โครงการ
การเรียนการสอนเพื่อเสริมประสบการณ์

ชื่อโครงการ ระบบแนะนำตามพื้นฐานความสนใจโดยประยุกต์ใช้การฝังตรึงเชิงพื้นที่
Attention-Based Recommender Systems by Applying Region Embedding

ชื่อนิสิต

นางสาวฐิติยา แซ่เตีย	593 36208 23
นางสาวอุมาพร ผดุงเกียรติวัฒนา	593 36696 23

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์
สาขาวิชาวิทยาการคอมพิวเตอร์

ปีการศึกษา 2562

คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ระบบแนะนำตามพื้นฐานความสนใจโดยประยุกต์ใช้การฝังตรึงเชิงพื้นที่

นางสาวฐิติยา แซ่เตีย
นางสาวอุมาพร ผดุงเกียรติวัฒนา

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์ ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2562

Attention-Based Recommender Systems by Applying Region Embedding

Ms. Thitiya Sae-diae
Ms. Umaporn Padungkiatwattana

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

หัวข้อโครงงาน	ระบบแนะนำตามพื้นฐานความสนใจโดยประยุกต์ใช้การฝังตรึงเชิงพื้นที่
โดย	นางสาวฐิติยา แซ่เตีย
	นางสาวอุมาพร ผดุงเกียรติวัฒนา
สาขาวิชา	วิทยาการคอมพิวเตอร์
อาจารย์ที่ปรึกษาโครงงานหลัก	รองศาสตราจารย์ ดร.ศรันญา มณีโรจน์
อาจารย์ที่ปรึกษาโครงงานร่วม	ผู้ช่วยศาสตราจารย์ ดร.กิติพร พลายมาศ

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับโครงงานฉบับนี้เป็นส่วนหนึ่ง ของการศึกษาตามหลักสูตรปริญญาบัณฑิต ในรายวิชา 2301499 โครงงานวิทยาศาสตร์ (Senior Project)

.....

(ศาสตราจารย์ ดร.กฤษณะ เนียมมณี) หัวหน้าภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

คณะกรรมการสอบโครงงาน

.....

(รองศาสตราจารย์ ดร.ศรันญา มณีโรจน์) อาจารย์ที่ปรึกษาโครงงานหลัก

.....

(ผู้ช่วยศาสตราจารย์ ดร.กิติพร พลายมาศ) อาจารย์ที่ปรึกษาโครงงานร่วม

.....

(อาจารย์ ดร.ทรรปณ์ ปณิธานะรักษ์) กรรมการ

.....

(อาจารย์ ดร.วุฒิชัย จงจิตเมตต์) กรรมการ

นางสาวฐิติยา แซ่เตีย, นางสาวอุมาพร ผดุงเกียรติวัฒนา: ระบบแนะนำตามพื้นฐานความสนใจโดยประยุกต์ใช้การฝังตรึงเชิงพื้นที่ (Attention-Based Recommender Systems by Applying Region Embedding)
อ.ที่ปรึกษาโครงงานหลัก: รองศาสตราจารย์ ดร.ศรันญา มณีโรจน์
อ.ที่ปรึกษาโครงงานร่วม: ผู้ช่วยศาสตราจารย์ ดร.กิติพร พลายมาศ, 78 หน้า.

ระบบแนะนำเป็นระบบที่ใช้ในการค้นหาสินค้าหรือผลิตภัณฑ์ที่ผู้ใช้งานน่าจะสนใจมาแนะนำให้กับผู้ใช้งาน ระบบแนะนำนิยมใช้ความสอดคล้องกับประวัติการใช้งานที่มีมาก่อนหน้าของผู้ใช้งานเป้าหมายและผู้ใช้งานคนอื่น ๆ ในระบบ หรือความชอบของผู้ใช้มาช่วยในการทำนายคะแนน อย่างไรก็ตาม ระบบแนะนำส่วนใหญ่ที่มีมาแต่ดั้งเดิมยังคงขาดการพิจารณาถึงความสัมพันธ์ที่แฝงอยู่ภายในลำดับโดยรอบของผู้ใช้งานเป้าหมาย และ ผลิตภัณฑ์เป้าหมาย ซึ่งอาจก่อให้เกิดผลกระทบต่อการทำนายคะแนนความพึงพอใจให้กับผู้ใช้งานเป้าหมายได้ ด้วยเหตุนี้ งานวิจัยนี้จึงเสนอขั้นตอนวิธีใหม่โดยการสกัดความสัมพันธ์แฝงภายในลำดับประวัติของ ผู้ใช้งานเป้าหมาย และ ผลิตภัณฑ์เป้าหมาย โดยอาศัยการฝังตรึงเชิงพื้นที่ กับ หน่วยบริบทเฉพาะแห่งมาใช้ เพื่อที่จะนำความสัมพันธ์แฝงที่ได้มาใช้ในการทำนายคะแนนส่วนบุคคลของผู้ใช้งานแต่ละคน อย่างไรก็ตามวิธีการดังกล่าวขาดการทำให้ชุดของเพื่อนบ้านมีความเฉพาะเจาะจงสำหรับผู้ใช้งานเป้าหมายแต่ละคน ดังนั้นงานวิจัยนี้จึงทำชุดของเพื่อนบ้านให้มีความเฉพาะเจาะจงกับผู้ใช้งานเป้าหมายแต่ละคน เพื่อให้คะแนนการแนะนำที่ได้มีความเฉพาะเจาะจงกับผู้ใช้งานเป้าหมาย แม้ว่าจะมีชุดของเพื่อนบ้านเป็นชุดเดียวกัน โดยการประยุกต์ใช้วิธีการความสนใจ วิธีการดังกล่าวจะคำนวณคะแนนความสนใจซึ่งเปรียบได้กับค่าความคล้ายคลึงระหว่างเพื่อนบ้านแต่ละคนกับผู้ใช้งานเป้าหมาย ดังนั้นชุดของเพื่อนบ้านจะถูกทำให้แตกต่างกันตามผู้ใช้งานเป้าหมายซึ่งแตกต่างกัน และ คำนวณคะแนนการแนะนำที่แตกต่างกันออกมาได้ ชุดข้อมูลที่ใช้ในการทดลองคือ MovieLens ในการวิจัย และเปรียบเทียบประสิทธิภาพของขั้นตอนวิธีที่นำเสนอมากับการแนะนำโดยใช้เทคนิคระบบประสาทแบบคัดกรองร่วม รวมถึงเปรียบเทียบผลการทดลองระหว่างวิธีการที่ประยุกต์ใช้วิธีการความสนใจ กับวิธีการที่ไม่มีการประยุกต์ใช้ความสนใจ โดยใช้ค่า NDCG@K และ HitRate@K ในการวัดผล ผลการทดลองได้แสดงให้เห็นว่าแบบจำลองที่เสนอให้ผลที่ดีกว่าแบบจำลองของการแนะนำโดยใช้เทคนิคระบบประสาทแบบคัดกรองร่วม นอกเหนือจากนี้ยังแสดงให้เห็นว่า แบบจำลองที่ใช้วิธีการความสนใจ ให้ผลที่ดีกว่าแบบจำลองที่ไม่มีการใช้วิธีการความสนใจรวมอยู่

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์ ลายมือชื่อนิสิต ฐิติยา แซ่เตีย

ลายมือชื่อนิสิต อุมาพร ผดุงเกียรติวัฒนา

สาขาวิชา วิทยาการคอมพิวเตอร์ ลายมือชื่อ อ.ที่ปรึกษาโครงงานหลัก ศรันญา มณีโรจน์

ปีการศึกษา 2562 ลายมือชื่อ อ.ที่ปรึกษาโครงงานร่วม กิติพร พลายมาศ

5933620823, 5933669623: MAJOR COMPUTER SCIENCE
KEYWORDS: COLLABORATIVE FILTERING, REGION EMBEDDING, ATTENTION MECHANISM, RECOMMENDER SYSTEM

MS. THITIYA SAE-DIAE, MS. UMAPORN PADUNGKIATWATTANA: ATTENTION-BASED RECOMMENDER SYSTEMS BY APPLYING REGION EMBEDDING. ADVISOR: ASSOC. PROF. SARANYA MANEEROJ, Ph.D., CO-ADVISOR: ASST. PROF. KITIPORN PLAIMAS, Ph.D., 78 pp.

Recommender System is a system that is used to search for an item that a user prefers and recommends it to another user. The Recommender Systems often uses the consistency with previous usage history of the target user and other users in the system or the user's preferences to predict scores. However, they still lack consideration of latent relations in the sequence around a target user and a target item. Hence, this research proposes a new recommendation method that can extract a latent relation in a historical sequence of a target user and a target item by applying Region Embedding with the Local Context Unit in order to utilize that latent relation for personalized rating predictions. However, this method does not serve for personalized recommendation for each target user. Therefore, this research also proposes a specific set of neighbors to different target users in order to differentiate among target users even if having the same set of neighbors by applying method of Attention mechanism. This method calculates Attention scores which are similar to the similarity score between neighbors and target users. Therefore, the proposed method can predict rating scores which are different for different target users. The dataset used in the experiment is MovieLens. This research compares the efficiency of the proposed method with Neural Collaborative Filtering recommendation. Moreover, the experimental results between our method with Attention was compared to our method without Attention. Finally, the models were evaluated with NDCG@K and HitRate@K. The results show that our model is better than NCF. Moreover, our model with Attention is better than our model without Attention.

Department: Mathematics and Computer Science Student's Signature Thitiya Sae-diae
Student's Signature Umaporn Padungkiatwattana
Field of Study : Computer Science Advisor's Signature Saranya Maneeroj
Academic Year : 2019 Co-Advisor's Signature K. Plaimas

ACKNOWLEDGEMENTS

First of all, we really appreciate everything that our senior project advisor, Assoc. Prof. Saranya Maneeroj, Ph.D. has done. This project would have never been accomplished without her kind support, understanding and dedicated involvement in every step throughout the research. Moreover, we would like to thank Asst. Prof. Kitiporn Plaimas, Ph.D., our senior project co-advisor, for her many suggestions.

We would like to express gratitude to our committee including Dr. Thap Panitanarak, and Dr. Wutichai Chongchitmate, who have participated in our senior project. They raised many important points that beneficial for improving our research.

We are also grateful to Mr. Padipat Sitkrongwong, a graduate student in Computer Science program, Chulalongkorn University, for his guidance. His suggestions are valuable for us.

We would also like to express our gratitude to the Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University that realizes the importance of our senior project and supports the budget for developing our work.

We also gratefully acknowledge to thank our friends and our family for their sincerely and consistently supports throughout the period of this project.

Finally, we are thankful to Chulalongkorn University, the place that gave many opportunities and cultivates a lot of knowledge for us.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1. Background and Rationale

The internet has many social roles in our lives, whether watching a movie, listening to music, shopping, updating news, etc. These activities make users feel more comfortable in their lives. Users of these various activities on the internet are increasing. Then, many entrepreneurs try to find ways that users can easily access their interests by using the internet. However, the large number of users makes it difficult to access what each user is interested in. Each user has a wide range of preferences, and product data are extensive and complex so the amount of data will be differently increased by the number of users. Therefore, finding things that users are interested in must rely on the engine, which is the recommender system. The recommender system has been developed to find what users are mostly interested in and recommend it to the target users. Therefore, we can see that the recommender system is available on many websites and applications such as Netflix, YouTube, Spotify, Shopee, Lazada, etc.

One of the popular methods in the recommender system is Collaborative filtering. Collaborative filtering creates recommendations by exploiting preference data from other users who are similar to a target user and predicting rating scores for the target user. For example, calculating a rating score for the movie that the target user has never seen before, Collaborative filtering uses the movie history of other users and that target user to find similarities between users and other users. Then, the rating scores of similar users are used to predict the rating score for that target user. Collaborative Filtering has many approaches such as using matrix factorization [1], using neural network [2], or using context information [3].

The main point of the Recommender System is to find latent relations among users and items in order to increase recommendation performance. Many collaborative filtering methods utilize historical relation of a user or item in order to find similarity between either a pair of users or a pair of items. However, they do not consider the order of these historical sequences. In our opinion, historical sequence has important latent relation inside, not only historical relation of users or items but also order of these

historical sequences can be used to find latent preference of users. Historical sequence in recommendation area can be divided into 2 types. Firstly, in term of a target user, system will consider the sequence of items that rated by that target user. Secondly, in term of target item, system will consider the sequence of users who have rated on that target item.

Table 1.1 User historical sequence of Amy

Rating	2	1	3	5	4
Movie	La La land	Romeo Juliet	50 first dates	Titanic	Me before you

Table 1.2 User historical sequence of Sara

Rating	5	4	4	1	2
Movie	Titanic	Peter Pan	Winnie the Pooh	Ted	Annabelle

Table 1.1 and Table 1.2 show sequence in term of a target user. In these table, there are 2 target users, Amy and Sara. Both of them rate Titanic (target item) with 5 score but movie sequence of them are different. Amy rates Titanic in the fourth order and Sara rates Titanic in the first order. Because Titanic is rated on different orders in movie sequence of Amy and Sara, movies around it should be different. Movies around Titanic in the movie sequence of Amy are the romantic movie as Titanic, whereas movies around Titanic in the movie sequence of Sara are cartoon movies. Therefore, Titanic rated by Amy and Titanic rated by Sara must be different because their movie sequences are different, even though it got the same rating score from Amy and Sara. It can be concluded that the order of movies in the historical sequence indicates users' interest and relation among these movies.

Table 1.3 Item historical sequence of Titanic rated by Sara

Rating	5	3	1	5	4
User	Sara	Joy	Amy	Paula	Jenny

Table 1.4 Item historical sequence of Avengers rated by Sara

Rating	3	1	5	2	5
User	Robert	David	Sara	John	Paul

Table 1.3 and Table 1.4 show sequences in term of target items. In these table, there are 2 target items, Titanic and Avengers. Both of them are rated by Sara (target user) with 5 score but their user sequences are different. Titanic is rated by Sara in the first order and Avengers is rated by Sara in the third order. Because Sara rates on different orders in user sequences of Titanic and Avengers, users around her should be different. Users around Sara in user sequences of Titanic are the same woman as Sara, whereas users around Sara in user sequences of Avengers are men. Therefore, Sara rating Titanic and Sara rating Avengers must be different because their user sequences are different, even though Sara rates the same rating score on these movies. It can be concluded that the order of users in the historical sequence indicate user characteristic of user groups for these movies and relation among these users.

The order consideration is often used in Natural Language Processing (NLP) because the order of words in the sentence has meaning. Moreover, the relation between words and words around the target word, i.e., context word, also have meaning too. Region Embedding is then introduced to capture that relation by using Local Context Unit (LCU). Local Context Unit is a weighting matrix that captures the interactions between a word and its neighbors in a text region [4]. Recently, Local Context Unit is applied to Recommendation work. For example, Local Context Unit is utilized in users' review for extracting contexts from review data. Then, the contexts is used to create a predictive model [5].

In this work, we propose a new recommendation method that is able to capture and extract latent relation in historical sequence of users and items by applying Region Embedding with the Local Context Unit in order to utilize that latent relation for personalized rating prediction.

Unfortunately, output rating from the method using only Local Context Unit that we propose in the previous paragraph did not serve for personalized recommendation. The model predicted the same rating score of the same movie to different target users because we specify the users' neighbors as people who rate the same target item, so users' neighbors of these target users are the same. Therefore, we also propose method to solve this problem by applying Attention [6]. Attention is a work that is widely used in Machine Translation to weight word that most similar to query word. For recommendation works, attention is used to weight item that user

should have interested most [7] or is used to model user's behavior based on users' interest [8]. In our work, we would like to differentiate among target users by applying Transformer from Attention. Transformer uses sequence of users who have rated on target item instead of word in our work. Therefore, we will get the output from Transformer as a list of neighbors personalized by each target user where each target user is the query. In other words, when we put different target user or different query into the transformer, we will get different list of neighbors according to different target users' perspective.

1.2. Objectives

1. To propose a new recommendation method that is able to capture relation in historical sequence of target user and target item by applying Region Embedding. Moreover, this new method is able to find personalized target users' neighbors by applying attention on target item sequence.
2. To compare the efficiency of our method with previous method on NDCG@K and HitRate@K.

1.3. Scope

1. Use publicly available datasets from MovieLens [9] dataset that is not more than 1,000,000 ratings, 4,000 movies and 6,000 users. The lowest rating score is 0.5 and the highest rating score is 5.0
2. Use value of rating in the range 1-5.
3. Predict only movie products.
4. Compare experimental results with the research of Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua in Neural Collaborative Filtering.

1.4. Project Activities

A. Study Plan

1. Study related works.
2. Analyze previous problems.
3. Design new methods for solving problems.

4. Analyze the proposed methods.
5. Develop the proposed methods.
6. Measure NDCG@K and HitRate@K of the proposed methods.
7. Experiment and compare the results.
8. Analyze the results.
9. Prepare project documentation.

B. Study Period

From the study plan, we can write Gantt chart as shown in Table I-5

Table 1.5 Timeline of Research Activities

Procedure	**2019**					**2020**			
	Aug	**Sep**	**Oct**	**Nov**	**Dec**	**Jan**	**Feb**	**Mar**	**Apr**
1. Study related works.									
2. Analyze previous problems.									
3. Design new methods for solving problems.									
4. Analyze the proposed methods.									
5. Develop the proposed methods.									

Procedure	2019					2020			
	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr
6. Measure NDCG@K and HitRate@K of the proposed methods.									
7. Experiment and compare the results.									
8. Analyze the results.									
9. Prepare project documentation.									

1.5. Benefits

A. In terms of knowledge and experience to the students

1. Gain knowledge and understanding of the process of implementing the system.
2. Practice analytical skill, working methodology, responsibility for work.
3. Gain knowledge about various models to create a recommendation system method.
4. Gain knowledge of programming and system development.

B. Knowledge and understanding that leads to solving problems of society or environment

1. Create an appropriate model for predicting movie ratings with more accuracy than previous methods.
2. Reduce errors from inaccurate recommendations from previous methods.

3. Recommend things that meet user expectation.

1.6. Report Outlines

The rest of this report is organized as followings:

1. Chapter 2 Literature Review: will present knowledge, related works in recommender system.
2. Chapter 3 Methodology: will explain the proposed method and its process.
3. Chapter 4 Results: will present results and evaluate the performance.
4. Chapter 5 Conclusion: will discuss our work

CHAPTER 2
LITERATURE REVIEW

In this chapter, the works related to our proposed method, including collaborative filtering, neural Collaborative Filtering, Region Embedding, Recurrent Neural Network, and Attention are introduced.

2.1. Collaborative Filtering

In Recommender System, Collaborative Filtering (CF) is one of the traditional methods that try to solve the serendipitous problem. The main step of the CF is to find the most users (neighbor) who have the past preference similar to the target user by using cosine similarity in the Equation (2.1) or (2.2). After that, the rating score is then predicting for that target user.

$$Cosine\ Similarity: Sim(u_i, u_k) = \frac{r_i \cdot r_k}{|r_i||r_k|} \tag{2.1}$$

$$Cosine\ Similarity: Sim(u_i, u_k) = \frac{\sum_{j=1}^{m} r_{ij} r_{kj}}{\sqrt{\sum_{j=1}^{m} r_{ij}^2 \sum_{j=1}^{m} r_{kj}^2}} \tag{2.2}$$

where u_i, u_k denote the target user and another user, respectively, i denotes the items, r_{ij} denotes rating that user u_i rated on item j, r_{kj} denotes rating that user u_k rated on item j.

For example, we want to predict rating score for target user Jenny which we can see her rating score and other users in the Table(2.1)

Table 2.1 User-Item rating matrix: rating score of users for each movie

User/Movie	Titanic	Winnie the pooh	Avengers	Me before You	Toy Story
Lisa	4	2	5	4	5
Joy	5	4	2		1
Ann	5	4	2		
Paula	2		3		
Jenny	5	4	**?**		1

At the first step, we have to find the similarity between Jenny and other users. Notice that Jenny is similar to the other 2 users, Joy (similarity score = 0.955) and Ann (similarity score = 0.943). Then, we will use user's rating from the user who similar to Jenny to predict Jenny's rating. Therefore, rating score of Jenny for Avengers is 2 like Joy who most similar to Jenny.

Matrix factorization (MF) is one of the methods in CF which gave the model has the most effective result. MF minimizing a loss function for represents users and items in high sparsity data in a lower-dimensional. For example, we have past usage history of each user, which provides information about the rating data of each item, a group of user u_i , where n is the size, a group of item v_j , where m is the size. So, we can create matrix $\hat{R} = P \times Q^T$, that their product approximates R, where P is the size of n_k, and Q is the size of m_k , as you can see in Figure 2.1. Since P has the same number of rows as the user. Therefore, it can be used to find similar users and items that users are more interested in, with scores based on the Equation (2.3).

$$r_i = p_i \times Q^T \tag{2.3}$$

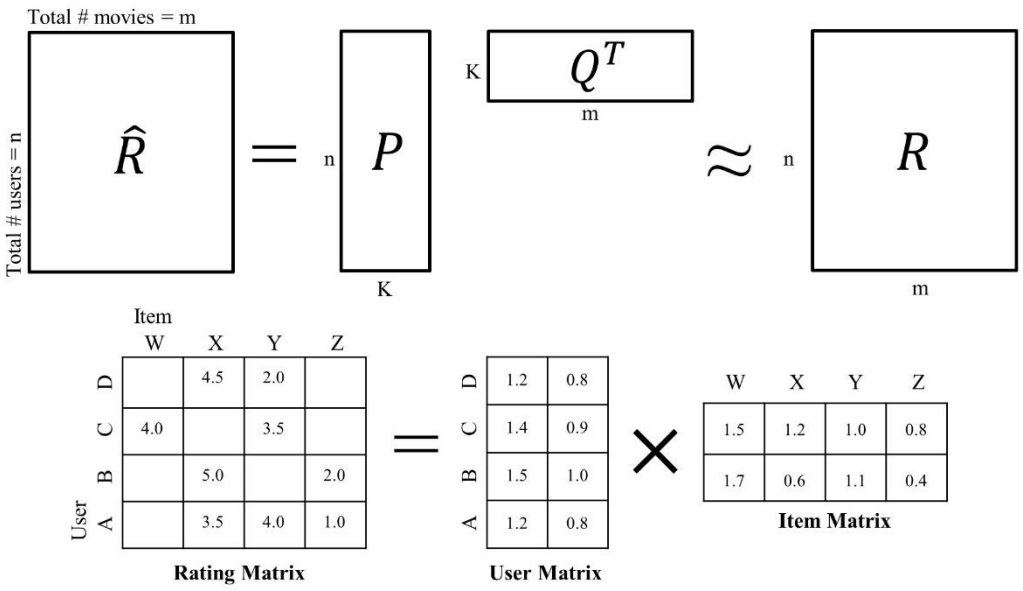


Figure 2.1 Matrix Factorization for Collaborative Filtering [10, 11]

For finding Matrix P and Q, use Gradient Descent to calculate with loss function of Mean Square Error or as in Equation (2.4)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (2.4)$$

There are various matrix factorization models that are commonly used:

2.1.1. Singular Value Decomposition (SVD)

SVD is a powerful technique for reducing dimensions. SVD of an $m \times n$ matrix A can be calculated by using an Equation (2.5)

$$SVD(A) = U\sum V^T \quad (2.5)$$

where U and V denote $m \times m$ and $n \times n$ orthogonal matrices, respectively, $\sum$ is the $m \times n$ singular orthogonal matrix with non-negative elements. You can see the illustrate of SVD in Figure 2.2.

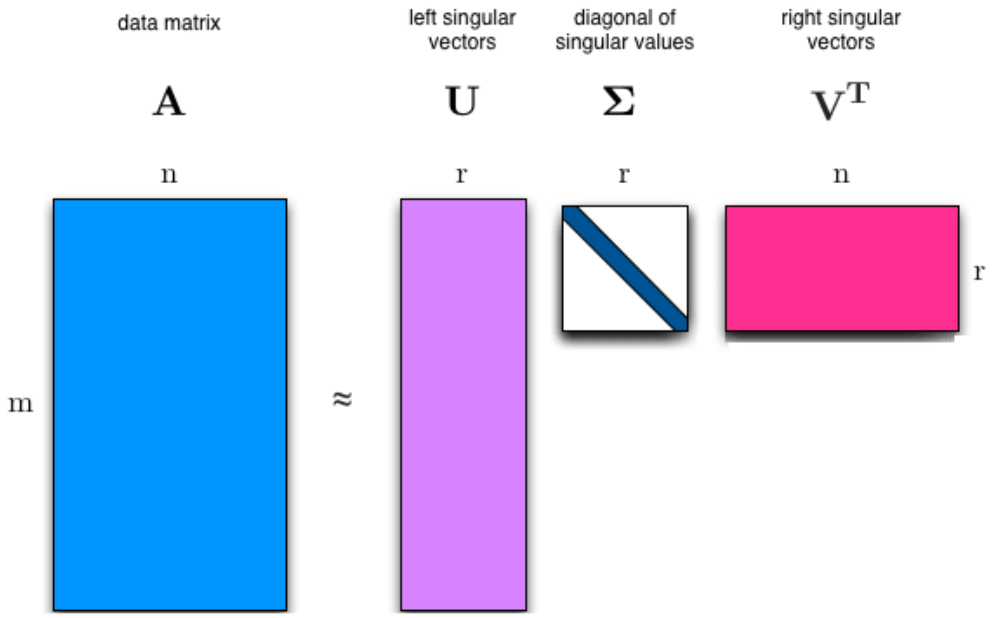


Figure 2.2 Singular Value Decomposition [12]

2.1.2. Principal Component Analysis (PCA)

PCA is also an effective technique for downsizing and is a unique method for factorization matrices. PCA is a statistical process that uses orthogonal transformations to transform observation sets of related variables. It can be a set of nonlinear variable values known as key components. The original number of variables greater than or equal to the main components. This transformation is defined in such a way that the linear projection of the high

dimensional data in the sub-areas below, such as the variance retained is maximized and the least square reconstruction error is minimized. The illustrate of PCA is shown in the Figure 2.3.

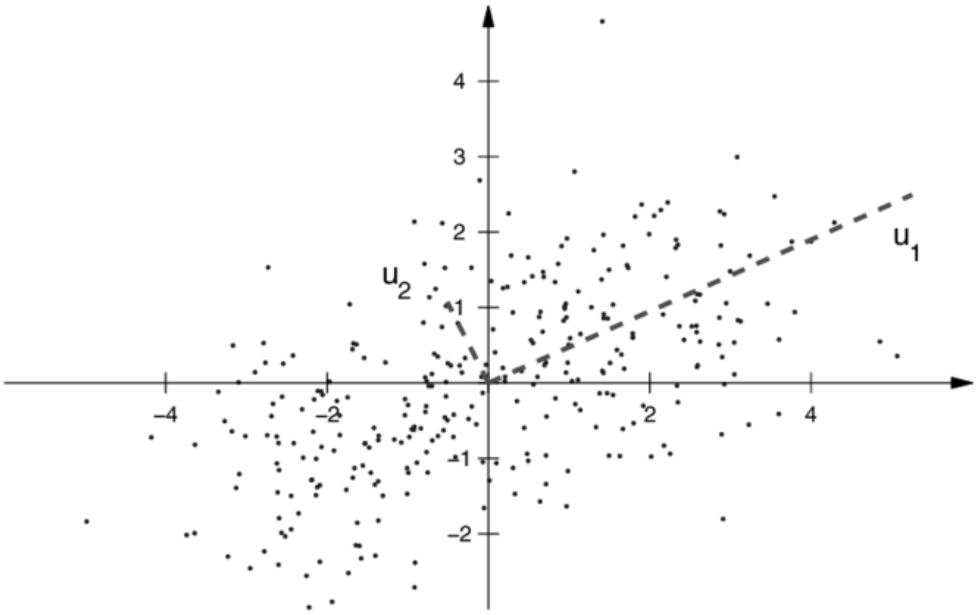


Figure 2.3 PCA analysis of a two-dimensional point cloud from a combination of Gaussians. (u1 and u2 are PCs) [13]

2.2. Neural Collaborative Filtering

Neural Collaborative Filtering (NCF) is the neural recommendation that simulate the operation of Collaborative Filtering. It was created to settle the limitation problem from Collaborative Filtering. Figure 2.4 shows an example illustrates MF's limitation.

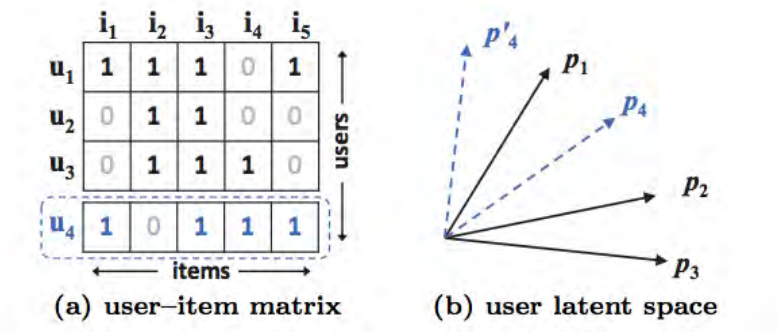


Figure 2.4 An example illustrates MF's limitation [2]

From data matrix (a), u_3 is more similar to u_4 than u_2. However, in the latent space (b), p_4 is closer to p_2 than p_3.

In the NCF's input layer, it uses only a binarized sparse vector with a one-hot encoding of the identity of a user and an item to create a model to predict user ratings as you can see in Figure 2.5. The next layer is embedding layer, which is a fully connected layer. This layer is then projected the sparse representation to a dense vector. Therefore, we get user and item embedding which can be seen as the latent vector for user or item from this layer. After that, user and item embedding are fed into a multi-layer neural architecture of Neural Collaborative Filtering layers. The multi-layer neural architecture is then map the latent vectors to predict rating scores. The last layer, the output layer, Equation 2.6 and 2.7 are used in order to predicts the rating score $\hat{Y}_{ui}$. After that, minimizing the pointwise loss between $\hat{Y}_{ui}$ and its target value Y_{ui} for training model.

$$\hat{y}_{ui} = f(P^T v_u^U, Q^T v_i^I | P, Q, \Theta_f) \tag{2.6}$$

where v_u^U , v_i^I are the feature vectors that describe user $u \in U$ and item $i \in I$, respectively. $P \in R^{M \times K}$, $Q \in R^{N \times K}$ denoting the latent factor matrix for users and items and, Θ_f is the model parameters of the interaction function f.

$$f(P^T v_u^U, Q^T v_i^I) = \phi_{out}(\phi_x(\dots \phi_2(\phi_1(P^T v_u^U, Q^T v_i^I))\dots)) \tag{2.7}$$

where ϕ_{out} and ϕ_x denoting the mapping function for the output layer and x-th neural collaborative filtering (CF) layer, and there are X neural CF layers in total.

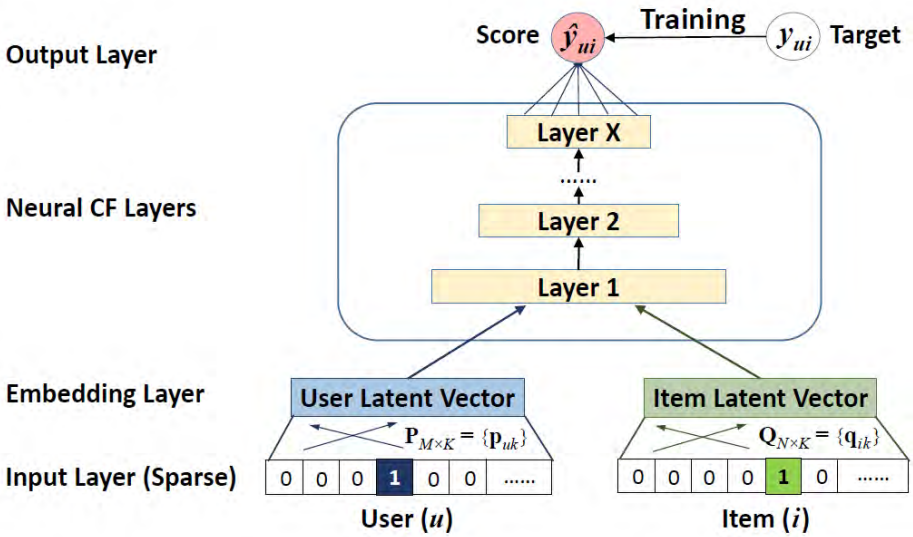


Figure 2.5 Neural Collaborative Filtering framework [2]

Pointwise squared loss is then using to learn model parameters as Equation (2.8).

$$L_{sqr} = \sum_{(u,i)\epsilon\gamma\cup\gamma^-} w_{ui}\,(y_{ui} - \hat{y}_{ui})^2 \tag{2.8}$$

where denotes the set of observed interaction in Y, and -denotes the set of negative instances of unobserved interactions, and w_{ui} is the weight of training instance (hyperparameter). Then, we need a probability method for learning the pointwise NCF that constrains the output y_{ui} in the range of [0,1]. It can be achieved by utilizing a probabilistic function as the activation function for the output layer out. The likelihood function is defined as Equation 2.9

$$p(\gamma, \gamma^-|P, Q, \Theta_f) = \prod_{(u,i)\epsilon\gamma} \hat{y}_{(u,i)} \prod_{(u,i)\epsilon\gamma^-} (1 - \hat{y}_{uj}) \tag{2.9}$$

Then, taking the negative logarithm of the likelihood function by using Equation (2.10)

$$L = -\sum_{(u,i)\epsilon\gamma} log\, \hat{y}_{ui} - \sum_{(u,i)\epsilon\gamma^-} log(1 - \hat{y}_{uj})$$

$$= -\sum_{(u,i)\epsilon\gamma\cup\gamma^-} y_{ui} log\, \hat{y}_{ui} + (1 - y_{ui}) log(1 - \hat{y}_{ui}) \tag{2.10}$$

This is a function whose purpose is to reduce the size for NCF methods and the optimization can be done by performing stochastic gradient descent (SGD).

In [2], they show that MF can be generalized under NCF framework by using a multi-layer perceptron (MLP) to learn the user–item interaction function. Moreover,

they combined MF and MLP under the NCF framework in order to create a new neural matrix factorization model.

2.2.1. Generalized Matrix factorization (GMF)

MF is popular model in the recommender system and it allows NCF to mock a large system of factorization models because MF being able to recover. The obtained embedding vector can be seen as the latent vector of user (item). The mapping function ϕ of the first neural CF layer can define as Equation 2.11.

$$\phi(p_u, q_i) = p_u \odot q_i \tag{2.11}$$

where the user latent vector p_u be $P^T v_u^U$, item latent vector q_i be $Q^T v_i^I$ and $\odot$ denotes the element-wise product of vectors. The vector is then projected to the output layer by using Equation (2.12)

$$\hat{y}_{ui} = a_{out}(h^T(p_u \odot q_i)) \tag{2.12}$$

where a_{out} and h denote the activation function and edge weights of the output layer, respectively.

2.2.2. Multi-Layer Perceptron (MLP)

In [2], they add hidden layers on the concatenated vector by using a standard MLP to learn the interaction between user and item latent features. The MLP model under their NCF framework is defined as Equation (2.13)

$$z_1 = \phi_1(p_u, q_i) = \begin{bmatrix} p_u \\ q_i \end{bmatrix},$$

$$\phi_2(z_1) = a_2(W_2^T z_1 + b_2),$$

$$\dots\dots$$

$$\phi_L(z_{L-1}) = a_L(W_L^T z_{L-1} + b_L),$$

$$\hat{y}_{ui} = \sigma(h^T \phi_L(z_{L-1})), \tag{2.13}$$

where ϕ_x W_x, b_x, and a_x denote the mapping function, weight matrix, bias vector, and activation function for the x-th layer's percertron,respectively

2.2.3. Fusion of GMF and MLP

From [2] solution, they fuse GMF and MLP under NCF framework by sharing the same embedding layer and combine the outputs of their interaction functions by using Equation (2.14). However, GMF and MLP must use the same size of embeddings.

$$\hat{y}_{ui} = \sigma(h^T a(p_u \odot q_i + W \begin{bmatrix} p_u \\ q_i \end{bmatrix} + b)) \tag{2.14}$$

Then, they allow GMF and MLP to learn separate embeddings, and concatenating their last hidden layer in order to combine the two models. Therefore, it has more flexibility to the fused model. Figure 2.6 is shown the fused model.

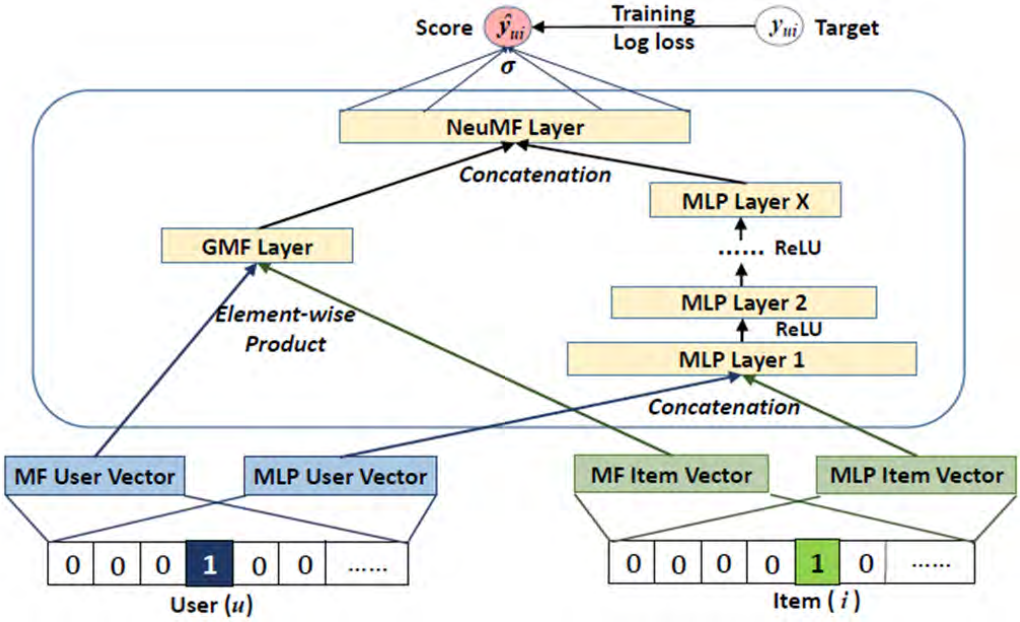


Figure 2.6 Neural matrix factorization model [2]

The formulation of this model is given as Equation (2.15), (2.16) and (2.17).

$$\phi^{GMF} = p_u^G \odot q_i^G \tag{2.15}$$

$$\phi^{MLP} = a_L(W_L^T(a_{L-1}(\dots a_2(W_2^T \begin{bmatrix} p_u^M \\ q_i^M \end{bmatrix} + b_2)\dots)) + b_L) \tag{2.16}$$

$$\hat{y}_{ui} = \sigma(h^T \begin{bmatrix} \phi^{GMF} \\ \phi^{MLP} \end{bmatrix}) \tag{2.17}$$

where p_u^G and p_u^M denote the user embedding for GMF and MLP parts, respectively, and the similar notations q_i^G and q_i^M for item embedding.

2.3. Region Embedding

Region Embedding is a method that considers the relation between the surrounding words and the middle word, target word, because the same word in each position of the document should not have the equal weight and have the same meaning. For example, if we have the document with the word ‘Apple’ that appears two times in different. The first ‘Apple’ appears with the sentence ‘She likes to eat apple that is fruit containing many vitamins’ at the beginning of the document. The second ‘Apple’ appears with the sentence ‘She bought a new phone, iPhone 11 that is the latest product launched by Apple’ at the last of the same document, as shown in Figure 2.7.

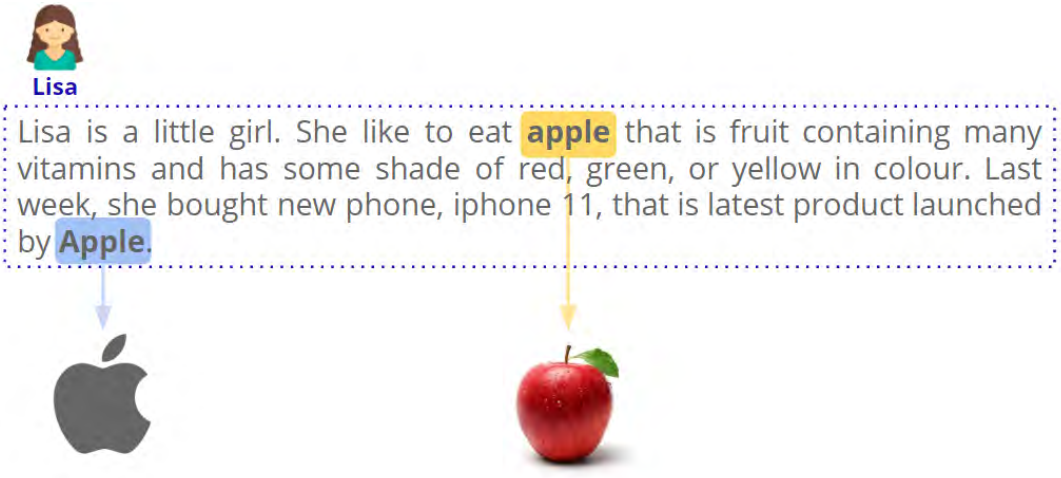


Figure 2.7 The document with the word ‘Apple’ that appears two times

It shows that word 'Apple' in different contexts has different meanings. That is, the first ‘Apple’ represents to the fruit, but the second ‘Apple’ represents to the company. Therefore, it is necessary to consider the surrounding context words in that position. Then, Region Embedding is developed and used in the following steps in order to find what is the meaning of the word in each position. The First step is learning the representations of the word as a small text region as shown in Figure 2.8.

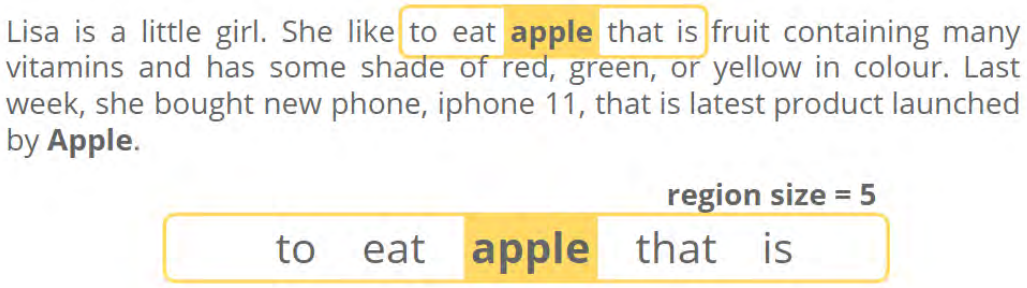


Figure 2.8 The word with a small text region

A region length of small text region can be calculated by using Equation (2.18).

$$r(i,c) = 2c + 1\text{r}, \tag{2.18}$$

where w_i stands for the i-th word of the document that starting from 0. For example, $r(4,2)$ is the sequence of 'to eat apple that is' of the sentence 'She likes to eat apple that is fruit containing many vitamins' in the document. Then, using word embedding to create a vector for each word in the region for represent the region and then, captures relations among the words and the target in the region in order to create a weighted matrix Local Context Unit (LCU), as shown in Figure 2.9.

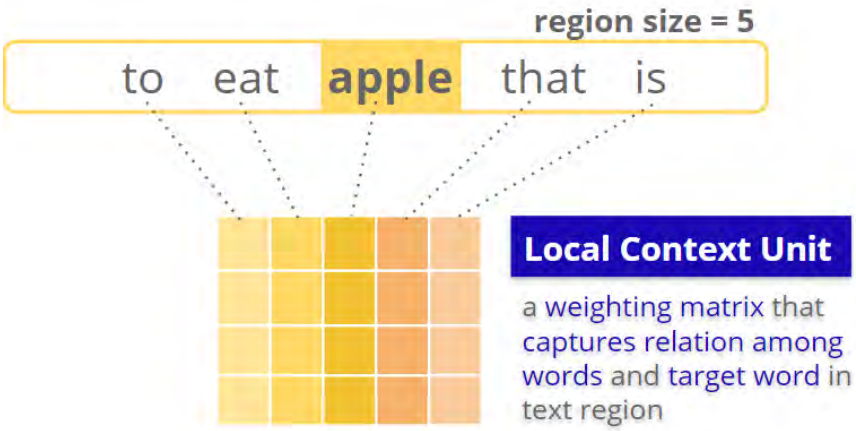


Figure 2.9 A Local Context Unit

After that, each word embedding is then projected with the Local Context Unit of the target word by element-wise multiplication as in Figure 2.10. Then, the result will let us know what the is the meaning of the word in the region. Finally, summarize the strength of each word together by max-pooling the output from the previous step together. Then, we will get the Region Embedding vector that representing this region.

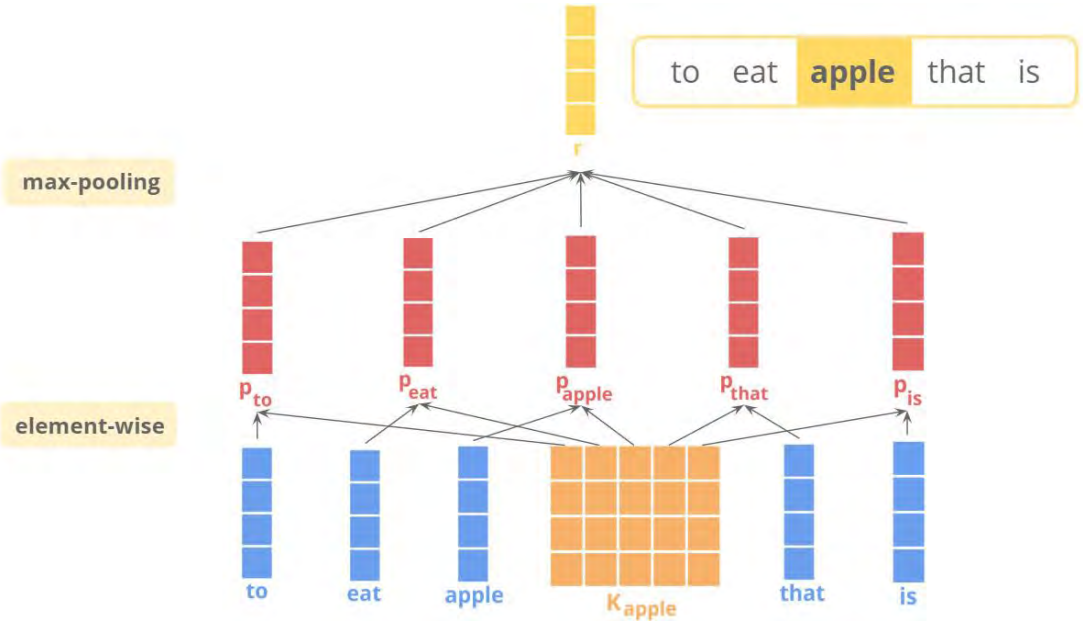


Figure 2.10 Working principle of the example

2.4. Recurrent Neural Network (RNN)

Recurrent neural network (RNN) is the artificial neural networks that designed to use with sequential task such as natural language processing (NLP), alphabetical order, and time-series data, etc. RNN has a multi-layer which can store an information in their node. Therefore, the model can receive data in the form of sequences and give results in the form of sequences. The outputs from the previous node is used to be an input in the next node. Between nodes in RNN there will be a hidden state, as shown in Figure 2.11. Therefore, the same input can produce different output depending on their previous inputs in the series.

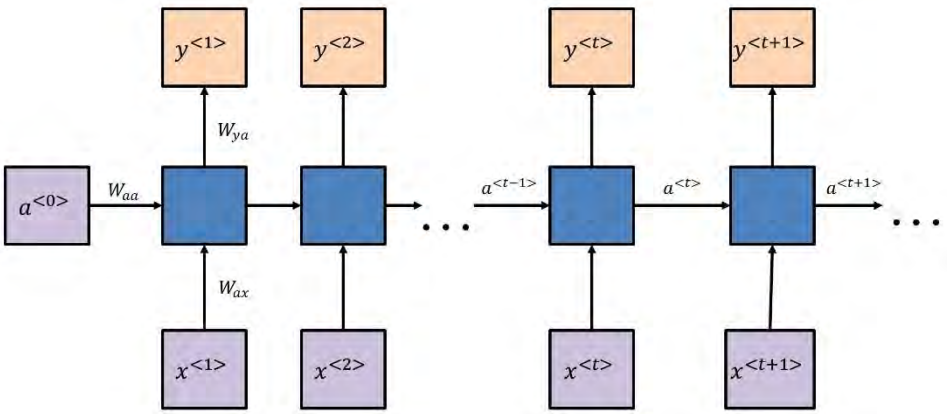


Figure 2.11 An architecture of RNN

From Figure 2.11, the activation $a^{<t>}$ and the output $y^{<t>}$ can be calculated by using the Equation (2.19) and (2.20).

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \quad (2.19)$$

$$y^{<t>} = g_2(W_{ya}a^{<t>} + b_y) \quad (2.20)$$

where W_{aa}, W_{ax}, W_{ya}, b_a, b_y are coefficients that are the same for each time and g_1, g_2 are activation functions.

In the RNN model, the input can be any length because RNN uses previous data as an input into the next node and the size of the model does not increase with the size of the input. However, RNN still has problems with a long time to computation because the input can be any length. The weight (W_{aa}, W_{ax}, W_{ya}) are still using over and over again to different items in the series. Moreover, it difficult to access information long ago because RNN uses only the earlier information in the sequence to make a prediction. Therefore, the gradient value will continually decrease while the data is longer until we hardly see the change in the gradient. Then, the vanishing gradient problem occurs.

2.4.1 Deep Recurrent Neural Networks (Deep RNNs)

RNN with single hidden layer is quite challenging to predict output while the model does not have enough flexibility to model the different types of interactions. Therefore, Deep RNNs was created to fix this problem by adding more layers instead of using a single perceptron as shown in Figure 2.12. The results from this mechanism are then more flexible because of the combination of several simple layers.

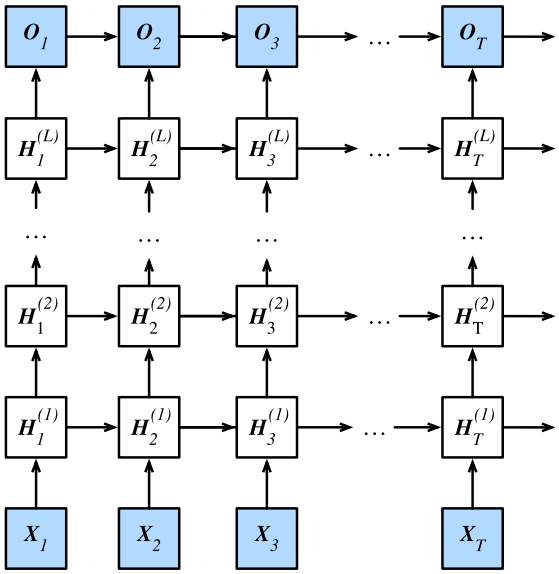


Figure 2.12 Architecture of a deep recurrent neural network [14]

2.4.2 Bidirectional Recurrent Neural Networks (Bidirectional RNNs)

In the traditional RNN, the model is unidirectional that use information from the past to predict the future. However, some tasks like speech recognition, handwriting recognition tasks, etc. need to look into the future to fix the past because it is often necessary to know what will happen next in order to understand the context and detect the present. Bidirectional RNNs are then occurs and putting two independent RNNs together. Therefore, the networks are fed the input sequence with normal time order, and reverse time order as shown in Figure 2.13.

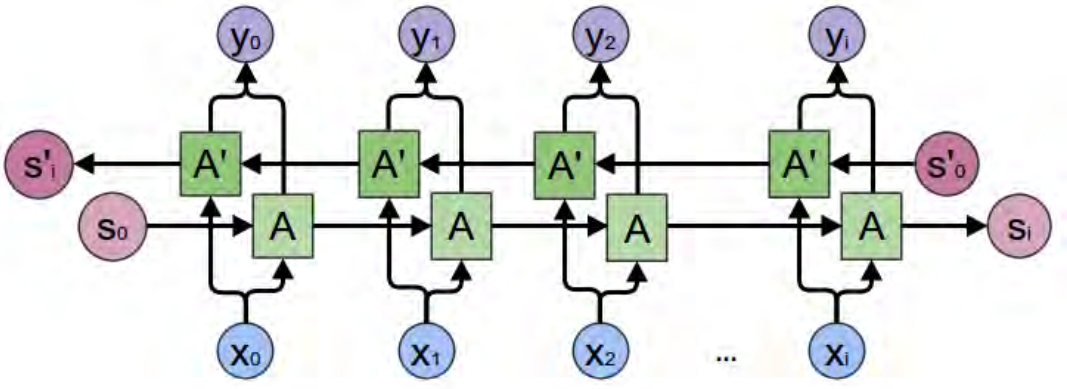


Figure 2.13 General Structure of Bidirectional Recurrent Neural Networks [15]

2.4.3 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks are a developed from RNN in order to resolve the vanishing gradient problem. LSTM is especially suitable for classifying, processing, and predicting time for a specified period of time without knowing the duration. Back-propagation is using to train this model. In the LSTM network, there are three gates as shown in Figure 2.14.

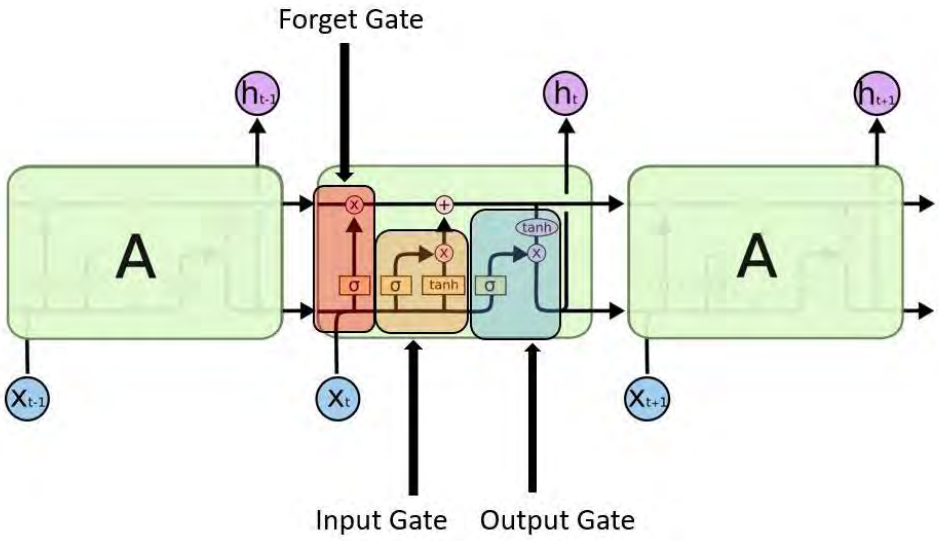


Figure 2.14 LSTM gates [16]

a) Input gate

This gate is used to find which value from input should be used to fix the memory. It uses the Sigmoid function, Equation (2.21), to decide which value to pass through 0,1 and use the tanh function, Equation (2.22), to weight values passed through their priority decisions from -1 to 1.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (2.21)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (2.22)$$

b) Forget gate

Forget gate is used to discover what details to be discarded from the block by using Sigmoid function as Equation (2.23).

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (2.23)$$

The previous state (h_{t-1}) and the content input(x_t) are then used to outputs the numbers which are 0 or 1 for each number in the cell state C_{t-1}. The

number 0 is means omit this number in the cell state and the number 1 is means keep this number in the cell state.

c) Output gate

The input and memory of the block are used to calculate the output by using Sigmoid function and tanh function as Equation (2.24) and Equation (2.25).

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (2.24)$$

$$h_t = o_t * \tanh(C_t) \quad (2.25)$$

2.5. Attention Mechanism

In the recommendation system, if we have item sequence and we want to know which items the user will rate in the future by using the information on the user's past preference, the previous recommendation tasks use RNN to predict it. However, RNN still has a problem with the weight of each item in the sequence are equal which in fact should not have equal weight. Therefore, attention is used to find for hidden relation in the past preference.

Attention mechanism is one of the neural machine translations that used to fix the problem from RNN**.** One of the network architectures of attention is the Transformer, Figure 2.15, which has an encoder-decoder structure.

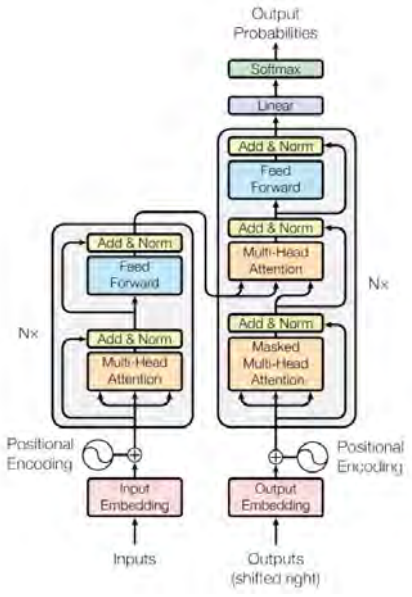


Figure 2.15 The Transformer – model architecture [6]

The encoder uses input and output as feature vectors. These feature vectors store the information representing the input in order to help the decoder to provide the closest match to the actual input. Therefore, the feature vectors from the encoder side are weighted according to the input before sending them to the decoder side. A set of multiple vectors comes from weighting relation between query word and each word in its context. After that, each vector in the decoder side receives different context vector inputs according to the interest in each encoding context. The output of attention can be calculated by using scaled dot-product as Equation (2.26) and compute multiple attention weighted sums as Equation (2.27) and (2.28).

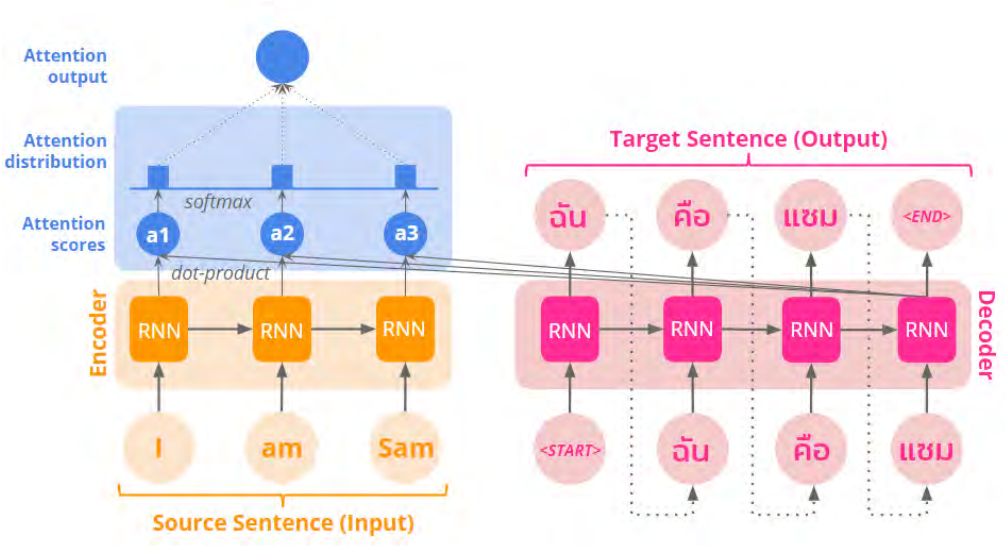
$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.26}$$

where$q$, k, v are denote query, keys and values respectively. Q, K, V is matrix of q, k, v. The d_k is dimension of vector k and $softmax$ is softmax function that uses to obtain the weights on the values .

$$head_i = Attention(QW^Q, KW^K, VW^V) \tag{2.27}$$

$$MultiHead(Q\ ,K,V) = Concat(head_1, \dots, head_h)W^0 \tag{2.28}$$

where h is number of head. It might be seen the output from attention as a sequence of behavioral substitution, taking into account the effects of others in each latent space. You can see the illustrate in Figure 2.16.



,

Figure 2.16 Attention model

2.6. ATRank

Recently, ATRank is used attention in recommendation tasks by [8]. They propose this model to preserve data integrity and avoid recommendations from unrelated user behavior. The framework of this model is shown in Figure 2.17.

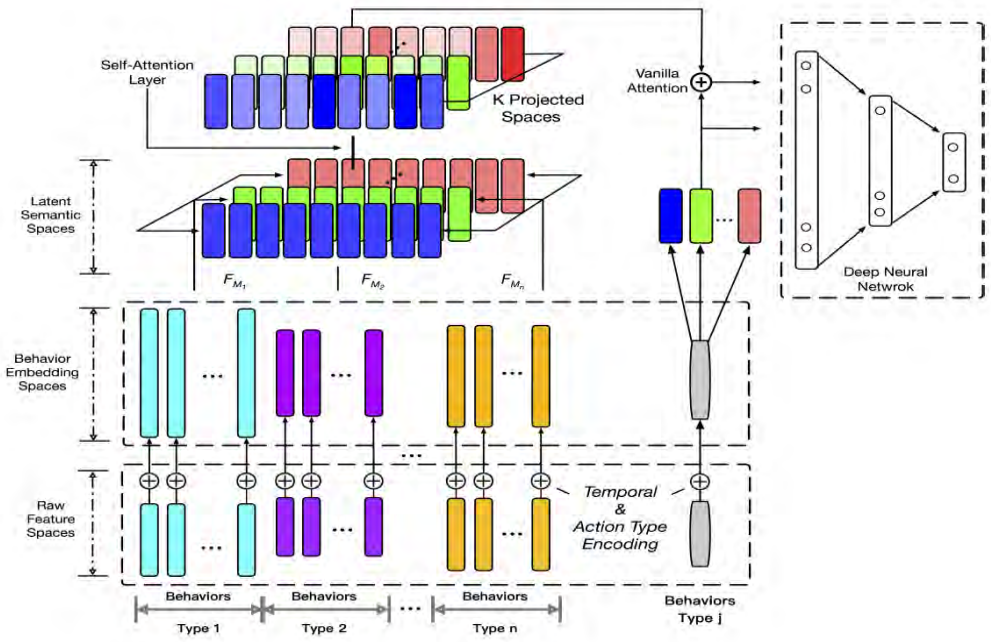


Figure 2.17 The framework for the ATRank behavior model [8]

ATRank is a framework for modeling user behavior based on attention. User behaviors pass through various elements within the model. Each element performs a specific function as follows:

a) Raw feature spaces

Raw feature spaces are used to separate behavior into groups of various behavior groups.

b) Behavior embedding spaces

After dividing the behavior into different groups, the Behavior embedding spaces is then using to embed raw features of user behaviors.

c) Latent semantic spaces

Latent semantic spaces are used to create connections between behaviors and used for comparing behaviors.

d) Self-attention layer

Self-attention is use in this model in order to capture relations between behaviors , e.g. buy items, search keywords, click ads, use coupons, watch videos offered by shops etc., in each semantic space of heterogeneous user behaviors.

e) Vanilla attention

Vanilla attention is used in this model in order to create the context vector that is relevant to the target user due to the embedding vector which is to be predicted.

2.7. BERT4Rec

BERT4Rec is the model that uses to predict the next item which target user prefer to. The illustrate of this model is shown in Figure 2.18.

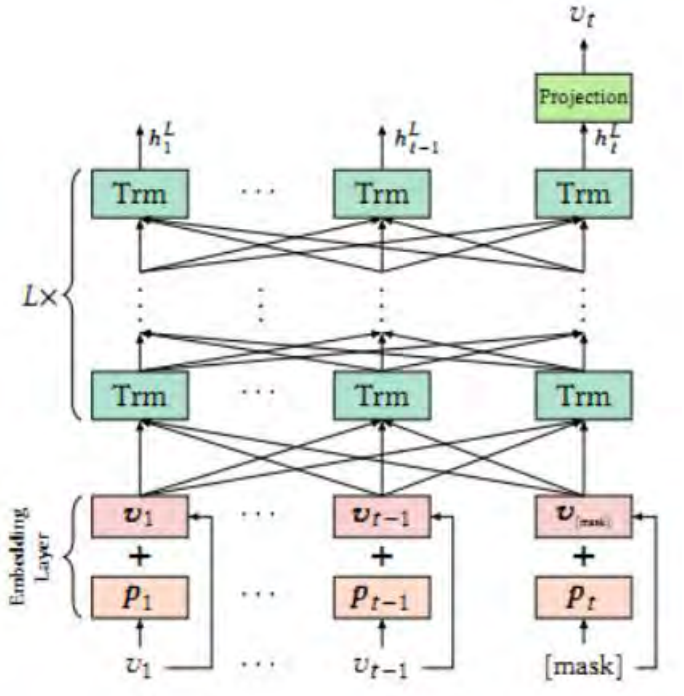


Figure 2.18 BERT4Rec model architecture [7]

Attention is used in this model in order to weight items in the item sequence based on the similarity level among the items. The Attention score, similar to the similarity score, is then calculated to finding the similarity between the items on the item's sequence. Therefore, BERT4Rec can capture inner-relation between items and it can use that inner-relation to predict the next item.

It can be seen that the related works that mention in this chapter, except for CF and NCF, are also content-based filtering which use only past preference of target user to predict rating. Even though in the attention, ATRank and BERT4Rec, are not the same weight on their items in the sequence, but they do not consider the latent relations in the sequence of users and items. And because of their works do not consider the sequence in term of collaborative filtering, the information from users' neighbor do not use to predict rating for the target user. Therefore, the prediction results are not diverse. Then, we would like to consider the information from neighbors which is a collaborative filtering method in order to resolve these problems and make predictive more various results. Moreover, we capture inner latent features between users and items in the sequence by utilizing LCU. Then, utilize attention in order to personalized target user's neighbor. So, our target user's neighbor will have a different important level of the different target users.

CHAPTER 3
METHODOLOGY

In this chapter, we will explain the proposed method for predicting the rating score of the target item $v_j \in Item$ for the target user $u_i \in User$. We divide our proposed method into two main parts. The first part presents the method that uses to personalize rating predictions by applying a Region Embedding with Local Context Unit (LCU). The second part presents the method that differentiates among target users by applying Attention.

3.1. Personalized Rating Predictions

We first present the personalized rating prediction method, which proposes to capture and extract latent relation in the historical sequence of users and items by applying Region Embedding with the Local Context Unit in order to utilize that latent relation for personalized rating predictions. This part comprises six steps which are create a user and item embeddings, create Local Context Units, project embeddings with Local Context Units, find rating scores of neighbors, compute a rating score of the target user, and finally feed into a fully connected layer. Figure 3.1 is shown the model architecture of this part.

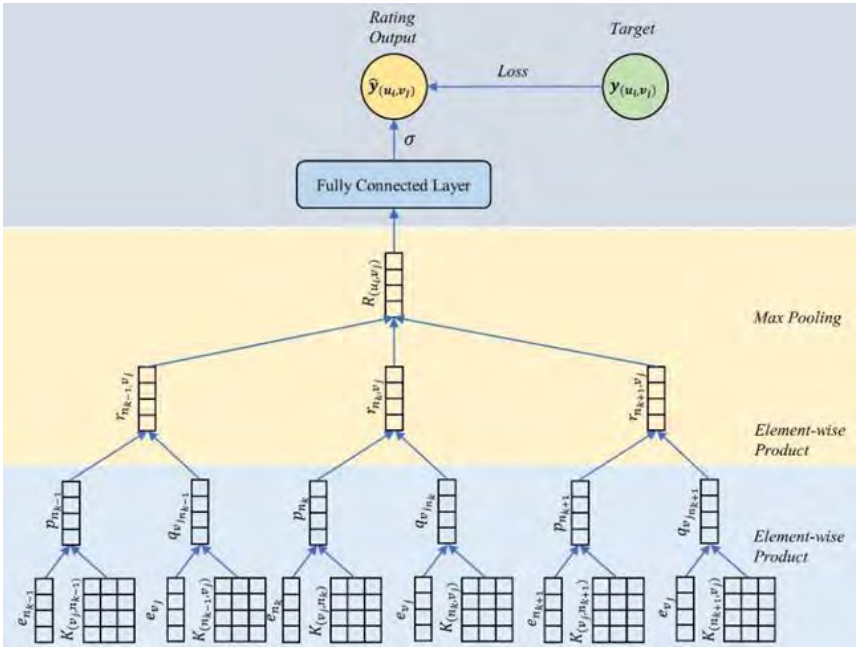


Figure 3.1 Model Architecture without Attention

3.1.1. Create User and Item Embeddings

At first, we define the sequence of items rated by u_i's neighbor (i.e., n_k) as user historical sequence $uhs_{n_k} \in UHS$ (as shown in Table 1.1 and 1.2) and define the sequence of users who have rated on target item v_j as item historical sequence $ihs_{v_j} \in IHS$ (as shown in Table 1.3 and 1.4). We define u_i's neighbor $n_k \in ihs_{v_j}$ as a user who has rated on target item v_j at the kth order in the ihs_{v_j}. Next, we create user neighbor's embedding by looking up in user embedding matrix $\boldsymbol{E_u} \in R^{|User| \times e}$, where e is the embedding size, that comes from user historical sequence. In the same way, we create item's embedding by looking up in item embedding matrix $\boldsymbol{E_i} \in R^{|Item| \times e}$ that comes from item historical sequence. Therefore, we will get the user neighbor's embedding as illustrates in Figure 3.2 and the item embedding as illustrate in Figure 3.3 from this step.

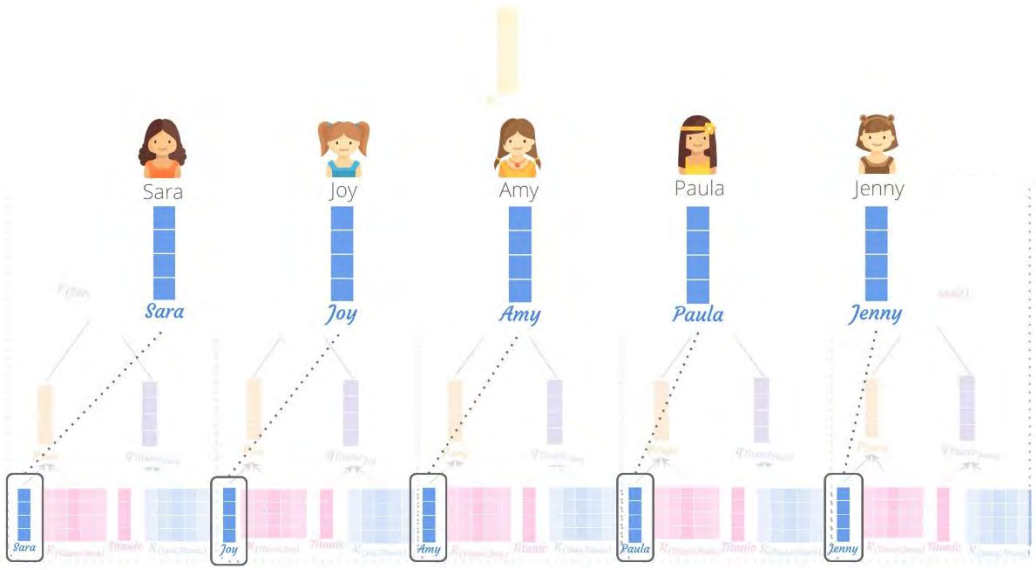


Figure 3.2 User Neighbor's Embedding

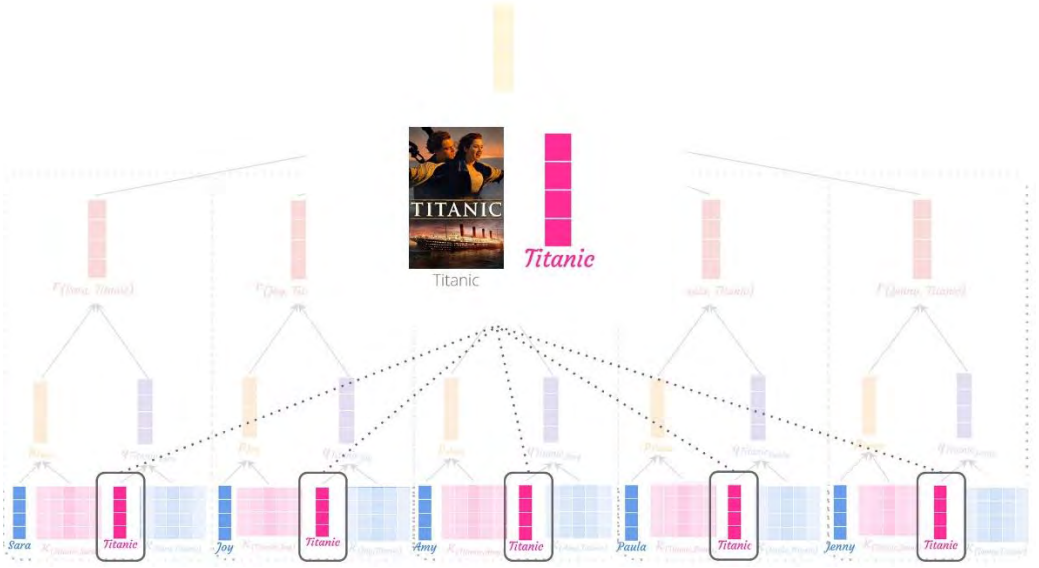


Figure 3.3 Item Embedding

3.1.2. Create Local Context Unit

We create two types of Local Context Unit in order to find latent relation from the historical sequence of users and items. We create Item-User Local Context Unit $K_{(v_j,n_k)}$ which is a weighting matrix that each column shows an important level on how surrounding users affect, have a latent relation, to the u_is' neighbor n_k who has rated on target item v_j from ihs_{v_j}. And we create User-Item Local Context Unit $K_{(n_k,v_j)}$ which is a weighting matrix that each column shows an important level on how surrounding items affect to target item v_j rated by n_k from uhs_{n_k}. For the target item v_j, we create $K_{(v_j,n_k)}$ for every n_k in ihs_{v_j}. For example, we assume that target item is Titanic. As you can see in Table 1.3, Titanic has 5 neighbors n_k who have rated on it. So, we create $K_{(n_k,v_j)}$ by looking up in uhs_{n_k} to find Local Context Unit of target item v_j for each uhs_{n_k}. It means that, if u_i's neighbor n_k are Sara and Amy, we will find Local Context Unit of Titanic $v_{Titanic}$ in uhs_{Sara} and uhs_{Amy} by selecting Local Context Unit of Sara and Amy from the user historical sequence uhs_{Sara} of Sara and uhs_{Amy} of Amy. After this step complete, we will get item-user Local Context Unit $K_{(v_j,n_k)}$ as shown in Figure 5 and User-Item Local Context Unit $K_{(n_k,v_j)}$ as shown in Figure 6.

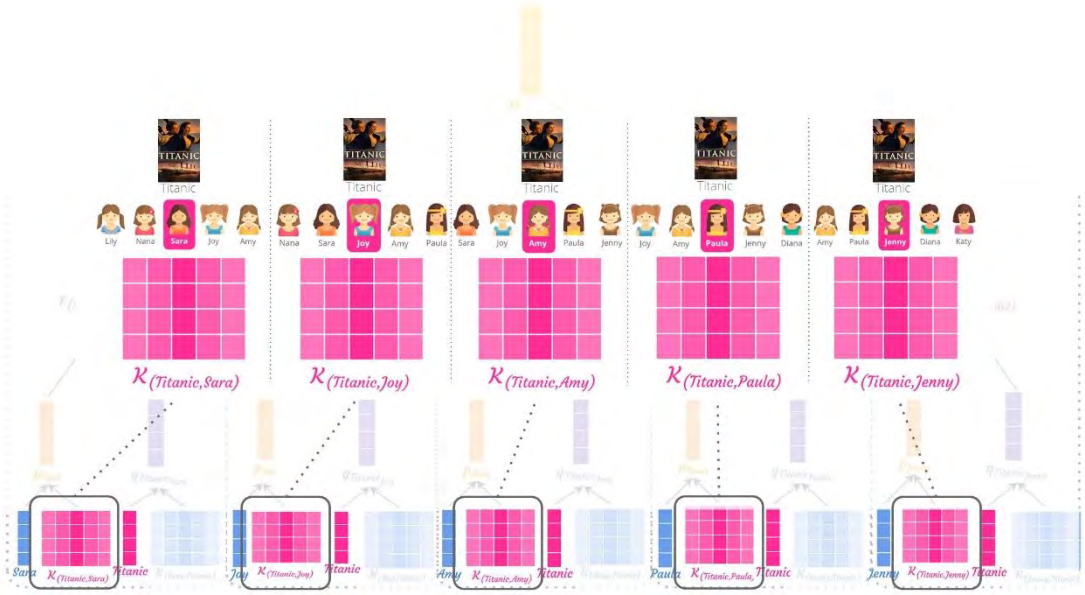


Figure 3.4 Item-User Local Context Unit

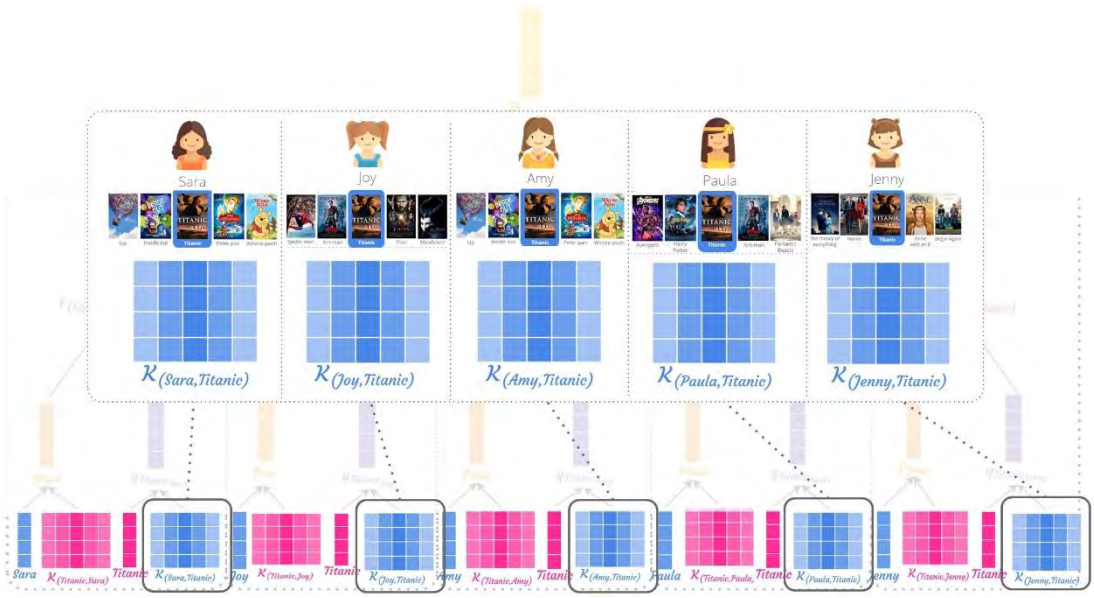


Figure 3.5 User-Item Local Context Unit

3.1.3. Project Embedding with Local Context Unit

After $K_{(v_j,n_k)}$ and $K_{(n_k,v_j)}$ are created, in order to find user n_k profile (i.e., p_{n_k}), we would like to project user characteristics with latent features surrounding that user. Therefore, we use an element-wise multiplication between user n_k embedding (i.e., e_{n_k}) and Item-User Local Context Unit $K_{(v_j,n_k)}$ by using Equation (3.1) .You can see the illustrated in Figure 3.6. In the

same way, in order to find item v_j profile on sequence of neighbor n_k (i.e., $q_{v_{j n_k}}$), we would like to project item characteristics with latent features surrounding that item. Therefore, we use an element-wise multiplication between item v_j embedding (i.e., e_{v_j}) and User-Item Local Context Unit $K_{(n_k, v_j)}$ by using Equation (3.2) as illustrated by Figure 3.7.

$$p_{n_k} = K_{(v_j, n_k)} \odot e_{n_k} \tag{3.1}$$

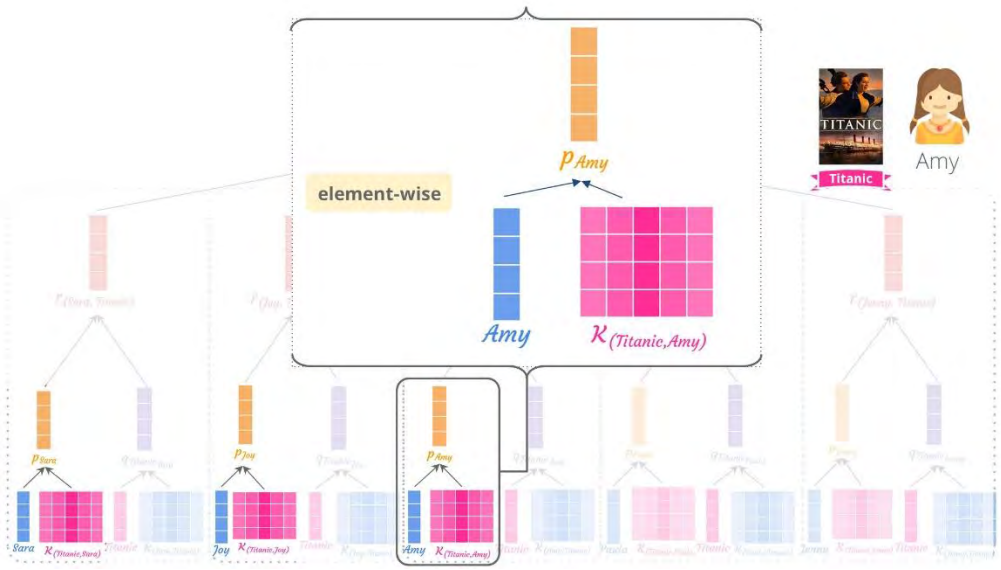


Figure 3.6 Find User Profile

$$q_{v_{j n_k}} = K_{(n_k, v_j)} \odot e_{v_j} \tag{3.2}$$

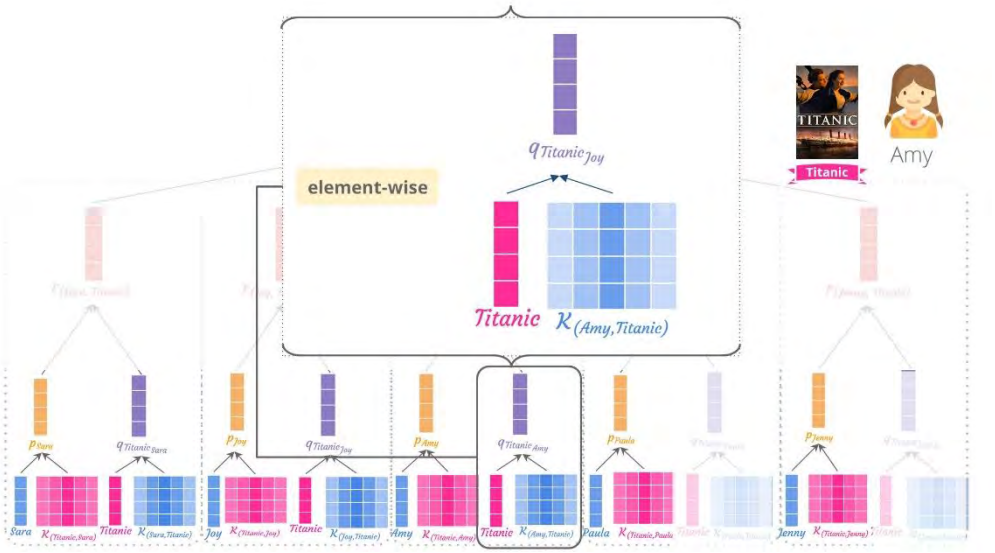


Figure 3.7 Find Item Profile

After we finish this step, we will get user profile p_{n_k} that shows the representation of n_k under the item v_j's historical sequence and get item profile $q_{v_{jn_k}}$ that represents characteristics of target item in term of u_i's neighbor n_k .

3.1.4. Find Rating Scores of Neighbors

After we have got the user profile and item profile which have considered the latent features from the previous step, we will create a rating score for each neighbor by using these profiles.

According to Neural Collaborative Filtering concept, rating score of each u_i's neighbor toward target item v_j (i.e., r_{n_k,v_j}) is computed from user profile p_{n_k} and item profile $q_{v_{jn_k}}$ by using an element-wise multiplication as Equation (3.3). The illustrate of this step is shown by Figure 3.8.

$$r_{n_k,v_j} = p_{n_k} \odot q_{v_{jn_k}} \tag{3.3}$$

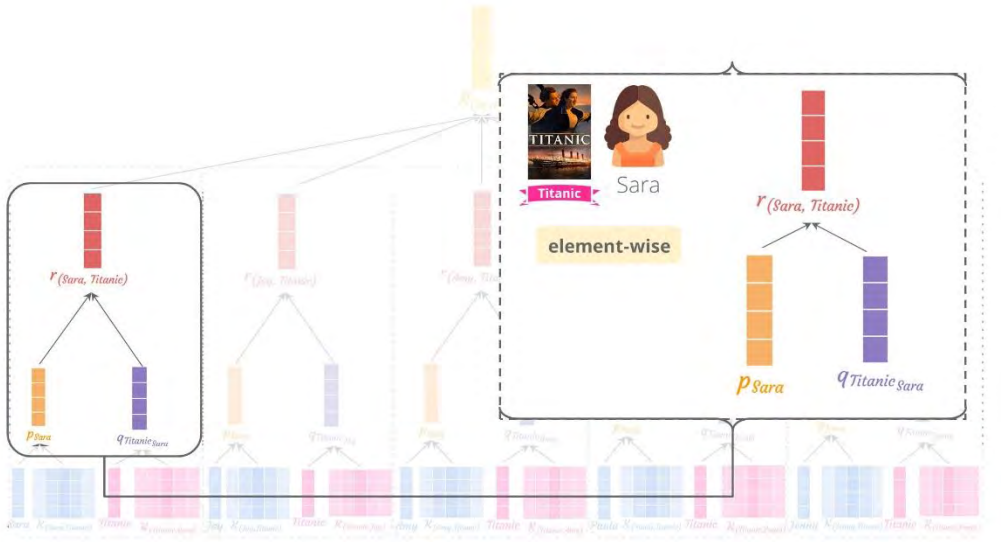


Figure 3.8 Find Rating Score of Neighbors.

3.1.5. Compute Rating Score of Target User

After we have rating scores of each neighbor, we then compute rating score for the target user. Inspired by rating prediction equation of Collaborative filtering which calculated rating score by using the summation of rating score from all neighbors of the target user. Therefore, we compute a rating score vector of target movie v_j for target user u_i (i.e., $R_{(u_i,v_j)}$) by applying max-pooling operation on all r_{n_k,v_j} for all the neighbors n_k of the target user u_i toward the target item v_j. It can be computed by using Equation (3.4) and the illustration of this step is shown in Figure 3.9.

$$R_{(u_i,v_j)} = max\left(\left[r_{n_0,v_j}\ r_{n_1,v_j}\ \dots\ r_{n_{m-1},v_j}\ r_{n_m,v_j}\right]\right);\ m\ is\ number\ of\ n_k \quad (3.4)$$

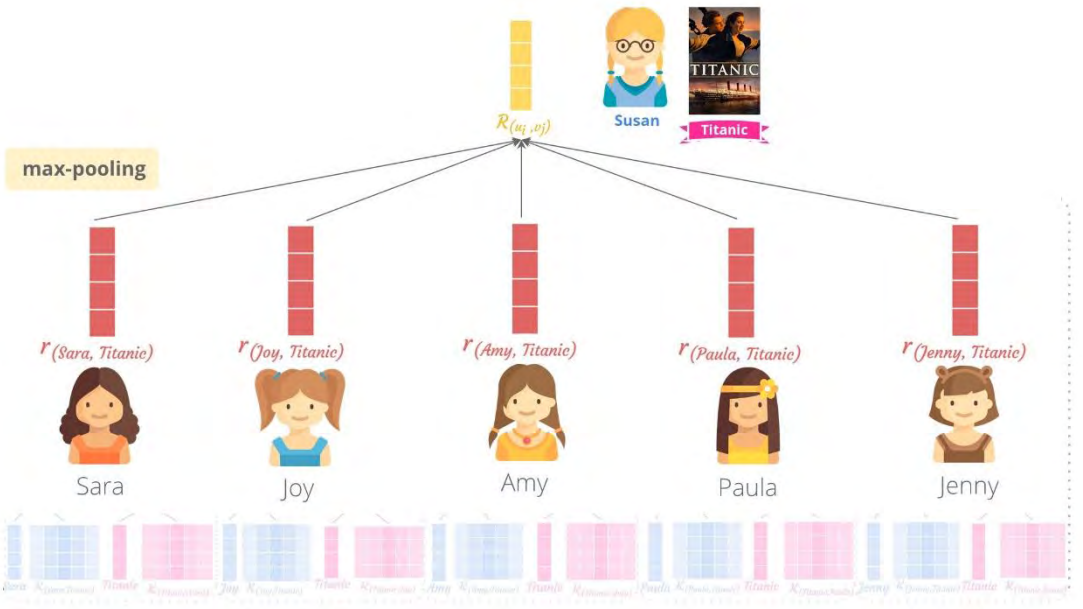


Figure 3.9 Compute Rating Score of Target User.

3.1.6. Prediction Rating Score Scalar

Finally, we would like to learn a rating score vector to be a rating score scalar. So, we feed $R_{(u_i,v_j)}$ of the target user u_i into the fully connected layer to predict the rating score output $\hat{y}_{(u_i,v_j)}$ of the target item v_j as Equation (3.5).

$$\hat{y}_{(u_i,v_j)} = \sigma(W \cdot R_{(u_i,v_j)} + b) \tag{3.5}$$

where W and b are the weight matrix and bias, respectively, and σ is the sigmoid function of the output layer. We illustrated this step by Figure 3.10.

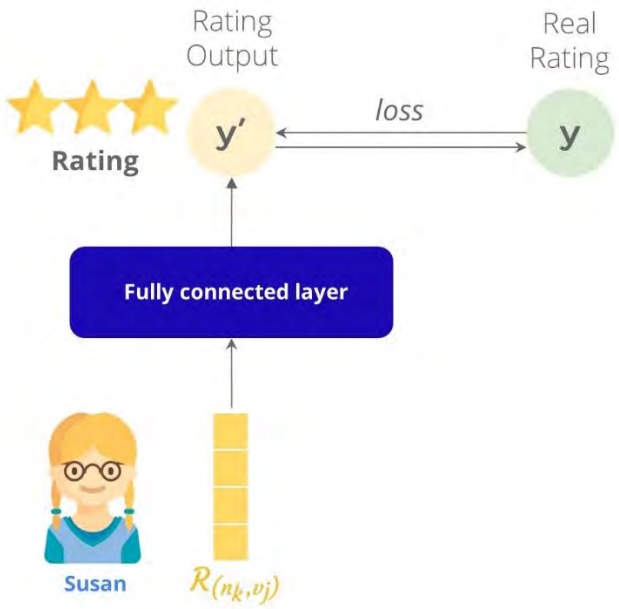


Figure 3.10 Feed into Fully Connected Layer

3.2. Differentiate among Target Users by Applying Attention

Because of output rating from 3.1 did not serve for a personalized recommendation. Therefore, different target users will receive the same rating score on the same target item because users' neighbors of these target users are the same. For example, if the model predicts a rating score of Titanic for Susan and Alice, both of them will receive the same rating score. Because the model of Susan and Alice have the same neighbor (Sara, Joy, Amy, Paula, Jenny) as shown in Figure3.11.

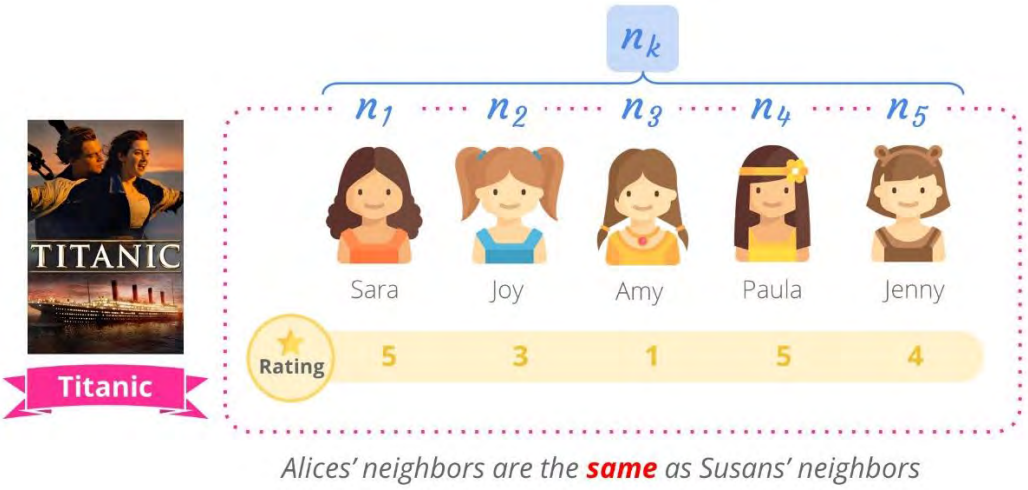


Figure 3.11 A list of neighbors personalized without Attention

They have the same neighbor embeddings and same Local Context Units, so these models predict the same result. Therefore, we would like to differentiate among target users by applying Attention. Transformer proposed by [1] in Attention is applied in our work by including a sequence of the user who has rated on the target item ihs_{v_j}. Therefore, the output from the Transformer is a list of neighbors personalized by each target user where each target user is the query. In other word, when we put different queries or different target users into the Transformer, you will get a different list of neighbors. Now, we get a list of neighbors according to each target users' perspective. The model architecture with attention is shown in Figure 3.12.

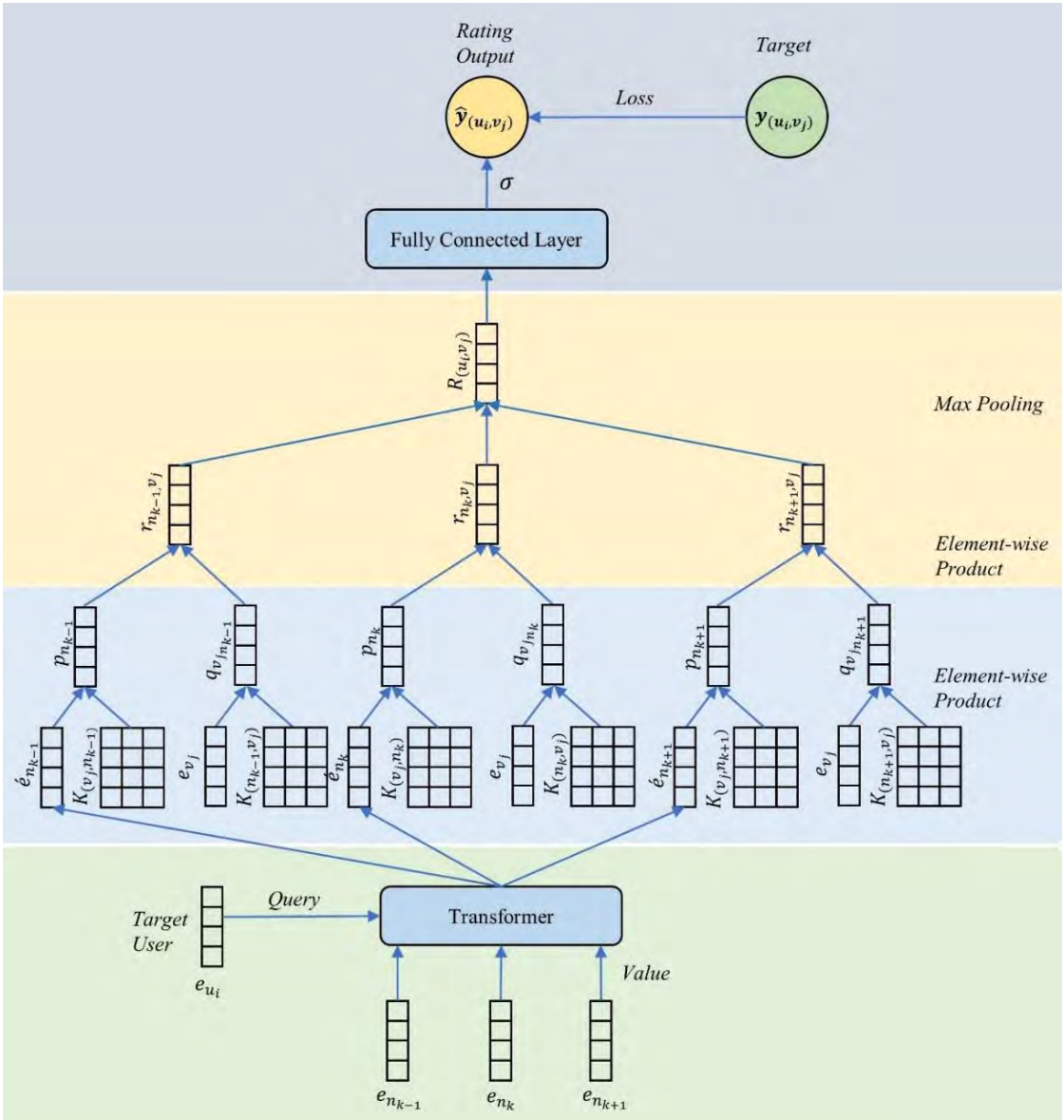


Figure 3.12 Model architecture with Attention

This model has the same working as the model in Figure 3.1 besides the attention in the first layer of the model. Therefore, if the model predicts the rating score of Titanic for Susan and Alice, as shown in Figure 3.13, neighbors of Susan and Alice are now different, as shown in Figure 3.13.

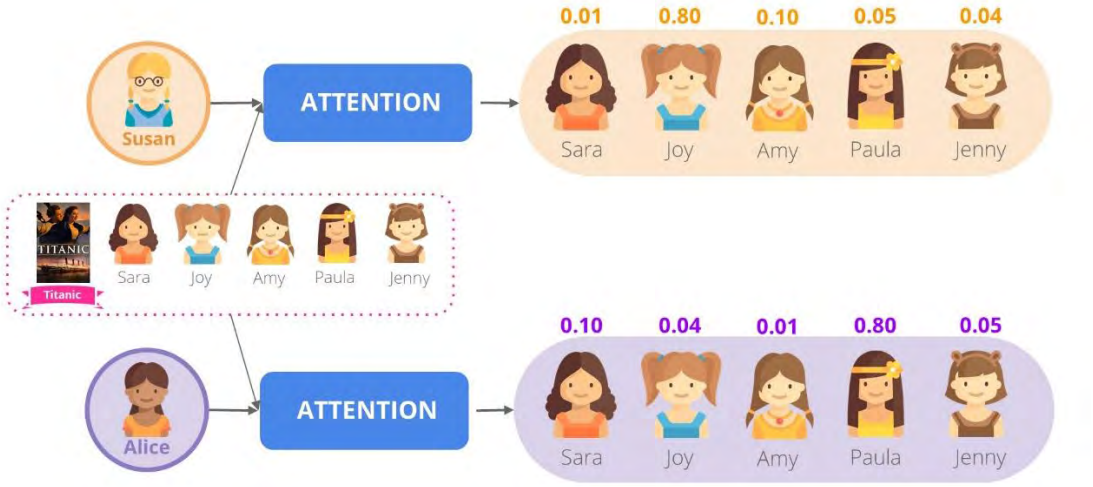


Figure 3.13 A list of neighbors personalized by using Attention

The neighbors of Susan and Alice will have a different attention score based on the target user. Neighbors of Susan are $Sara_{Susan}$, Joy_{Susan}, Amy_{Susan}, $Paula_{Susan}$ and $Jenny_{Susan}$ and neighbors of Alice are $Sara_{Alice}$, Joy_{Alice}, Amy_{Alice}, $Paula_{Alice}$ and $Jenny_{Alice}$. Each neighbor has its own attention score and this attention score is similar to the similarity score between the neighbor and the target user. Therefore, when we input neighbors into the encoding side and input query user, i.e. target user into the decoding side, then the model generates the new neighbors that personalize to each target user.

CHAPTER 4
EXPERIMENTAL EVALUATION

In this chapter, we will prove our two assumption points. First, we believe that user historical sequence and item historical sequence both have important latent relations inside that can affect to user preference. So, we propose a new method that captures relation inside both sequences and we will compare experimental results of our proposed model with Neural Collaborative Filtering (NCF) that does not use and consider latent relation in both user historical sequence and item historical sequence. Second, we apply attention to our proposed method in order to find personalized target users' neighbors because the method without attention did not serve for a personalized recommendation. So, we will compare the result between the model that applies attention and model that do not apply attention. Therefore, the organization of this chapter is as follows. First, the details of the datasets that are used in these experiments are explained. Second, evaluation metrics in these experiments are introduced including NDCG@K and HitRate@K. Finally, experiment results between the proposed method and NCF and between the model with attention and model without attention are compared.

4.1. Datasets

The dataset that we use in our experiment is MovieLens. MovieLens is a public movie rating dataset provided by GroupLens organization. This dataset has been widely used to evaluate the collaborative filtering method. The dataset has many versions. The version that we use in this experiment is ml-latest-small. It contains 9,000 movies, 600 users, and 100,000 ratings with a rating range of 0.5 to 5. This dataset consists of four columns which are userId, movieId, rating, and timestamp. The sample of the ml-latest-small dataset is shown in Table 4.1. The first record means userId *1* rated movieId *47* with rating *5.0* at timestamp *964983815*.

Table 4.1 The sample of MovieLens dataset

userId	movieId	rating	timestamp
1	47	5.0	964983815
1	70	3.0	964982400
1	110	4.0	964982176
2	47	4.0	964984041
2	110	5.0	964984100
3	70	3.0	964983500

For data preprocessing, we first split the data into 60 percent for the training set and 40 percent for test set in order to train and test the model. Our proposed method considers both user historical sequence and item historical sequence. So, we prepare user historical sequence for each user by extract all items rated by that user and ascending sort these items by timestamp. In the same way, we prepare item historical sequence by extract all raters of the item and ascending sort these raters by timestamp. Table 4.2 shows user historical sequences and item historical sequences that create from sample data in Table 4.1.

Table 4.2 User historical sequence and Item historical sequence

userId	User historical sequence	movieId	Item historical sequence
1	110, 70, 47	47	1, 2
2	47, 110	70	1, 3
3	70	110	1, 2

The original data is very sparse. Most users interact with few items and most items are interacted by a few users. Since sparse data make the model more difficult to evaluate, we filter the dataset by choosing only the user who has at least 20 interactions and choosing only the movie which has at least 20 interactions. The characteristics of the dataset after preprocessing are summarized in Table 4.3.

Table 4.3 The characteristics of the dataset

Number of users	478
Number of items	789
Number of interactions	46784
Number of rating	10
Rating range	0.5-5
Max user historical sequence	220
Max item historical sequence	186

4.2. Evaluation Metrics

In this section, we will present evaluation metrics that evaluate our model. To compare the efficiency, we choose Normalized Discounted Cumulative Gain (NDCG@K) and HitRate@K. These two-evaluation metrics can be described as follows.

4.2.1. NDCG@K

NDCG@K measures the ranking efficiency of the model. NDCG@K considers not only predicted rating scores but also the order of items in the recommendation list predicted from the model. The highly relevant item could rank in the topper rank than the lowly relevant item. If the model has high NDCG value, it means that model has high efficiency in item ranking. Let K denotes the top number of items on the recommendation list and U denotes the user set. NDCG@K can be computed as Equation(4.1).

$$NDCG@K = \sum_{u \in U} \frac{DCG_{uk}}{IDCG_{uk}} \tag{4.1}$$

where DCG_{uk} is Discounted Cumulative Gain of user u for top k items. DCG_{uk} can be computed as Equation(4.2), and $IDCG_{uk}$ denotes ideal discounted cumulative gain which is highest DCG value among the possible ranked item list. $IDCG_{uk}$ can be computed as Equation(4.3).

$$DCG_{uk} = \sum_{i=1}^{k} \frac{2^{rel_i - 1}}{\log_2(i+1)} \quad (4.2)$$

$$IDCG_{uk} = \sum_{i=1}^{k} \frac{2^{rel_i - 1}}{\log_2(i+1)} \quad (4.3)$$

where i is rank index of items in top K and rel_i is actual rating score of items at rank i in top K recommendation list.

Table 4.4 Actual rating and Predicted rating of target user u

Item	Actual rating	Predicted rating
9	2	3
10	5	4
13	1	1
15	3	3
17	3	3
20	3	5

To clarify more about NDCG@K, for target user u, let Table 4.4 be actual rating and predicted rating of target user u. First, actual items are descending sorted by their actual ratings and predicted items are descending sorted by their predicted ratings. Let $K = 3$, Top three actual items that have highest actual rating are selected to actual rank item. In the same way, predicted items that have highest predicted rating are selected to predicted rank item as shown in Table 4.5.

Table 4.5 Sorted actual rating and predicted rating of target user u

Sorted actual item	Sorted actual rating	Sorted predicted item	Sorted predicted rating
10	5	20	5
15	3	10	4
20	3	9	3
17	3	15	3

Sorted actual item	Sorted actual rating	Sorted predicted item	Sorted predicted rating
9	2	17	3
13	1	13	1

To compute NDCG@K, we need to find predicted rank rating and actual rank rating. For predicted rank rating, we loop in actual rank item. From Table 4.6, first item in actual rank item is item *10*, we search in predicted rank item to find the position of item *10*. The position of item *10* in predicted rank item is position two. Backing to actual rank item, we bring the rating of actual rank item at position two to be predicted rank rating. The rating of actual rank item at position two is rating *3*, so, rating *3* is brought to be predicted rank rating of item *20*. If there is actual item in actual rank item that is not in predicted rank item, we will set predicted rank rating at that position to 0. Item *15* is not in predicted rank item, so, set predicted rank rating in position two to 0. After loop all actual rank item, the predicted rank rating of item *20, 10, 9* are *3, 0, 5* respectively. For actual rank rating, actual rating is commonly bring to actual rank rating. The actual rank rating and predicted rank rating are shown in Table 4.7.

Table 4.6 Actual rank item and Predicted rank item of target user u

Actual rank item	Actual rating	Predicted rank item	Predicted rating
10	5	20	5
15	3	10	4
20	3	9	3

Table 4.7 Actual rank rating and Predicted rank rating of target user u

Actual rank item	Actual rank rating	Predicted rank item	Predicted rank rating
10	5	20	3
15	3	10	0
20	3	9	5

Apart from predicted rank rating and actual rank rating, rank index is also used to compute DCG_{u3} and $IDCG_{u3}$. Rank index is rank number of items in top K recommendation list. If same ratings appear more than one, rank index of those same rating will assign in the same rank as shown in Table 4.8.

Table 4.8 Rank index of items in Top K recommendation list

Actual rating	5	3	3
Rank Index	1	2	2

Lastly, predicted rank rating and rank index are used to compute DCG_{u3} while actual rank rating and rank index are used to compute $IDCG_{u3}$ as shown in Figure 4.1

$$DCG_{u3} = \frac{2^3-1}{log_2(1+1)} + \frac{2^0-1}{log_2(2+1)} + \frac{2^5-1}{log_2(2+1)} = 26.55$$

$$IDCG_{u3} = \frac{2^5-1}{log_2(1+1)} + \frac{2^3-1}{log_2(2+1)} + \frac{2^3-1}{log_2(2+1)} = 39.83$$

$$NDGC_{u3} = \frac{DCG_{u3}}{IDCG_{u3}} = \frac{30.98}{39.83} = 0.67$$

Figure 4.1 NDCG@3 Computation

Noted that the value of NDCG are in range of 0 to 1, a higher value of NDCG indicates better performance of model ranking.

4.2.2. HitRate@K

HitRate@K measure efficiency of model prediction. Let K denotes the number of items on the recommendation list. If predicted item in top K recommendation list are rated by target user, we consider that item is a hit. HitRate@K consider only a hit of predicted item. It does not consider order of

predicted item like NDCG@K. We apply cut off rating threshold to HitRate@K in order to calculate only item that have rating more than cut off rating threshold, because higher rating indicates more user interest and we do not focus on low rating.

Table 4.9 Actual rating and Predicted rating of target user u

Item	Actual rating	Predicted rating
9	2	3
10	5	4
13	1	1
15	3	3
17	3	3
20	3	5

To compute HitRate@K for target user u, let Table 4.9 be actual rating and predicted rating of target user u. First, we descending sort actual item and predicted item by actual rating and predicted rating respectively. Let $K = 3$, we selected only top three items which have highest rating from both actual item and predicted item as shown in Table 4.10.

Table 4.10 Sorted actual rating and predicted rating of target user u

Sorted actual item	Sorted actual rating	Sorted predicted item	Sorted predicted rating
10	5	20	5
15	3	10	4
20	3	9	3
17	3	15	3
9	2	17	3
13	1	13	1

Next, we loop in actual item, if actual item appears in predicted rank item, it is a hit. Let 1 means actual item hits in predicted rank item and 0 means

actual item does not hit in predicted rank item. After finding all items in actual item, we compute HitRate@3 by averaging of all hits value together as shown in Table 4.11.

Table 4.11 HitRate@3 Computation

Sorted actual item	Hits
10	1
15	0
20	1
Hit rate	$\frac{2}{3} = 0.67$

4.3. Experimental Results

For parameter settings, we used the region size of 7 and we also added a zero padding of length 3 to both the head and tail of each user historical sequence and item historical sequence. The embedding size is 16, the batch size is 64 and the learning rate is 0.0001. We used the Adam as the optimizer and the L2 as the regularizer.

Our proposed method has two assumptions. First, in our opinion, sequence around the target user and sequence around the target item have some important latent relations. We proposed a method applying Region Embedding that able to extract latent relations from user historical sequence and item historical sequence. To evaluate this assumption, we will compare experimental results with Neural Collaborative Filtering (NCF) that do not consider latent relations in both user historical sequence and item historical sequence. Second, we apply attention to personalize neighbors for each target user in order to serve for personalized recommendations. To evaluate this second assumption, we will compare the results between our proposed model with attention and our proposed model without attention. These three methods, a proposed method, a proposed method without attention, and NCF are implemented on the same dataset to avoid bias. We vary Top-K recommendation lists to 3, 5, and 10. The experimental results are shown in the following.

Table 4.12 Comparison of NDCG@K [3,5,10] on the proposed method, the proposed method without attention and NCF

Model	**NDCG@K**		
	K=3	**K=5**	**K=10**
The proposed method	**0.675269**	**0.717785**	**0.766534**
The proposed method without attention	0.664182	0.688507	0.725360
NCF	0.658600	0.674100	0.706900

Table 4.13 Comparison of HitRate@K [3,5,10] on the proposed method, the proposed method without attention, and NCF

Model	**HitRate@K**		
	K=3	**K=5**	**K=10**
The proposed method	**0.503972**	**0.526275**	**0.605291**
The proposed method without attention	0.481094	0.501176	0.577592
NCF	0.460500	0.487800	0.550300

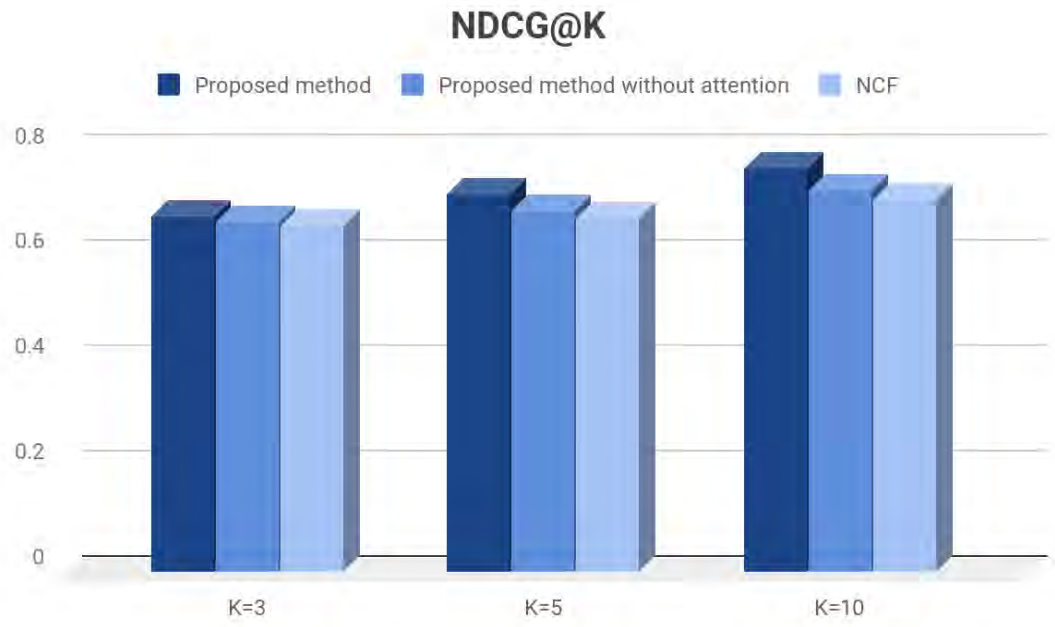


Figure 4.2 Comparison of NDCG@K [3, 5, 10] on the proposed method, the proposed method without attention and NCF

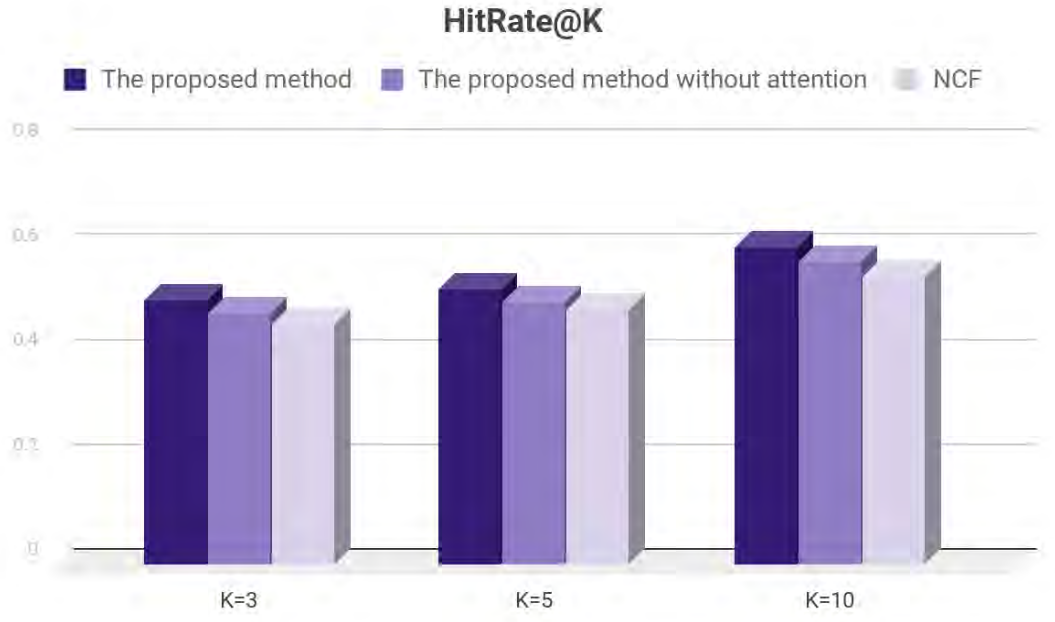


Figure 4.3 Comparison of HitRate@K [3, 5, 10] on the proposed method, the proposed method without attention and NCF

Table 4.12 and Figure 4.2 show a comparison of NDCG@K on the proposed method. The model that has the highest NDCG@K is the proposed method, the proposed method without attention and NCF respectively.

Table 4.13 and Figure 4.3 show a comparison of HitRate@K on the proposed method. The model that has the highest HitRate@K is the proposed method, the proposed method without attention and NCF respectively.

Results of NDCG@K and HitRate@K show that the performance of our proposed method and our proposed method without attention are all higher than NCF because our proposed method captures latent relations in both user historical sequence and item historical sequence but NCF does not consider any historical sequences. For efficiency of attention, our proposed method that applies attention gain higher NDCG@K and HitRate@K than our proposed method that does not apply attention because attention personalizes neighbors for each target user.

CHAPTER 5
CONCLUSION

5.1. Conclusion

Attention-based recommender systems by applying Region Embedding is proposed to capture and extract latent relation in the historical sequence of users and items by applying Region Embedding with the Local Context Unit in order to utilize that latent relation for personalized rating predictions. Moreover, we apply Attention to our work in order to differentiate among target users. Then, compare experimental results with NCF, which does not consider the user relation and item relation in the sequence. From the experimental results, it can be concluded that the proposed method which considers latent relation in the historical sequence of users and items is better than NCF. And because we find an important level between users' neighbors and target users by utilizing Attention, so the model with Attention is better than the model without Attention.

REFERENCES

[1] Koren, Y., Bell, R., & Volinsky, C. 2009. Matrix Factorization Techniques for Recommender Systems. Computer, 42(8), 42-49. http://doi.org/10.1109/MC.2009.263

[2] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. *Proceedings of the 26th International Conference on World Wide Web (WWW '17).* http://dx.doi.org/10.1145/3038912.3052569

[3] L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. *Proceedings of the third ACM conference on Recommender systems, 2009.*

[4] Chao Qiao, Bo Huang, Guocheng Niu, Daren Li, Daxiang Dong, Wei He, Dianhai Yu, and Hua Wu. 2018. A new method of region embedding for text classification. *International Conference on Learning Representations* (2018).

[5] Padipat Sitkrongwongn, and Atsuhiro Takasu. 2019. Unsupervised context extraction via region embedding for context-aware recommendations. *Proceedings of the 23rd International Database Applications & Engineering Symposium (IDEAS '19).* ACM, Athens, Greece.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762.*

[7] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. *arXiv:1904.06690v2.* https://doi.org/10.1145/3357384.3357895

[8] Chang Zhou, Jinze Bai, Junshuai Song, Xiaofei Liu, Zhengchao Zhao, Xiusi Chen, and Jun Gao. 2017. ATRank: An Attention-Based User Behavior Modeling Framework for Recommendation. *arXiv preprint arXiv:1711.06632.*

[9] GroupLens gratefully acknowledges the support of the National Science Foundation under research grants, *MovieLens Latest Datasets* [Online].

Available from: https://grouplens.org/datasets/movielens/latest [2019, December 20]

[10] Titipat Achakulvisut, *Collaborative filtering ฟีเจอร์การแนะนำเพลงของ Spotify* [Online]. Figure from: https://tupleblog.github.io/spotify [2020, January 15]

[11] Kevin Liao, *Prototyping a Recommender System Step by Step Part 2: Alternating Least Square (ALS) Matrix Factorization in Collaborative Filtering* [Online]. Figure from: https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-2-alternating-least-square-als-matrix-4a76c58714a1[2020, January 15]

[12] Mukesh Mithrakumar, *Singular Value Decomposition with Tensorflow 2.0* [Online]. Figure from: https://medium.com/@mukesh.mithrakumar/singular-value-decomposition-with-tensorflow-2-0-af36fa31c772 [2020, January 16]

[13] Nuria Oliver, *Data Mining Methods for Recommender Systems* [Online]. Figure from: https://www.researchgate.net/figure/PCA-analysis-of-a-two-dimensional-point-cloud-from-a-combination-of-Gaussians-The_fig2_225924875 [2020, January 16]

[14] Aston Zhang, Zack C. Lipton, Mu Li, and Alex J. Smola. 2020. *Deep Recurrent Neural Networks* [Online]. Figure from: https://d2l.ai/chapter_recurrent-modern/deep-rnn.html [2020, March 25]

[15] Christopher Olah. 2015. *Neural Networks, Types, and Functional Programming* [Online]. Figure from: http://colah.github.io/posts/2015-09-NN-Types-FP/ [2020, March 25]

[16] Aditi Mittal. 2019. *Understanding RNN and LSTM* [Online]. Figure from: https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e [2020, March 25]

APPENDIX

APPENDIX A

The Project Proposal of Course 2301399 Project Proposal

Academic Year 2019

Project Title (Thai)	ระบบแนะนำตามพื้นฐานความสนใจโดยประยุกต์ใช้การฝังตรึงเชิงพื้นที่
Project Title (English)	Attention-Based Recommender Systems by Applying Region Embedding
Project Advisor	1. Associate Professor Saranya Maneeroj, Ph.D.
	2. Assistant Professor Kitiporn Plaimas, Ph.D.
By	1. Miss Thitiya Sae-diae ID 5933620823
	2. Miss Umaporn Padungkiatwattana ID 5933669623
	Computer Science Program, Department of Mathematics and Computer Science
	Faculty of Science, Chulalongkorn University

Background and Rationale

In the present, the internet has many social roles. We can observe from the number of activities happening on social networks such as shopping, watching movie, listening to music, updating news, etc. These activities make things more convenient for users, as a result, the number of users who use these activities is increased dramatically. Then, many entrepreneurs try to find a way that users can easily access to their own interests, but each user has a wide range of preferences and product data are extensive and complex. Therefore, recommender system has been developed to find what users are mostly interested in and recommend it to other users.

Collaborative Filtering is one of the popular methods in the recommender system. This method creates recommendations by exploiting preference data from other users who are similar to a target user and predicting rating scores for that target user. For example, predicting rating scores of unseen movies for a target user, Collaborative

Filtering utilizes movie history of other users and that target user to find the similarity of users' preference. It then uses the opinion of the similar users to predict rating scores for that target user. Collaborative Filtering has many approaches such as using matrix factorization [5], using neural network [8], or using context information [6].

The main point of Recommender Systems is finding latent relations among users and items to increase recommendation performance. Many collaborative filtering methods utilize historical relation of a user or item in order to find similarity between either a pair of users or a pair of items. However, they do not consider the order of these historical sequences. In our opinion, historical sequence has important latent relation inside, not only historical relation of users or items but also order of these historical sequences can be used to find latent preference of users. Historical sequence in recommendation area can be divided into 2 types. Firstly, in term of a target user, system will consider the sequence of items that rated by that target user. Secondly, in term of target item, system will consider the sequence of users who have rated on that target item.

Table 1: *User historical sequence of Amy and Sara*

Amy

Rating	2	1	3	5	4
Movie	La La land	Romeo Juliet	50 first dates	Titanic	Me before you

Sara

Rating	5	4	4	1	2
Movie	Titanic	Peter Pan	Winnie the Pooh	Ted	Annabelle

Table 1 shows sequence in term of a target user. In this table, there are 2 target users, Amy and Sara. Both of them rate Titanic (target item) with 5 score but movie sequence of them are different. Amy rates Titanic in the fourth order and Sara rates Titanic in the first order. Because Titanic is rated on different orders in movie sequence of Amy and Sara, movies around it should be different. As shown in Table 1, movies

around Titanic in the movie sequence of Amy are the romantic movie as Titanic, whereas movies around Titanic in the movie sequence of Sara are cartoon movies. Therefore, Titanic rated by Amy and Titanic rated by Sara must be different because their movie sequences are different, even though it got the same rating score from Amy and Sara. It can be concluded that the order of movies in the historical sequence indicates users' interest and relation among these movies.

Table 2: *Item historical sequence of Titanic and Avengers*

Titanic

Rating	5	3	1	5	4
User	Sara	Joy	Amy	Paula	Jenny

Avengers

Rating	3	1	5	2	5
User	Robert	David	Sara	John	Paul

Table 2 shows sequences in term of target items. In this table, there are 2 target items, Titanic and Avengers. Both of them are rated by Sara (target user) with 5 score but their user sequences are different. Titanic is rated by Sara in the first order and Avengers is rated by Sara in the third order. Because Sara rates on different orders in user sequences of Titanic and Avengers, users around her should be different. As shown in Table 2, users around Sara in user sequences of Titanic are the same women as Sara, whereas users around Sara in user sequences of Avengers are men. Therefore, Sara rating Titanic and Sara rating Avengers must be different because their user sequences are different, even though Sara rates the same rating score on these movies. It can be concluded that the order of users in the historical sequence indicate user characteristic of user groups for these movies and relation among these users.

The order consideration is often used in Natural Language Processing (NLP) because the order of words in the sentence has meaning. Moreover, the relation between words and words around the target word, i.e., context word, also have meaning too. Region Embedding is then introduced to capture that relation by using Local Context Unit (LCU). Local Context Unit is a weighting matrix that captures the interactions

between a word and its neighbors in a text region [3]. Recently, Local Context Unit is applied to Recommendation work. For example, Local Context Unit is utilized in users' review for extracting context from review data and uses it to create a predictive model [7].

In this work, we propose a new recommendation method that is able to capture and extract latent relation in historical sequence of users and items by applying Region Embedding with the Local Context Unit in order to utilize that latent relation for personalized rating predictions. We explain this method in Figure 1.

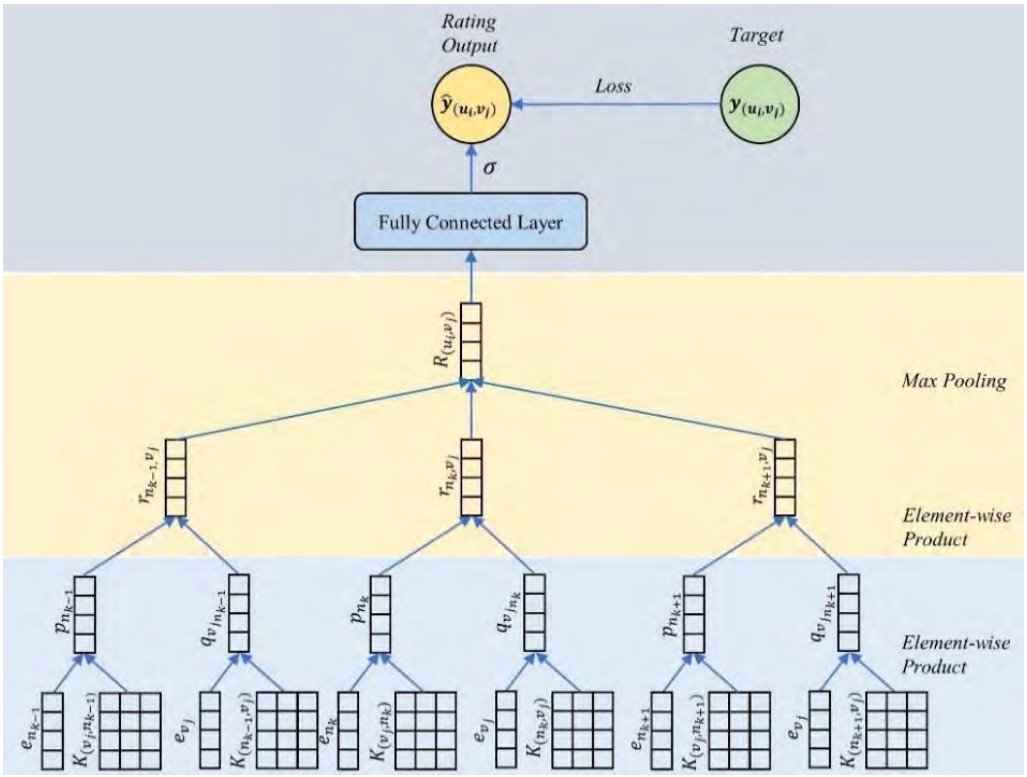


Figure 1 *Model Architecture without Attention*

For predicting rating score of target item $v_j \in Item$ for target user $u_i \in User$, we first create item historical sequence $ihs_{v_j} \in IHS$ (as shown in Table 2) that is the sequence of users who have rated on target item v_j and create user historical sequence $uhs_{n_k} \in UHS$ (as shown in Table 1) that is the sequence of items rated by u_is' neighbor (i.e., n_k). We define u_is' neighbor $n_k \in ihs_{v_j}$ as a user who has rated on target item v_j at the kth order in the ihs_{v_j}. To find latent relation from historical sequence of users and items, we create 2 types of Local Context Unit. From ihs_{v_j}, we create Item-User

Local Context Unit $K_{(v_j,n_k)}$ which is a weighting matrix that each column shows an important level on how surrounding users affect to the u_is' neighbor n_k who has rated on target item v_j while from uhs_{n_k}, we create User-Item Local Context Unit $K_{(n_k,v_j)}$ which is a weighting matrix that each column shows an important level on how surrounding items affect to target item v_j rated by n_k. For the target item v_j, we create $K_{(v_j,n_k)}$ for every n_k in ihs_{v_j}. For example, we assume that target item is Titanic. As you can see in Table 2, Titanic has 5 neighbors n_k who have rated on it. So, we create $K_{(n_k,v_j)}$ by looking up in uhs_{n_k} to find Local Context Unit of target item v_j for each uhs_{n_k}. For example, if n_k are Sara and Amy, we will look up in the user historical sequence of Sara uhs_{Sara} and user historical sequence of Amy uhs_{Amy} to find Local Context Unit of Titanic $v_{Titanic}$ in uhs_{Sara} and uhs_{Amy}, respectively. After $K_{(v_j,n_k)}$ and $K_{(n_k,v_j)}$ are created, in order to find user n_k profile (i.e., p_{n_k}), we would like to project user characteristics with latent features surrounding that user. Therefore, we use an element-wise multiplication between user n_k embedding (i.e., e_{n_k}) and Item-User Local Context Unit $K_{(v_j,n_k)}$ by using Equation 1. In the same way, in order to find item v_j profile on sequence of neighbor n_k (i.e., $q_{v_{j_{n_k}}}$), we would like to project item characteristics with latent features surrounding that item. Therefore, we use an element-wise multiplication between item v_j embedding (i.e., e_{v_j}) and User-Item Local Context Unit $K_{(n_k,v_j)}$ by using Equation 2.

$$p_{n_k} = K_{(v_j,n_k)} \odot e_{n_k} \tag{1}$$

$$q_{v_{j_{n_k}}} = K_{(n_k,v_j)} \odot e_{v_j} \tag{2}$$

According to Neural Collaborative Filtering concept, rating score of each u_is' neighbor toward target item v_j (i.e., r_{n_k,v_j}) is computed from user profile p_{n_k} and item profile $q_{v_{j_{n_k}}}$ by using an element-wise multiplication as Equation 3.

$$r_{n_k,v_j} = p_{n_k} \odot q_{v_{j_{n_k}}} \tag{3}$$

Since rating prediction equation of collaborative filtering is the summation of rating score from all neighbors of target user, so we compute rating score of target

movie v_j for target user u_i (i.e., $R_{(u_i,v_j)}$) by applying max-pooling operation on all r_{n_k,v_j} for all the neighbors n_k of target user u_i toward target item v_j. It can be expressed as Equation 4.

$$R_{(u_i,v_j)} = max\left(\left[r_{n_0,v_j}\, r_{n_1,v_j}\, \ldots\, r_{n_{m-1},v_j}\, r_{n_m,v_j}\right]\right);\, m\ is\ number\ of\ n_k \quad (4)$$

Finally, we would like to learn a rating score vector to be a rating score scalar. So, we feed $R_{(u_i,v_j)}$ into the fully connected layer to predict the rating score output $\hat{y}_{(u_i,v_j)}$ of target item v_j for target user u_i as Equation 5.

$$\hat{y}_{(u_i,v_j)} = \sigma(W \cdot R_{(u_i,v_j)} + b) \quad (5)$$

where W and b are the weight matrix and bias, respectively, and σ is the sigmoid function of the output layer.

We show an example of predicting rating score of target item Titanic $v_{Titanic}$ to target user Hwain u_{Hwain}. Firstly, we create Hwain historical sequence uhs_{Hwain} and Titanic historical sequence $ihs_{Titanic}$. From Table 2, users who rate Titanic are Sara, Joy, Amy, Paula and Jenny. These users will be neighbors of Hwain. Next, for each neighbor such as Joy, we create Item-User Local Context Unit $K_{(v_{Titanic},n_{Joy})}$ by finding Local Context Unit of Joy in historical sequence of Titanic $ihs_{Titanic}$. In contrast, we create User-Item Local Context Unit $K_{(n_{Joy},v_{Titanic})}$ by finding Local Context Unit of Titanic in historical sequence of Joy uhs_{Joy}. After creating $K_{(v_{Titanic},n_{Joy})}$ and $K_{(n_{Joy},v_{Titanic})}$ of every neighbors of Hwain u_{Hwain}, we will create Joy profile p_{Joy} by using an element-wise multiplication between Item-User Local Context Unit $K_{(Titanic,Joy)}$ and Joy embedding vector e_{Joy} as Equation 1 and we will find Titanic profile on the sequence of Joy $q_{Titanic_{Joy}}$ by using an element-wise between User-Item Local Context Unit $K_{(Joy,Titanic)}$ and Titanic embedding $e_{Titanic}$ as Equation 2. Next, the model computes rating score of Joy toward Titanic by element-wise multiplying Joy profile p_{Joy} and Titanic profile on sequence of Joy $q_{Titanic_{Joy}}$. After obtaining rating scores of all neighbors of Hwain, the model applies max pooling operation to all of rating scores in order to summarize rating scores from all of the

neighbors. Finally, rating score is fed into fully connected layers to calculate rating score output of Titanic for Hwain.

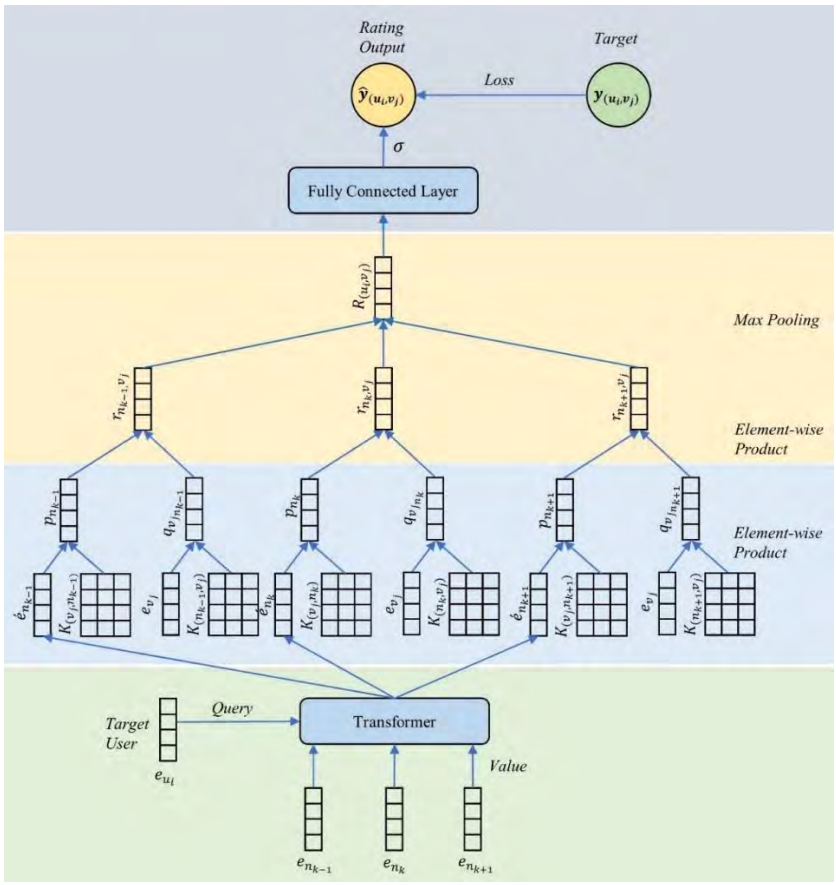


Figure 2 *Model architecture with Attention*

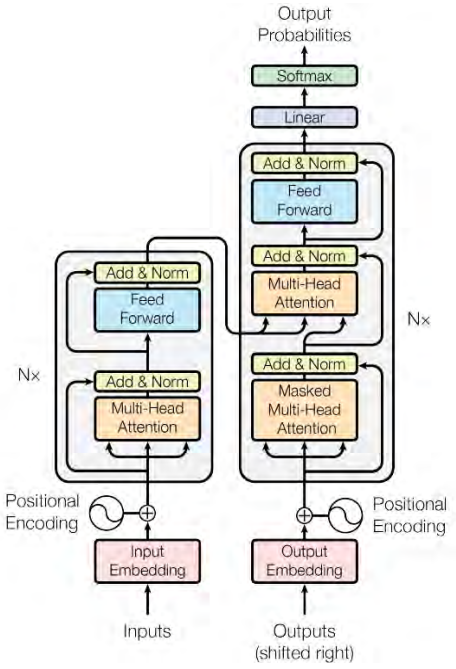


Figure 3 *Model Architecture of Transformer Attention*

Unfortunately, output rating from this method did not serve for personalized recommendation. It means that if the model predicts rating score of Titanic for different target users, these different target users will receive the same rating score because users' neighbors of these target users are the same. For example, if model predict rating score of Titanic for Susan and Alice, both of them will receive the same rating score. Because model of Susan and Alice have same neighbor embeddings (Sara, Joy, Amy, Paula, Jenny) and same Local Context Units, so these models predict the same result. Therefore, we also propose method to solve this problem by applying Attention. Attention is a work that is widely used in Machine Translation to weight word that most similar to query word. For recommendation works, attention is used to weight item that user should have interested most [4] or is used to model user behavior based on users' interest [2]. In our work, we would like to differentiate among target users by applying attention. Transformer proposed by [1] in attention is applied in our work by including sequence of user who have rated on target item ihs_{v_j} instead of word as shown in Figure 3. Therefore, the output from Transformer is a list of neighbors personalized by each target user where each target user is the query. In other word, when we put different query or different target user into the transformer, you will get different list of neighbors. Now, we get list of neighbors according to each target users' perspective. The model architecture with attention is shown in Figure 2. This model has the same working as the model in Figure 1 besides the attention in the first layer of model. Therefore, from above example, neighbors of Susan and Alice are now different. Neighbors of Susan are $Sara_{Susan}$, Joy_{Susan}, Amy_{Susan}, $Paula_{Susan}$ and $Jenny_{Susan}$ and neighbors of Alice are $Sara_{Alice}$, Joy_{Alice}, Amy_{Alice}, $Paula_{Alice}$ and $Jenny_{Alice}$. We input neighbors into encoding side and input query user i.e., target user into decoding side, then the model generates the new neighbors that personalize to each target user.

Objectives

1. To propose a new recommendation method that is able to captures relation in historical sequence of target user and target item by applying region embedding.

Moreover, this new method is able to find personalized target users' neighbors by applying attention on target item sequence.

2. To compare the efficiency of our method with previous method on NDCG@K and HitRate@K.

Scope

1. Use publicly available datasets from MovieLens dataset that is not more than 1,000,000 ratings, 4,000 movies and 6,000 users. The lowest rating score is 0.5 and the highest rating score is 5.0
2. Use value of rating in the range 1-5.
3. Predict only movie products.
4. Compare experimental results with the research of Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua in Neural Collaborative Filtering.

Project Activities

A. Study Plan

1. Study related works.
2. Analyze previous problems.
3. Design new methods for solving problems.
4. Analyze the proposed methods.
5. Develop the proposed methods.
6. Measure NDCG@K and HitRate@K of the proposed methods.
7. Experiment and compare the results.
8. Analyze the results.
9. Prepare project documentation.

B. Study Period

Timeline of Research Activities

Procedure	2019					2020			
	Aug	**Sep**	**Oct**	**Nov**	**Dec**	**Jan**	**Feb**	**Mar**	**Apr**
1. Study related works.									
2. Analyze previous problems.									
3. Design new methods for solving problems.									
4. Analyze the proposed methods.									
5. Develop the proposed methods.									
6. Measure NDCG@K and HitRate@K of the proposed methods.									
7. Experiment and compare the results.									
8. Analyze the results.									
9. Prepare project documentation.									

Benefits

A. In terms of knowledge and experience to the students

1. Gain knowledge and understanding of the process of implementing the system.
2. Practice analytical skill, working methodology, responsibility for work.
3. Gain knowledge about various models to create a recommendation system method.
4. Gain knowledge of programming and system development.

B. Knowledge and understanding that leads to solving problems of society or environment

1. Create an appropriate model for predicting movie ratings with more accuracy than previous methods.
2. Reduce errors from inaccurate recommendations from previous methods.
3. Recommend things that meet user expectation.

Equipment

A. Hardware

- Computer with Microsoft Windows 10 Intel(R) Core(TM) I5 1.80 GHz, Ram 8 GB,
- Computer with Microsoft Windows 10 Intel(R) Core(TM) I7 2.20 GHz, Ram 8 GB,

B. Software

- Python library for data processing and machine learning such as numpy, scipy, pandas
- Google Colab
- Visual Studio Code version 1.39.0

Budget

1. SSD 1 TB	1 piece	4590	Baht
2. SSD 500 GB	1 piece	2000	Baht
3. Ram 16 GB	1 piece	3000	Baht
4. Micro SD 64 GB	1 piece	410	Baht
	Total	10000	Baht

Note: Budget may change as appropriate.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762.*

[2] Chang Zhou, Jinze Bai, Junshuai Song, Xiaofei Liu, Zhengchao Zhao, Xiusi Chen, and Jun Gao. 2017. ATRank: An Attention-Based User Behavior Modeling Framework for Recommendation. *arXiv preprint arXiv:1711.06632.*

[3] Chao Qiao, Bo Huang, Guocheng Niu, Daren Li, Daxiang Dong, Wei He, Dianhai Yu, and Hua Wu. 2018. A new method of region embedding for text classification. *International Conference on Learning Representations* (2018).

[4] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. *arXiv:1904.06690v2*. https://doi.org/10.1145/3357384.3357895

[5] Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. Computer, 42(8), 42-49. http://doi.org/10.1109/MC.2009.263

[6] L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. *Proceedings of the third ACM conference on Recommender systems, 2009.*

[7] Padipat Sitkrongwongn, and Atsuhiro Takasu. 2019. Unsupervised context extraction via region embedding for context-aware recommendations. *Proceedings of the 23rd International Database Applications & Engineering Symposium (IDEAS '19).* ACM, Athens, Greece.

[8] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. http://dx.doi.org/10.1145/3038912.3052569

BIOGRAPHY

Miss Thitiya Sae-diae
16 September 1997 Bangkok, Thailand
Department of Mathematics and Computer Science
Faculty of Science, Chulalongkorn University
Email: Thitiya.SaeDiae@gmail.com

Miss Umaporn Padungkiatwattana
26 August 1997 Bangkok, Thailand
Department of Mathematics and Computer Science
Faculty of Science, Chulalongkorn University
Email: umaploy@gmail.com