAE). DAE takes the corrupted input data by adding noise, then the output layer aims to reconstruct the real input data from the latent representation of corrupted data to make the model robust. SDAE is the extended version of DAE by adding more layers symmetrically. Encoding layers of SDAE are corrupted to make the model more robust than DAE.

There are two ways of using autoencoders in the recommender system: predicting rating from the user-item rating matrix and learning either user representation or item representation at the bottleneck layer [3]. In the first approach, Sedhain et al. (2015) proposed AutoRec [4] which takes user-item rating vectors or item-user rating vectors as input and reconstructs it in the output layer to fill missing values in the rating matrix (Figure A.3). This figure shows two approaches which are depending on the input: user-based AutoRec takes user-item rating vector as the input (the green square on the left of figure A.3) and item-based AutoRec takes item-user rating vector as the input (the orange square on the right of figure A.3).

Movie User	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5
User 1	1	0	1	0	0
User 2	0	1	1	0	0
User 3	0	0	0	1	1
User 4	1	1	1	0	0
User 5	0	0	0	0	1

Movie User	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5
User 1	1	0	1	0	0
User 2	0	1	1	0	0
User 3	0	0	0	1	1
User 4	1	1	1	0	0
User 5	0	0	0	0	1

Figure A.3 : Two approaches of autorec: user-based AutoRec (left) which takes user-item rating vector as the inputs and item-based AutoRec (right) which takes item-user rating vector as the input

Another model called CDAE (Collaborative Denoising Auto-Encoders) [5], proposed after AutoRec, uses DAE to corrupt input data, which is implicit feedback instead of the rating. The output layer aims to reconstruct the real input data from the latent representation of corrupted data. An extension from CDAE called CFN (Hybrid Recommender System based on Autoencoders) [6] takes the input similar to AutoRec, but a corrupted version. It also incorporates side information (e.g., user profiles and item descriptions) to increase the prediction accuracy.

In the second approach, there are many models proposed. First, Hybrid Collaborative Filtering Model with Deep Structure for Recommender [7] aimed to learn user representation from user-item rating matrix and item representation from item-user rating matrix for Matrix Factorization Recommender using SDAE. Second, DHA-RS (A Novel Deep Hybrid Recommender System Framework based on Autoencoders) [8] improved recommendation accuracy by using SDAE to learn latent representation from users and items side information, and then they concatenated the embedding of the user ID one-hot vector with the user representation to form the user profile and the embedding of the item ID one-hot vector with the item representation to form the item profile. They fed the user profile and item profile to DHA-RS framework: GMF++ (Generalized Matrix Factorization++) and MLP++ (Multi-Layer Perceptron++) to learn user-item relation and predict the rating.

According to previous methods, the uses of autoencoder in the recommender system have shown that they used only the target user data and the target item data, but

there was no information of other users or other items considered. This means they did not consider the relations among users or among items.

In this work, we extend the use of autoencoder for user and item representation by including the advantages of user-based CF and item-based CF. Therefore, we propose to apply two more autoencoders that aim to learn user similarity representation and item similarity representation on user-user similarity matrix and item-item similarity matrix respectively (Figure A.4). The user-user similarity matrix is the matrix of cosine similarity between user-user ratings, and the item-item similarity matrix is the matrix of cosine similarity between item-item ratings. The first autoencoder aims to learn the user representation from the user-item rating matrix (Figure A.4, top left). The second autoencoder, which is the main part of the proposed model, aims to learn the user similarity representation from the user-user similarity matrix (Figure A.4, top right). The output vectors from the first autoencoder and the second autoencoder are concatenated to form the user profile. The third autoencoder aims to learn the item representation from the item-user rating matrix (Figure A.4, bottom left). The fourth autoencoder aims to learn the item similarity representation from the item-item similarity matrix (Figure A.4, bottom right). The output vectors from the third autoencoder and the fourth autoencoder are concatenated to form the item profile. Then, we do concatenation on the user profile and the item profile. After that, we feed the concatenated vector in the NCF [9] framework: MLP (Multi-Layer Perceptron) to learn user-item relations and predict the rating. Finally, we use Hit Rate (HR) and Normalized Discounted Cumulative Gain (NDCG) as the evaluation metrics to measure the accuracy of the model.

Movie User	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5
User 1	1	0	1	0	0
User 2	0	1	1	0	0
User 3	0	0	0	1	1
User 4	1	1	1	0	0
User 5	0	0	0	0	1

Figure A.4 : Both user-based and item-based collaborative filtering based on Autoencoders with deep neural networks

In this section, we will breakdown through the calculation of the proposed model. From figure A.4, we split the proposed model into two modules: Learning representation module and NCF framework module. The learning representation consists of four autoencoders; all of them generate the output as in equation 1:

$$O_l = \sigma(W_l^T O_{l-1} + b_l) \quad (1)$$

where O_l , σ , W_l , O_{l-1} and b_l denote the output of autoencoder at layer l , activation function, the weight matrix at layer l , the output of autoencoder from the previous layer and the bias vector at layer l , respectively. For the first layer, the computation will be defined as in equation 2:

$$O_1 = \sigma(W_1^T X + b_1) \quad (2)$$

where X is the input which is different among the four autoencoders. X is the user-item rating vector for the first autoencoder, user-user similarity vector for the second autoencoder, item-user rating vector for the third autoencoder and item-item

similarity vector for the fourth autoencoder. The loss function of autoencoders can be defined as in equation 3:

$$L_{An} = \|O_y - X\|_F^2 + \lambda_W \|W\|_F^2 + \lambda_b \|b\|_F^2 \quad (3)$$

where L_{An} is the loss of autoencoder at number n (Figure A.4), O_y is the output from last layer of the autoencoder, λ_W and λ_b is a regularization-term parameter of weight matrix and bias vector, respectively.

The NCF framework module consists of MLP which is used to learn user-item relations and predict the rating. The calculation of the NCF can be defined as in equation 4:

$$\begin{aligned} Z &= [U ; I] \\ O_1 &= \sigma(W_1^T Z + b_1) \\ O_2 &= \sigma(W_2^T O_1 + b_2) \\ &\dots \\ O_l &= \sigma(W_l^T O_{l-1} + b_l) \\ \hat{r}_{ui} &= \sigma(H^T O_l) \end{aligned} \quad (4)$$

where Z is the concatenated vector between user profile (U) and item profile (I), O_l is the output of MLP at layer l , W_l is the weight matrix at layer l , b_l is the bias vector at layer l , σ is the activation function, H^T is the weight matrix of rating prediction layer and \hat{r}_{ui} is the predicted rating of user U and item I . The loss function of NCF is defined as equation 5:

$$L_{NCF} = \sum_{(u,i) \in R \cup R^-} -(1 - r_{ui}) \log_2(1 - \hat{r}_{ui}) + r_{ui} \log_2(\hat{r}_{ui}) + \lambda_\theta \|\theta\|_F^2 \quad (5)$$

which is a binary cross-entropy, where R is the set of observed ratings, R^- is the set of unobserved ratings, λ_θ is a regularization-term of parameters, θ is the parameters, r_{ui} is an actual rating of user U and item I . Therefore, the total loss is defined as equation 6:

$$L_{total} = L_{NCF} + \alpha L_{A1} + \beta L_{A2} + \gamma L_{A3} + \delta L_{A4} \quad (6)$$

where α , β , γ and δ denote the hyperparameters of the loss function.

Objectives

1. To propose to apply two more autoencoders that aim to learn to get user similarity representation and item similarity representation from user-user similarity matrix and item-item similarity matrix respectively.
2. To use the NCF framework: MLP (Multi-Layer Perceptron) to learn to get user-item relations and predict the rating.

Scope

1. Using a dataset named MovieLens-1M, which contains 6040 users, 3706 items, and a million ratings. The max rating is 5. The min rating is 1. Zero means there is no rating for the movie.
2. Evaluation metrics are HR and NDCG.
3. The dataset has to contain users, items, and ratings.

Project Activities

A. project Plan

1. Do a literature review on the recommender system.
2. Identify the problems and limitations of previous works.
3. Design and analyze the improved method of the recommender system.
4. Implement the proposed system.
5. Evaluate the accuracy of the proposed system.
6. Analyze and discuss the experimental results.
7. Do the documentation.

B. Schedule

Table A.1 : the Gantt Chart that explains the processes of methods in the timeline.

Method	Year 2019						Year 2020			
	Jul.	Aug.	Sep.	Oct.	Nov.	Dec.	Jan.	Feb.	Mar.	Apr.
1. Read research papers and academic articles about the recommender system.										
2. Identify the problems and limitations of previous works.										
3. Design and analyze the improvement method of the recommender system.										
4. Develop and test the efficiency of the proposed system.										
5. Experiment to evaluate the accuracy of the proposed system.										
6. Analyze and discuss the experiment result.										

7. Provide the report documentation.										
--------------------------------------	--	--	--	--	--	--	--	--	--	--

Benefits

1. Benefits for users

- 1.1 Users get a new recommender system, which should be better in accuracy than the previous work.
- 1.2 Increase the product sales.
- 1.3 Increase the income of the user's company.
- 1.4 Users can access the content efficiently.

2. Benefits for the system developers

- 2.1 Achieved both theoretical and practical knowledge in the recommender system field.
- 2.2 Learn new tools and programs, which are significant to develop the system.
- 2.3 Practice how to plan the work properly.
- 2.4 Improve problem-solving skills.

Equipment

1. Hardware

- 1.1 Dell Inspiron 7559 with Windows 10 64-bit Operating System, 2.30 GHz Intel ® Core (TM) i5-6300HQ Processor, 8 GB Ram, and 1 TB HDD Storage.
- 1.2 Personal Computer with Windows 10 64-bit Operating System, 3.40 GHz quad-core Intel Core I5 Processor, 16 GB Ram, and 1 TB HDD Storage.
- 1.3 Macbook Pro 2017 with macOS Mojave Operating System, 2.30 GHz dual-core Intel Core I5, 8 GB Ram, and 256 GB SSD Storage.

2. Software

- 2.1 Jupyter Notebook
- 2.2 Google Colab
- 2.3 sagemaker on AWS

Budget

1. SSD 500 GB	3070	Baht
2. Apple Airpods	4708	Baht
	Total	<u>7778</u> Baht

References

[1] Dawen Liang; Rahul G. Krishnan; Matthew D. Hoffman; and Tony Jebara, Los Gatos, Cambridge, San Francisco, Variational Autoencoders for Collaborative Filtering. Proceedings of the 2018 World Wide Web Conference, 23–27 April, 2018, Lyon, France. Copyright 2018 ACM ISBN 978-1-4503-5639-8/18/04.

[2] Hao Wang; Naiyan Wang; and Dit-Yan Yeung, Hong Kong University of Science and Technology, Collaborative Deep Learning for Recommender Systems. Proceedings of the 21th ACM SIGKDD International, 10-13 August, 2015, Sydney, NSW, Australia. Copyright 2015 ACM ISBN 978-1-4503-3664-2/15/08.

[3] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2018. Deep Learning based Recommender System: A Survey and New Perspectives. Journal of ACM Computing Survey. 1, 1, Article 1 (July 2018), 35 pages.

[4] Suvash Sedhain; Aditya Krishna Menon; Scott Sanner; and Lexing Xie, Australian National University/NICTA, AutoRec: Autoencoders Meet Collaborative Filtering. Proceedings of the 24th International Conference on World Wide Web, 18-22 May, 2015, Florence, Italy. Copyright 2015 ACM 978-1-4503-3473-0/15/05.

[5] Yao Wu; Christopher DuBois; Alice X. Zheng; and Martin Ester, Simon Fraser University, Dato Inc., Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, 22-25 February, 2016, San Francisco, California, USA. Copyright 2016 ACM ISBN 978-1-4503-3716-8/16/02.

[6] Florian Strub; Jer´emie Mary; and Romaric Gaudel, Univ. Lille, CNRS, Centrale Lille, Inria, Hybrid Recommender System based on Autoencoders. Proceedings of the

1st Workshop on DLRS 2016, 15-15 September, 2016, Boston, MA, USA. Copyright 2016 ACM ISBN 978-1-4503-4795-2/16/11.

[7] Xin Dong; Lei Yu; Zhonghuo Wu; Yuxia Sun; Lingfeng Yuan; and Fangxi Zhang, Ctrip Travel Network Technology (Shanghai) Co., Limited., A Hybrid Collaborative Filtering Model with Deep Structure for Recommender Systems. Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, 04-09 February, 2017, San Francisco, California, USA. Copyright 2017, Association for the Advancement of Artificial Intelligence (www.aaai.org).

[8] Yu Liu, Shuai Wang, M. Shahrukh Khan, and Jieyu He. 2018. A novel deep hybrid recommender system based on auto-encoder with neural collaborative filtering. Journal of Big Data Mining and Analytics. 1, 3, Article 1 (September 2018), 211-221 pages.

[9] Xiangnan He; Lizi Liao; and Hanwang Zhang, National University of Singapore, Columbia University, Neural Collaborative Filtering. Proceedings of the 26th International Conference on WWW '17, 03-07 April, 2017, Perth, Australia. Copyright 2017 ACM 978-1-4503-4913-0/17/04.

BIOGRAPHY



Nattapon Napasai

Department of Mathematics and Computer Science

Faculty of Science, Chulalongkorn University

Email : Nattapon.n4p@gmail.com



Patcharapol Promanee

Department of Mathematics and Computer Science

Faculty of Science, Chulalongkorn University

Email : nut3870@hotmail.com