Journal of Information Science and Engineering 38, 547-570 (2022)
DOI: 10.6688/JISE.202205_38(3).0004

Hybrid Embedding of Multi-Behavior Network and Product-Content Knowledge Graph for Tourism Product Recommendation

Li-Pin Xiao[1], Po-Ruey Lei[2,+] and Wen-Chih Peng[3]
[1]*KKday, Taipei, Taiwan*
[2]*Department of Electrical Engineering*
ROC Naval Academy, Kaohsiung, 813 Taiwan
[3]*Department of Computer Science*
National Yang Ming Chiao Tung University
Hsinchu, 300 Taiwan
E-mail: felix.xiao@kkday.com; barry@cna.edu.tw; wcpeng@cs.nctu.edu.tw

In the last decade, recommendation systems have gradually become the most important service for online business, which serve as sales assistants for e-commerce business increasing their profits. However, the conventional recommendation systems are usually confronted with two challenges. First, in online shopping contexts, users often browse products that they do not go on to order. The majority of action sequences are browsing-browsing rather than browsing-order. As a result, user actions are not a direct reflection of user preferences. Second, the popularity of sold products creates a skewed distribution that results in the problem of cold-start product for recommendation. In this paper, we present our research on developing a two-stage framework of hybrid recommendation system to tackle these two challenges for tourism product recommendation. In order to extract knowledge from users' implicit feedback, we develop the neighborhood structure of users and products in the multi-behavior interaction network that simultaneously incorporates the browsing and order behaviors. To ensure the coverage of cold products, we considered the metadata associated with products and extracted more features from the textual content to form a product-content knowledge graph. By embedding the multi-behavior network and product-content knowledge graph within the recommendation system, we were able to capture user preferences from implicit feedback and the relationships among products. To evaluate the proposed model, we conducted experiments on a real-world dataset. Experimental results indicate that the proposed approach outperforms several widely-used recommendation systems.

Keywords: tourism product recommendation, hybrid embedding system, multi-interaction behaviors, knowledge graph, network embedding

Received January 31, 2021; revised May 12, 2021; accepted August 25, 2021.
Communicated by Chia-Hui Chang.
[+]Corresponding author was supported in part by the Ministry of Science and Technology of Taiwan, Project MOST 109-2221-E-012-004-MY2.

1. INTRODUCTION

E-commerce has achieved great success over the Internet in the last two decades. Users not only can obtain a huge amount of information but can also access more and more services online. A growing sector of e-commerce is the tourism products. Especially for independent travelers, there are many unknowns. They face questions such as 'What are points of interest (POIs) worth visiting?', 'Which hotel best suits my needs?', and 'Which mode of transport should I use when I reach my destination?' Answers to these questions can be found on sites such as Facebook, YouTube, or blogs. In addition, travelers can buy almost all the travel products that they needed (including SIM cards, theme park tickets, transportation tickets, *etc.*) on online travel platforms before leaving for their trip.

The huge number of travel related products available online, as well as the complexity of their content, means that it can be difficult and time-consuming for users to find what they are looking for. Therefore, it is important for e-commerce travel platforms to filter their product offerings through well-designed recommendation systems. The simplest and least effective algorithms merely recommend the most popular products. However, the results with poor diversity may be recommended and then users may always see the same products. This leads to the lack of personalization tends to undermine the user's experience of the platform.

Providentially, the websites and apps of travel platforms are able to collect users' online behavior when they browse the website and purchase something.

To analyze information on user behavior data can gain a better understanding of user preference from the products they browsed and ordered. However, there are still some challenges we need to face for exploring knowledge from implicit feedback data.

The majority of research into recommendation systems focuses on increasing the click-through rate [1–3]. However, increasing the click-through rate may not increase the order rate. Browsing is the process of discovering new products and includes the decision-making process for which products to purchase. Ordering is the purchase action, which is usually performed after a significant period of browsing. Conversion happens when the recipient of a marketing message performs a desired action [4] (*i.e.*, ordering after browsing). The conversion rate describes the proportion of website visitors that makes a purchase. Fig. 1 presents statistics on user-based and product-based conversion rates from the popular travel platform KKday. The user-based conversion rate measures how many products a user has bought relative to how they have browsed, while the product-based conversion rate measures how many users have bought the product after browsing it. Both types of conversion rates tend to be low. Increasing the browsing rate may not necessarily have a growth on the ordering rate. Therefore, browsing is not an accurate measure of user preferences.

Ordering is more powerful in terms of measuring user interest, but often, the quantity of available data on orders tends to be insufficient to build a reliable recommendation system due to the data sparsity. Therefore, the incorporation of data on both browsing and ordering behaviors likely creates a more robust algorithm.

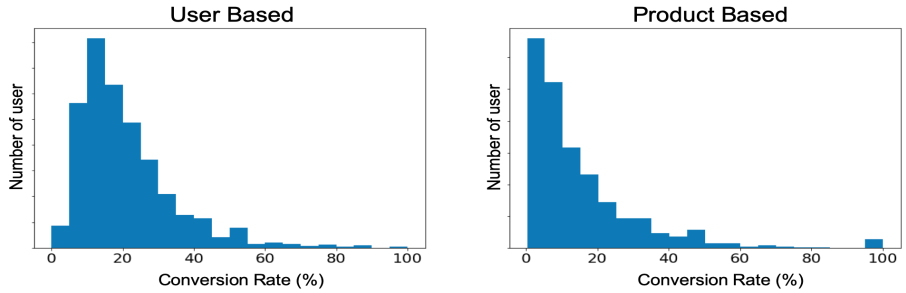

Fig. 1. Conversion rate analysis.

Another common issue facing recommendation systems is the right-skewed distribution of product popularity. In Fig. 2, we present a plot of the browsing behavior of KKday users against product popularity. The distribution of this graph is highly skewed, which means that the most popular products are significantly more popular than the other products. Therefore, if recommendation systems rely on product popularity and consistently recommend top-selling products over others, the recommendation results will lack diversity.

The products that are rarely purchased and never purchased are considered as the cold products. A good recommendation system should consider both the needs of customers and the company. Customers want to see more popular products that meet their preferences. The company's demand is to hope that more cold products can be seen by customers. Thus, a good recommendation system should not only recommend popular products but also recommend a selection of cold-start products which are likely to suit user preferences.

In this paper, we extend the previous work in [5] and propose a two-stage framework of hybrid recommendation system based on the implicit feedback between users and products as well as the relationships among products. Before entering the recommending stage, user segmentation and profiling is developed for differentiating among users based on users' specific characteristics. To capture implicit behaviors, we applied graph embedding so that a product's representation contains users who have browsed and ordered the product. This overcomes the problem of high dimensionality in representing a vertex in a graph and also provides better representation because it takes neighborhood structure into account. To find cold-start products similar to those frequently browsed by users, we incorporated product information with the knowledge graph. The novel contributions made by this paper are summarized in the following:

- We combine browsing and ordering behaviors in the user-product interaction network and incorporate information segment of product with the knowledge graph to increase performance.
- The proposed model recommends products according to different application scenarios, such as only browsing or only ordering.
- The proposed model considers collaborative filtering and content-based methods simultaneously to increase the accuracy of recommendations as well as product coverage.

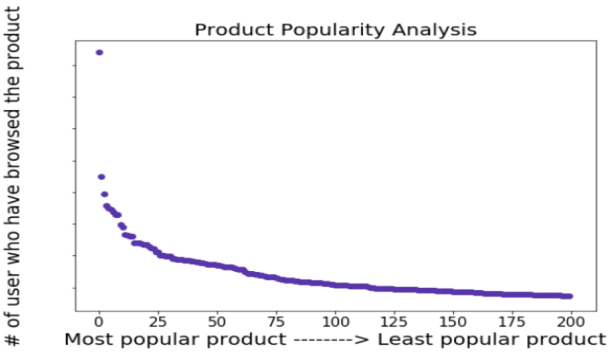

Fig. 2. Skew of product popularity.

2. RELATED WORK

Recommendation systems predict users' preferences to recommend items or services that users may like. Without loss of generality, recommendation lists are generated based on users' historical data, item context, and the related information of the similar users. Collaborative filtering, content-based, and hybrid approaches are widely used by current commercial recommendation system [6–8].

Collaborative filtering recommendation system explores a correlation or similarity between users or items to generate a preference [9, 10]. The authors in [11] model the collaborative relations from the direct and in-direct interactions between users and products. However, data sparsity and cold start problems because of different users and inadequate rating information make it hard to provide accurate prediction [12].

Content-based recommendation approach analyzes textual information of items previously rated by a user, and builds a model or profile of user interests based on the features of the items that user rated [13]. By comparing each item's features with the user profile, the items that have a high degree of similarity with items previously preferred by the user will be recommended. The method proposed in [14] provides the recommendation by predicting the user preference score from the embedding of news title which is a content-based recommendation. Content-based recommendation has ability to capture the specific preference of a user but cannot afford to expand on the users' existing interests. The limited content analysis and over-specialization are the problems of a content-based recommendation system [8].

Although collaborative filtering systems are able to deal with the problem of overspecialization occurred in the content-based filtering approach, it also has cold start and the sparsity problems. To tackle the problems, hybrid recommender systems are proposed to adopt a combination of multiple recommendation strategies to benefit from their complementary advantages [15–18]. The hybrid recommendation model proposed in [19] adjusts recommendation list dynamically by updating similarity table of items incrementally in incremental update item-based collaborative filtering and combining content-based algorithm ensures the

relevance of recommendation. In order to increase the accuracy of recommendations, the hybrid algorithm approach is adopted in our proposed framework. Furthermore, user-item interaction behavior model and embedding methods of network and knowledge graph are considered for improving the problem of cold product coverage.

2.1 Multi-Behavior Based Models

Most studies of recommendation system [11, 20–23] do not take the interactions between users and products (*i.e.*, implicit feedback) into account. However, there are many interactions that users perform on products that can indicate their preferences, such as browsing, sharing, adding to favorites lists, and ordering. It's obvious that different behaviors provide different meanings to the users. For example, research has shown that increasing the click-through rate (*i.e.*, browsing) does not necessarily increase the order rate and may not even positively affect overall revenue [24].

In order to learn the latent representation of products, prod2vec [20] treat the products as words, sequences of products purchased as sentences, and embed the products based on the language model Word2Vec [25, 26]. Behavior2Vec [27] is similar to prod2vec but instead focuses on multi-behavior interactions (*i.e.*, viewing and purchasing) to learn product representations. They replace the products in the sequences with different kinds of behaviors and also apply Word2Vec model on the sequences to generate the viewing vectors and purchasing vectors. The learnt vectors can represent the specific action to the product. They conducted the experiment on four kinds of tasks: viewing to viewing, viewing to purchasing, purchasing to viewing, and purchasing to purchasing to demonstrate the effectiveness of the proposed model.

2.2 Network Embedding

A network graph is a data structure widely applied to real-world applications, such as social networks, technological networks, and information networks [28–32]. A graph consists of nodes and edges, the edges between two nodes indicate that there is a connection between them. Due to this kind of characteristic, we can use such a graph data structure to represent a complex network data easily. Furthermore, modeling the interactions between entities as graphs has provided the opportunity to understand the various network systems in a systematic manner [33]. For instance, in social networks, classifying users into meaningful social groups is useful for many important tasks, such as user search, targeted advertising, and recommendation for friendship or product.

The embedding based representation is able to learn the dense and continuous representations of nodes in a low dimensional space, so that the noise or redundant information can be reduced and the intrinsic structure information can be preserved [34]. With Graph Embedding, it can transform a complex network data structure to a low dimensional vector representation while preserving the properties of the raw network structure. Some approaches like node2vec [35] and deepwalk [36] borrow the concept of natural language processing model like

Word2Vec to generate the embedding vectors of nodes, they treat the nodes in the graph as words, and generate sentences by random walking on the graph.

The growth of deep learning techniques has also improved the research of network embedding. SDNE (Structural Deep Network Embedding) [37] preserves the first-order and second-order proximities by applying a semi-supervised model. To model the first-order proximity, SDNE uses the Laplacian Eigenmaps technique to penalize the similar nodes that are mapped far away from each other in the embedding space. For the second-order proximity, the autoencoder is applied to find the representation for a node that can reconstruct the neighborhood.

2.3 Knowledge Graph Embedding

Knowledge graph (KG) has experienced rapid development in recent years [38]. KG is a multi-relational graph consists of entities (*i.e.*, nodes) and relations (*i.e.*, edges), which provides a structure to store the human knowledge. It is widely used in applications such as recommendation systems, word classification, and question-answering [39–42]. In the recommendation scenario, items or users are mapped to one or more entities in the knowledge graph.

Knowledge graph embedding is used to learn the low-dimension latent representation of entities and relations in a knowledge graph and projecting the elements in KG into the continuous vector space. TransE [43], TransH [44], and TransR [45] are three well known knowledge graph embedding methods, they measure the plausibility of facts by the distance between two entities and exploit the distance-based scoring functions to generate the embeddings. In TransE, all the entities and relations are embedded into a same vector space while this will have trouble when the relations are 1-to-N, N-to-1, or N-to-N. TransH solves this problem by introducing a hyperplane for each relation in the vector space, and the entities vectors are projected to the hyperplane before calculating the distance in the scoring function. TransR introduces relation-specific spaces for each relation, the entities are projected into a relation-specific spaces before calculating the distance.

3. PROBLEM STATEMENT

The aim of this paper was to construct a framework of recommendation system for promotion users and retention users. The target users who have a specific destination for next trip are called as promotion users and the users who may not have used the services of the selected site for a long period are considered as retention users. In order to achieve our goal, the framework development is unfolded in two stages: *User Segmentation and Profiling* and *Hybrid Recommendation*.

3.1 User Segmentation and Profiling

In the first stage, we segment the users into different groups and create a profile for each user. Specifically, for user $u \in U$, given historical logs Log^u and product set P, the output is user class $Class^u$ and the user profile $Profile^u$.

The input historical log $Log^u = \{log_1^u, log_2^u, ..., log_N^u\}$ describes the historical behaviors of user u, where $log_i^u = \{Action_i^u, Product_i^u, Time_i^u\}$. $Action_i^u$ can be a browsing or order action, $Product_i^u$ indicates on which product the user performed the action, and the time at which the action was taken is represented by $Time_i^u$. The product p_i in product set $P = \{p_1, p_2, ..., p_M\}$ keeps a 3-tuple information $p_i = (Title_i, Category_i, Country_i)$.

The two types of output user classes are promotion and retention. For users classified into promotion class, their profiles $Profile^u = \{Country : c\}$ include information on their destination country c. For retention users, we want to extract the day of the week on which the user is most likely to browse the site. For this class, $Profile^u = \{Weekday : w\}$ is extracted, which means that the user u is most likely to browse KKday in the days of the week w.

3.2 Hybrid Recommendation

In the second stage, our task is to predict what products the users may like and make recommendation P_{rec}^u based not only on historical log Log^u ut also on product information in P. In addition to the original dataset, we include the outputs from the first stage (*i.e.*, $Class^u$ and $Profile^u$ are also taken into consideration) in the second stage. Two hyperparameters, h_ratio and top_k, balance the trade-off between recommendation accuracy and product coverage and affect how many products are recommended.

4. FRAMEWORK

Fig. 3 provides an overview of the proposed framework. The framework is divided into two stages: the first stage is user segmentation and profiling, and the second stage is to recommend the products by a hybrid recommendation, which combines the concept from multi-behavior interaction and the product information.

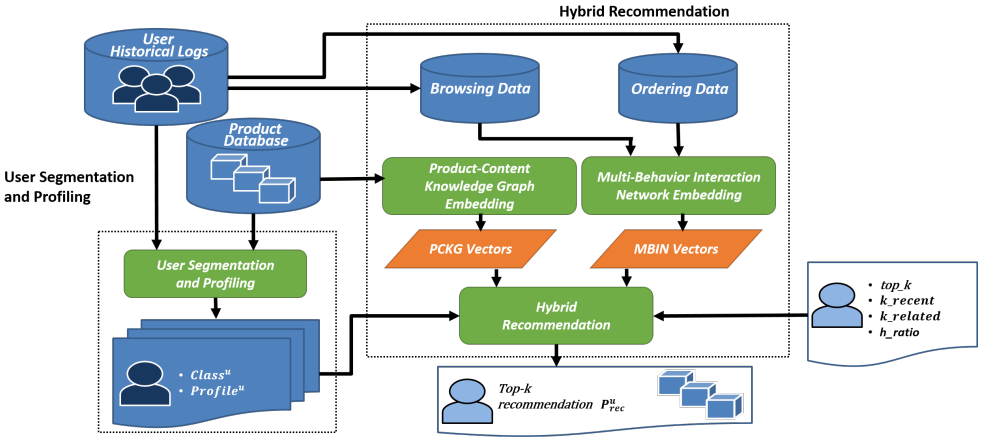

Fig. 3. Framework overview.

4.1 User Segmentation and Profiling

For users who have recently browsed the site, there are available data on users' intended destinations; however, this is not the case for users who have not browsed the site for a while. An effective recommendation system differentiates among users based on user characteristics. Hence, we segment users into two classes: promotion and retention. User class affects the information retained for the user profile. Based on the specific class of the user, our framework will take the corresponding profile for recommendation.

Promotion users seek products for a particular destination, and the relevant products must be marketed directly to them. We need to promote the users to make the orders when we know that they are interested in some countries and looking for some products belong to the countries. To find these users, we reference historical browsing behaviors. If they only browsed a few products from one country, then we classify them as less likely to order these products. If the number of products that the user browsed from a specific country is between the lower bound and the upper bound, then we identify this user as interested in that country. This user is then segmented to the promotion class. Users who have browsed beyond the upper bound exhibit a high probability of ordering, which makes direct promotion unnecessary. The destination country of promotion user u is revealed by their browsing history; the country with the highest number of products browsed by the user is their selected destination c. This information is extracted for the user profile $Profile^u = \{Country : c\}$ and will be taken into consideration in hybrid recommendation.

For some users have become infrequent browsers of a travel platform, the goal is to devise a strategy to attract them to return to browsing and even potentially ordering a product. Based on the collected data, travelers are unlikely to make several trips within the same year; we hence defined a general trip gap. If users have no record of orders for the length of the general trip gap, then it is assumed that they are in the market for a new destination; these users are considered retention users. A good way to attract users to return to browsing is to send an electronic direct mail (EDM) to targeted users. Because emails can be easily overlooked as most users receive a high volume of emails, it is important to send EDMs on the right day of the week when the user is more likely to browse for travel products (based on historical data). As some examples shown in Fig. 4, the temporal analysis of one-month browsing behavior for three users is presented. Each user has specific browsing behavior in temporal feature. Retention users therefore are profiled as $Profile^u = \{Weekday : w\}$ for recommendation. After recommended products are predicted for retention users, EDMs can be sent to the users according to their temporal characteristic of browsing behavior.

4.2 Product-Content Knowledge Graph Embedding

For product information modeling, we build a knowledge graph for products. A knowledge graph with lots of entities (nodes) connected by different types of relations (edges) can easily keep the connection between products and get the structural relatedness between products. The product features we have from

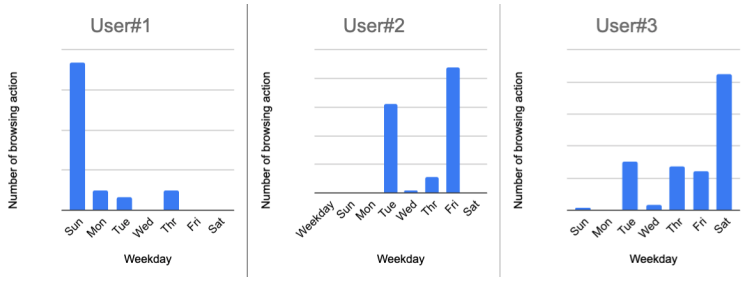

Fig. 4. Temporal analysis of browsing behavior.

Product	Product Name	Category	Country	Text Segmentation
1	台北內湖｜Craftzy Sports 滑雪模擬體驗	Ticket	Taiwan	台北/ 內湖/ Craftzy/ Sports/ 滑雪/ 模擬/ 體驗
2	【日本滑雪一日遊】佐久滑雪花園 PARADA	Package	Japan	日本/ 滑雪/ 一日/ 遊/ 佐久/ 滑雪/ 花園/ PARADA
3	首爾滑雪一日遊｜江原道 WELLI HILLI PARK 滑雪度假村	Package	Korea	首爾/ 滑雪/ 一日/ 遊/ 江原道/ WELLI/ HILLI/ PARK/ 度假村

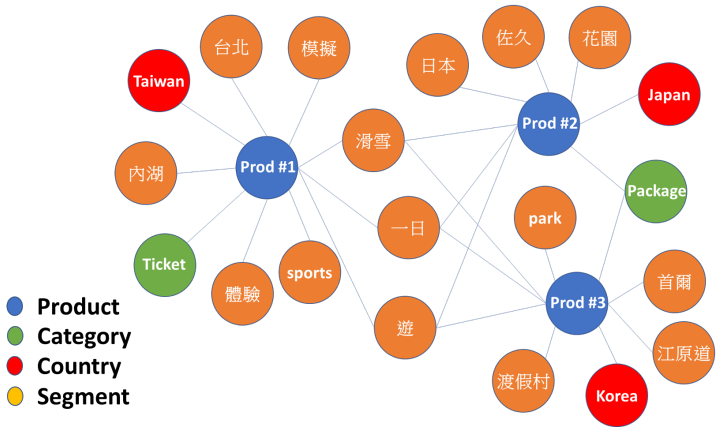

Fig. 5. An example of product-content knowledge graph.

KKday are name, category, and country, and these features can directly be treated as an entity node in the knowledge graph.

However, recommendation by such knowledge graph still suffers the problems of the skewed distribution of product popularity and the cold-start. In order to effectively overcome the problems, increasing connectivity between products is necessary. To enrich the product information and connection between products, the content based features of products should be discovered. We apply the text segmentation method on product name to extract the meaningful segments in the name and use these segments as the content based features to represent the detail knowledge of the corresponding products.

Jieba [46] is a well known text segmentation package that receives a sentence as input and then outputs the words of the sentence in segments. Our proposed algorithm feeds product names into the Jieba module and the product segments

are returned. Some segments will only appear in one product and do not therefore present a relationship to any other product. These segments are helpless to enrich our recommendation result because the connection to any other products is not provided by them. Then, all segments are compared across all the products and thus filtered out so that only key segments remain. In the proposed knowledge graph, we have products, countries, categories, and segments as our entities. As an example shown in Fig. 5, we connect the related entities with three types of relations: product-country, product-category, and product-segment to finish the knowledge graph construction procedure.

The constructed knowledge graph is a powerful tool that allows us to easily calculate the relatedness of every product pair based on the product characteristics of country, category, and segmented product name. For this calculation, we need to first transform the product entities to a vector representation. Several methods for knowledge-graph embedding exist, including TransE, TransH, and Trans R. These methods embed entities in the knowledge graph to a continuous vector space. As the proposed algorithm deals with only three kinds of relations, TransH is the most efficient choice for the purposes of this paper.

The scoring function of TransH [44] is defined as follows:

$$f_r(h,t) = \|h_\perp + d_r - t_\perp\|_2^2. \tag{1}$$

In TransH, each relation has it's relation-specific hyperplane w_r and a translation vector d_r in the hyperplane. The projected vector $h_\perp$ and $t_\perp$ of h and t are connected by d_r on the hyperplane if the triple (h,r,t) exists in the knowledge graph. Let's take a look at an example, there are a similar product pair p_1 and p_2 that have the same country, category, segment, and a dissimilar product pair p_3 and p_4 which only have the same country. Suppose that we have a perfect embedding, the projected vector of p_1 and p_2 will be exactly the same on country hyperplane, category hyperplane, and segment hyperplane, while the projected vector of p_3 and p_4 will only be the same on country hyperplane. Hence the raw vector of p_1 and p_2 will be closer than p_3 and p_4. Based on this characteristic, we can treat the distance between two product vectors as the similarity measurement.

4.3 Multi-Behavior Interaction Network Embedding

Network embedding is an effective method of modeling user-product interactions because it preserves structural characteristics such as the neighborhood structure of users and products. Browsing and ordering are two common interactions performed by users on the products featured on e-commerce platforms. While browsing may not necessarily lead to orders, order logs are sparse for travel platforms. Hence, we incorporate both actions in a single network to increase the performance of recommendation.

From historical logs of all users, we build a multi-behavior interaction network (MBIN) as $G = \{V, E\}$, where V is a set of nodes and E is a set of edges connecting two nodes in V. There are three types of nodes in V: $\{U : user, BP : browsing - for - product, OP : order - for - product\}$, where we call $B(P)$ and $O(P)$ "behavior nodes" that means the browsing or order action on the specific

User	Action	Product
1	Browse	1
1	Browse	2
1	Browse	3
1	Order	1
1	Order	3
2	Browse	2
2	Browse	3
2	Order	2

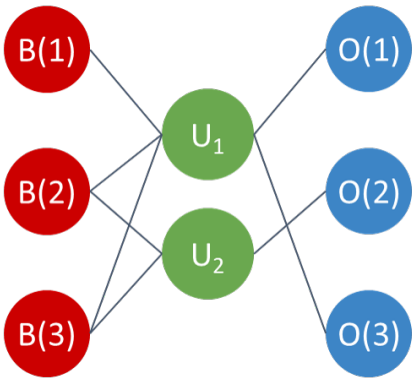

Fig. 6. Example of multi-behavior interaction network.

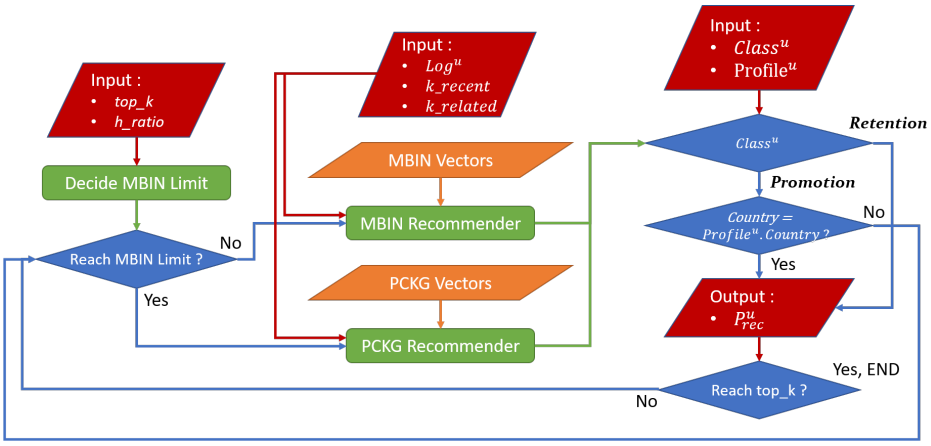

Fig. 7. Hybrid recommendation flow chart.

product. The node in U_i is connected with the node in $B(P)$ or $O(P)$ when the user has browsed or ordered the product. For example, Fig. 6 presents some user-product interaction behaviors and the generated network.

To generate the latent representation vectors of products, Structural Deep Network Embedding (SDNE) [37] is applied to our system. The embedding model takes the user-product interaction network G as its input and learns latent representation vectors that well preserved the first-order and second-order structure for every node in the network.

4.4 Hybrid Recommendation

The proposed recommendation system is a hybrid model that combines data from user-product interactions with product information. User-product interactions provide collaborative information, so preferences can be predicted by identifying users with similar behavioral patterns. Identifying similarities in the content-based information of product descriptions helps overcome the problem

of cold-start products and skewed popularity distributions. Fig. 7 presents the procedure of the hybrid model in detail. We first decide how many products recommended for browsing or order should be chosen from the Multi-Behavior Interaction Network (MBIN) Recommender by $top\_k * h\_ratio$ and select the corresponding number of products from it. We then select $top\_k * (1 - h\_ratio)$ products from the Product-Content Knowledge Graph (PCKG) Recommender.

Algorithm 1 : Hybrid Recommendation

Input: a user u; user class $Class^u$; user profile $Profile^u$; top_k products to recommend; hybrid ratio h_{ratio}

Output: recommendation result P^u_{rec}

1: **if** $Class^u == promotion$ **then**
2: $target\_country = get\_country\_profile(Profile^u)$
3: **end if**
4: $mbin\_limit = top\_k * h\_ratio$
5: $P^u_{rec} = list()$
6: **while** $len(P^u_{rec}) < mbin\_limit$ **do**
7: $product = pop(MBIN\_Recommender())$
8: **switch** $(Class^u)$
9: **case** *promotion*:
10: **if** $get\_country(product) == target\_country$ **then**
11: $P^u_{rec}.append(product)$
12: **end if**
13: **case** *rentention*:
14: $P^u_{rec}.append(product)$
15: **end switch**
16: **end while**
17: **while** $len(P^u_{rec}) < top\_k$ **do**
18: $product = pop(PCKG\_Recommender())$
19: **if** $Class^u == promotion$ **then**
20: **if** $get\_country(product) == target\_country$ **then**
21: $P^u_{rec}.append(product)$
22: **end if**
23: **else**
24: $P^u_{rec}.append(product)$
25: **end if**
26: **end while**
27: **return** recommendation result P^u_{rec}

By adjusting the hybrid ratio h_ratio, we balance recommendation accuracy with product coverage. Algorithm 1 presents the procedure of hybrid recommendation, which generates a recommended list composited from the Multi-Behavior Interaction Network (MBIN) Recommender and Product-Content Knowledge Graph (PCKG) Recommender. The procedures of the MBIN Recommender and PCKG Recommender are respectively presented in Algorithms 2 and 3.

In Algorithm 1, in lines 1 to 3 we obtain the target country if the user is

a promotion user. The number of products that should be recommended from the MBIN Recommender is calculated in line 4. Then we start the procedure of extracting products from the MBIN Recommender in lines 6 to 16. The loop stops when the number of recommended products meets the limit. Every time we extract a product from the MBIN Recommender for a promotion user, we examine the product country (lines 8 to 14). The same extraction process is applied for the PCKG Recommender in lines 17 to 26. To generate the final output, if the user is a promotion user, we check the countries associated with the recommendations made by the MBIN and PCKG recommenders. If the country matches the target country stored in $Profile^u$, then the product becomes recommendation result P^u_{rec}. The procedure is repeated until there are top_k products in P^u_{rec}.

Algorithm 2 : MBIN Recommender

Input: a user $u \in U$; historical records Log^u; embedding vectors V_{BP} and V_{OP} of all browsing and order behaviors; prediction source type S_{type}(browsing or order); prediction target type T_{type}(browsing or order); last K_{recent} behaviors to consider;top $K_{related}$ related products to consider

Output: a sorted product list P^u_{mbin} for user u

1: **switch** (S_{type})
2: **case** *browsing*:
3: $S = list$(last K_{recent} browsing behaviors of u)
4: **case** *order*:
5: $S = list$(last K_{recent} order behaviors of u)
6: **end switch**
7: $S\_weight = softmax(freq(S))$
8: **for** $i = 1$ to K_{recent} **do**
9: **switch** (T_{type})
10: **case** *browsing*:
11: $RP^{(i)} = list$(top $K_{related}$ browsing behaviors of $S^{(i)}$)
12: **case** *order*:
13: $RP^{(i)} = list$(top $K_{related}$ order behaviors of $S^{(i)}$)
14: **end switch**
15: $RP\_weight^{(i)} = softmax(RP)$
16: **for** $j = 1$ to $K_{related}$ **do**
17: $SCORE^{(RP^{(i,j)})} = SCORE^{(RP^{(i,j)})} + S\_weight^{(i)} * RP\_weight^{(i,j)}$
18: **end for**
19: **end for**
20: **return** sorted product list P^u_{mbin} according to $SCORE$

4.5 MBIN/PCKG Recommender

The recommendation is generated by both the MBIN Recommender and PCKG Recommender. The MBIN recommender considers user-product interactions, and the PCKG recommender takes into account the connection between products. Algorithms 2 and 3 respectively present the procedures of these two

recommenders.

In Algorithm 2, the MBIN Recommender takes browsing and ordering behaviors as input to predict a sorted list of products associated with certain behavioral patterns. The recent K_{recent} behaviors and products browsed or ordered by a target user u are designated as the source S from the historical logs Log^u of user (lines 1 to 6). The interaction counts of the sources are then normalized to act as weight S_{weight}(line 7). For every source $S^{(i)}$ interacted with by the user, we find related behaviors $RP^{(i)}$ by comparing the distance between the embedding vectors of $S^{(i)}$ and other behaviors in the MBIN, and normalizing the vector similarities to function as the weight of the related behaviors to the corresponding source $S^{(i)}$ (lines 8-15). In order to determine which related behavior is the most meaningful, we update the $SCORE^{(RP^{(i,j)})}$ of the related behavior $RP^{(i,j)}$ by adding $S\_weight^{(i)} * RP\_weight^{(i,j)}$(lines 16-18). When the procedure is completed for all K_{recent} sources, the $SCORE$ contains the score of all related behaviors. This enables the algorithm to return the sorted list of candidate products related to these behaviors for recommendation (line 20).

Algorithm 3 : PCKG Recommender

Input: a user u; historical records Log^u; embedding vectors V_P of all products in the PCKG; prediction source type T(browsing or order); last K_{recent} interacted products to consider; top $K_{related}$ related products to consider

Output: a sorted product list P^u_{kg} for user u

1: **switch** (S_{type})
2: **case** *browsing*:
3: $S = list($ last K_{recent} browsed products of $u)$
4: **case** *order*:
5: $S = list($ last K_{recent} ordered products of $u)$
6: **end switch**
7: $S\_weight = softmax(freq(S))$
8: **for** $i = 1$ to K_{recent} **do**
9: $RP^{(i)} = list($top $K_{related}$ products of $S^{(i)})$
10: $RP\_weight^{(i)} = softmax(RP^{(i)})$
11: **for** $j = 1$ to $K_{related}$ **do**
12: $SCORE^{(RP^{(i,j)})} = SCORE^{(RP^{(i,j)})} + S\_weight^{(i)} * RP\_weight^{(i,j)}$
13: **end for**
14: **end for**
15: **return** sorted product list P^u_{kg} according to $SCORE$

In Algorithm 3, the input of the PCKG Recommender is a list of products with which the user recently interacted, and the output is a sorted product list based on similarity scores among products. The first step is represented by lines 1 to 6, where we obtain source S, which represents the products most recently interacted with by the target user u. It is obtained from historical logs Log^u of the user. To assign weights S_{weight}(line 7) for the products in S, we use

their normalized interaction counts. We find the most related products $RP^{(i)}$ for every source $S^{(i)}$ interacted with by the user by calculating the distance between the embedding vectors of $S^{(i)}$ and other products in the PCKG. The weights of the related products $RP^{(i)}$ to the corresponding source $S^{(i)}$ is the normalized the vector similarities(lines 8-10). In order to determine the most meaningful recommendation, we add $S\_weight^{(i)} * RP\_weight^{(i,j)}$ to the $SCORE^{(RP^{(i,j)})}$ of the related product $RP^{(i,j)}$ (lines 11-13). When all tasks for all K_{recent} sources are completed, the $SCORE$ will contain the score of all the related products. A sorted list is then returned to the user (line 15).

5. EXPERIMENT

5.1 Dataset and Setting

To verify the efficacy of the proposed framework, the experimental dataset is provided by a traveling e-commerce platform KKday. This dataset contained browsing and order records (*i.e.*, implicit feedback) as well as product information. Each browsing/order record contained the timestamp, user_id, and product_id. Each set of product information contained product_id, product_name, category, country, and selling_time. We collected implicit feedback from January 1, 2018, to March 31, 2019, designating one-year data from 2018 as the training dataset and three-month data from 2019 as the testing dataset. We only considered the records of users who ordered five or more kinds of products to avoid outliers. Based on this dataset, we constructed a multi-behavior interaction network and product-content knowledge graph. The hyperparameters of network and knowledge graph embedding were fine-tuned for optimal performance: K_{recent}, $K_{similar}$ and $K_{related}$ were respectively set to 3, 20, and 20.

5.2 Experiment Tasks

We conducted five experiments to demonstrate the feasibility of the proposed framework. In the first two experiments, we evaluated the effectiveness of the multi-behavior interaction network (MBIN) and product-content knowledge graph (PCKG). In the last two experiments, we compared the performance of the proposed framework to four baseline algorithms (Collaborative Filtering(CF) [47], BiNE [48], SDNE [37], TransH [44]).

- Experiment 1: Effective Evaluation of MBIN Features
- Experiment 2: Effective Evaluation of PCKG Features
- Experiment 3: Hybrid Recommendation
- Experiment 4: Recommendation Scenarios
- Experiment 5: Recommendation for Promotion Users

5.3 Evaluation Measurements

To evaluate the framework, we used precision, recall, f1-score, average precision, and product coverage as the evaluation metrics for recommendation.

- **Precision and Recall**
  - Precision: The fraction of products that are successfully predicted relevant to our prediction.

$$Pre^u@k = \frac{P^u_{rec} \cap P^u_{tru}}{k},\ mean\text{-}Precision@k = (\sum_{u \in U} Pre^u@k)/|U|$$

  - Recall: The fraction of products that are successfully predicted relevant to the user's ground truth

$$Rec^u@k = \frac{P^u_{rec} \cap P^u_{tru}}{|P^u_{tru}|},\ mean\text{-}Recall@k = (\sum_{u \in U} Rec^u@k)/|U|$$

  where P^u_{rec} is the products recommended to the user u and P^u_{tru} denotes the products browsed or ordered by the user u.

- **F1-score: Harmonic mean of precision and sensitivity**

$$F1^u@k = \frac{2 * Pre^u@k * Rec^u@k}{Pre^u@k + Rec^u@k},\ mean\text{-}F1\text{-}score@k = (\sum_{u \in U} F1^u@k)/|U|$$

- **Average Precision: Ranking-aware evaluation metrics**

$$AP^u@k = \frac{\sum_{i=1}^{k}(P(k) * rel(k))}{|P^u_{tru}|},\ mean\text{-}AP@k = (\sum_{u \in U} AP^u@k)/|U|$$

- **Product Coverage: including the product coverage on training or testing dataset for cold-start product evaluation**
  - Training-Coverage@k: The fraction of products that are recommended relevant to the products in the training set

$$Training - Coverage@k = \frac{P_{rec} \cap P^{train}_{tru}}{P^{train}_{tru}}$$

  - Testing-Coverage@k: The fraction of products that are recommended relevant to the products in the testing set

$$Testing - Coverage@k = \frac{P_{rec} \cap P^{test}_{tru}}{P^{test}_{tru}}$$

  where P_{rec} denotes the products that have been recommended to any of the users, P^{train}_{tru} is the products appeared in the training data, and P^{test}_{tru} is products appeared in the testing data.

5.4 Effective Evaluation of MBIN Features

For modeling user-product interaction behavior on recommendation, multi-behavior interaction network (MBIN) is developed by incorporating browsing and ordering actions into a single network. To demonstrate the effectiveness of MBIN features, we trained the embedding vectors from three different kinds of networks: browsing only, order only and a combination of two. The vectors trained by the browsing only dataset were used to predict the future browsing. These were compared with the vectors trained by the combined dataset. A similar approach was taken with the ordering-only dataset. Results are shown in Table 1. For both tests, the vectors trained by the combined dataset outperformed those considering only one type of action.

Table 1. Experiment result of multi-behavior interaction network.

Recommendation Target	Network Type	Precision @k	Recall @k	F1 @k	mAP @k	training coverage @k	testing coverage @k
Recommend Browsing	Browsing Only	0.0257	0.0571	0.031	0.0195	0.2815	0.3595
	Combined Dataset	**0.0267**	**0.0618**	**0.0324**	**0.0203**	**0.3091**	**0.394**
Recommend Order	OrderOnly	0.0057	0.045	0.0097	0.0139	0.0971	0.1509
	Combined Dataset	**0.0124**	**0.1164**	**0.0218**	**0.0365**	**0.2026**	**0.3633**

5.5 Effective Evaluation of PCKG Features

In this section, we determine the effectiveness of including name segments in the knowledge graph. To overcome the problem of cold products and skewed popularity distribution, product information product-content knowledge graph (PCKG) is constructed by product name segmentation. Results are presented in Table 2. Clearly, adding segments to the knowledge graph led to superior performance because it allowed product attributes to be modeled more precisely.

Table 2. Experiment result of product-content knowledge graph features.

Traing Knowledge Graph Dataset	Precision @k	Recall @k	F1 @k	mAP @k	training coverage @k	testing coverage @k
without segment	0.0026	0.0225	0.0045	0.0041	0.6034	0.7189
with segment	0.0051	0.0474	0.0088	0.0147	0.6074	0.7537

5.6 Hybrid Recommendation

The final step of the proposed algorithm assembles the multi-behavior model and the product information model into a single framework. Hyperparameter h_{ratio} adjusts the weight granted to each model. By decreasing this hyperparameter from 1 to 0, we can increase product coverage by incorporating more results from the product information model. Results are presented in Fig. 8. The results are demonstrated under the task that is using browsing to recommend order. It shows that setting h_{ratio} to 0.8 increased the coverage of products while only slightly negatively impacting recall and precision. Hence, by fine-tuning h_{ratio}, a

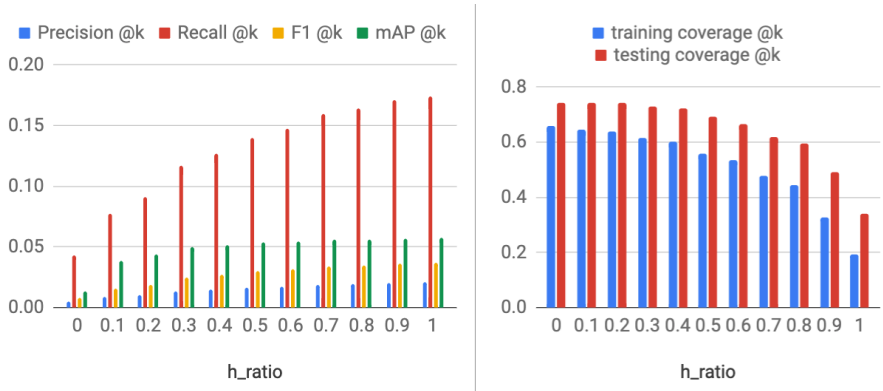

Fig. 8. Metrics for hybrid recommendation.

company can balance the trade-off between product coverage and other evaluation metrics.

5.7 Recommendation Scenarios

Because the proposed system treats browsing and ordering as different actions but models them simultaneously, we have the ability to recommend products based on either browsing or ordering behaviors. There creates four recommendation scenarios: browsing-recommend-browsing, browsing-recommend-order, order-recommend-browsing, and order-recommend-order. In this section, we conducted experiments for these four scenarios to compare the performance of the proposed framework with that of baseline approaches. Results are shown in Table 3. In most of the tested scenarios, the proposed system outperformed baseline methods in terms of precision, recall, f1, and mAP. Although the performance of the proposed method is similar to that of BiNE for the order-recommend-browsing scenario, its performance for product coverage is superior. TranH performs best in terms of product coverage, but the proposed method is superior in other aspects.

The proposed system is aimed at recommending both popular and cold products based on user preferences. Thus, accurate prediction of user preferences and high hit rates for cold products are two important considerations. In order to measure the effectiveness of hybrid recommendation, we evaluated the F1-score of precision and mean cold product hit rate (mCPH) of the recommendation result. We define the cold product hit rate as the number of cold products that are recommended to the user divided by the total number of products recommended to the user. Cold products are those which have a browsing count of less than 50%. The experimental result shown in Fig. 9 indicates that our method outperforms baseline methods in all scenarios.

5.8 Recommendation for Promotion Users

To perform an offline evaluation of the results from promotion recommendation, we first defined the ground truth. For the selected platform, promotions run daily. Therefore, a promotion was defined as (date, user, country). The ground

Table 3. Experiment result for recommendation scenarios comparison.

		Precision @k	Recall @k	F1 @k	mAP @k	training coverage @k	testing coverage @k
	CF	0.0206	0.0501	0.0252	0.0165	0.2294	0.2946
Browsing	BiNE	0.0187	0.039	0.0222	0.0139	0.007	0.009
recommend	SDNE	0.0079	0.0151	0.0092	0.0044	0.3067	0.3832
Browsing	TransH	0.0089	0.0199	0.0108	0.0063	0.5067	0.7088
	OUR	0.0212	0.0494	0.03	0.0181	0.4775	0.6595
	CF	0.0195	0.1668	0.0337	0.0566	0.2542	0.4371
Browsing	BiNE	0.0115	0.0932	0.02	0.0406	0.0083	0.0191
recommend	SDNE	0.005	0.0431	0.0085	0.0109	0.1827	0.3313
Order	TransH	0.0051	0.0474	0.0088	0.0147	0.6074	0.7537
	OUR	0.0201	0.172	0.0361	0.0578	0.4175	0.601
	CF	0.0186	0.047	0.023	0.0156	0.1942	0.2491
Order	BiNE	0.0187	0.039	0.0222	0.0139	0.007	0.009
recommend	SDNE	0.0079	0.0151	0.0092	0.0044	0.3067	0.3832
Browsing	TransH	0.0076	0.0181	0.0094	0.0049	0.4132	0.5776
	OUR	0.014	0.0347	0.02	0.0105	0.3494	0.4569
	CF	0.0122	0.1114	0.0212	0.0397	0.2257	0.3885
Order	BiNE	0.0115	0.0932	0.02	0.0406	0.0083	0.0191
recommend	SDNE	0.005	0.0431	0.0085	0.0109	0.1827	0.3313
Order	TransH	0.0048	0.0462	0.0085	0.0117	0.5204	0.6764
	OUR	0.012	0.1137	0.0219	0.0362	0.368	0.5629

truth of the promotion is defined as whether the user browsed or ordered the promoted product within one month of the promotion. Because user segmentation for promotion is achieved by checking browsing behaviors, the recommendation scenarios for this experiment were browsing recommend browsing and browsing recommend order. Results are shown in Table 4. Comparing the performance of the other baseline methods, most of the experimental results are better than other methods except for the coverage measurement. The coverage performance of TransH and the proposed system is very competitive.

Table 4. Experiment result for promotion recommendation comparison.

		Precision	@k	F1 @k	mAP @k	training coverage @k	testing coverage @k
	CF	0.0487	0.0978	0.0581	0.0453	0.3462	0.4532
Browsing	BiNE	0.0565	0.1139	0.0674	0.0536	0.1249	0.1640
recommend	SDNE	0.0341	0.0644	0.04	0.0249	0.3852	0.48
Browsing	TransH	0.0213	0.0438	0.0254	0.018	0.5536	0.7843
	OUR	0.0505	0.1034	0.0606	0.045	0.4502	0.6372
	CF	0.0391	0.2341	0.0642	0.1054	0.3802	0.6161
Browsing	BiNE	0.0398	0.2239	0.0653	0.1034	0.1152	0.2154
recommend	SDNE	0.0274	0.1661	0.0451	0.0585	0.3416	0.5114
Order	TransH	0.0094	0.0636	0.0158	0.0225	0.6056	0.7894
	OUR	0.0404	0.2415	0.0664	0.1014	0.39	0.6209

6. CONCLUSIONS

In this paper, we proposed a framework of hybrid recommendation model for tourism product recommendation. We first addressed that an effective recommendation system should differentiate among users based on user characteristics and developed the module of user segmentation and profiling for recommending specific users. To extract knowledge from implicit feedback, we developed the neighborhood structure of users and products in an interaction network to simul-

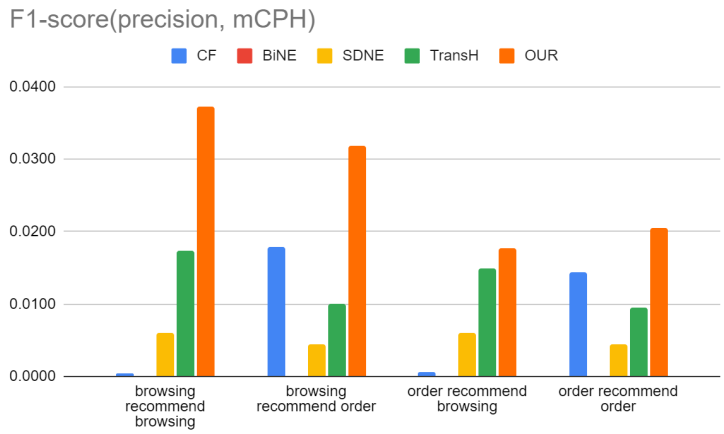

Fig. 9. Effectiveness of hybrid recommendation.

taneously incorporate both browsing and ordering behaviors. The multi-behavior characteristic allows the system to recommend products based on either browsing or ordering, or a combination of both. To deal with the problem of skewed popularity distribution in which cold products are overlooked, we take product information into consideration using metadata to form a knowledge graph based on textual content. By applying embedding algorithms to the multi-behavior interaction network and the product-content knowledge graph, we are able to capture user preferences from implicit feedback as well as the relationships among products. To evaluate the performance of the proposed framework, we conducted experiments on a real-world dataset. Experimental results indicate that the proposed approach outperforms other widely used recommendation systems.

REFERENCES

1. P. Wang, Y. Jiang, C. Xu, and X. Xie, "Overview of content-based click-through rate prediction challenge for video recommendation," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 2593-2596.
2. Q. Chen, H. Zhao, W. Li, P. Huang, and W. Ou, "Behavior sequence transformer for e-commerce recommendation in alibaba," in *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*, 2019, pp. 1-4.
3. M. Naumov, D. Mudigere, H.-J. M. Shi, J. Huang, N. Sundaraman, J. Park, X. Wang, U. Gupta, C.-J. Wu, A. G. Azzolini *et al.*, "Deep learning recommendation model for personalization and recommendation systems," *arXiv Preprint*, 2019, arXiv:1906.00091.
4. B. Suh and E. Koh, "Identifying web pages that are likely to guide browsing viewers to improve conversion rate," US Patent 9,104,767, 2015,

5. L. Xiao, P. Lei, and W. Peng, "Personalized product recommendation based on embedding of multi-behavior network and product information knowledge," in *Proceedings of Conference on Technologies and Applications of Artificial Intelligence*, 2020, pp. 125-130.
6. G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge & Data Engineering*, 2005, pp. 734-749.
7. S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Computing Surveys*, Vol. 52, 2019, p. 5.
8. K. Patel and H. Patel, "A state-of-the-art survey on recommendation system and prospective extensions," *Computers and Electronics in Agriculture*, Vol. 178, 2020, p. 105779.
9. Y. Cai, H. Leung, Q. Li, H. Min, J. Tang, and J. Li, "Typicality-based collaborative filtering recommendation," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 26, 2014, pp. 766-779.
10. Z. Cui, X. Xu, F. Xue, X. Cai, Y. Cao, W. Zhang, and J. Chen, "Personalized recommendation system based on collaborative filtering for iot scenarios," *IEEE Transactions on Services Computing*, Vol. 13, 2020, pp. 685-695.
11. C.-M. Chen, C.-J. Wang, M.-F. Tsai, and Y.-H. Yang, "Collaborative similarity embedding for recommender systems," in *Proceedings of ACM World Wide Web Conference*, 2019, pp. 2637-2643.
12. H. Li, K. Li, J. An, and K. Li, "MSGD: A novel matrix factorization approach for large-scale collaborative filtering recommender systems on gpus," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 29, 2018, pp. 1530-1544.
13. J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, "Recommender system application developments: a survey," *Decision Support Systems*, Vol. 74, 2015, pp. 12-32.
14. H. Wang, F. Zhang, X. Xie, and M. Guo, "DKN: Deep knowledge-aware network for news recommendation," in *Proceedings of ACM World Wide Web Conference*, 2018, pp. 1835-1844.
15. E. Çano, "Hybrid recommender systems: A systematic literature review," *Intelligent Data Analysis*, Vol. 21, 2017, pp. 1487-1524.
16. G. Geetha, M. Safa, C. Fancy, and D. Saranya, "A hybrid approach using collaborative filtering and content based filtering for recommender system," in *Journal of Physics: Conference Series*, Vol. 1000, 2018, p. 012101.
17. S. Yang, M. Korayem, K. AlJadda, T. Grainger, and S. Natarajan, "Combining content-based and collaborative filtering for job recommendation system: A cost-sensitive statistical relational learning approach," *Knowledge-Based Systems*, Vol. 136, 2017, pp. 37-45.
18. Y. Deldjoo, M. F. Dacrema, M. G. Constantin, H. Eghbal-Zadeh, S. Cereda, M. Schedl, B. Ionescu, and P. Cremonesi, "Movie genome: alleviating new item cold start in movie recommendation," *User Modeling and User-Adapted Interaction*, Vol. 29, 2019, pp. 291-343.

19. H. Wang, P. Zhang, T. Lu, H. Gu, and N. Gu, "Hybrid recommendation model based on incremental collaborative filtering and content-based algorithms," in *Proceedings of IEEE 21st International Conference on Computer Supported Cooperative Work in Design*, 2017, pp. 337-342.
20. M. Grbovic, V. Radosavljevic, N. Djuric, N. Bhamidipati, J. Savla, V. Bhagwan, and D. Sharp, "E-commerce in your inbox: Product recommendations at scale," in *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 1809-1818.
21. X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han, "Personalized entity recommendation: A heterogeneous information network approach," in *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, 2014, pp. 283-292.
22. Z.-H. Deng, L. Huang, C.-D. Wang, J.-H. Lai, and P. S. Yu, "DeepCF: A unified framework of representation learning and matching function learning in recommender system," *arXiv preprint*, 2019, arXiv:1901.04704.
23. H. Wang, F. Zhang, M. Zhao, W. Li, X. Xie, and M. Guo, "Multi-task feature learning for knowledge graph enhanced recommendation," in *Proceedings of ACM World Wide Web Conference*, 2019, pp. 2000-2010.
24. H.-H. Chen, C.-A. Chung, H.-C. Huang, and W. Tsui, "Common pitfalls in training and evaluating recommender systems," *ACM SIGKDD Explorations Newsletter*, Vol. 19, 2017, pp. 37-45.
25. T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv Preprint*, 2013, arXiv:1301.3781.
26. T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, 2013, pp. 3111-3119.
27. H.-H. Chen, "Behavior2vec: Generating distributed representations of users' behaviors on products for recommender systems," *ACM Transactions on Knowledge Discovery from Data*, Vol. 12, 2018, p. 43.
28. P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge-Based Systems*, Vol. 151, 2018, pp. 78-94.
29. D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Network representation learning: A survey," *IEEE Transactions on Big Data*, Vol. 6, 2018, pp. 3-28.
30. X. Wang, P. Cui, J. Wang, J. Pei W. Zhu, and S. Yang, "Community preserving network embedding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31, 2017, pp. 203-209.
31. Y. Wen, L. Guo, Z. Chen, and J. Ma, "Network embedding based recommendation method in social networks," in *Companion Proceedings of the Web Conference*, 2018, pp. 11-12.
32. C. Shi, B. Hu, W. X. Zhao, and S. Y. Philip, "Heterogeneous information network embedding for recommendation," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 31, 2018, pp. 357-370.
33. J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Transactions on Knowledge Discovery from Data*, Vol. 1, 2007, pp. 2-es.

34. P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 31, 2018, pp. 833-852.
35. A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 855-864.
36. B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 701-710.
37. D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1225-1234.
38. F. Lu, P. Cong, and X. Huang, "Utilizing textual information in knowledge graph embedding: A survey of methods and applications," *IEEE Access*, Vol. 8, 2020, pp. 92 072-92 088.
39. F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 353-362.
40. A. Bordes, X. Glorot, J. Weston, and Y. Bengio, "Joint learning of words and meaning representations for open-text semantic parsing," in *Artificial Intelligence and Statistics*, 2012, pp. 127-135.
41. A. Bordes, J. Weston, and N. Usunier, "Open question answering with weakly supervised embedding models," in *Proceedings of Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2014, pp. 165-180.
42. A. Bordes, S. Chopra, and J. Weston, "Question answering with subgraph embeddings," *arXiv Preprint*, 2014, arXiv:1406.3676.
43. A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in Neural Information Processing Systems*, 2013, pp. 2787-2795.
44. Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, 2014, pp. 1112-1119.
45. Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, 2015, pp. 2181-2187.
46. J. Sun, "Jieba chinese word segmentation tool," 2012.
47. X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in Artificial Intelligence*, 2009, p. 19.
48. M. Gao, L. Chen, X. He, and A. Zhou, "Bine: Bipartite network embedding," in *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018, pp. 715-724.

Li-Pin Xiao received the master's degree in Institute of Computer Science and Engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2019. Since October 2019, he has been working in KKday as a data scientist. His research interests include data mining and machine learning.

Po-Ruey Lei received the MS degree in Electrical Engineering from University of Southern California, USA in 2000, and the Ph.D. degree in the Department of Electrical and Electronic Engineering, National Defense University, Taiwan, ROC in 2012. Currently, he is an Associate Professor at the department of Electrical Engineering, ROC Naval Academy, Taiwan. His research interests include trajectory data mining, machine learning, and AIS data analysis for anomaly detection and collision avoidance.

Wen-Chih Peng received the BS and MS degrees from the National Chiao Tung University, Taiwan, in 1995 and 1997, respectively, and the Ph.D. degree in Electrical Engineering from the National Taiwan University, Taiwan, in 2001. Currently, he is a Professor at the Department of Computer Science, National Yang Ming Chiao Tung University, Taiwan. Dr. Peng has the best paper award in ACM Workshop on Location-Based Social Network 2009 and the best student paper award in IEEE International Conference on Mobile Data Management 2011. His research interests include mobile computing, network data management and data mining.