University Kasdi Merbah OUARGLA ALGERIA

Faculty of New Technologies of Information and Communication

Department of Computer Science and Information Technology

Domain: Mathematics and Computer science

Track: Computer science

Specialty: Fundamental

THESIS of ACADEMIC MASTER

Presented by:

Boubellouta Islam

Titled:

Multi-criteria Collaborative recommendation system with Self- Attention : study of impact of self-Attention in Recommendation Phase

Publicly discussed on:

20/06/2022

Committee Members:

Mr. Cheriet Abdelhakim	President	UKM	Ouargla
Mrs. Ameur Khadidja	Supervisor	UKM	Ouargla
Mr. Belhadj Mourad	Examiner	UKM	Ouargla

Academic Year: 2021 / 2022

Acknowledgment

Above all, i would like to thank ALLAH the most gracious the most merciful for giving us health, faith and strength to complete my studies and the desire to begin and end this work.

I am overwhelmed in all humbleness and gratefulness to acknowledge my depth to all those who have helped me to put these ideas, well above the level of simplicity and into something concrete.

I would like to express my special thanks of gratitude to my supervisor who gave me the golden opportunity to do this wonderful work, which also helped me in doing a lot of study and i came to know about so many new things.

I am really thankful to Dr.Ameur Khadidja.

Any attempt at any level can‘t be satisfactorily completed without the support and guidance of my parents and friends.

I would like to thank my parents who helped me a lot in gathering different information, collecting data and guiding me from time to time in making this thesis, despite of their busy schedules, they gave me different ideas in making this work unique.

Dedication

I dedicate this modest work to:

My beloved mother,

You have been an outstanding inspiration to me.

Getting to this stage in my life has taken a lot of work, but it is nothing compared to how you worked and sacrificed for me.

You are the number one reason I am where I am today. Without your continued support I could never have accomplished so much.

I love you.

Abstract

Recommender systems have been very useful tools in different applications such as online marketing, e-commercial services, and social networking applications. They are information filtering technologies used to recommend products to users using specific techniques. The most common ones are collaborative filtering techniques. They are usually used on single rating datasets; however, they have been extended to work on multi-criteria datasets, which have been proved more accurate.

Deep learning has achieved impressive results in many domains such as natural language processing Natural Language Processing (NLP). Recently, deep learning for recommender systems started to receive great interest, and there are many proposed models based on deep learning.

Self-Attention is one of the deep learning methods that was first used in NLP domain, and then it was adapted in sequential recommender systems. However, as far as we know, there is not yet any study, which gathers multi-criteria recommendation and collaborative filtering with Self-Attention.

In this work, we propose a novel multi-criteria collaborative filtering model with Self-Attention. Our experimental evaluation show a side-by-side comparison between multi-criteria collaborative filtering model with and without the use of Self-Attention. Results of our experiments show the success of using Self-Attention in multi-criteria collaborative filtering recommender systems.

Key words: Multi-criteria, recommendations system, self-attention, collaborative filtering, Deep learning. Sparsity data.

ملخص

كانت أنظمة التوصية أدوات مفيدة للغاية في تطبيقات مختلفة مثل التسويق عبر الإنترنت وخدمات التجارة الإلكترونية وتطبيقات الشبكات الاجتماعية. إنها تكنولوجيات تصفية المعلومات المستخدمة للتوصية بالمنتجات للمستخدمين باستخدام تقنيات معينة. الأكثر شيوعًا بينها تقنيات التصفية التعاونية. يتم استخدامها عادةً في مجموعات بيانات التصنيف الفردي؛ ومع ذلك، فقد تم تمديدها للعمل على مجموعات بيانات متعددة المعايير، والتي ثبت أنها أكثر دقة.

حقق التعلم العميق نتائج مبهرة في العديد من المجالات مثل معالجة اللغة الطبيعية. في الآونة الأخيرة، بدأ التعلم العميق لأنظمة التوصية يحظى باهتمام كبير، وهناك العديد من النماذج المقترحة القائمة على التعلم العميق.

الاهتمام الذاتي هو إحدى طرق التعلم العميق التي تم استخدامها لأول مرة في مجال معالجة اللغة الطبيعية، ثم تم تكييفها في أنظمة التوصية. ومع ذلك، على حد علمنا، لا توجد حتى الآن أي دراسة تجمع توصيات متعددة المعايير والتصفية التعاونية مع الاهتمام الذاتي.

في هذا العمل، نقترح نموذج جديد لتصفية التعاونية متعدد المعايير مع الاهتمام الذاتي. يُظهر تقييمنا التجريبي مقارنة جنبًا إلى جنب بين نموذج التصفية التعاوني متعدد المعايير مع استخدام الاهتمام الذاتيْ أو بدونه. تظهر نتائج تجاربنا نجاح استخدام الاهتمام الذاتي في أنظمة التوصية بالترشيح التعاوني متعدد المعايير.

الكلمات المفتاحية: متعدد المعايير ، نظام التوصيات ، الاهتمام الذاتي ، التصفية التعاونية ، التعلم العميق. بيانات متفرقة.

Résumé

Les systèmes de recommandation ont été des outils très utiles dans différentes applications telles que le marketing en ligne, les services de commerce électronique et les applications de réseautage social. Ce sont des technologies de filtrage d'informations utilisées pour recommander des produits aux utilisateurs en utilisant certaines techniques. Les plus courantes sont les techniques de filtrage collaboratif. Ils sont généralement utilisés sur des ensembles de données de notation uniques; cependant, ils ont été étendus pour travailler sur des ensembles de données multicritères, qui se sont avérés plus précis.

L'apprentissage en profondeur a obtenu des résultats impressionnants dans de nombreux domaines tels que le traitement du langage naturel TLN. Récemment, l'apprentissage en profondeur pour les systèmes de recommandation a commencé à susciter un grand intérêt, et il existe de nombreux modèles proposés basés sur l'apprentissage en profondeur.

Self-Attention est l'une des méthodes d'apprentissage en profondeur qui a d'abord été utilisée dans le domaine de la TLN, puis elle a été adaptée dans les systèmes de recommandation séquentiels. Cependant, à notre connaissance, il n'existe pas encore d'étude rassemblant recommandation multicritère et filtrage collaboratif avec Self-Attention.

Dans ce travail, nous proposons un nouveau modèle de filtrage collaboratif multicritères avec Self-Attention. Notre évaluation expérimentale montre une comparaison côte à côte entre le modèle de filtrage collaboratif multicritères avec et sans l'utilisation de Self-Attention. Les résultats de nos expériences montrent le succès de l'utilisation de l'attention personnelle dans les systèmes de recommandation de filtrage collaboratif multicritères.

Mots clés: Multi-critères, système de recommandations, self-attention, filtrage collaboratif, Deep learning. Données de parcimonie.

Contents

List of Figures

List of Tables

List of Abbreviations

RS Recommender System

RSs Recommender Systems

MCRSs Multi-Criteria Recommender Systems

NLP Natural Language Processing

ACAE Attention Collaborative Auto Encoder

MCCFRS Multi-Criteria Recommender Systems Collaborative Filtering

CF Collaborative Filtering

RNN Recurrent Neural Network

MCDM Multi-Criteria Decision Making

List of Equations

General Introduction

1 Introduction

With the rise of online shopping services and other technologies that need user feedback analysis and relevant product suggestions, recommender systems Recommender System (RS) have become critical and unavoidable in our day-to-day life. Recommender systems Recommender Systems (RSs) are information-filtering systems that aim to create a relationship between items and users by learning what items are more suitable to which users [1][16][18]. To achieve this objective, many techniques are used. The most common and efficient one being the collaborative filtering technique Collaborative Filtering (CF). Collaborative filtering is the process of measuring the similarity between either users or items. Meaning that similar behavior shows similar taste [5][11][20].

Since we are interested in the recommendation phase of the recommender system there are three approaches to use. The first one is designing a total order for item recommendations. The second approach is Finding Pareto optimal item recommendations that we can learn about in [7]. Another approach is using a selected criterion as a filter while converting other criteria into constraints. These techniques were used on the traditional or single criterion recommender systems [1][7]. They are implemented on datasets with an overall rating, meaning every user has only one non-negative integer evaluation for each item they rate like the work of [11][18]. This was considered as a limitation to the state-of-the-art recommender models because it does not collect enough information about the user's opinion [11][16][18]. Therefore, adding multiple criteria into account has led to a more accurate result. That is when multi-criteria recommender systems were introduced with more deep learning methods that help extract the factors of each criterion to make up the model. More on these methods can be found in these documents [7][20].

Although these approaches proved to be most efficient, the approach they use is not the optimal in the literature. In addition, they could not perform very well when using sparse datasets because they tend to solve the problem in a linear approach. To solve this linear problem, different deep learning algorithms are being used to capture nonlinear latent factors. Many proposed deep learning methods succeeded in capturing the nonlinearity and managed to yield good performance. At the same time, they fail to put more weight on the special users or items that although they can be more credible are treated aimlessly. Resulting in poor accuracy.

One of deep learning methods that have been introduced lately called self-attention has managed to overcome the problem of ignoring special users/items by making the model perform in a nonlinear manner and still focusing on the local relations in the user-item matrix [3]. Self-attention performed well in the natural language processing NLP field. The idea behind it was to make the decoder for the NLP focus on the relevant input sequence in a flexible manner by attributing the most relevant parts with the highest weight [9][15]. It was adapted in the sequential recommender system to allow the model to capture long-term sequences by assigning weights on each time step [10]. In [2], a state-of-the-art solution was introduced called attention collaborative auto encoder Attention Collaborative Auto Encoder (ACAE) that uses self-attention. It had a good performance when it comes to single criterion recommender systems.

Our objective is to apply self-attention in a multi-criteria recommender system model using collaborative filtering techniques. We will study its effect on particularly the recommendation phase and testing with the Top-N items evaluation method, in order to achieve better results and minimize problems such as sparsity and cold start.

This document's structure is as follows. In the first chapter, we explain in details the recommendation phase and its challenges in recommender systems. As well as the reason behind choosing multi-criteria recommender systems instead of the traditional ones. Also the application of self-attention in the recommender system. In the second chapter, we explain our methodology in a step-by-step guide on how to implement the proposed model. The third chapter will contain the experiment study and the evaluation of the impact when using Self-Attention in recommendation phase with multi-criteria datasets, and a discussion about the obtained results. Finally, we conclude our work in the final chapter.

Chapter 01

State of The Art

1 Introduction

The impact brought by self-attention in different fields made it interesting to implement in recommendation systems [9][10]. Collaborative multi-criteria recommendation system has gained plenty of recognition in recent years. It has a major influence in the most important tools in our lives such as e-shopping, e-learning, e-library and many more. The goal of this chapter is to give an overview of the multi-criteria recommender systems. It begins by defining the recommendation problem as a multi-criteria decision making Multi-Criteria Decision Making (MCDM) problem. Then goes over several MCDM methodologies and strategies that can help with multi-criteria recommender implementation [1][19]. Then extend this by laying out the challenges that exist in the recommendation phase, which is a part of two phases in the recommender system. Later on, we are going to view a detailed explanation concerning self-attention. Finally, we discuss the previous works done on the subjects that are of interest to us.

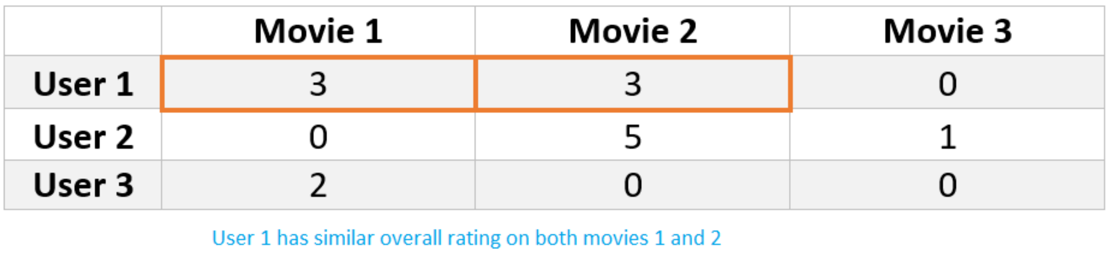
2 Multi-Criteria Recommender Systems

The idea behind recommender systems is to discover the items that have higher chances of a chosen user liking them. It does that by analyzing their preferences from either an explicit or implicit feedback on other items they have seen or rated. To be more specific, there is a set of Users containing users of a system. A set of items that contains items to the same system [11].

For each $item\, i \in Items$ and $user\, u \in Users$ there is a positive integer R_{ui} representing the rating of $useru$ on $itemi$. For the case where the user has not rated the item, the R_{ui} stays unknown and represented with zero. The goal of recommender systems is to predict the unknown ratings and select Top-N items from the most suggested items to each user [1].

In many cases, users give only a single rating for the items they have obtained. Many other systems have the option to give multiple ratings according to many criteria to judge the item based upon [7]. If we take for example a movie streaming application where every movie has to be judged based on storytelling and acting. We call these two the criteria of a movie. Having multiple criteria for each movie helps extract more information about the users taste [16].

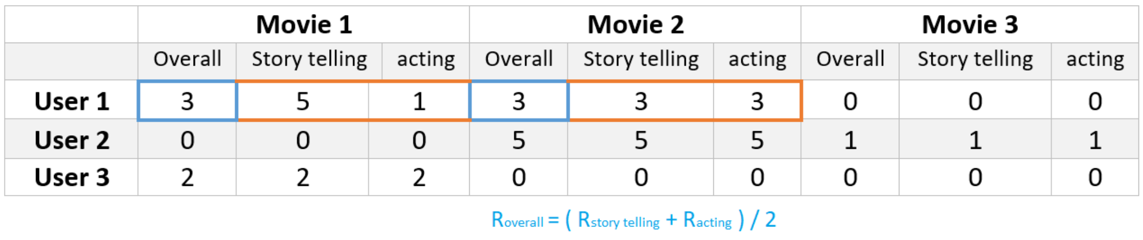
To understand this more let us take the example in figure 1 below where two items item1 and item2 that user1 has given each an overall rating of 3. We would then assume that these two items are both of similar taste to the user. However, in the next figure 2, that considers multiple criteria such as storytelling and acting, the rating for item1 and item2 are completely different despite having a similar overall rating.



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

User 1 has similar overall rating on both movies 1 and 2

Figure 1: Example of single criterion user-item matrix



	Movie 1			Movie 2			Movie 3		
	Overall	Story telling	acting	Overall	Story telling	acting	Overall	Story telling	acting
User 1	3	5	1	3	3	3	0	0	0
User 2	0	0	0	5	5	5	1	1	1
User 3	2	2	2	0	0	0	0	0	0

$R_{overall} = (R_{story\ telling} + R_{acting}) / 2$

Figure 2: Example of multi-criteria user-item matrix

3 Recommender system techniques

There are many techniques used in recommender systems and we are interested in the collaborative filtering techniques. These techniques apply in two major phases that the recommender system goes through (prediction and recommendation phases) [11]. Although we are interested in the recommendation phase, every recommender system model must have the prediction phase. To be able to choose the right technique in each phase, we need to define them first. Give a brief explanation about the prediction phase techniques. Then go into more details for the recommendation phase.

3.1 Prediction phase

This phase aims to predict the rating or score of unseen/unknown items for a specific user. Its techniques are grouped into two approaches: heuristic (memory-based) and Model-based approaches. Memory-based approach uses similarity computation methods to find a set of neighbors for each user to predict their ratings. Model-based approach creates a predictive model that learns from the given data to make its prediction [1]. Figure 3 shows the different approaches in collaborative filtering in the prediction phase.

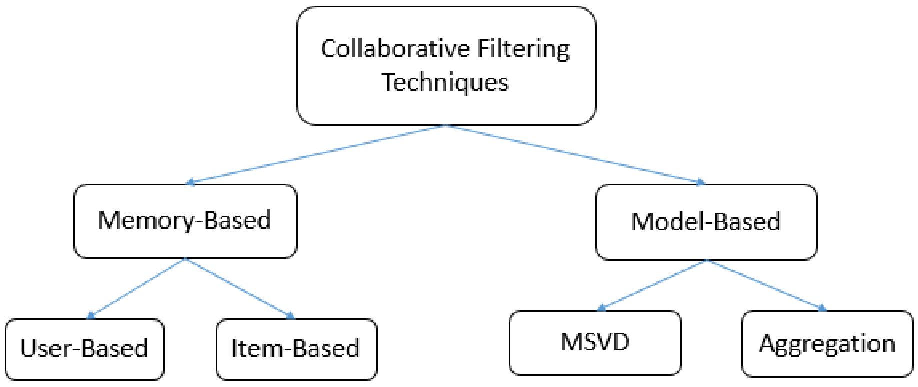
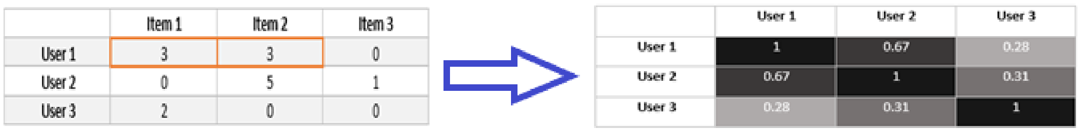


Figure 3: Example of result of using cosine similarity on user-item matrix

We choose cosine-based similarity method, which is used in heuristic based on both user-based and item-based approaches [6]. Since we are interested in evaluating list recommendation, this approach is better to use [4][8]. Cosine-based similarity is a function that measures the similarity between users/items of a system based on similar values of other users/items [11]. Figure 4 show an example on how to portray the results of cosine similarity. The matrix on the left is the user-item matrix showing the rating of each user on every item. A value of zero means the user did not rate that item. In the right matrix, we have the results after computing similarity between users, which is NxN matrix where N is the number of users. The values are between zero and one, where the bigger the value the more similar users are.



	Item 1	Item 2	Item 3
User 1	3	3	0
User 2	0	5	1
User 3	2	0	0

	User 1	User 2	User 3
User 1	1	0.67	0.28
User 2	0.67	1	0.31
User 3	0.28	0.31	1

Figure 4: Example of result of using cosine similarity on user-item matrix

3.2 Recommendation phase

This phase is an extension of the prediction phase where there is various techniques applied to support the user's decision by filtering the most suitable items [19][20]. It recommends/proposes new items to the user (i.e. a set of Top-N items with the highest-predicted ratings) that the user will most likely be interested in. There are many techniques used in this phase [1]:

-Designing a total order for item recommendations: this technique focuses on taking a linear combination of multiple criteria and reducing the problem to the single-criterion optimization problem. One of the methods used here is the Lakiotaki et al's UTilities

Additive method (UTA) which first calculates the marginal utility for each criteria-rating value on an item i then summing the results to have the overall rating of that item. Manouselis and Costopoulou also proposed a method similar to the last one but in addition to that, they weight the predicted ratings that the user would give on each criterion.

- Finding Pareto optimal item recommendations: The goal of this approach is to find the best performing items among a set of candidates on several criteria. One of the methods used to achieve this is Data envelopment analysis (DEA). This method uses linear programming to reduce the number of items from a large set by choosing the ones with the highest rating over multiple criteria. Lee and Teng also proposes a method called skyline queries which uses Pareto optimal points to eliminate the candidate items that are dominated (has lower rating) by other candidates.

- Using multi-criteria ratings as recommendation filters: The goal here is optimizing only the most important criterion (user's choice) and converting other criteria to constraints.

We choose the Pareto optimal solution. The goal of this approach is to find the best performing items among a set of candidates on several criteria. Taking figure 5 for example, to determine the Top-N movies that dominates all other movies on multi-criteria movie recommending system. The points in orange are the set of movies that perform better on both criteria [7].

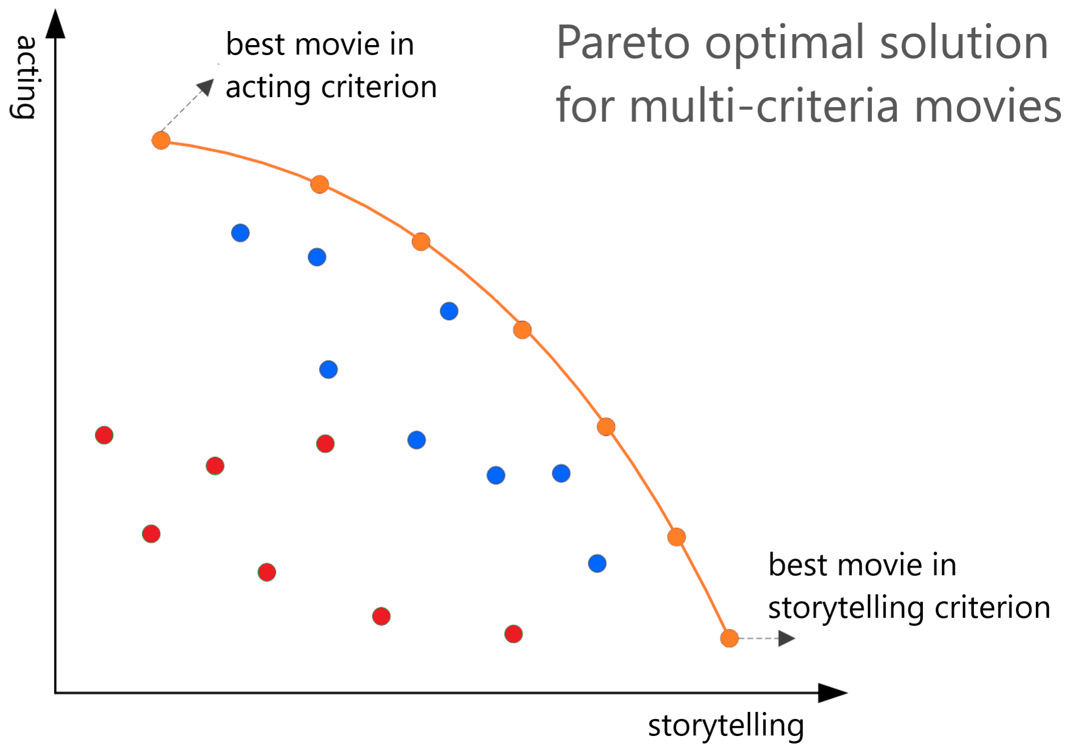


Figure 5: Example of result of using Pareto optimal on multi-criteria movies dataset

Constructing Pareto points comes right after computing the similarity of users in prediction phase. It is the process of selecting Top-N items for each user based on the user similarity matrix. If we take the previous figure 5 as example. The first row represents the similarity array of user 1 and all other users, which have values as follows: [1, 0.67, 0.28]. Meaning it is completely similar to itself, and more similar to second user then the third user. The goal here is ranking all items that other users have rated while the selected user did not rate (i.e. has the value 0) [1][7].

4 Challenges in recommendation phase

Multi-criteria recommender systems can model user's rating for an item by including both the overall rating and ratings of the item criteria or model it including only ratings of the criteria [19].

If it is the first choice (i.e. both overall rating and each criteria rating), the recommendation process is very straightforward. Meaning after the prediction phase the recommender system uses the overall rating of items to select the most relevant items for a specific user. Same as what happens in single-criterion recommender systems.

However, in the other case (i.e. without an overall rating), the recommendation process becomes more difficult to compare the items and select the Top-N items for each user. For example, let us say we have a two-criterion movie recommender system as shown in figure 1, where users judge movies based on their storytelling and acting. Also, suppose that only one of the movies can be chosen among these two movie 1 and movie 2, where despite having different criteria ratings, they have similar overall rating. In this case, a normal ranking approach is not possible. This is why the techniques we talked about in the previous section were introduced [1]. However, other challenges such as sparsity and cold start problems still affect the recommender systems [12], and in order to improve them even better the Self-Attention was recently introduced to this filed and had achieved promising results [14].

5 Self-Attention

Self-attention is an attention mechanism that connects distinct points of a same sequence in order to calculate a representation of it. Reading comprehension, abstractive summarization, textual entailment, and learning task-independent sentence representations have all been successfully employed with self-attention [3].

Self-attention was before known as attention mechanism when in computer science fields such as natural language processing NLP, and was used with the recurrent neural network Recurrent Neural Network (RNN) models. To our knowledge, it was then used as transformer model and became self-attention [3][9].

A self-attention module takes in n inputs, and returns n outputs. In non-technical language, the self-attention mechanism allows the inputs to interact with each other ("self") and find out whom they should pay more attention to ("attention"). The outputs are aggregates of these interactions and attention scores [3].

In the example below we can see the steps that self-attention go through in order to calculate attention scores and modify our input matrix to make it self-aware. It takes T number of inputs and goes through each array $x^{(i)}$ of the input to calculate dot product of that array with the input matrix resulting in another array. Using the formula $dot(x^{(i)}) = [x^{(i)^T} * x^{(j)}]_{j \in [0,T]}$. We then normalize the values of resulting array using softmax function $W_{ij} = softmax(dot(x^{(i)}))$. Finally, the output is the sum of all the products of normalization values with input sequence $output(i) = Sum(W_{ij} * x^{(j)})_{j \in [0,T]}$.

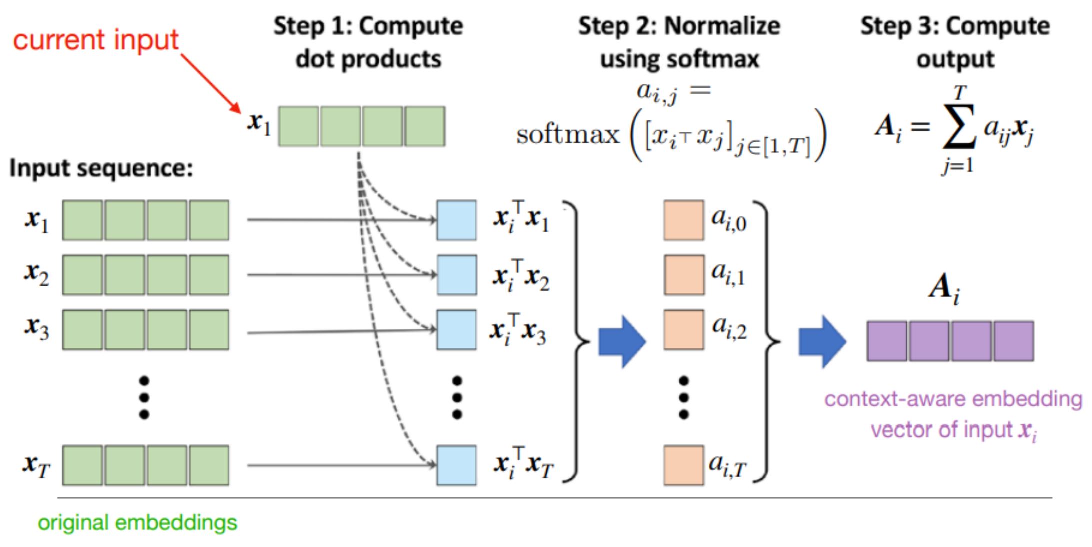
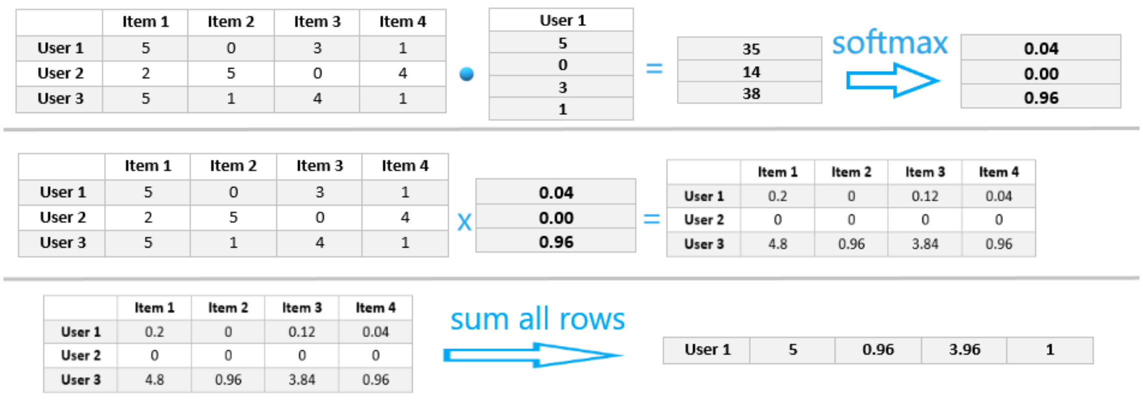

current input
Step 1: Compute dot products
Step 2: Normalize using softmax
Step 3: Compute output
$a_{i,j} = \text{softmax}\left([x_{i^\top} x_j]_{j \in [1,T]}\right)$
$A_i = \sum_{j=1}^{T} a_{ij} x_j$
x_1
Input sequence:
x_1
x_2
x_3
x_T
$x_i^\top x_1$
$x_i^\top x_2$
$x_i^\top x_3$
$x_i^\top x_T$
$a_{i,0}$
$a_{i,1}$
$a_{i,2}$
$a_{i,T}$
A_i
context-aware embedding vector of input x_i
original embeddings

Figure 6: Self-Attention Steps [21]

In order to understand the process of this method we take the figure 7 as an example. We can think of dot-product attention as three steps process for each row of the matrix.



	Item 1	Item 2	Item 3	Item 4
User 1	5	0	3	1
User 2	2	5	0	4
User 3	5	1	4	1

User 1
5
0
3
1

35
14
38

0.04
0.00
0.96

	Item 1	Item 2	Item 3	Item 4
User 1	5	0	3	1
User 2	2	5	0	4
User 3	5	1	4	1

0.04
0.00
0.96

	Item 1	Item 2	Item 3	Item 4
User 1	0.2	0	0.12	0.04
User 2	0	0	0	0
User 3	4.8	0.96	3.84	0.96

User 1	5	0.96	3.96	1

Figure 7: Example of applying Self-Attention on user-item matrix

In the example shown in figure 7, we take the first row, which is user 1 ratings. After transposing the user 2 ratings array, we calculate the dot product between the array and the original matrix. The result is Nx1 array, where N is number of users. After that, we run the array through softmax function to normalize the values. The second step is a normal multiplication between the resulting array values and the original matrix. That results in another matrix with same dimension as the original. In the last step, we sum the values of all rows of the matrix to have one array with same columns and one row. That array represents the ratings of user 1 after adding attention scores and becoming context aware. Notice that the rating of user 1 on the second item is close to one. That is because user 1 is more similar to user 3; making it have a close rating on the item, that user 1 did not rate yet.

6 Previous Work

Self-Attention has gained a lot of recognition in recent years. It is used in different fields of machine learning including recommender systems [9]. Collaborative filtering techniques are popular in recommender systems. To help improve these techniques by implementing Self-Attention we need to have a look on the previous works on both CF techniques in Multi-Criteria Recommender Systems (MCRSs) and Self-Attention.

The paper published by the google team ”Attention is all you need” [1] explains in details all there is to using Self-Attention and how it was adapted from the original attention mechanism. The paper explains all forms of attention from the basic scaled dot product attention to transformers.

An example of implementing this method is the work of Shuo Chen and Min Wu [2]. They create a Self-Attention recommender model called attention collaborative auto encoder ACAE. This model implements Self-Attention on single rating recommender system and evaluates the prediction phase.

However, our goal is to evaluate the recommendation phase and the paper [3] has done an excellent job of explaining different evaluation metrics of recommender systems including the metrics that we use in the recommendation phase precision and recall.

Since we want to implement CF techniques in MCRSs, in the recommender system handbook [1] we can find all the different deep learning methods to use in recommender systems. It explains all the techniques used for multi criteria rating system such as finding optimal Pareto point, which is of interest to us.

7 Conclusion

Throughout this chapter, we presented and explained the important concepts related to our research. Starting with multi-criteria recommender systems and how they emerged to achieve better result than the traditional single rating RSs. After that, we have seen the two phases in RSs prediction and recommendation; and the techniques used in both of them to reach the overall goal of RSs. Then, we take interest in the recommendation phase and lay down the challenges that face it. Another concept we went through is Self-Attention as we see when it first was used and how it reached recommender systems by explaining its process step-by-step. Finally, we talk about previous works that have been conducted which concerns both MCRSs and Self-Attention.

Chapter 02

Our Methodology

1 Introduction

This chapter explains our methodology, which we have relied on. In the first section, we will see how we implement the proposed model. Meaning the collaborative filtering techniques chosen and the thought process behind choosing them. In the second section, we are going to explain the process of integrating self-attention to the model. In the third section, here is the comparative study between the proposed model with and without self-attention. In the last section, we will conclude our methodology.

2 Proposed Model

In this section, we discuss the approach we choose and techniques used to build our model. Our model derive from multi-criteria collaborative filtering recommender systems Multi-Criteria Recommender Systems Collaborative Filtering (MCCFRS), which is the modern type of recommender systems. Before integrating Self-Attention in the MCCFRS we first need to use it on single criterion recommender system (i.e. traditional RS). We will go through both the user-based and item-based approaches in this single criterion RS. After that, we will integrate the Self-Attention on the MCCFRS also on both user-based and item-based approaches.

2.1 Self-Attention in single-criterion RS

In this section, we explain the different strategies that we are going to use in building our model on single criterion recommender system. We are using both approaches in memory based collaborative filtering (item based and user based). We will see how to construct these models and include Self-Attention later on.

2.1.1 User-based collaborative filtering approach

This approach apply cosine similarity on the user item matrix creating another symmetric matrix that contains the similarity between all users. Using the newly made matrix the unknown ratings of each user are predicted based upon the known ratings of the most similar users.

$$\mathrm{sim(u,\ u')} = \frac{\sum_{i\in \mathrm{I(u,\ u')}} \mathrm{R(u,\ i)R(u',\ i)}}{\sqrt{\sum_{i\in \mathrm{I(u,\ u')}} \mathrm{R(u,\ i)}^2}\sqrt{\sum_{i\in \mathrm{I(u,\ u')}} \mathrm{R(u',\ i)}^2}} \tag{1}$$

Equation 1 show the formula for calculating the similarity between two users u and u' where R(u, i) is the rating value of user u on item i, and I(u, u') is the similar rated items of both user u and user u' [11][17].

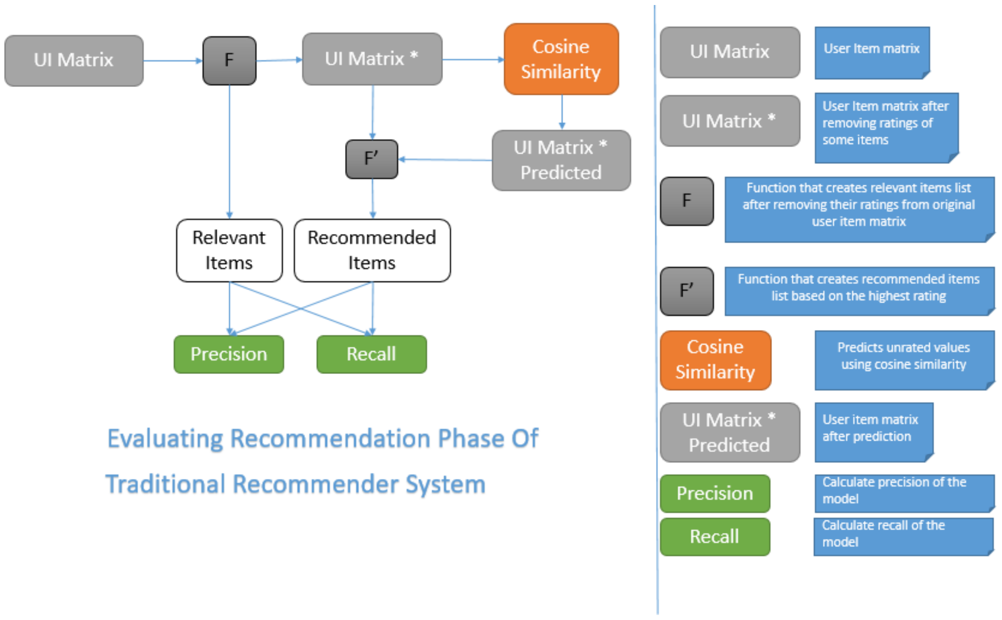
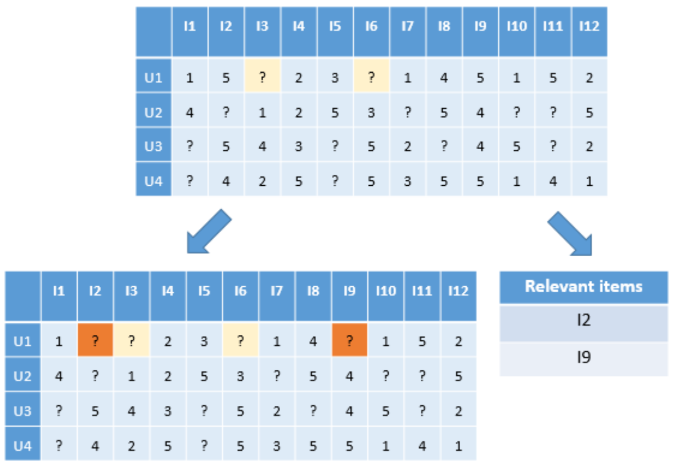

UI Matrix
F
UI Matrix *
Cosine Similarity
UI Matrix * Predicted
F′
Relevant Items
Recommended Items
Precision
Recall
Evaluating Recommendation Phase Of Traditional Recommender System
UI Matrix
User Item matrix
UI Matrix *
User Item matrix after removing ratings of some items
F
Function that creates relevant items list after removing their ratings from original user item matrix
F′
Function that creates recommended items list based on the highest rating
Cosine Similarity
Predicts unrated values using cosine similarity
UI Matrix * Predicted
User item matrix after prediction
Precision
Calculate precision of the model
Recall
Calculate recall of the model

Figure 8: Process of evaluating single rating recommender system using collaborative filtering user based approach

Figure 8 explains the steps done to evaluate this approach using the decision support metrics, precision and recall.

The user item matrix is processed by a function F, which after selecting a user, removes chosen amount of ratings of that user on items, which he gave good rating on. These items are the relevant items and are saved in a list. Figure 9 shows an example of this process.



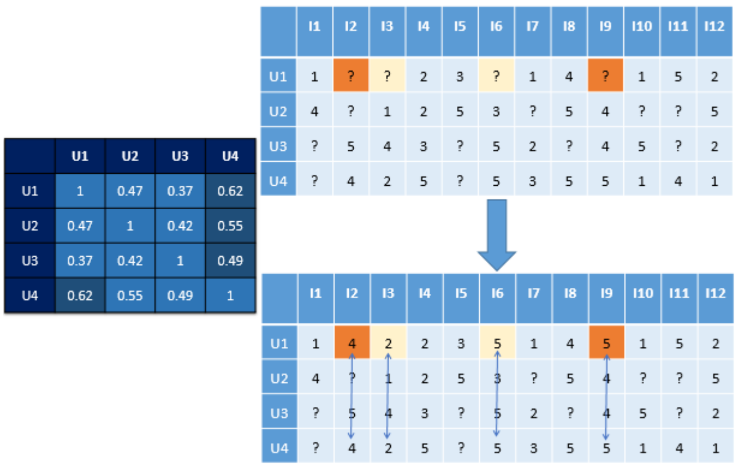
	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12
U1	1	5	?	2	3	?	1	4	5	1	5	2
U2	4	?	1	2	5	3	?	5	4	?	?	5
U3	?	5	4	3	?	5	2	?	4	5	?	2
U4	?	4	2	5	?	5	3	5	5	1	4	1

	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12
U1	1	?	?	2	3	?	1	4	?	1	5	2
U2	4	?	1	2	5	3	?	5	4	?	?	5
U3	?	5	4	3	?	5	2	?	4	5	?	2
U4	?	4	2	5	?	5	3	5	5	1	4	1

Relevant items
I2
I9

Figure 9: Example of creating relevant items from user item matrix

After that, the user based similarity prediction is made using cosine similarity, which we already explained. This process is explained in figure 10.



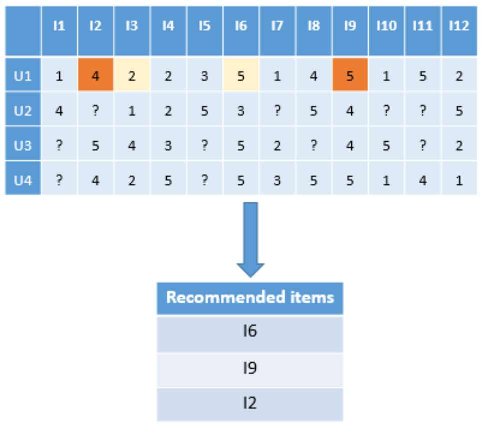
	U1	U2	U3	U4
U1	1	0.47	0.37	0.62
U2	0.47	1	0.42	0.55
U3	0.37	0.42	1	0.49
U4	0.62	0.55	0.49	1

	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12
U1	1	?	?	2	3	?	1	4	?	1	5	2
U2	4	?	1	2	5	3	?	5	4	?	?	5
U3	?	5	4	3	?	5	2	?	4	5	?	2
U4	?	4	2	5	?	5	3	5	5	1	4	1

	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12
U1	1	4	2	2	3	5	1	4	5	1	5	2
U2	4	?	1	2	5	3	?	5	4	?	?	5
U3	?	5	4	3	?	5	2	?	4	5	?	2
U4	?	4	2	5	?	5	3	5	5	1	4	1

Figure 10: Example of using cosine similarity to predict unknown ratings for one user

Using the predicted user item matrix, we apply another function that creates list of top N items from the previously unrated items. This process is explained in figure 11. Finally, using relevant items and recommended items lists precision and recall are calculated to give a score on the recommendation.



	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12
U1	1	4	2	2	3	5	1	4	5	1	5	2
U2	4	?	1	2	5	3	?	5	4	?	?	5
U3	?	5	4	3	?	5	2	?	4	5	?	2
U4	?	4	2	5	?	5	3	5	5	1	4	1

Recommended items
I6
I9
I2

Figure 11: Example of creating recommended items list from the predicted user item matrix

After we have built our model and the evaluation process, integrating Self-Attention in the model is displayed in the figure 12. It is applied on the predicted user item matrix which will make it self aware.

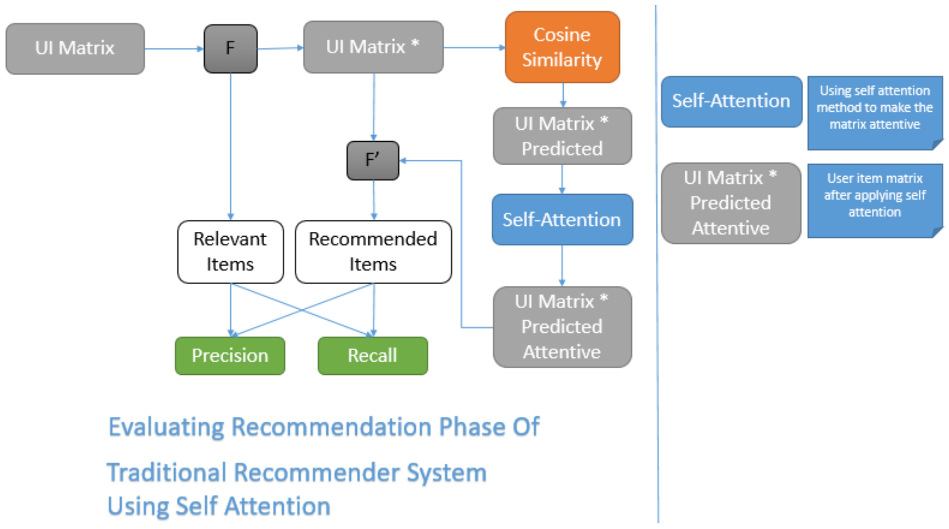
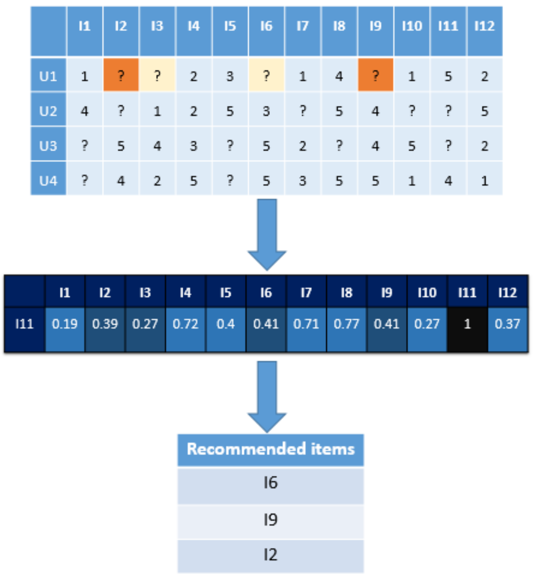

UI Matrix
F
UI Matrix *
Cosine Similarity
UI Matrix * Predicted
F'
Self-Attention
Relevant Items
Recommended Items
UI Matrix * Predicted Attentive
Precision
Recall
Evaluating Recommendation Phase Of Traditional Recommender System Using Self Attention
Self-Attention
Using self attention method to make the matrix attentive
UI Matrix * Predicted Attentive
User item matrix after applying self attention

Figure 12: Process of evaluating single rating recommender system using collaborative filtering user based approach with Self-Attention

2.1.2 Item based collaborative filtering approach

In this approach, we will apply cosine similarity on the user item matrix to extract the relation between items instead of users. We know that in order to evaluate the recommendation of any model we need list of relevant items and a list of recommended items. This approach and the previous one (user-based approach) obtain the relevant items list in a similar manner. However, they differ on the method of obtaining the recommended items list. Therefore, in order to understand construction step of this model, we will explain how the process of obtaining the later list happens. Figure 13 shows an example of how to obtain the recommended items list using item based collaborative filtering. After using cosine similarity on the user item matrix (we obtain this by passing the user item matrix transposed to cosine similarity function), we choose the item that the selected user likes the most (i.e. highest rated), then we rank the unrated items by the selected user based on which is more similar to the chosen item.



	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12
U1	1	?	?	2	3	?	1	4	?	1	5	2
U2	4	?	1	2	5	3	?	5	4	?	?	5
U3	?	5	4	3	?	5	2	?	4	5	?	2
U4	?	4	2	5	?	5	3	5	5	1	4	1

	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12
I11	0.19	0.39	0.27	0.72	0.4	0.41	0.71	0.77	0.41	0.27	1	0.37

Recommended items
I6
I9
I2

Figure 13: Example of creating recommended items list from the predicted user item matrix using item-based approach

2.2 Self-Attention in Multi-Criteria RS

In this section, we explain the different strategies that we are going to use in building our model on multi-criteria recommender system. We are using both approaches in memory based collaborative filtering (item based and user based). We will see how to construct these models and include Self-Attention later on.

2.2.1 User-based collaborative filtering approach

To use our methodology on multi-criteria user-based as shown in figure 14 we need to use the pareto optimal points method which we explained in the previous chapter. Since we have multiple criteria, we would have multiple user item matrices. Therefore, we would apply our methods on multiple matrices. We need pareto optimal method to obtain the recommended items list based on multiple user item matrices.

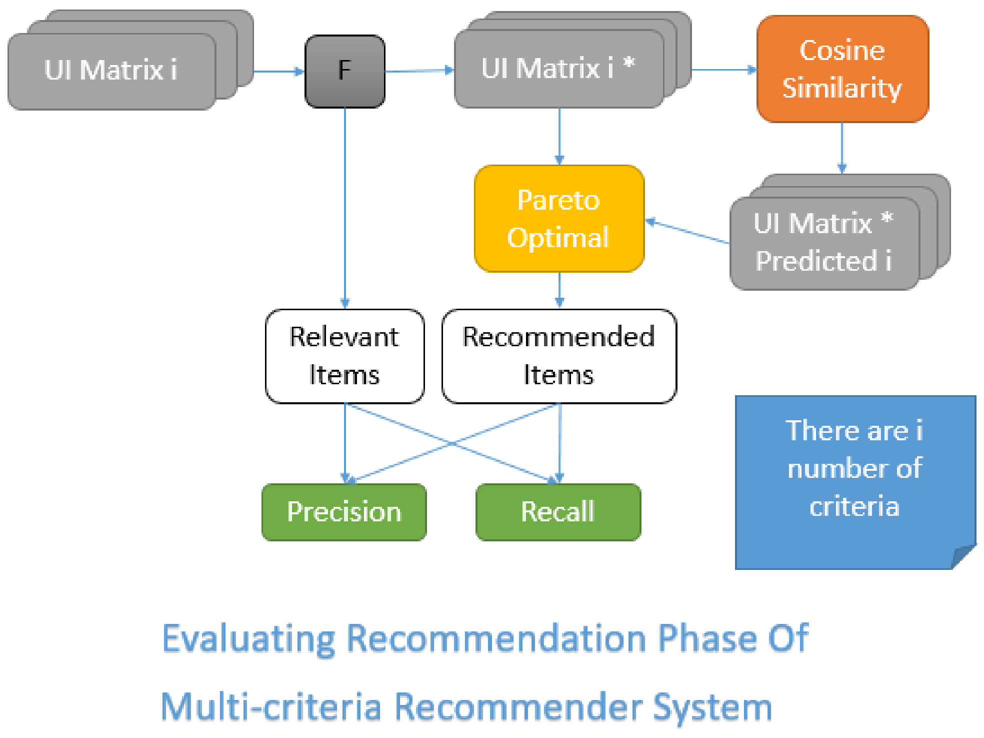


Figure 14: Process of evaluating multi-criteria recommender system using collaborative filtering user based approach

Figure 14 explains the steps done to evaluate this approach using the decision support metrics, precision and recall.

The user item matrix is processed by a function F, which after selecting a user, removes chosen amount of ratings of that user on items, which he gave good rating on. Since we have multiple criteria this process is done separately on each criteria having chosen the best performing items on all criteria. These items are the relevant items and are saved in a list. Figure 15 shows an example of this process

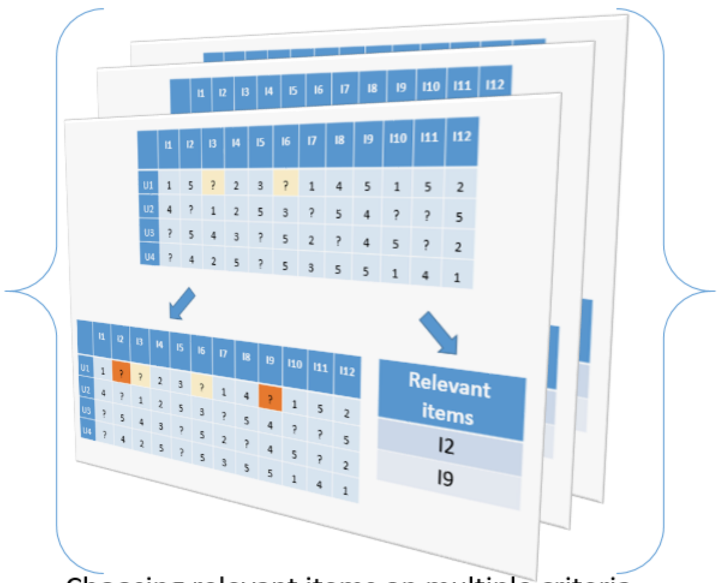


Figure 15: Example of creating relevant items from user item matrix for multi-criteria user-based approach

After that, the user based similarity prediction is done on each criteria using cosine similarity, which we already explained. This process is explained in figure 16.

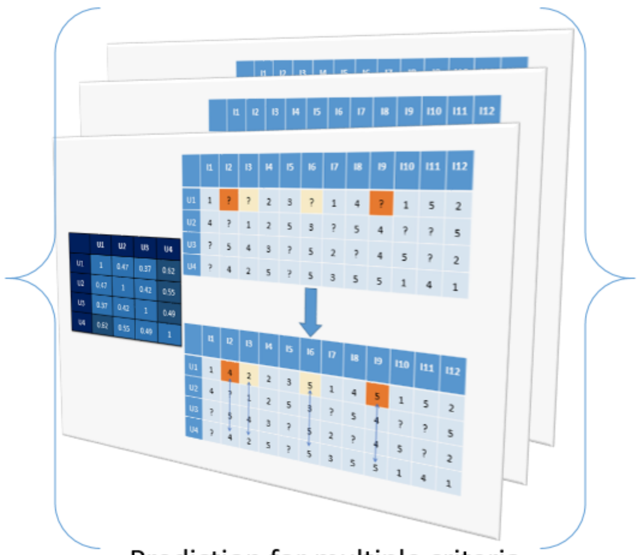
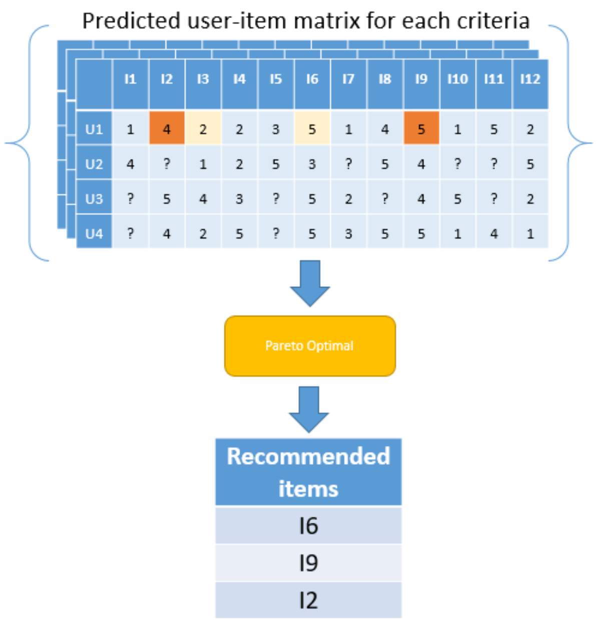


Figure 16: Example of using cosine similarity to predict unknown ratings for one user

Using the predicted user item matrix, we apply Pareto optimal to create list of top N items from the previously unrated items. This process is explained in figure 17.



	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12
U1	1	4	2	2	3	5	1	4	5	1	5	2
U2	4	?	1	2	5	3	?	5	4	?	?	5
U3	?	5	4	3	?	5	2	?	4	5	?	2
U4	?	4	2	5	?	5	3	5	5	1	4	1

Figure 17: Example of creating recommended items from user item matrix for multi-criteria user-based approach

Adding Self-Attention in multi-criteria recommender system is no different from the single rating one. Except we would apply the method on each of the user item matrices as shown in the figure 18.

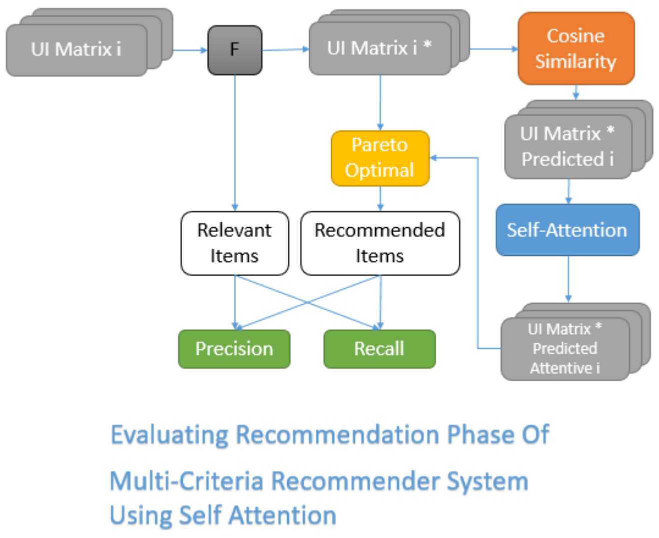


Figure 18: Process of evaluating multi-criteria recommender system using collaborative filtering user based approach with Self-Attention

2.2.2 item-based collaborative filtering approach

For multi-criteria item-based approach we will use the same pareto optimal points method as the user-based approach, but with different input. Figure 19 shows an example of using multiple item-to-item similarity lists as an input to Pareto optimal method to determine the top performing items, where each list is calculated from different criteria.

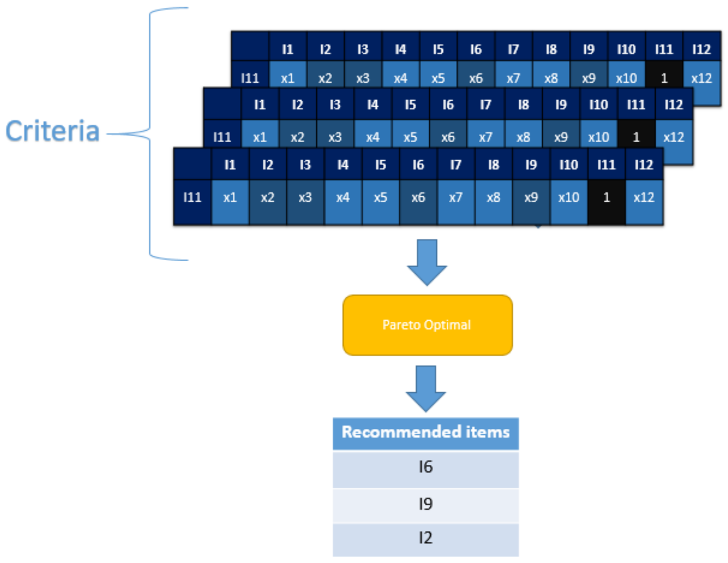

Criteria
I1 I2 I3 I4 I5 I6 I7 I8 I9 I10 I11 I12
I11 x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 1 x12
Pareto Optimal
Recommended items
I6
I9
I2

Figure 19: Example of creating recommended items list from the predicted user item matrix using item-based approach with multi-criteria

3 Conclusion

Throughout this chapter, we presented our proposed model by first explaining the techniques we use in both the prediction and recommendation phases. After that, we give an example on the process of applying Self-Attention on a user-item matrix to show briefly how it works. Next, we went through the process of constructing our models using the previously explained techniques in both user-based and item-based approaches, and the difference in the construction of the model in single rating RSs and multi-criteria RSs.

Chapter 03

Experiments & Results Discussions

1 Introduction

In this chapter, we explain the methodology of out experiment and discuss the results obtained from them. The first section describes the nature of the datasets used and their content. In the second section, we explain the metrics used to evaluate the models. The demonstration of experiments and obtained results is in the third section. We conclude this chapter and explain how we can further improve the model in the last section.

2 Experimental Design

In this section, we start by describing the architecture of the datasets used. Then, we go through the metrics that we use to evaluate our model. Finally, we define the tool we use in programming.

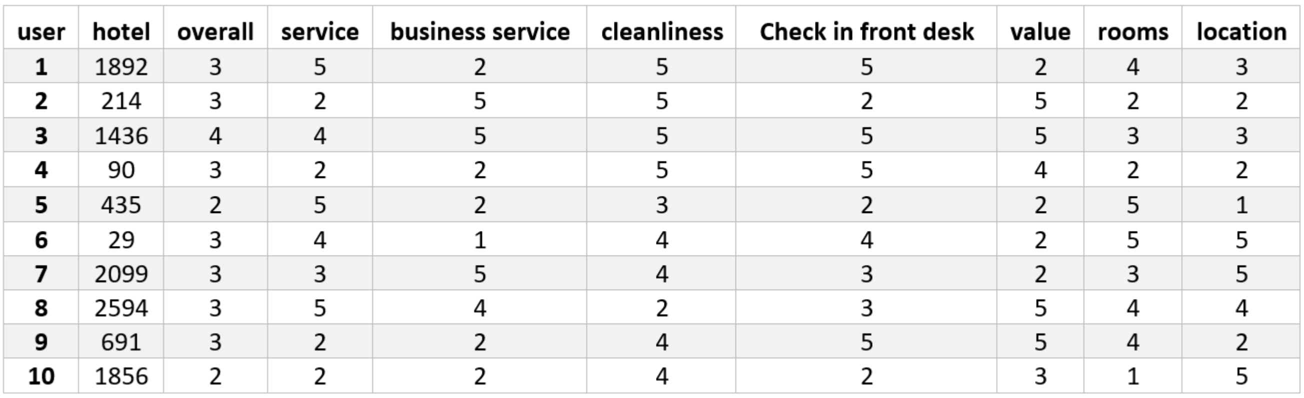
2.1 Datasets

For our experiment, we are using two different multi-criteria datasets. The first one comes from hotel rating system from the popular website TripAdvisor. The other dataset is movies ratings, and it is from MovieLens website.

2.1.1 TripAdvisor dataset

TripAdvisor is an e-commerce website that specialize in creating reservations to hotels for their clients as well as the ability to review them on many aspects such as service, business service, cleanliness, check in front desk, value, rooms and location [13]. These are the seven criteria provided in this dataset. The ratings range from one to five, where the bigger the number the more user likes the hotel. We represent the missing ratings with zero when constructing the user-item matrix. The table 1 shows a sample of the dataset, where we can see the ratings of ten users to different hotels based on our seven criteria and overall rating. This dataset is available at [1].

[1] https://www.tripadvisor.com/



user	hotel	overall	service	business service	cleanliness	Check in front desk	value	rooms	location
1	1892	3	5	2	5	5	2	4	3
2	214	3	2	5	5	2	5	2	2
3	1436	4	4	5	5	5	5	3	3
4	90	3	2	2	5	5	4	2	2
5	435	2	5	2	3	2	2	5	1
6	29	3	4	1	4	4	2	5	5
7	2099	3	3	5	4	3	2	3	5
8	2594	3	5	4	2	3	5	4	4
9	691	3	2	2	4	5	5	4	2
10	1856	2	2	2	4	2	3	1	5

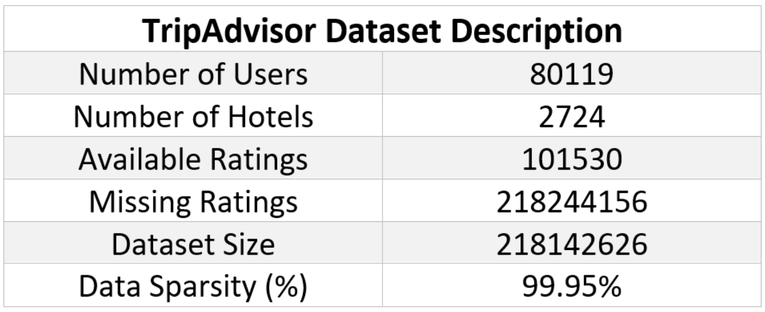
Table 1: Data sample of TripAdvisor dataset

We use basic mathematical formulas to extract information from the dataset. The equation 2 shows the rule we use to calculate the dataset size. The equation 3 is the formula that determines the sparsity of data.

$$\text{Dataset size} = \text{Number of Users} \times \text{Number of Items} \tag{2}$$

$$\text{Data Sparsity} = 1 - \frac{\text{Number of Ratings}}{\text{Dataset size}} \tag{3}$$

The table 2 is a description of the information obtained from applying previous formulas as well as MS excel formulas on our dataset.



TripAdvisor Dataset Description	
Number of Users	80119
Number of Hotels	2724
Available Ratings	101530
Missing Ratings	218244156
Dataset Size	218142626
Data Sparsity (%)	99.95%

Table 2: Description of TripAdvisor Dataset

2.1.2 MovieLens dataset

MovieLens provides a huge database of movie ratings by building on collaborative filtering techniques. It has multi-criteria recommendations on five different criteria with ratings ranging from one to 13. The table 3 shows a sample of the dataset, where we can see the ratings of ten users to different movies based on our four criteria and overall rating. This dataset is available at [2].

User id	Criteria_1	Criteria_2	Criteria_3	Criteria_4	Overall	Movie id
1	6	6	8	12	8	2
2	3	8	7	12	2	655
3	12	10	12	10	10	42
4	6	9	9	12	9	436
5	9	10	9	9	9	626
6	10	12	11	11	11	144
7	11	9	12	13	11	17
8	1	10	1	10	1	83
9	10	11	12	13	12	2
10	5	11	8	9	5	262

Table 3: Data sample of MovieLens dataset

We use the previous formulas to extract this dataset information. We can see the result in the table 4. We can see that the dataset is also very spare with the value of 98 percent.

MovieLens Dataset Description	
Number of Users	6078
Number of Movies	976
Available Ratings	62156
Missing Ratings	5869972
Dataset Size	5932128
Data Sparsity (%)	98.95%

Table 4: Description of MovieLens Dataset

[2]https://movielens.org/

2.2 Evaluation metrics

Evaluating the performance of a recommender system model can be done by many metrics. These metrics are grouped by two major categories of model evaluation: decision support metrics and statistical accuracy metrics [8][11][13]. Statistical Accuracy Metrics evaluates the quality of the recommender system by calculating evaluation scores in order for the recommender system to generate ratings for the unrated items. Decision support metrics helps the recommender system to choose the most relevant items to the user from the set of non-rated items.

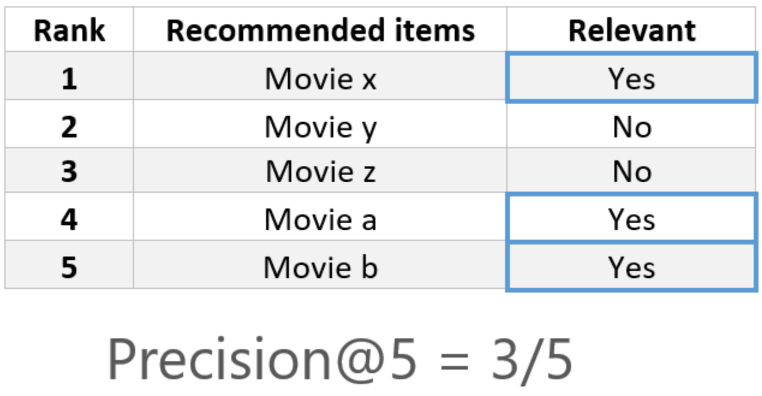
Since we are evaluating our model in the recommendation phase of the recommender system, this means we are using the decision support metrics. The most popular and effective among these metrics are precision and recall [11].

2.2.1 Precision

Precision helps measure the quality of the model and its ability to recommend only relevant items. We calculate the precision by the number of relevant recommended items divided by the total number of recommended items. Equation 4 demonstrates its formula [8].

$$\text{Precision} = \frac{|I_e| \cap |I_r|}{|I_r|} \tag{4}$$

We take the figure 20 as an example of the result of precision on top five recommended items for a specific user. The number of recommended and relevant items is three divided by the number of total recommended items five.



Rank	Recommended items	Relevant
1	Movie x	Yes
2	Movie y	No
3	Movie z	No
4	Movie a	Yes
5	Movie b	Yes

Precision@5 = 3/5

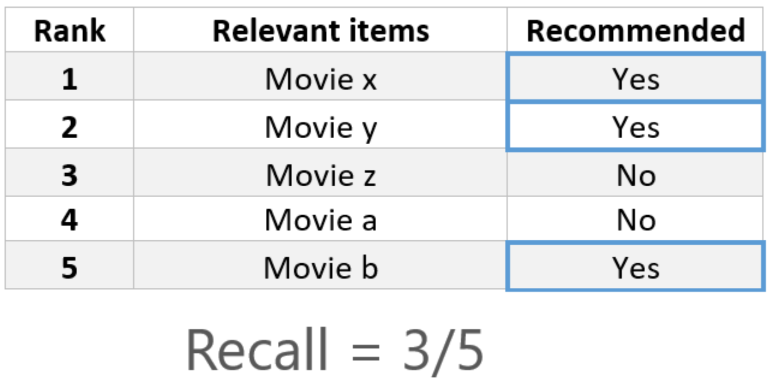
Figure 20: Example of calculating precision

2.2.2 Recall

Recall is the measure of quantity and the ability of the model to recommend all the relevant items. We calculate the recall by the number of relevant recommended items divided by the total number of relevant items. Equation 5 demonstrates its formula [8].

$$\text{Recall} = \frac{|I_e| \cap |I_r|}{|I_e|} \tag{5}$$

We take the figure 21 as an example of the result of recall on three recommended items for a specific user. The number of recommended and relevant items is three divided by the total number of relevant items five.



Rank	Relevant items	Recommended
1	Movie x	Yes
2	Movie y	Yes
3	Movie z	No
4	Movie a	No
5	Movie b	Yes

Recall = 3/5

Figure 21: Example of calculating recall

2.3 Tools

We are using Google Collaborator Platform, which provides Jupyter notebooks in the cloud, and using python 3 as a programming language. We use some python machine learning' libraries such as Tensorflow, Numpy and Pandas.

3 Discussion of the results

In this section, we experiment with our proposed models in the previous chapter and compare the obtained results to determine the most performing model. We start by determining whether Self-Attention does indeed reduce sparsity of our datasets. Then we discuss the expirement of comparative study between our proposed model and the basic collaborative filtering RS model.

3.1 Does Self-Attention reduce sparsity?

"One of the major problems that complicate the item ranking process is data sparsity because items cannot be reliably linked to users, causing a limitation in the recommendation's effectiveness and limited coverage of recommendation space. This problem occurs due to Insufficient or missing information of either the user or item or both in the dataset" [12].

Before we start implementing self-attention in multi-criteria collaborative filtering techniques to solve problems such as sparsity, we first need to deduct an experiment to prove that self-attention does in fact have a positive impact on sparse datasets.

We have seen in the previous section the shape of datasets that we are using. We have also seen the formula in which we calculate sparsity of the dataset.

In this experiment, we calculate the sparsity level of our datasets before and after applying self-attention method on them. We are applying the Self-Attention on one criterion user-item matrix.

Datasets	**Sparsity level (%)**	
	Before Self-Attention	After Self-Attention
Movie Lens	98.95%	85.50%
TripAdvisor	99.49%	91.62%

Table 5: Sparsity level before and after applying self-attention

The table 5 show the results obtained from this experiment on both datasets movie lens and tripadvisor. We can see the sparsity level is reduced after applying self-attention.

We conclude from this experiment that applying the deep learning method self-attention on sparse datasets does indeed reduce their sparsity level.

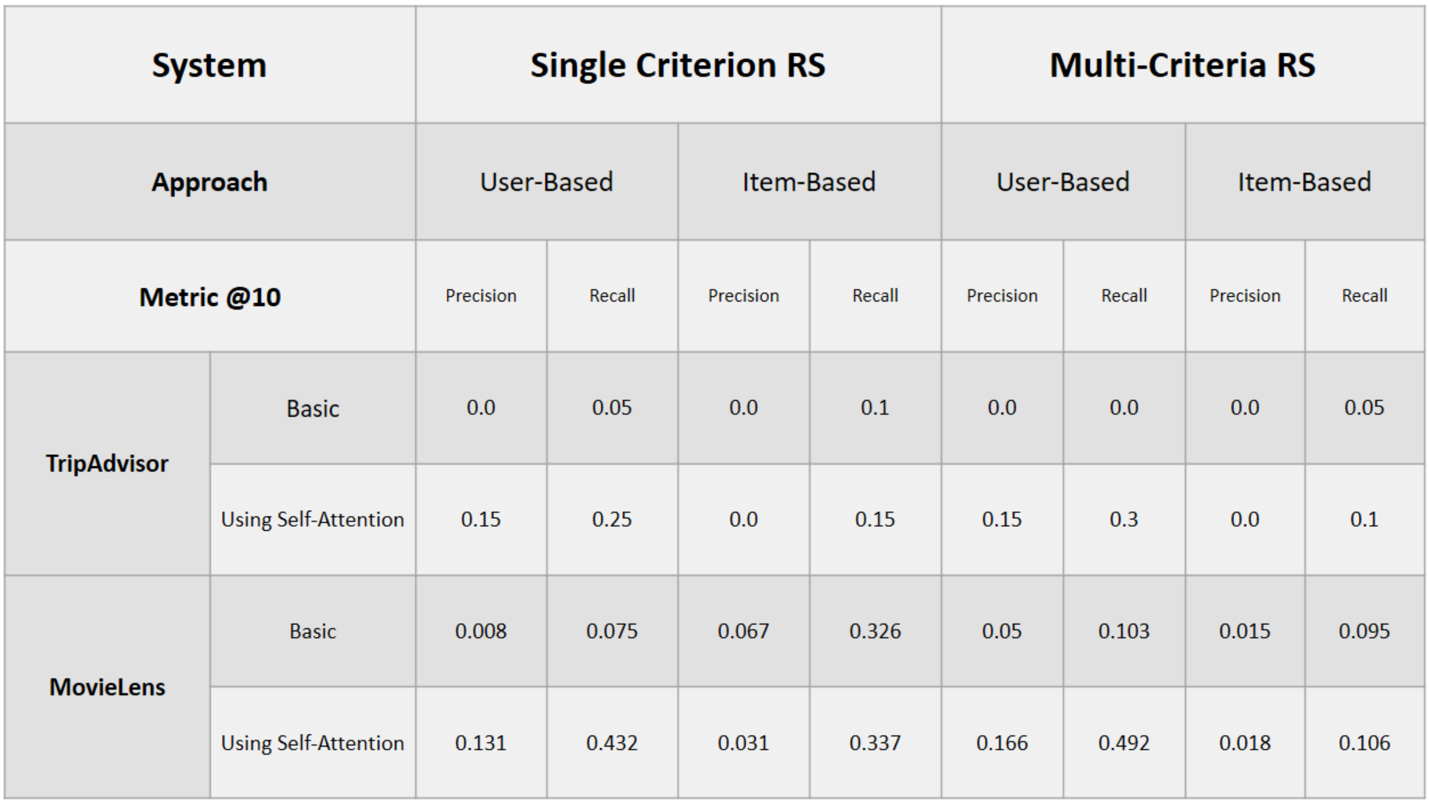
3.2 Does applying Self-Attention improve recommendation on sparse datasets?

After show casing that self-attention reduces sparsity, we now can move to the next point and use the model that we provided in the previous chapter to evaluate the performance of the model with and without the later method.

First, we compare the results of single criterion model with and without self-attention. We start by defining all the necessary functions that define our models as seen in the previous chapter for traditional single rating RS model. We read our dataset, which is of type csv and turn it into user item matrix using only overall ratings. We define relevant items and recommended items numbers. These numbers are necessary to calculate precision and recall as seen in evaluation metrics section. We use the relevant items number, user item matrix to determine the modified user items matrix, and relevant items list. We predict the unrated items of each user using cosine similarity function. We create top N recommended items list using the user items matrix after prediction, as well as the matrix before prediction to select only the unrated items. We also pass the recommended items number to choose the length of the list. Finally, we use both the recommended items, relevant items lists to calculate the precision and recall. For our second Model (i.e. with self-attention), we use the self-attention function on the predicted user item matrix before creating the top N recommended items list.

Next, we compare the results of multi-criteria model with and without self-attention. We start by defining all the necessary functions that define our models as seen in the previous chapter multi-criteria RS model. We read our dataset, take into consideration all criteria, and create k user item matrices (k being number of criteria there is). We define relevant items and recommended items numbers. These numbers are necessary to calculate precision and recall as seen in evaluation metrics section. We use the user item matrices and the relevant items number to create modified user items matrices and a relevant items list. We predict the unrated items of each user using cosine similarity function for all criteria. We create top N recommended items list using the user items matrices after prediction, as well as the matrices before prediction to select only the unrated items, and the recommended items number. We then calculate precision and recall using the resulted lists. For our second Model (i.e. with self-attention), we use the self-attention function on the predicted user item matrices (for each criteria) before creating the top N recommended items list.

We should note that for TripAdvisor dataset, due to hardware limitation problem we could not perform the experiment on the full dataset. Therefore the experiment is done on a smaller sample of the dataset.



System		Single Criterion RS				Multi-Criteria RS			
Approach		User-Based		Item-Based		User-Based		Item-Based	
Metric @10		Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
TripAdvisor	Basic	0.0	0.05	0.0	0.1	0.0	0.0	0.0	0.05
	Using Self-Attention	0.15	0.25	0.0	0.15	0.15	0.3	0.0	0.1
MovieLens	Basic	0.008	0.075	0.067	0.326	0.05	0.103	0.015	0.095
	Using Self-Attention	0.131	0.432	0.031	0.337	0.166	0.492	0.018	0.106

Table 6: Results of precision and recall before and after using Self-Attention using both memory-based approaches on both our datasets

The final results of our experiments can be seen in the table 6. We can see that using Self-Attention yields better results. We conclude from these two experiments, that self-attention does in fact improve the performance of the collaborative filtering technique chosen when it comes to sparse datasets.

4 Conclusion

In this chapter, we started by presenting the datasets that we used TripAdvisor and MovieLens, explaining their architecture and giving a brief description. After that, we learn about the metrics used for evaluating the recommendation phase of our model. Then, the tools that we use for our programming. Finally, we have the results of our experiments where we see that using Self-Attention in Multi-Criteria recommender systems improve the quality of recommendation.

Conclusion And Future work

Conclusion

Over the chapters of this thesis we have introduced multi-criteria recommender system, and the collaborative filtering techniques used in both prediction and recommendation phases. We have as well explained the deep learning innovative method called self-attention. In order to implement this method alongside collaborative techniques, we bring to the discussion the previous works concerning this subject such as using the self-attention in the traditional single rating recommender systems. We implemented our model using heuristic-based approach in prediction phase, while using the most effective deep learning method in recommendation phase in case of our study. We find that Pareto optimal points is the method used for multi-criteria recommender systems. For making our experiment happen, we use TripAdvisor and Movies Lens datasets. While using decision Support Accuracy measurements such as precision and recall to evaluate our models. We compare the results of our study to determine the more performing model out of the models discussed before.

Future work

We were unable to achieve the results that we desired due to lack of hardware. However, we hope to achieve numerous ideas that can improve our model. We want to use the full dataset on our experiments and not just samples of them. That way we can achieve more accurate results. We also want to make more experiments and answer further questions such as: ”Can we classify sparse criteria as a problem?”, ”Does sparse criteria dataset reduce accuracy of recommendation?”, ”How can we solve sparse criteria problem?” and ”Does multi-head self-attention improve execution time while maintaining similar results as normal self-attention?”. We would also like to include the other collaborative filtering techniques to make the complete study of Self-Attention on all collaborative filtering techniques.

References

[1] Adomavicius, Gediminas, Nikos Manouselis, and YoungOk Kwon. ”Multi-criteria recommender systems.” Recommender systems handbook. Springer, Boston, MA, 2011. 769-803.

[2] Chen, Shuo, and Min Wu. ”Attention collaborative autoencoder for explicit recommender systems.” Electronics 9.10 (2020): 1716.

[3] Vaswani, Ashish, et al. ”Attention is all you need.” Advances in neural information processing systems 30 (2017).

[4] Karypis, George. ”Evaluation of item-based top-n recommendation algorithms.” Proceedings of the tenth international conference on Information and knowledge management. 2001.

[5] Lu, Jie, et al. ”Recommender system application developments: a survey.” Decision Support Systems 74 (2015): 12-32.

[6] Lakiotaki, Kleanthi, Stelios Tsafarakis, and Nikolaos Matsatsinis. ”UTA-Rec: a recommender system based on multiple criteria analysis.” Proceedings of the 2008 ACM conference on Recommender systems. 2008.

[7] Lee, Hsin-Hsien, and Wei-Guang Teng. ”Incorporating multi-criteria ratings in recommendation systems.” 2007 IEEE International Conference on Information Reuse and Integration. IEEE, 2007.

[8] Periyasamy, Kola, et al. ”Analysis and Performance Evaluation of Cosine Neighbourhood Recommender System.” International Arab Journal of Information Technology (IAJIT) 14.5 (2017).

[9] Kim, Jaeyoung, Mostafa El-Khamy, and Jungwon Lee. ”T-gsa: Transformer with gaussian-weighted self-attention for speech enhancement.” ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2020.

[10] Li, Jiacheng, Yujie Wang, and Julian McAuley. ”Time interval aware self-attention for sequential recommendation.” Proceedings of the 13th international conference on web search and data mining. 2020.

[11] Adomavicius, Gediminas, and YoungOk Kwon. ”New recommendation techniques for multicriteria rating systems.” IEEE Intelligent Systems 22.3 (2007): 48-55.

[12] Al-Ghuribi, Sumaia Mohammed, and Shahrul Azman Mohd Noah. ”Multi-criteria review-based recommender system–the state of the art.” IEEE Access 7 (2019): 169446-169468.

[13] Nassar, Nour, Assef Jafar, and Yasser Rahhal. ”A novel deep multi-criteria collaborative filtering model for recommendation system.” Knowledge-Based Systems 187 (2020): 104811.

[14] Lv, Yanxia, et al. ”AICF: Attention-based item collaborative filtering.” Advanced Engineering Informatics 44 (2020): 101090.

[15] Yang, Baosong, et al. ”Context-aware self-attention networks for natural language processing.” Neurocomputing 458 (2021): 157-169.

[16] Adomavicius, Gediminas, and Alexander Tuzhilin. ”Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions.” IEEE transactions on knowledge and data engineering 17.6 (2005): 734-749.

[17] Adomavicius, Gediminas, et al. ”Incorporating contextual information in recommender systems using a multidimensional approach.” ACM Transactions on Information systems (TOIS) 23.1 (2005): 103-145.

[18] Manouselis, Nikos, and Constantina Costopoulou. ”Analysis and classification of multi-criteria recommender systems.” World Wide Web 10.4 (2007): 415-441.

[19] Roy, Bernard. Multicriteria methodology for decision aiding. Vol. 12. Springer Science & Business Media, 1996.

[20] Manouselis, Nikos, and Constantina Costopoulou. ”Experimental analysis of design choices in multiattribute utility collaborative filtering.” International Journal of Pattern Recognition and Artificial Intelligence 21.02 (2007): 311-331.

[21] Raschka, Sebastian, and Vahid Mirjalili. Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2. Packt Publishing Ltd, 2019.