Heterogeneous Collaborative Filtering for Recommender System Case Study at Business

Final Year Project Report

A 4th Year Student Name

Vũ Hồng Quân

Instructor Dr. Phan Duy Hùng

Bachelor of Computer Science

Hoa Lac Campus – FPT University

29 December 2020

ACKNOWLEDGEMENT

I would like to thank my instructor, Dr. Phan Duy Hùng, for his patience and time, and for instructing and advising me enthusiastically.

I would like to thank all at my University, FPT, for giving me the best environment to study and grow over the years.

I always remember our family's encouragement and support. Thank to them, I have the target to make great efforts every day.

ABSTRACT

In recent years, lots of techniques applied and optimized to make the recommender system better. Most of them mainly focus on the target interaction between users and items or a part of auxiliary information but hardly leverage other useful information relevant to transactions of customers. In this paper, we want to propose a new model named Heterogeneous neural collaborative filtering (HNCF) for learning recommender systems from two important parts: multi-aspects and multi-behaviors. The HNCF model proposed is divided into four parts: Commuting similarity matrix, Multi-Layer Perceptron, Fusion by the Attention Mechanism, Multi-behavior Prediction. The model exploits characteristics of customers and properties of products from different aspects besides the aspect of purchase by building meta-paths then com-muting similarity matrices. Aspect-level latent factors fusion gives results as factors representing each user and item. These factors synthesizing with multi-behaviors prediction layers is the highlight of the model to make a better recommender system.

Keywords: multi-aspects, multi-behaviors, recommender system, heterogeneous network.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

1.1 Recommender System Overview

Nowadays, with the development of the global Internet network, customers of information systems, especially e-commerce websites, gradually get acquainted and regularly use electronic transactions in many fields such as securities, insurance, finance, banking, technology, telecommunications, and so on [1], [2]. However, with a huge amount of information, they often feel difficult to find and choose suitable information. For example, customers want to deposit money in the bank but hardly know which bank products to choose, whether they should choose an insured deposit product or not? What are the benefits and possible risks and what is the best product for the customer? Customers have lots of choices but not enough time or knowledge to make optimal decisions.

On the other hand, from the aspect of product suppliers, agencies always want the best products to reach potential customers. So how to assist customers in purchasing while they always have a comparison rating between products to make the best choice. Once the customer has been interested in a certain product, other products are needed to introduce to the customer [3]. Moreover, it is necessary to propose to the customer how many products are reasonable. Suppliers are concerned about the process of introducing new products and curious about which types of products will be suitable that have not had customer reviews.

Stemming from the problems in interaction between customers and enterprise's products in e-commerce, the recommender system was introduced and developed to better support customers and businesses. Websites with an electronic trading system need a good advisory system to make a big profit because customers can immediately find the products they like and they will tend to buy. On the contrary, if they cannot find the products they are interested in, they may leave and look for better service providers.

The recommender system has become an important area of research stemming from the first studies. Based on the algorithm, there are two main types of recommender system [4], [5], [6]:

- Content-based Filtering (CB): The system will take into account the content and characteristics of the current item, then suggest the user to similar items [7]. More specific types of content-based recommendation systems consist of CB by Content Similarity, by Latent Factor Modeling, Topic Modeling, or by Popular Content Promotion.

- Collaborative Filtering (CF): the system will analyze users with the same rating and purchase current items. Thereafter, it finds a list of other items rated by these users, rates, and suggestions for the user. The idea of this approach is based on the similarities of preferences between users to make suggestions. Variations on this type include CF by User Similarity or by Association [8]

Besides, the taxonomy of the recommender system can add two more categories:

- Knowledge-based: Users are suggested products meeting their requirements.
- Hybrid recommendation: Combining different approaches [9], [10].

Every method has its strength and weakness. An important issue of content-based filtering is whether the system is able to find out user preferences from user actions concerning a content source and use them on other types of content. When the system is restricted from putting forward the content of the same type as current users, the value from the recommender system is appreciably lower than other types of content from other services that may be offered. For example, recommend news articles based on browsing useful news but it is much more useful when you can suggest music, videos, products, discussions, and so on from different services based on browsed news.

On the other hand, Collaborative filtering also has some limitations:

- Scalability: Most big recommender systems have millions of users and products so a large amount of computational power is needed to compute.
- Sparsity: The number of items sold on major e-commerce websites is extremely large. The most active users will rate only a small subset of the overall database. Therefore, even the most popular items have very little ratings.
- Cold Start: These systems often require large amounts of user data to make exact recommendations [11].

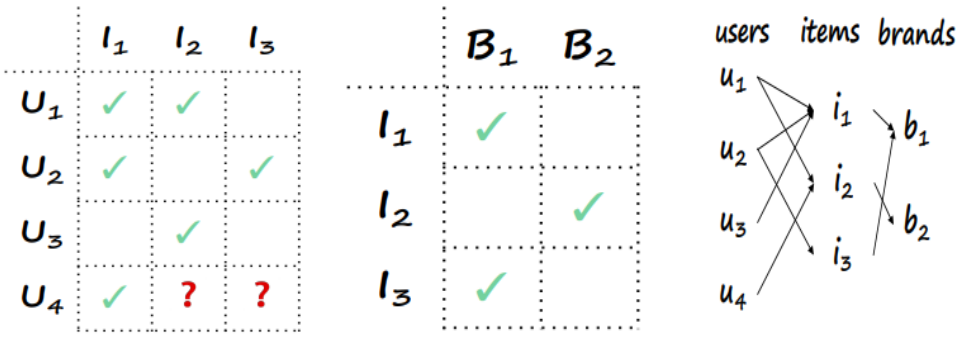
Taking everything into consideration, which technique should be used depends on the situation adopted. But generally speaking, one of the downsides to content filtering is the difficulty in the collection of information, while most of the models relying on collaborative filtering only need 3 information (user id, item id, feedback) to be able to function properly. Therefore, most researchers currently favor collaborative filtering groups.

1.2 Idea and motivation

Current recommender system algorithms, whether Collaborative Filtering (CB), Content-Based Filtering (CF) or hybrid algorithms that combine both methods, predominantly focus on the interactions between the users and items, such as purchase behavior in E-commerce which associating with the Key Performance Indicator (KPI) of conversation rate. However, they hardly leverage other useful information that can influence customer decisions and can increase the accuracy of the algorithm, such as views, clicks, the actions of adding a product to shopping carts, or the different characteristics of users and the properties of items, such as their brands or categories.

In recommender systems, users' behaviors play an integral part in learning and mining users' preferences to personalize delivery or advertisement. For a typical online system, users' behaviors are usually in different forms. For example, when shopping on e-commerce sites, "view" and "click" may happen before "adding-to-cart" and "purchase". We may add some products to the cart but buy only a few things, or view a few trailers and reviews before deciding to eat popcorn and watch a movie. However, many systems today only capture the information of users' purchase history which only reflects user preferences and item characteristics from one aspect as well as has a lack of in-depth investigation of relationships between user behavior types [12], [13].

Besides, the aspects of the product also play an important role e.g., director and actor of a movie, category of product. For example, we look at Figure 1. Let U_i denote i_{th} user and I_j denote j_{th} item from the dataset. Suppose the recommender system wants to predict whether U_4 will buy I_2 or I_3 or not. Based on known data, we can deduce that U_4 is likely to buy I_2 and I_3. But if you consider the item-brand relationship, we see that I_1 and I_3 have the same brand so we should recommend I_3 for U_4.



(a) User-Item purchase relation (b) Item-Brand relation (c) heterogeneous network

Figure 1: A example of aspect-level

There are two auxiliary factors I want to emphasize are user's multi-behaviors and other different aspects of items. Both have important implications for the final customer decision. In general, current models either focus on handling the shallow projection or consider only one of the two factors above [14], [15]. Therefore, I want to devise a model that combines both important factors: multi-aspect information and the translation relationship among different behaviors.

1.3 Related works

- Neural Collaborative Filtering (2017)

This model replaces the inner product between users and items by presenting a neural network architecture named Neural Collaborative Filtering (NCF). Besides, NCF leverages a multi-layer perceptron to learn the user-item interaction function instead of the traditional inner product.

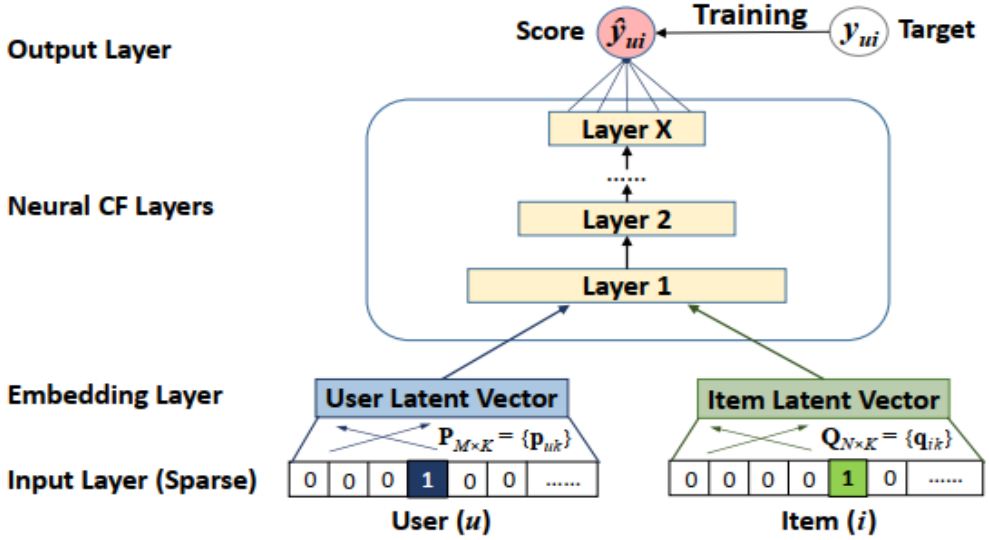


Figure 2: NCF

- Learning to Recommend with Multiple Cascading Behaviors (2019).

A neural network method named NMTR, which combines the recent advances of NCF modeling and the efficacy of multitask learning was built. This correlates the model prediction of each behavior type in a cascaded way.

- Efficient Heterogeneous Collaborative Filtering without Negative Sampling for Recommendation (2020)

In this paper, the prediction of each behavior is correlated in a transfer way to capture the complicated relations among different behaviors. Furthermore, a newly designed optimization method is used for efficient whole-data based model learning.

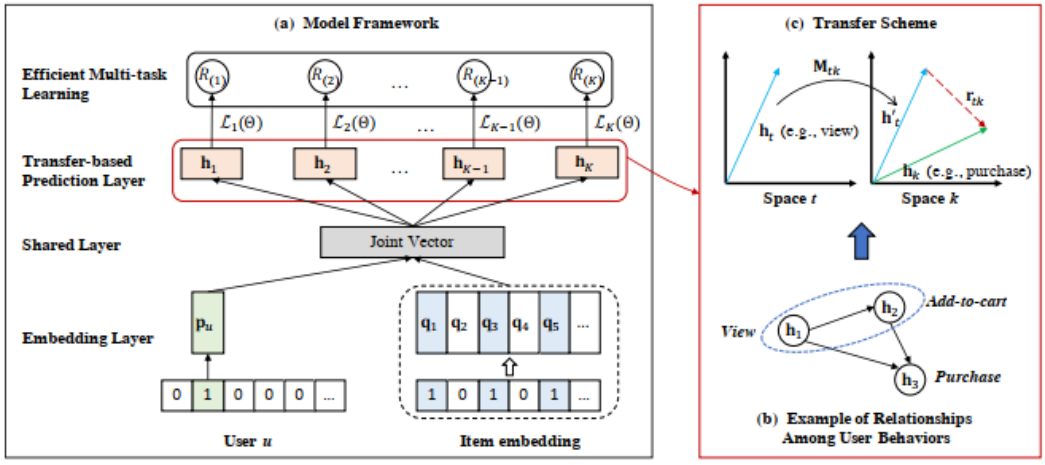


Figure 3: EHCF

- PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks

A novel and practical notion of meta path-based similarity were proved to be efficient for heterogeneous information networks. They comparatively and systematically examine different semantics of similarity measures in such networks and introduce a new meta path-based similarity measure to find similar objects of the same type in such networks.

- Deep Collaborative Filtering with Multi-Aspect Information in Heterogeneous Networks (2019)

Authors propose a novel neural network named NeuACF based on different-aspect features extracted from heterogeneous networks with meta-paths. NeuACF can learn the aspect-level latent factors and then fuses them with the attention mechanism. They also propose Neu-ACF++ which employs the self-attention mechanism to learn the importance of different aspects.

1.4 Challenges

Each user action in context has its meaning and has a close relationship with other actions. Most existing latent factor models predominantly make use of the rating information between users and items but are not concerned about the different useful aspects [16], or just use ineffective methods to commute similarity [30].

Aside from that, existing approaches for multi-behavior recommendation can be divided into two categories [16]. Collective matrix factorization (CMF) is the first category [17], [18], [19], [20], which jointly factorizes multiple behavior matrices based on extending the matrix factorization (MF) method. Another category approaches the problem from the perspective of learning [21], [22]. It is natural to assume that a user's interacted items should be more preferable over the non-interacted items when the system wants to learn recommender models from the (implicit) data of interactions. However, these data often lack behavior semantics such as the add-to-cart behavior cannot happen before a view or a view can happen without purchase. Chen et al. [16] leveraged these factors and correlated the model prediction of each behavior type in a cascaded way but fixed the relational line in behaviors.

Along with that, the characteristic of e-commerce is sparsity (a large number of users and goods but a small number of transactions) [23]. There are even users who have not made a purchase yet and also have products that have not yet been purchased [11]. Therefore, considering

only the transactions already done, we may not be able to come up with an RS that is good enough for customer satisfaction as well as the profitability of the company. Moreover, due to a large number of users and items, the data to be processed is too large. Meanwhile, the model we use has to consider both time efficiency and accuracy.

1.5 Contribution

This work proposes a new model named Heterogeneous Neural Collaborative Filtering (HNCF) which combines the two important factors: multi-aspect information and the translational relationships be-tween different behaviors.

From a defined Heterogeneous Information Network (HIN), the model first determines the meta paths based on the structure of the dataset, then use PathSim [24] to compute the similarity matrices between users with other users and items with other items. Aspect-level latent factors are learned through a standard Multi-Layer Perceptron (MLP) then fused by the attention mechanism. This paper focuses on aggregating these result factors with multi-behaviors prediction layers to improve the performance.

The data for training and testing was obtained from an online shop in the Middle East at the end of 2019. The efficient optimization method was implemented with TensorFlow and optimized using mini-batch Adagrad [25]. The proposed algorithmic process is tested to verify their performance and effectiveness is confirmed using two indexes: Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG).

2 Collecting data and preprocessing

2.1 Dataset

This model leverages a real-world E-commerce dataset from a large multi-category online store in the Middle East. The data was collected from October to November 2019. Table 1 gives 9 properties of the dataset:

Table 1: Data properties

Property	Description
event_time	The time when the event happened at (in UTC).
event_type	Only one kind of event: purchase
product_id	The ID of a product
category_id	Product's category ID
category_code	Product's category taxonomy (code name) if it was possible to make it. Usually present for meaningful categories and skipped for different kinds of accessories
brand	Downcase string of brand name. Can be missed
price	The price of a product. Present
user_id	Permanent user ID
user_session	Temporary user's session ID. Same for each user's session. It changes every time a user comes back to the online store from a long pause

Auxiliary information is the category and brand of items. There are also 3 kinds of user behaviors: view, add-to-cart, and purchase:
- View: a user viewed a product
- Cart: a user added a product to the shopping cart
- Purchase: a user purchased a product

Because of material facilities and research problems, the model will get 10000 users and 6676 items to train and test.

2.2 Exploration

- Visitors daily trend

This figure shows how the traffic fluctuates by date. We realize that the

number of accesses gradually decreases from 2019-10-01 to the beginning of next month.

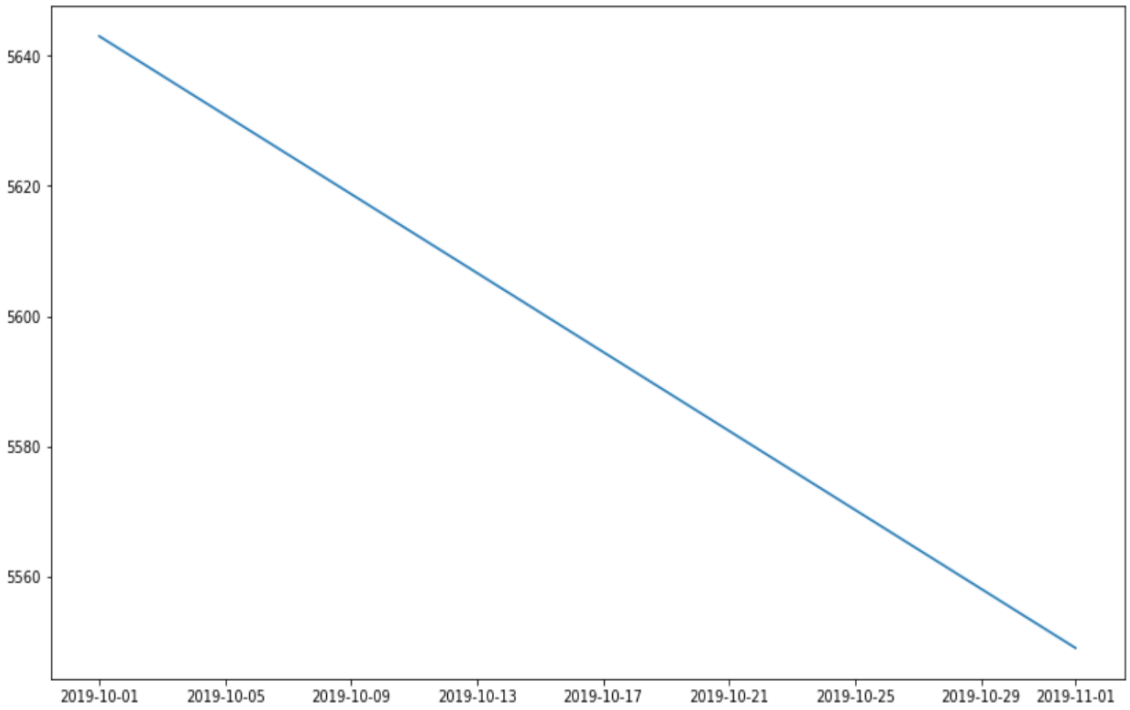


Figure 4: Traffic fluctuation

- Top Category

This figure clearly points out that electronics is the category customers interact with the most. The two next categories are the undefined category and appliances. And meanwhile, country_yard, stationery, and medicine are categories hardly purchased. From this observation, we recognize the long-tail distribution in the commerce domain.

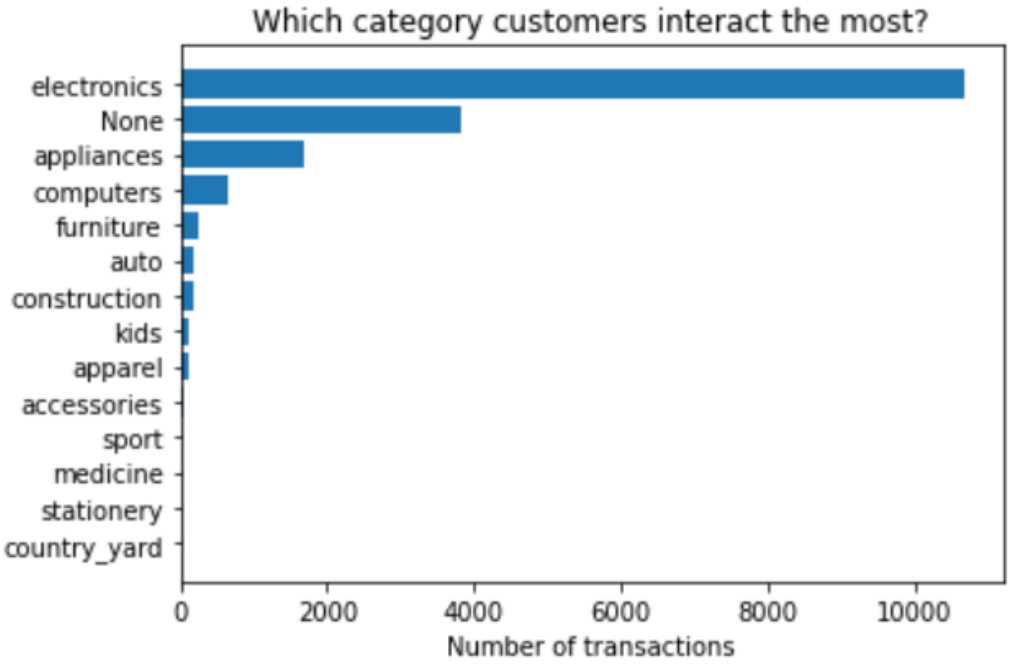


Figure 5: Categories Rank

- Top brand

In addition, Samsung, Apple, Xiaomi, Huawei, and Lucente are the most five brands users take an interest in. Contrary to most people's predictions that Apple is the most popular, Samsung is the brand that is ideal for lots of people. We saw the remarkable rise of 4 mid- and mid-range brands in the top 5 positions. Therefore, the customer segmentation our online shop should concentrate on is individuals from the middle and lower classes.

brand	count
samsung	4267
apple	3606
xiaomi	1430
huawei	647
lucente	311
oppo	287
lg	237
cordiant	183
artel	174
sony	151

Figure 6: Brand Rank

- Top behavior

The number of views of users is much higher than both behaviors. Modern technology development makes online shopping easier than ever for everyone. Thanks to the clicks, customers can ‘view’ a large number of products in a short time. They can ‘add to cart’ and ‘remove from cart’ presently without the appraising eyes as they go shopping in traditional stores. The variety of categories and prices is also an advantage if we want to develop our online shop.

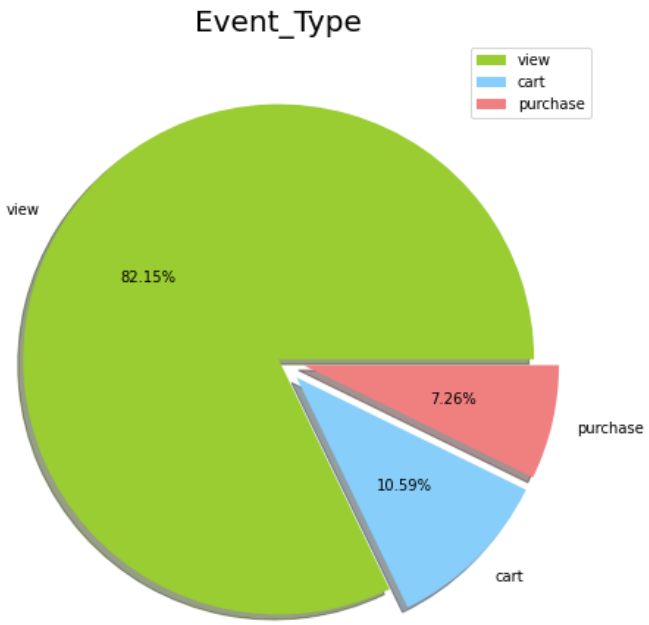


Figure 7: Multi-behavior

- Discovery some session

Select any session to analyze, we recognize the typical behaviors of users when shopping online. First of all, we will look at products in the same category, compare them with each other. They add to the cart the most satisfactory products, think carefully before deciding to buy a certain product.

	event_time	event_type	product_id	category_code	brand	user_id
1040070	2019-11-01 14:47:58 UTC	view	3789	electronic	samsung	321541268
1040071	2019-11-01 14:48:19 UTC	view	3789	electronic	samsung	321541268
1040072	2019-11-01 14:48:52 UTC	view	3789	electronic	samsung	321541268
1040073	2019-11-01 14:49:04 UTC	view	3855	electronic	samsung	321541268
1040074	2019-11-01 14:49:34 UTC	cart	3855	electronic	samsung	321541268
1040075	2019-11-01 14:50:04 UTC	purchase	3855	electronic	samsung	321541268
1040076	2019-11-01 14:50:38 UTC	view	3855	electronic	samsung	321541268

Figure 8: Discovery some session

3 Methodology

3.1 Problem formulation

As we all know, to work or build a new information system, we need to figure out which components are essential to create them. Simply, great things are built of small things. If we talk about a recommender system approached by the machine learning method, there are three basic characteristics as follows:

- The first thing to consider is the “user”.
- We need to pay attention to items. These items can be products on the sales page, songs on music sites, or other “friends” on social networks.
- Leveraging the feedback of each user on that item is significant. It could be the rating, an indicator of a user's interest in the item, and so on.

After we have gathered the above information of the system, we need to represent that information in a computable form [26]. A great idea is to use matrices. A generated matrix showing each user preference for the respective items is represented as follows.

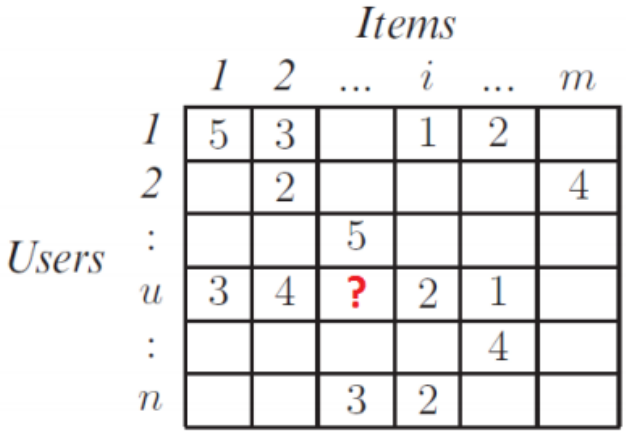


Figure 9: Interaction between users and items

In the matrix, there are weighted cells that will show the preference of each user on items. On the other hand, there are also empty boxes to show that the user has never accessed the item. This shows the role of a recommender system. Based on information known in the user’s past, the system will suggest to that user information the user does not know. It means predicting the values in the empty cells in the matrix above and sorting in descending order of preference.

Formally:

Let U be a set of n users, |U| = n, and u be a specific user ($u \in U$)

Let I be a set of m items, |I| = m, and i be a specific item ($i \in I$)

Let R be a set of user response values (these are usually preferences of the user) and $r_{ui} \in R(R \subset \Re)$ is the rating of user u on item i.

Let $r : U \times I \mapsto R \qquad (u,i) \mapsto r_{ui}$

The target of the recommender system is to find a function $\hat{r} : U \times I \mapsto R$ so that $\zeta(r, \hat{r})$ satisfies some conditions. For example, if ζ is a function that evaluates accuracy (R_2) then it needs to be maximized if ζ is a function for measuring error like Root Mean Squared Error (RMSE):

$$RMSE = \sqrt{\frac{1}{D^{test}} \sum_{u,i,r \in D^{test}} (r_{ui} - \hat{r}_{ui})^2}$$

then it should be minimized.

Let $\{y_1, y_2, ..., y_K\}$ denote the user-item interaction matrix for all K types of behaviors, where $y_k = [y_{k-uv}] \in \{0,1\}$ ($u \in U$ and $i \in I$) indicates whether user u has interacted with item v under behavior k. Generally, a model has a target behavior to be optimized, which is denoted as y_K. In this model, the target behavior is purchase and other behaviors are click, view, and add-to-cart.

3.2 Model overview

First of all, the meta path [27], [28] containing objects is created based on the rich user-item interaction information in the recommendation, and PathSim is used to compute similarity matrices for each meta path of HIN. Then these matrices are used as inputs for a neural network to learn the aspect-level latent factors separately. They will be combined with the attention mechanism and each user or item will have its final latent factor. Simultaneously, we apply the transfer-based multi behavior prediction to generate the prediction layer for the K^{th} behavior (the target behavior) from other behaviors. Therefore, the likelihood that user u (p_u) will purchase item v (q_v) under k^{th} behavior is estimated with.

$$\hat{y}_{K-uv} = h_K^T \times (p_u \odot q_v)$$

3.3 Heterogeneous Information Network

We need to clarify some basic definitions to go deep into the heterogeneous information network.

Definition 1: Information Network. An information network is defined as a directed graph G = (V, E) with an object type mapping function φ : V → A and a link type mapping function ψ : E → R, where each object v ∈ V belongs to one particular object type φ(v) ∈ A, and each link e ∈ E belongs to a particular relation ψ(e) ∈ R

It is important to realize that the relationship is reversible. It means if we have A R B, a relationship exists from type A to type B, the inverse relation R^{-1} holds naturally for B R^{-1} A. For most of the times, R and inverse R^{-1} are not equal, unless the two types are the same and R is symmetric.

For example, the eCommerce information network (dataset) is a form of the heterogeneous network which contains four object types: user (U), item (I), brand (B), and category (C) and links from each item to other objects. Links are assigned by dissimilar relations.

Definition 1.2: Heterogeneous/Homogeneous information network. The information network is called a heterogeneous information network if the types of objects |A| > 1 or the types of relations |R| > 1; otherwise, it is a homogeneous information network.

The semantics underneath different paths, in essence, indicate different similarities. For instance, two users can be linked through "user-item-user" (UIU) or "user-item-brand-item-user" (UIBIU) path. Let define these paths as the meta path.

Definition 2: Meta path. A meta path P is a path defined on the graph of network schema TG = (A, R), and is denoted in the form of $A_1 \to A_2 \to ... \to A_l$, which defines a respectively composite relation $R = R_1 \circ R_2 \circ ... \circ R_l$ between type A_1 and A_l, where ∘ denotes the composition operator on relations.

After obtaining meta paths, the target is computing similarity between users vs. users and items vs. items based aforesaid paths. In fact, there are several methods. Personalized PageRank [29] is an asymmetrical measure of similarity which evaluates the likelihood of visiting object y from object x by walking randomly on the network with a restart. ObjectRank [30] and PopRank [31] recognize that heterogeneous relationships could have an impact on random walking, and assigned different propagation factors to each form of object relationship, either to derive a revised version of P-PageRank (ObjectRank) or a global PageRank (PopRank). Nevertheless, such methods only grant one specific combina-

tion of all the possible meta paths utilizing the fixed weights decided by the damping factor and propagation factors between different types.

Motivated by Sun et al. [28], this model uses PathSim similarity which finds peer objects in the network to measure the similarity between users vs. users and items vs. items based on their meta paths.

Definition 3: PathSim: A Meta path-based similarity measure. Given a symmetric meta path P, PathSim between two objects of the same type x and y is

$$s(x,y) = \frac{2 \times |p(x,y)|}{|p(x,x)| + |p(y,y)|}$$

where p(x, y) is a path instance between x and y, p(x, x) is that between x and x, and p(y, y) is that between y and y.

Let M as the commuting matrix for a meta path $P = (A_1, A_2, ..., A_l)$ then $M = W_{A_1A_2} W_{A_2A_3} ... W_{A_{l-1}A_l}$ where $W_{A_iA_{i+1}}$ is the adjacency between two types A_i and A_{i+1}. From that, $s(x_i, x_j) = \frac{2M_{ij}}{M_{ii} + M_{jj}}$ is PathSim between two objects x_i and x_j.

For example, table 2 shows the adjacency matrix W_{UC} implying how many items a user purchased in each category.

Table 2: Adjacency matrix W_{UC}

	Electronics	**Furniture**	**Appliance**	**Accessories**
Quan	2	1	0	0
Chung	50	20	0	0
Mai	2	0	1	0
Nghia	2	1	0	0
Quang	0	0	1	1

We have $W_{CU} = (W_{UC})^T$. The commuting matrix is $M = W_{UC} W_{CU}$ or:

$$M = \begin{bmatrix} 5 & 120 & 4 & 5 & 0 \\ 120 & 2900 & 100 & 120 & 0 \\ 4 & 100 & 5 & 4 & 1 \\ 5 & 120 & 4 & 5 & 0 \\ 0 & 0 & 1 & 0 & 2 \end{bmatrix}$$

Now we compute the similarity between users *Quan* and *Chung*:

$$s(Quan, Chung) = s(x_1, x_2) = \frac{2M_{12}}{M_{11} + M_{22}}$$

$$= \frac{2 \times 120}{5 + 2900} = 0.08261618$$

Finally, the similarity matrix between users underneath the meta path UCU is indicated as:

Table 3: Similarity matrix

	Quan	**Chung**	**Mai**	**Nghia**	**Quang**
Quan	1	0.08261618	0.8	1	0
Chung	0.08261618	1	0.06884682	0.08261618	0
Mai	0.8	0.06884682	1	0.8	0.28571429
Nghia	1	0.08261618	0.8	1	0
Quang	0	0	0.28571429	1	1

To be stricter, two theorems need to be proved:

Theorem 1: Properties of PathSim:

1. Symmetric: $s(x_i, x_j) = s(x_j, x_i)$
2. Self-maximum: $s(x_j, x_i) \in [0,1]$ and $s(x_j, x_i) = 1 \; \forall i$
3. Balance of Visibility: $s(x_i, x_j) \leq \frac{2}{\sqrt{\frac{M_{ii}}{M_{jj}}} + \sqrt{\frac{M_{jj}}{M_{ii}}}}$

Theorem 2: Limiting behavior of PathSim under infinity length meta path.

Let meta path $P^{(k)} = (P_l P_l^{-1})^k$, M_p be the commuting matrix for meta path P_l, and $M^{(k)} = (M_P M_P^T)^k$ be the commuting matrix for $P^{(k)}$, then by PathSim, the similarity between objects x_i and x_j as $k \mapsto \infty$ is:

$$\lim_{k \to \infty} s^{(k)}(i, j) = \frac{2r(i)r(j)}{r(i)r(i) + r(j)r(j)} = \frac{2}{\frac{r(i)}{r(j)} + \frac{r(j)}{r(i)}}$$

where r is the primary eigenvector of M, and r(i) is the i_{th} item.

3.4 Multi-Layer Perceptron

To have a network that is able to learn different aspect-level latent factors separately, this model employs a Multi-layer Perceptron (MLP) which gives it a large level of flexibility and non-linearity.

In addition to the Input and Output layers, an MLP can have multiple hidden layers in between [32], [33], [34]. Hidden layers in the order from the input layer to the output layer are numbered as hidden layer 1, hidden layer 2, and so on. Figure 10 is an example with 2 hidden layers.

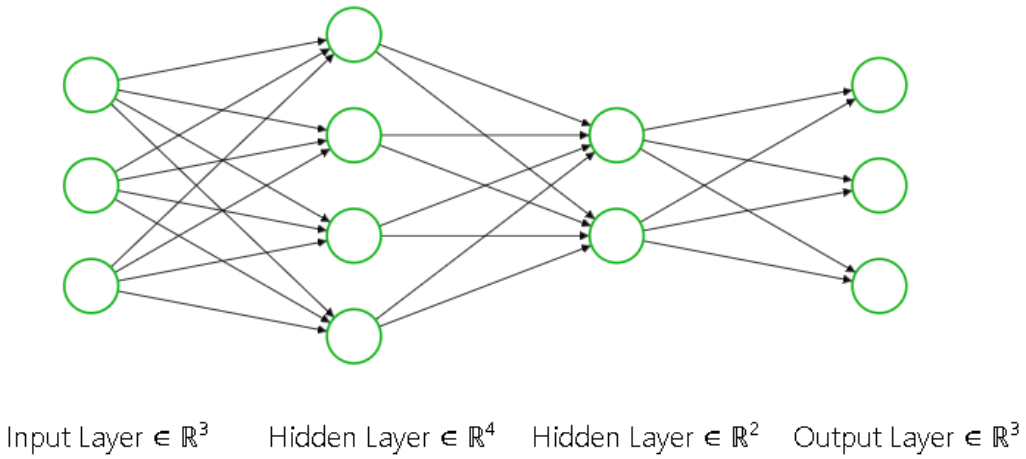


Figure 10: MLP with 2 hidden layers

The inputs of the hidden layers are denoted by z, the output of each unit is usually denoted a (representing activation, which is the value of each unit after we apply the activation function to z). The output of the i^{th} unit in layer l is denoted by $a^{l}_{(i)}$. Activation function can be considered among Sigmoid, Tanh, or ReLU functions.

Sigmoid function is formed as $\frac{1}{1+e^{-s}}$. If the input is large, the function will give an output close to 1 [35], [36], [37]. For a small (very negative) input, the function will give an output close to 0. This function has been used a lot in the past because it has a very nice derivative but it has one fundamental drawback: Sigmoid saturates and kills gradients. If the input has a very large absolute value (very negative or very positive), the gradient of this function is almost zero. This means that the coefficients which correspond with the units being learned will be nearly not updated. Tanh (hyperbolic tangent) also has the same disadvantage of having very small gradients with large absolute value inputs [38].

ReLU (Rectified Linear Unit) is widely used recently because of its simplicity. Its gradient is computed extremely fast with a gradient of 1 if the input is greater than 0, equal to 0 if the input is less than 0. Krizhevsky et al. [39] proved that ReLU makes training of Deep Net-works much faster. Therefore, we apply ReLU as the activation function of MLP.

We also leverage similarity matrices computed above as inputs of MPL. The number of layers in 1 MLP is equal to the number of hidden

layers plus 1 (the total number of layers except for the input layer). For example, we get some meta-paths as UICIU. The similarity matrix which presents the similarity between users is a $n \times n$ matrix S^C with the property that the i_{th} user U_i is represented as an n-dimensional vector S_i^C (n is the total number of users in a dataset). After each layer, the output vector from the previously hidden layer is mapped into a new vector in a new space. It means that the initial similarity vector between user U_i and other users is projected to a low-dimensional aspect-level latent factor:

$$
\begin{aligned}
a_0 &= S_i^C \\
a_1 &= RELU(\mathrm{W}_1^T \times a_0 + b_1) \\
a_2 &= RELU(\mathrm{W}_2^T \times a_1 + b_2) \\
&\dots \\
u_i^B &= RELU(\mathrm{W}_n^T \times a_{n-1} + b_n)
\end{aligned}
$$

where W_i and b_i are the weight matrix and bias for the layer, respectively.

Similarity matrices between items under meta-paths are implemented analogously.

3.5 Attention Mechanism

After applying MLP to learn the aspect-level latent factors separately for users and items within their own domain, we need to incorporate them together to acquire upholstery factors. Naturally, we think of average all the latent factors. Another way is to concatenate all the factors. However, both strategies cannot show different levels of importance in the information presentation of aspects. Therefore, I decided to use attention-based aspect-level latent factors fusion. Attention mechanism's effectiveness has been proved in various domains such as machine translation or image captioning [40], [41], [42].

In fact, our brain also has its attention mechanism. For example, our eyes have a field of view of 120 degrees in both vertical and horizontal directions. However, most of the time we process only a small part of the image information. Did you notice that when we drive and turn left, or right, we only pay attention to the part of the space on the rearview mirror and then think to decide to move next? This mechanism of the brain helps us not to need a lot of energy to make decisions but still provides reliable results.

In general, there are two kinds of attention mechanisms: hard attention (reinforcement learning to learn where to pay attention) and soft at-

tention (weighted learning using the backpropagation algorithm). The soft attention is chosen for our model due to the ease of optimization. The mechanism will learn the weights to pay attention to on all the informative parts of the image, the sentence, or whatever is necessary to synthesize all the information to make a prediction. This aggregate of information is calculated by a weighted average of all pieces of information. These weights are easily learned by the model by backpropagation. Specifically, given the user's category-aspect latent factor u_i^C, we use a two-layer network to compute the attention score s_i^C by the following equation:

$$s_i^C = \mathrm{W}_2^T \times f(\mathrm{W}_1^T \times u_i^C + b_i) + b_2$$

where W_* and b_* are the weight matrix and bias, respectively.

Finally, we can apply the Softmax function and a normal ratio to obtain the attention value for the aspect-level latent factors:

$$w_i^C = \frac{\exp(s_i^C)}{\sum_{k=1}^{A} \exp(s_i^k)}$$

where A is the number of aspects.

With each k-aspect latent factor of user u_i, we have attention weights w_i^k respectively, the aggregated latent factor u_i can be calculated by the following equation:

$$u_i = \sum_{k=1}^{A} \mathrm{w}_i^k \times u_i^k$$

3.6 Multi-Behavior Prediction

To demonstrate the importance of other behaviors of users besides target behavior is the purchase, we denote h_k as the prediction layer for the k_{th} behavior.

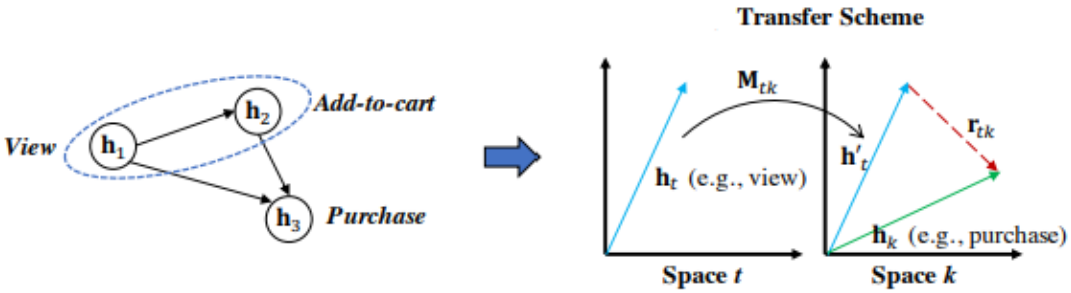


Figure 11: Transfer scheme

As mentioned above, behaviors of users as view, add-to-cart, remove-from-cart, or purchase do not exist independently but are closely related. The prediction of a behavior is dependent on the predictions of its precedent behaviors (Chen et al. 2020) [40]. Simultaneously, a transfer mechanism is also designed in knowledge representation [41]. Motivated by that mechanism, the transfer scheme of two relational behaviors (from h_t to h_k) is formulated as:

$$f_{h_t \to h_h} = h_t \times M_{tk} + r_{tk}$$

where M_{tk} is a transfer matrix projecting h_t from the t_{th} behavior space to the k_{th} behavior space and r_{tk} is the bias.

When we aggregate all of the behaviors on a sequence of behaviors to some considered behavior, we obtain the equation:

$$h_k = \sum_t (h_t \times M_{tk} + r_{tk})$$

where t is considered as the precedent behaviors of that k_{tk} behavior. Of course, the first behavior (view) is randomly initialized.

Because of the sparsity of most E-commerce datasets, this prediction layer can make the recommender system better for users who have few behaviors and transactions.

3.7 Objective Function

Let u_i and v_j denote the aggregated latent factors of users U_i and V_j, respectively. We take the problem of the Logistic problem form. We try to predict whether the user u_i interacts with item v_j or not under the k^{th} behavior h_k in the range of [0, 1], especially the target behavior y_{K-ij} (purchase). To ensure that the resulting results are properly formatted, Sigmoid is used as an activation function.

The probability of the interaction between the user u_i and item v_j is formulated according to the equation:

$$y_{k-ij} = sigmoid(h_k \times (u_i \odot v_j)) = \frac{1}{1 + \exp(-h_k \times (u_i \odot v_j))}$$

Further assuming that the data points are randomly generated independently of each other, we could write:

$$p(\alpha, \alpha^- \mid w) = \prod_{i,j \in \alpha} y_{ij} \prod_{i,k \in \alpha^-} (1 - y_{ik})$$

$$p(\alpha, \alpha^- \mid w) = \prod_{i,j \in \alpha \cup \alpha^-} (y_{ij})^{y_{ij}} \prod_{i,k \in \alpha^-} (1 - y_{ik})^{1 - y_{ij}}$$

where w is the parameter, α and α^- are the positive and negative instance sets, respectively.

=> Loss function respectively to k_{tk} behavior:

$$L_k = \sum_{i,j \in \alpha \cup \alpha^-} (y_{k-ij} \times \log y_{k-ij}) + (1 - y_{k-ij}) \times \log(1 - y_{k-ij}))$$

Multi-task learning (MTL) is a paradigm that performs joint training on the models of different but correlated tasks, to obtain a better model for each task [42]. Loss function should be fused by a sum weighted to maintain the different level of behaviors:

$$L = \sum_{k=1}^{K} \lambda_k L_k$$

where $\sum_{k=1}^{K} \lambda_k = 1$.

This is the overall objective function of our model. We will use Tensor-flow, a modern machine learning tool, to implement our efficient optimization method. To optimize the objective function, we use mini-batch Adagrad [43], [44] as the optimizer. There are some advantages when applying Adagrad. It eliminates the need to manually tune the learning rate. Convergence is faster and more reliable than simple SGD when the scaling of the weights is unequal. It is not very sensitive to the size of the master step.

3.8 Implement the algorithm

The model is implemented by Tensorflow [45], [46]. The dataset is used to train and test from an online shop in the Middle East at the end of 2019. We set the batch size to 256 and set the learning rate to 0.0005. With each couple user u_i and item v_j which interact with each other, I add 10 items more to which user u_i do not have any relation-ship for negative sampling.

The original data is very large but highly sparse. Each user interacts with a few items which make it difficult to design a sufficiently efficient recommender system for a large number of customers.

Table 4: Statistics of evaluation dataset

Interaction	Items	Users	Sparsity
82533	6676	10000	99.88%

To avoid overfitting, the model adopts regularization. In a sense, regularization is to slightly alter the model to avoid overfitting while pre-

serving its generality (generality is a lot of data descriptiveness, in both training and test sets). More specifically, we will try to move the solution of the loss function optimization problem to a point near it. The direction of the movement will be the one that makes the model less complicated, although the value of the loss function will increase slightly.

4 Analysis and Improvement

4.1 Evaluation

To study how the model works for the recommender system, the model will apply the leave-one-out evaluation which was introduced former [47], [48], [49]. Therefore, what is the leave-one-out evaluation?

In many cases, we have a very limited amount of data to build a model. If taking too much data in the training set as validation data, the rest of the training set data is not enough to build the model. At this point, the validation set must be very small to keep the amount of data for training large enough. However, another problem arose. When the validation set is too small, overfitting can happen with the remaining training set. The solution is cross-validation.

Cross-validation is an improvement of the validation with the amount of data in the validation set is small, but the quality of the model is assessed on many different validation sets. A common way to use this is to divide the training set into k subsets that do not have a common element and have approximately the same size. At each test, called run, one of the k subsets is taken as a validation set. The model will be built based on the aggregation of k-1 remaining subsets. The final model was determined based on average train errors and validation errors. This approach is also known as k-fold cross-validation.

When k is equal to the number of elements in the original training set (each subset has exactly 1 element), we call this technique leave-one-out. Specifically, the model keeps the final interaction of the user for testing and the remaining for training. For each user, the model randomly gets 99 items that are unrelated to the user as negative samples. Then for each user, we will have 1 set of 100 items for testing (100 couples of users and items). Both Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG) are leveraged to evaluate the model performance. They are defined as:

$$HR = \frac{|\text{keys}|}{|\text{users}|}$$

$$NDCG = \frac{1}{|users|}\sum_{i=1}^{|users|}\frac{\log(p_i+1)}{\log 2}$$

where |hits| is the number of users whose test item appears in the recommended list and p_i is the position of the test item in the list for the i_{th} test user.

In this evaluation, I will proceed with HR@10 and NDCG@10. It means that HR and NDCG measure whether the test item is presented on the top-10 list.

4.2 Compare NeuCF and HNCF

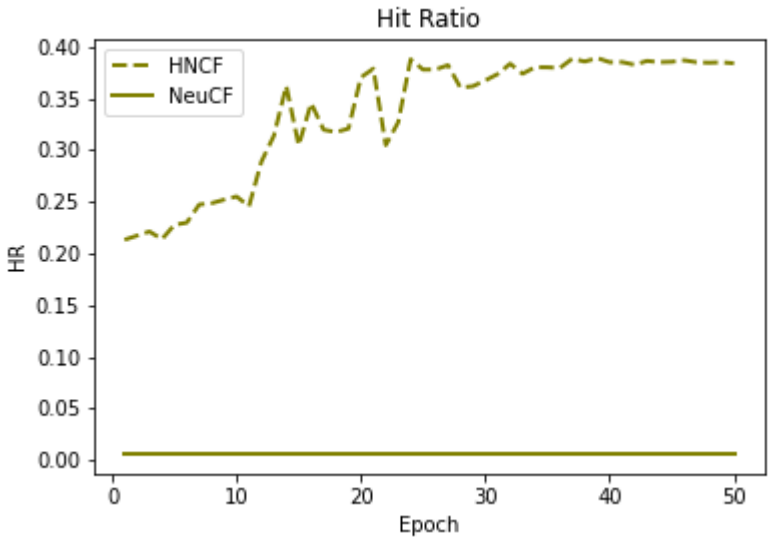


Figure 12: HR

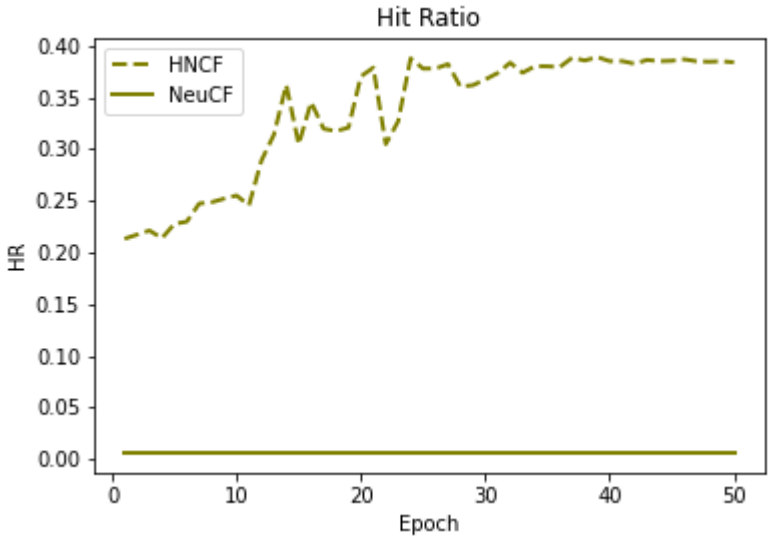


Figure 13: NDCG

The NeuCF model [50] was implemented to test on this dataset. Both HR@10 and NDCG@10 show that HNCF outperformed NeuCF although NeuCF did perform effectively on the movieLen-1M dataset. The most significant reason for this is due to the sparsity of this online shop (99.88%) being higher than the sparsity of MovieLen-1M (95.53%).

4.3 Analysis of Attention Mechanism

When fusing latent factors, two methods are compared:

- Average: Averaging all the aspect-level latent factors
- Attention mechanism: weighted averaging of all pieces of information

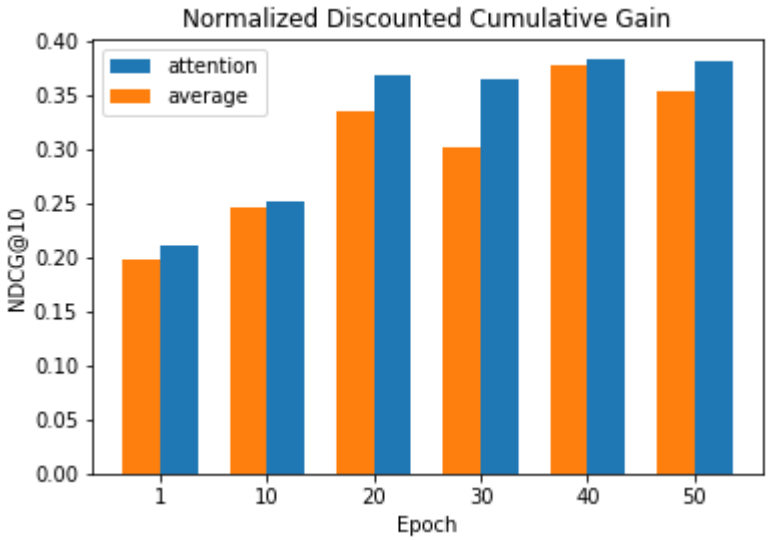


Figure 14: NDCG

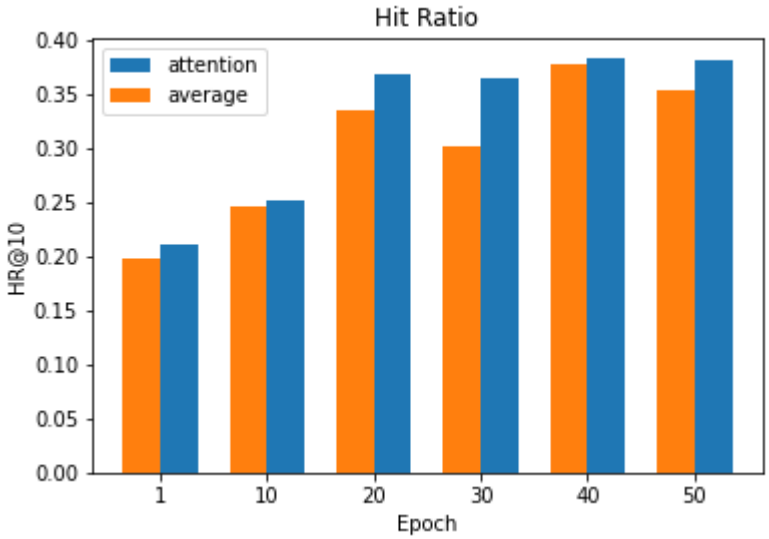


Figure 15: HR

The two charts show that the attention mechanism operates more effectively than average. This is due to the average method normalizing all of the important levels while the attention mechanism concentrates on what is significant.

4.4 Compare the number of layers

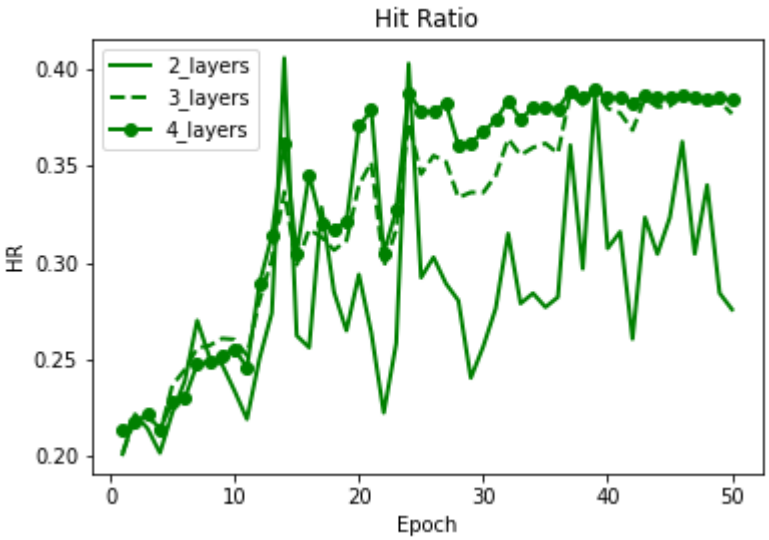


Figure 16: HR

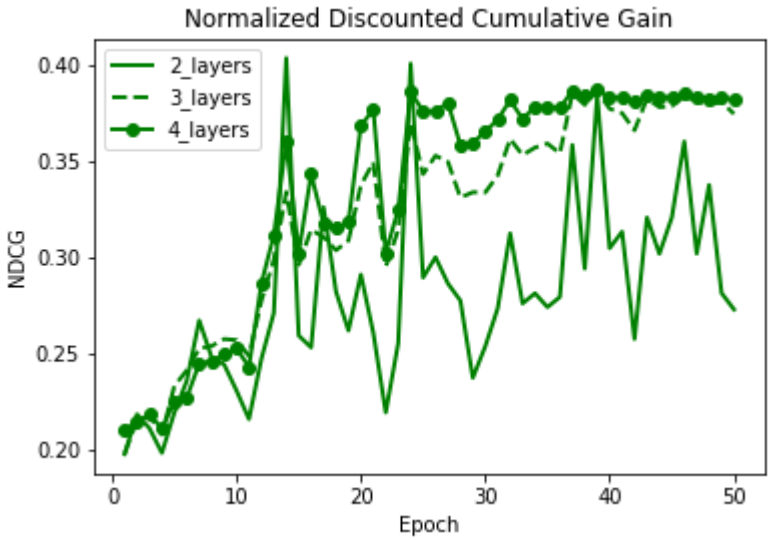


Figure 17: NDCG

The figure demonstrates that the more the number of layers is used, the more HR@10 and NDCG@10 also increased. Therefore, for an attention mechanism, we should use at least 4 layers to achieve the algorithm's effectiveness.

5 Conclusion and Perspective

Nowadays, after some clicks, they can choose the most suitable products from all over the world without having to leave home half a step. Besides convenience, the point attracting users to e-commerce sites is the fulfillment of human's greatest desire: freedom of choice. The variety of models and prices make customers feel comfortable and feel like they "have the right to choose". However, over choice or choice overload [Paul Whitmore, 2001] will lead to indecision and lower sales. Therefore, to increase the efficiency of e-commerce sites and online shops, eCommerce sites integrate the recommender system into the main system to be able to suggest to users the most suitable choices for them, leading to increasing the number of successful transactions. Processing a huge amount of data as well as applying machine learning and deep learning techniques play an important role in making a better system.

Most of the models focus on explicit interaction between users and items and forget other useful information. The others either focus on different aspects (e.g., the category-aspect, brand-category of items) or only handle heterogeneous feedback (e.g., view, click, add to cart) without leveraging all of the useful properties of the amount of that data.

This paper proposes a new model named HNCF that incorporates processing techniques to increase the efficiency of the recommender system. The model first calculates the similarities of users to each other and items through aspect-level latent factors. After aggregating the results, the factors are processed together with the multi-behavior prediction layers to give a probability of the interaction between the user u_i and item v_j is the highlight of the model to enhance the performance of the recommender system. Each user is given then 100 items to test the performance of the algorithm.

This study can be applied to most e-commerce sites because it solves the basic problems of online shopping such as Cold-start and sparsity. Multi-behaviors property enables it to be applied in the Movie recommendation domain. Besides, techniques of exploiting lots of aspects can be helpful in social networks to suggest friends and content to users.

Appendix

Theorem 1: Properties of PathSim
PROOF.

(1): Because of

$$M_{ij} = M_P(i,:) \bullet M_l(j,:) = M_l(j,:) \bullet M_l(i,:) = M_{ji}$$

We have:

$$s(x_i, x_j) = \frac{2M_{ij}}{M_{ii} + M_{jj}} = \frac{2M_{ji}}{M_{ii} + M_{jj}} = s(x_j, x_i)$$

where $\bullet$ means the dot product of two vectors.

(2): Let $M_l(i,:) = (a_1, a_2, ..., a_P)$, $M_l(j,:) = (b_1, b_2, ..., b_P)$

It is easy to recognize that a_k, b_k are nonnegative for $\forall 1 \le k \le p$, then

$M_{ij} = \sum_{k=1}^{p} a_k b_k \ge 0$, $M_{ii} = \sum_{k=1}^{p} a_k^2 > 0$ and $M_{jj} = \sum_{k=1}^{p} b_k^2 > 0$ so $s(x_i, x_j) \ge 0$

We also have $\sum_{k=1}^{p} a_k^2 + \sum_{k=1}^{p} b_k^2 \ge 2\sum_{k=1}^{p} a_k b_k$,

"=" happens when $a_k = b_k \ \forall k$

$\Rightarrow s(x_i, x_j) \le 1$ and $s(x_i, x_i) = 1 \ \forall i$

(3): Adopting Cauchy-Schwarz inequality, we get:

$$M_{ij} = \sum_{k=1}^{p} a_k b_k \le \sqrt{\sum_{k=1}^{p} a_k \sum_{k=1}^{p} b_k} = \sqrt{M_{ii} M_{jj}}$$

$$\Rightarrow s(x_i, x_j) \le \frac{2}{\sqrt{\frac{M_{ii}}{M_{jj}}} + \sqrt{\frac{M_{jj}}{M_{ii}}}}$$

Theorem 2: Limiting behavior of PathSim under infinity length meta path
PROOF.

Because $M = (M_P M_P^T)$ is really symmetric, it can be decomposed as $M = PDP^T$, where D is a diagonal matrix with the values of eigenvalues of M, P is an orthogonal matrix composed of eigenvectors corresponding to eigenvalues in D.

Let r be the first column in P, then $M^k = PD^k P^T$.

Let $s_{ij}^{(k)} = \frac{2M_{(i,j)}^{k}}{M_{(i,i)}^{k} + M_{(j,j)}^{k}}$, λ_1 is the largest eigenvalue of M

$$\Rightarrow s_{ij}^{(k)} = \frac{2(PD^{k}P^{T} / \lambda_1^{k})(i,j)}{(PD^{k}P^{T}(i,i) + PD^{k}P^{T}(j,j))/\lambda_1^{k}}$$

$$\text{and} \lim_{k\to\infty} s^{(k)}(i,j) = \frac{2r(i)r(j)}{r(i)r(i) + r(j)r(j)} = \frac{2}{\frac{r(i)}{r(j)} + \frac{r(j)}{r(i)}}$$

References

1. C. J. U. J. M. a. S. L. Hoofnagle, "Mobile Payments: Consumer benefits and new," BCLT Research Paper, 2012.
2. R. Alt, "Evolution and perspectives of electronic markets," Electron Markets 30, p. 1–13, 2020.
3. Y. T. &. C. C.-H. Chong, "Customer needs as moving targets of product development: A review. International Journal of Advanced Manufacturing Technology," 48. 395-406. 10.1007/s00170-009-2282-6, 2009.
4. S. &. K. M. &. A. K. &. G. T. &. N. S. Yang, "Combining content-based and collaborative filtering for job recommendation system: A cost-sensitive Statistical Relational," 2017.
5. P. &. G. R. &. B. S. B.Thorat, "Survey on Collaborative Filtering, Content-based Filtering and Hybrid Recommendation System," 2015.
6. R. &. L. A. Glauber, "Collaborative Filtering vs. Content-Based Filtering: differences and similarities," 2019.
7. B. &. S. K. &. K. A.-C. &. D. J. Barz, "Finding Relevant Flood Images on Twitter using Content-based Filters".
8. C. &. C. S. &. C. F.-L. Leung, "A collaborative filtering framework based on fuzzy association rules and multiple-level similarity," 2009.
9. J. M. F.-L. J. F. H. M. A. R.-M. Luis M. de Campos, "Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks," 2010.
10. S. &. A. B. Khorshidi, "A Review of Hybrid Recommender Systems" 2017.
11. K. Falk, Practical Recommender Systems, The United States of America: Manning Publications Co., 2019.
12. C. F. Z. G. a. L. S.-T. S. Rendle, "Bpr: Bayesian personalized ranking from implicit feedback," UAI, p. 452–461, 2009.
13. L. L. H. Z. L. N. X. H. a. T.-S. C. X. He, "Neural collaborative filtering," WWW, p. 173–182.
14. X. D. J. Z. S. H. a. J. C. H.-J. Xue, "Deep matrix factorization models for recommender systems," International Joint Conferences on Artificial Intelligence, p. 3203–3209, 2017.
15. H. Y. M. R. L. a. I. K. H. Ma, "Sorec: social recommendation using probabilistic matrix factorization," Proceedings of the 17th ACMCIKM, p. 931–940, ACM 2008.
16. C. &. J. D. &. H. X. &. G. D. &. C. X. &. F. F. &. L. Y. &. C. T.-S. &. Y. L. &. S. Y. Gao, "Learning to Recommend with Multiple Cascading Be-

haviors," IEEE Transactions on Knowledge and Data Engineering, pp. 1-1, 2019.
17. A. P. S. a. G. J. Gordon, "Relational learning via collective matrix factorization," SIGKDD, p. 650–658, ACM 2008.
18. D. K. J. O. a. H. Y. C. Park, "Do also-viewed products help user rating prediction?," p. 1113–1122, WWW 2017.
19. L. D. C. F. a. L. S.-T. A. Krohn-Grimberghe, "Multi-relational matrix factorization using bayesian personalized ranking for social network data," p. 173–182, WSDM 2012.
20. Z. C. L. H. a. E. H. C. Z. Zhao, "Improving user topic interest profiles by behavior factorization," p. 1406–1416, WWW 2015.
21. R. P. M. L. a. A. H. B. Loni, "Bayesian personalized ranking with multi-channel user feedback," RecSys, p. 361–364, 2016.
22. Y. L. G. G. Z. S. J. Z. a. H. T. N. H. Qiu, "Bprh: Bayesian personalized ranking for heterogeneous implicit feedback," Information Sciences, p. 80–98, 2018 vol. 453.
23. J. &. J. D. Bauer, "Optimal Pricing in E-Commerce Based on Sparse and Noisy Data," Decision Support Systems, 2017.
24. R. L. H. &. Y. K. &. M. H. Uu, "PathSimExt: Revisiting PathSim in Heterogeneous Information Networks," 10.1007/978-3-319-08010-9_6, pp. 38-42.
25. J. &. H. E. &. S. Y. Duchi, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," Journal of Machine Learning Research, pp. 2121-2159, 15 January 2010.
26. V. Pereira, "DS-Recommender Systems," 2020.
27. Y. Y. C. G. a. Y. S. X. Liu, "Meta-path-based ranking with pseudo relevance feedback on heterogeneous graph for citation recommendation," Proceedings of the 23rd CIKM, p. 121–130, ACM 2014.
28. Y. &. H. J. &. Y. X. &. Y. P. &. W. T. Sun, "PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks," International Conference on Very Large Data Bases, pp. 992-1003, 2011.
29. G. J. a. J. Widom, "Scaling personalized web search," WWW, p. 271–279, 2003.
30. V. H. a. Y. P. A. Balmin, "Objectrank: authority-based keyword search in databases," VLDB, p. 564–575, 2004.
31. Y. Z. J.-R. W. a. W.-Y. M. Z. Nie, "Object-level ranking: bringing order to web objects," WWW, p. 567–574, 2005.
32. S. Mirjalili, "Evolutionary Multi-layer Perceptron," 10.1007/978-3-319-93025-1_7, 2019.
33. S. Marsland, "The Multi-layer Perceptron," 10.1201/b17476-4, 2014.

34. C. &. P. C. &. N. M.-L. &. N. G. &. F. A. &. M. F. Voyant, "Multi-layer Perceptron and Pruning," Turkish Journal of Physics, 2017.
35. F. &. B. D. &. A. P. Baione, "An application of Sigmoid and Double-Sigmoid functions for dynamic policyholder behaviour," Decisions in Economics and Finance, 2020.
36. M. Ayyavoo, "Sigmoid function," 2019.
37. M. &. G. R. &. R. P. &. L. B. Goyal, "Activation Functions," 10.1007/978-3-030-31760-7_1, 2020.
38. B. &. H. M. &. D. X. &. L. T. Yuen, "Universal Activation Function For Machine Learning," 2020.
39. A. &. S. I. &. H. G. Krizhevsky, "ImageNet Classification with Deep Convolutional Neural Networks," Neural Information Processing Systems, 2012.
40. C. &. M. Z. &. Z. Y. &. M. W. &. L. Y. &. M. S. Chen, "Efficient Heterogeneous Collaborative Filtering without Negative Sampling for Recommendation," Proceedings of the AAAI Conference on Artificial Intelligence, 2020.
41. A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston and O. and Yakhnenko, "Translating embeddings for modeling multi-relational data," In Proceedings of NeuIPS, p. 2787–2795, 2013.
42. A. E. T. &. P. M. Argyriou, "Convex multi-task feature learning," Mach Learn 73, p. 243–272, 2008.
43. A. &. B. L. &. B. F. &. U. N. Defossez, "On the Convergence of Adam and Adagrad," 2020.
44. A. &. F. S. Lydia, "Adagrad - An Optimizer for Stochastic Gradient Descent," Volume 6, pp. 566-568, 2019.
45. X. Z. S. R. a. J. S. K. He, "Deep residual learning for image," Proceedings of the IEEE conference on computer vision and pattern recognition, p. 770–778, 2016.
46. N. Silaparasetty, "Programming with Tensorflow," 10.1007/978-1-4842-5967-2_9, 2020.
47. X. H. B. K. a. S. R. I. Bayer, "A generic coordinate descent framework for learning from implicit feedback," WWW, 2017.
48. H. Z. M.-Y. K. a. T.-S. C. X. He, "Fast matrix factorization for online recommendation with implicit feedback," SIGIR, pp. 549-558, 2016.
49. C. F. Z. G. a. L. S.-T. S. Rendle, "Bpr: Bayesian personalized ranking from implicit feedback," 452-461, p. UAI, 2009.
50. C. Shi et al.: Deep Collaborative Filtering with Multi-Aspect Information in Heterogeneous Networks. In: IEEE Transactions on Knowledge and Data Engineering (2019).