Towards Purchase Prediction: A Transaction-based Setting and A Graph-based Method Leveraging Price Information

Zongxi Li[a], Haoran Xie[b,*], Guandong Xu[c], Qing Li[d], Mingming Leng[b], Chi Zhou[e]

[a]*Department of Computer Science, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong SAR*
[b]*Department of Computing and Decision Sciences, Lingnan University, 8 Castle Peak Road, Tuen Mun, New Territories, Hong Kong SAR*
[c]*School of Computer Science and Advanced Analytics Institute, University of Technology Sydney, Broadway NSW 2007, Australia*
[d]*Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong SAR*
[e]*School of Management, Tianjin University of Technology, Tianjin, 300384, P.R. China*

Abstract

Targeting at boosting business revenue, purchase prediction based on user behavior is crucial to e-commerce. However, it is not a well-explored topic due to a lack of relevant datasets. Specifically, no public dataset provides both price and discount information varying on time, which play an essential role in the user's decision making. Besides, existing learn-to-rank methods cannot explicitly predict the purchase possibility for a specific user-item pair. In this paper, we propose a two-step graph-based model, where the graph model is applied in the first step to learn representations of both users and items over click-through data, and the second step is a classifier incorporating the price information of each transaction record. To evaluate the model performance, we propose a transaction-based framework focusing on the purchased items and their context clicks, which contain items that a user is interested in but fails to choose after comparison. Our experiments show that exploiting the price and discount information can significantly enhance prediction accuracy.

Keywords: purchase prediction, graph-based method, e-commerce, transaction-level data

1. Introduction

With the explosive growth of online shopping, predicting customer's behavior has become an essential topic in e-commerce customer management, since it provides decisive instructions for efficient resource allocation, strategy making at sales, and marketing of the online business [1]. Compared with the traditional click prediction and recommendation task of other online

*Corresponding author
Email address: `hxie2@gmail.com` (Haoran Xie)

applications (*e.g.*, video watching, music listening, and social networking), purchase prediction is of more special value to the e-commerce as recommending items that are more likely to be purchased can explicitly boost the revenue of the business. Meanwhile, an accurate demand forecasting is also helpful to the supply chain management by optimizing operational cost.

Although the research related to customer behavior has received attention for a long time [2], the purchase prediction task in the deep learning era is not as prevalent as click prediction, mostly caused by the difficulty of accessing real-world datasets and incomplete information of users and items. Consequently, existing approaches towards purchase prediction mainly depend on user's click data and are based on the learn-to-rank paradigm [3]. The evaluation methods also employ a ranking-oriented setting by comparing all items and selecting top-N as recommendation candidates [4, 5, 6].

We argue that the learn-to-rank framework is not suitable for purchase prediction task and restrict the application under the real scenario. The ranking-based approaches generate outputs in a contrastive manner, so the model cannot directly predict the binary label that indicates if an item will be purchased. Moreover, applying top-N pitching universally to all users overlooks the fact that users have different levels of purchase desire since it evaluates the model performance without taking the click-context per purchase prediction into account. Consequently, the prediction results cannot be employed as a reference for demand forecasting.

To address the problems stated above, we reformulate the purchase prediction problem and propose a new transaction-based modelling and evaluation framework that focuses on each transaction's context instead of the whole click records of each user. The goal is to predict the probability of a specific item being purchased within the context of each transaction. By setting a window for each transaction record, we treat the clicked but not purchased items within a time window as negative samples and drop items out of the windows due to the nature of implicit feedback. A model can be built and evaluated under transaction-based settings to correctly classify the purchased and non-purchased items by explicitly predicting the purchase possibility.

Another challenge of current purchase prediction research is insufficient available information, which narrows researchers' scope down to identifying patterns from click data only. The representations of user and item learned based on click records encode a user's interest or taste towards different items, which is useful for recommending items that the user is also interested in. However, such a preference does not necessarily mean that the user will eventually place an order on the items. A natural intuition is that customers do not buy commodities solely based on interest. On the contrary, they will also consider different aspects and compare it with other similar items before making a final decision. During the decision making of every user, the price and available discount are the most decisive factors. However, to the best of our knowledge, there is no previous study focusing on the pricing strategy due to the lack of relevant information from the dataset.

Graph Neural Network (GNN) has been verified to be influential in various domain applications because of its outstanding structural representation learning ability [7, 8, 9], for example, human pose estimation [10], relation reasoning [11], and multi-modality processing [12]. The graph-based approach has a potential to integrate and exploit various sources and aspects of information in the dataset for identifying some latent relationships for predictions. The node representations learned by a graph-based method embed aggregated information of local neighbourhood and topological structure of graph, and have been exploited in various downstream tasks, such as social recommendation [13], user profile enhancement [14], personalized recommendation [15], and so on. However, we notice that there is no previous work addressing purchase prediction via a graph-based approach. The challenge is possibly due to that the sparse purchase data and anonymized user-item information make the graph construction more difficult. To address this issue, we propose a graph-based model for transaction-based purchase prediction by incorporating the price features of each transaction. We conduct extensive experiments based on a newly available e-commerce dataset with transactional price information provided to demonstrate the significance of pricing and promotion information in the purchase prediction.

The main contributions of this paper are summarized as follows:

- To evaluate the model performance under a more realistic scenario, we define a new form of transaction-based purchase prediction task;
- To the best of our knowledge, we are the first to leverage price and discount features explicitly for purchase prediction task;
- To the best of our knowledge, we are the first to exploit the graph-based model for purchase prediction task;
- We conduct extensive experiments on a real-world e-commerce dataset. The results show that our proposed model incorporating price information enhances prediction performance significantly.

2. Related Work

Past researches regarded the purchases and clicks as implicit feedback [4], because the negative feedback (*e.g.*, viewed but not clicked and clicked but not purchased) was not directly observed. A common solution was to treat all non-purchased items of an user as negative feedback based on the *All Missing As Negative* (AMAN) assumption [16]. However, the AMAN assumption neglected the goal-specificity and temporal range of user's intent [17], which means the interactions that were chronologically close to a purchase can be more informative towards understanding the user's decision making before placing an order. In many recommendation-related application domains, multiple user-item interactions of different types can be recorded over time. Therefore, numerous works have leveraged time information, such as session partition and timestamp, to build session-based sequence-aware recommender system by enriching individual user models in the recommendation process [18]. Relevant works are surveyed in [19, 20]. Jannach and Zanker (2018) comprehensively reviewed collaborative filtering models in session-based recommendation scenarios [21]. These approaches adopt pairwise ranking loss function and produce Top-*N* ranked items as output.

Click records with simple user-item interaction have been exploited for various tasks such as click-through rate (CTR) prediction in online advertising [22] and user intent prediction [23]. On purchase-relevant datasets, analysis over click records has been the most fundamental component of purchase prediction model. Several researches focusing on predicting customer's purchase behavior in e-commerce have been made based on click records. Iwata *et al.* (2009) [24] adopted a new topic model for tracking time varying consumer purchase behavior , in which consumer interests and item trends change over time. Liu *et al.*(2016) [25] proposed a repeat buyer prediction model on sales data of "Double 11" dataset from Tmall, which utilizes different semantic features generated from user-merchant interactions. Park *et al.* (2020) [4] presented a model-based framework that leverages the historical click records of users to compensate for the missing user-item interactions of purchase records, *i.e.*, non-purchased items.

Apart from mining the click logs, there have been efforts leveraging knowledge of different forms. Zhang and Pennacchiotti (2013) [26] predicted user's purchase behavior on e-commerce websites employing the user's social media profile. Zhao *et al.* (2016) [27] focused on brand-based purchase prediction using platform-specific feature groups, including click features, purchase features, and collect-and-cart features. Jannach *et al.* (2017) [28] discussed the role of discounts in session-based item recommendation in e-commerce, where items in promotion were marked as being discounted. They found, by doing this, recommended items can be more attractive to users and lead to higher conversion rates. Chen *et al.* (2019) [29] presented a context-based purchase prediction framework based on user's sequential searching and clicking actions to improve purchase prediction accuracy, in which the connection between searching and clicking were exploited to investigate user's ultimate intention. Zhao *et al.* (2020) [30] examined and interpreted various user inaction factors in recommendation process.

All works stated adopted a learn-to-rank approach to build and test their proposed methods, and none of them considered that prices and discounts are varying over time and examined the effect of price factor in purchase prediction due to the availability of price information. Moreover, no previous effort has leveraged graph-based method for purchase prediction task.

The remainder of this article is organized as follows. In Section 3, we formally define the proposed transaction-based purchase prediction problem. In Section 4, we present the proposed graph-based model for purchase prediction. In Section 5, we describe our experiments on an e-commerce dataset and present the results. The summary and future research directions of this paper are discussed in Section 7.

3. Problem Statement & Formulation

Symbol	Description
$\mathcal{G}, \mathcal{V}, \mathcal{E}$	Graph, Vertices, and Edges
$\mathcal{U}, \mathcal{I}$	Set of Users, set of Items
n, m	Number of users and items
$\mathcal{T}$	Set of all Transactions
$t_{<u,i>}$	Transaction with user u and item i involved
$\mathrm{d}_{t<u,i>}$	Tuple of price and discount of transaction $t_{<u,i>}$
$\mathrm{T}_{t<u,i>}$	Timestamp of transaction $t_{<u,i>}$
$\mathbf{p}_u$	Items purchased by user u
$\mathbf{c}_u$	Items clicked by user u
$\mathbf{c}^{(+/-)}_{t<u,i>}$	Items clicked by user u before transaction $t_{<u,i>}$, the superscription denotes the label
K	Dimension of "interest" embeddings
M	Dimension of "decision" embeddings
$\boldsymbol{\alpha} \in \mathbb{R}^{n \times K}$	User embedding matrix
$\boldsymbol{\beta} \in \mathbb{R}^{m \times K}$	Item embedding matrix
τ	Threshold of purchase prediction

Table 1: Notation

We first introduce the notations used throughout the paper (as shown in Table 1). Let $\mathcal{U}$ and $\mathcal{I}$

denote the set of users and items, respectively, and n, m be the number of users and items. Each user or item is a vertex of the graph G, thus we have $(n + m)$ nodes on the graph. All transaction records of users in $\mathcal{U}$ on items in $\mathcal{I}$ are denoted by $\mathcal{T}$. Each transaction record $t_{<u,i>}$ stands for user u buying item i, additionally with a timestamp $\mathrm{T}_{t<u,i>}$ and a tuple with six elements encoding price and discount information, which concretely include original unit price, final unit price, and discount quantity of four promotion strategies, *i.e.,* direct discount, quantity discount, bundle discount, and coupon discount. $\mathbf{p}_u$ and $\mathbf{c}_u$ denote the sets of items purchased and clicked by user u, respectively.

Especially, we use $\mathbf{c}_{t<u,i>}$ to signify items clicked by user u before a transaction $t_{<u,i>}$, which includes items purchased within the transaction (as user must click to take further action), denoted as $\mathbf{c}^{(+)}_{t<u,i>}$, and items clicked-but-not-bought ahead of the transaction, denoted as $\mathbf{c}^{(-)}_{t<u,i>}$. For transaction-based purchase prediction task, we regard the purchased item(s) as positive sample(s) with label **1** and not-purchased items as negative samples with label **0**. Intuitively, the negative samples in $\mathbf{c}^{(-)}_{t<u,i>}$ can be concluded into two types: 1) items that user u is highly interested in, but decide not to purchase after comparing from different aspects, such as price and available discounts; and 2) items that user u casually click through without clear purchase intention, which are less relevant to the purchased item(s). To simplify the scenario without loss of generality and accelerate both training and testing by making the model parallelizable, we truncate the $\mathbf{c}_{t<u,i>}$ into item sequence of a fixed length according to their reversed click order, which should contain most of the highly-relevant items and some of the low-relevant items. We believe that 10 is a suitable length because the ratio of records of purchased item to click records is 1:9. To be specific, we have $455,383$ records of items being purchased and $3,986,902$ click records regarding those purchased items, which leads to an average of 8.755 clicks before a transaction (note that users with less than 20 click records and items that had never been purchased are removed). For notation simplicity, we use $\mathbf{c}_{t<u,i>}$ to denote the fix-length item sequence in the rest of this paper.

We formally define the research problem of this paper. Under the settings stated above, the transaction-based purchase prediction task is to correctly predict the label of each item in $\mathbf{c}_{t<u,i>}$ according to their possibilities of getting purchased. The prediction performance is evaluated by using widely-adopted evaluation metrics, *i.e.*, accuracy and F1 score.

4. Methodology

In this section, we first introduce the price information exploited in the proposed model, and then demonstrate our two-step framework in terms of its components: in the first step, an undirected graph is constructed based on click data for learning a general representation of both users and items; the proposed purchase prediction model incorporating price and discount information as a new feature is presented in the second step.

4.0. Step 0: Discount as Features

Given a transaction record $t_{<u,i>}$, we can retrieve the corresponding information about product pricing and promotional activities. As shown in Table 2, the *original_unit_price* is the list price of the item, which is the same for all customers at any given time but can change from time to time; the *final_unit_price* is the actual price paid by the customer, which can vary among customers even at the same time owing to different discounts or promotions. The gap between the original price and the final price is the summation of all discounts associated with the item, which are *direct discount*, *quantity discount*, *bundle discount* and *coupon discount*. Direct discount is a

Field	Description
order_ID	Unique identifier for each transaction
user_ID	Unique identifier for each user
item_ID	Unique identifier for each item
order_time	Specific time at which the order gets placed (format: yyyy-mm-dd HH:MM:SS)
original_unit_price	Original list price
final_unit_price	Final purchase price
direct_discount	Discount due to direct price deduction
quantity_discount	Discount due to purchase quantity
bundle_discount	Discount due to bundle promotion
coupon_discount	Discount due to coupon used by customer

Table 2: Description of each transaction record

promotion strategy that the seller offers an unconditional price cut regarding the list price, which reflects the reduction in the original price stated on the product detail page. Quantity discount is a promotion strategy that the seller provides an extra discount to entice the customer to buy more, which can have different forms including “get an RMB 100 discount if the total price of order exceeds RMB 199” or “buy 3 to get an RMB 10 discount on each”. Bundle discount means that a discount is provided if the customer purchases a pre-specified collection of items. Coupon discount allows customers to use coupons obtained in advance to gain extra price reduction.

Exploiting the order details elaborated above, we manipulate the price and discount tuple using the feature engineering technique to generate the transaction-based feature vector containing the following elements:

- Price-aware attributes, a six-dimension vector, of which the attributes are the proportion of final price, total discount and discounts of four promotion strategies regarding the original price, respectively.
- Discount-aware attributes, a four-dimension vector, of which the attributes are the proportion of discounts of four promotion strategies regarding the total price reduction.

Intuitively, the price-aware attributes describe a pricing profile by highlighting the overall discount percentage, and the discount-aware attributes depict the promotion strategies by using the weight of each strategy. We concatenate these two vectors to form the final feature vector $\mathrm{d}_{t<u,i>}$. We visualize a virtual example to better explain the process to generate $\mathrm{d}_{t<u,i>}$ in Figure 1.

By doing this, we can construct the price features for all purchased items. However, we cannot explicitly access the price information of those clicked-but-not-bought items, because click records do not include the promotion information of the item when it is clicked. An alternative is to conclude a price pattern based on all orders of an item. It is notable that the price attribute of an item, including its original price and final price, can fluctuate throughout the period, and the promotion strategies can be different even within the same day. Considering the fact that $\mathrm{d}_{t<u,i>}$ is a transaction-specific feature regarding such a dynamic situation, we hope the price feature $\mathrm{d}^{i'}_{t<u,i>}$ for $i \in \mathbf{c}^{(-)}_{t<u,i>}$ can also be transaction-specific. To address this issue, we collect the transaction

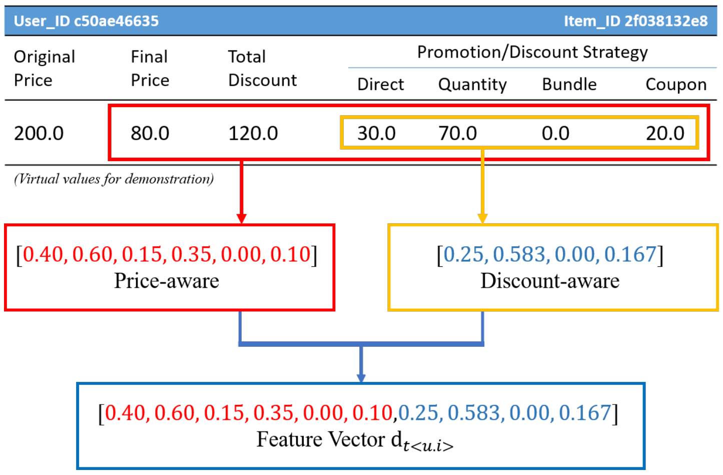


Figure 1: A virtual example of generating feature vector of discount information. Given a transaction record, we extract price-aware attributes and discount-aware attributes accordingly and then concatenate both vectors to form the feature vector.

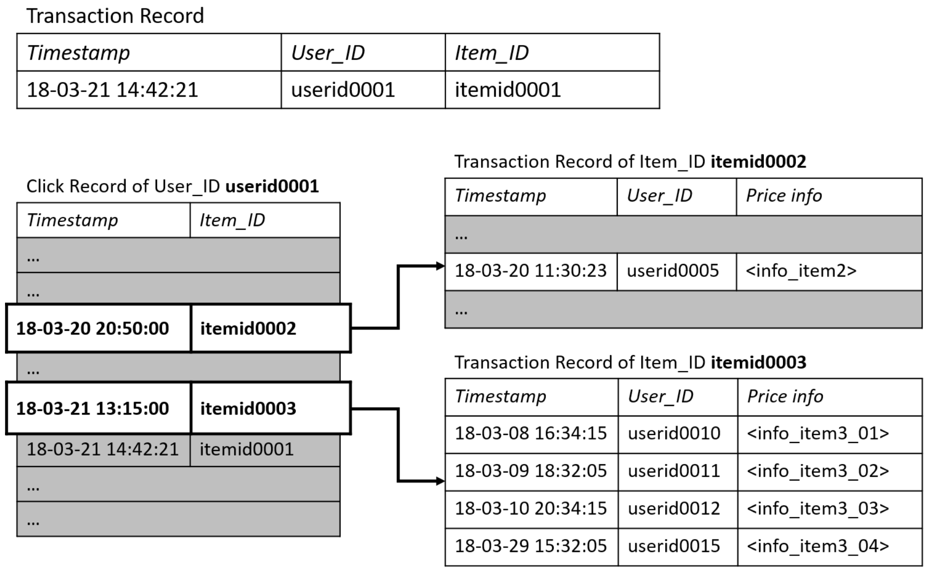
records for items $i \in \mathbf{c}^{(-)}_{t<u,i>}$ that are chronologically close to the time $\mathrm{T}_{t<u,i>}$ that $t_{(u,i)}$ got placed, and retrieve the closest record(s) to generate $\mathrm{d}^{i'}_{t<u,i>}$ for item i'. If there are multiple transactions records with different pricing patterns within a reasonable time window, say one hour or one day, we calculate the average values of the original price and final price, and pick the mode values of promotional discounts to generate $\mathrm{d}^{i'}_{t<u,i'>}$ for item i'. If the closest record is more than one day away, we will generate the price feature based on all historical transaction records following the same approach with the previous situation. Note that items without any purchase record are removed.

In Figure 2, we present a virtual example to illustrate the process of retrieving transactional price information for `itemid002` and `itemid003`, which are clicked-but-not-purchased items given a transaction record involving `itemid001`. For `itemid002`, we can find a relevant purchase record on the same day. Therefore, we can exploit `<info_item2>` to generate $\mathrm{d}^{i'}_{t<u,i'>}$ for `itemid002`. As for `itemid003`, we find all purchase records are chronologically far from the timestamp. Thus, we generate $\mathrm{d}^{i'}_{t<u,i'>}$ for `itemid003` based on all available records.

To this end, we successfully produce price feature for all items, which will be utilized in Step 2 (Section 4.2) for purchase prediction. Next, we exploit a graph-based node embedding approach to map each user and item into a higher-dimensional space as latent representation.

4.1. Step 1: General Embedding Learning

Clicks are the primary user-item interaction, which are informative on profiling user-item relationship and modeling the preference and interests of a user. Especially, frequently repeated clicking on a certain item is a telltale signal that the user is more likely to buy this item than other items. To exploit such information, we construct the graph with weighted edges encoding click frequency. We define the graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \mathcal{U} \cup \mathcal{I}$ is the set of vertices, and $\mathcal{E}$ is the set of edges between the vertices. Each edge $e \in \mathcal{E}$ represents a user-item pair (u, i) for user u and item i, which is assigned with a weight $w_{u,i} > 0$ indicating the relatedness. More concretely,



Transaction Record

Timestamp	*User_ID*	*Item_ID*
18-03-21 14:42:21	userid0001	itemid0001

Click Record of User_ID **userid0001**

Timestamp	*Item_ID*
…	
…	
18-03-20 20:50:00	**itemid0002**
…	
18-03-21 13:15:00	**itemid0003**
18-03-21 14:42:21	itemid0001
…	
…	

Transaction Record of Item_ID **itemid0002**

Timestamp	*User_ID*	*Price info*
…		
18-03-20 11:30:23	userid0005	<info_item2>
…		

Transaction Record of Item_ID **itemid0003**

Timestamp	*User_ID*	*Price info*
18-03-08 16:34:15	userid0010	<info_item3_01>
18-03-09 18:32:05	userid0011	<info_item3_02>
18-03-10 20:34:15	userid0012	<info_item3_03>
18-03-29 15:32:05	userid0015	<info_item3_04>

Figure 2: A virtual example of retrieving transactional information for clicked items.

for a user u, the weight of edge connecting with item $i^k \in \mathbf{c}_u$ is calculated as

$$w_{u,i^k} = \frac{1}{1 + e^{-\epsilon \times f_{u,i^k}}}, \tag{1}$$

where f_{u,i^k} is the normalized clicks on item i^k,

$$f_{u,i^k} = \frac{\text{\# of clicks on } i^k}{\sum_j \text{\# of clicks on } i^j \text{ for } i^j \in \mathbf{c}_u}, \tag{2}$$

and ϵ is a modulating hyperparameter controlling the range of output value by adjusting the curve of sigmoid function shown in Figure 3.

Eqn. 1 maps the value of edge weight into the range of $[0.5, 1)$, which can effectively avoid the situation that a less reasonable representation is learned when the edge weight is too small if a user has enormous click records. Moreover, the graph $\mathcal{G}$ is undirected, which implies that we have $(u, i) \equiv (i, u)$ and $w_{u,i} \equiv w_{i,u}$.

Different from social network graphs where a depth-first searching model can work well, in practice, the click graph focuses on the local pairwise relationship between the vertices that share similar neighbours. We thus adopt a breadth-first searching method to learn the node presentation. We assume that users having more co-clicked items tend to share similar interests or tastes as shown in Figure 4. Therefore, we follow the same way as in [31] to embed the graph into a low-dimensional space with *second-order* proximity network structure preserved. In general, the *second-order* proximity between a pair of vertices (v_i, v_j) in the graph measures the similarity between their neighbourhood network structures and is determined by the pairwise similarity of *first-order* proximity. Vertices with similar distributions over the neighbourhood vertices are assumed to be similar. In order to encode the *second-order* proximity information,

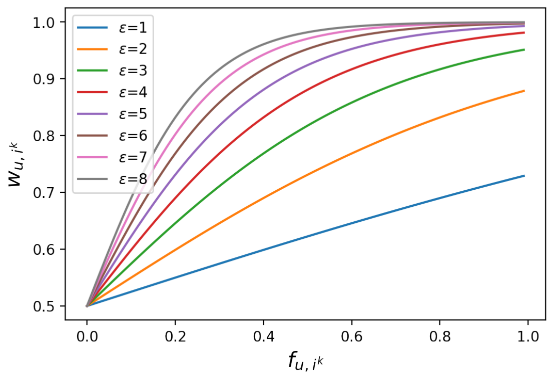


Figure 3: Relationship between w_{u,i^k} and f_{u,i^k} in Eqn. 1 with different values of ϵ

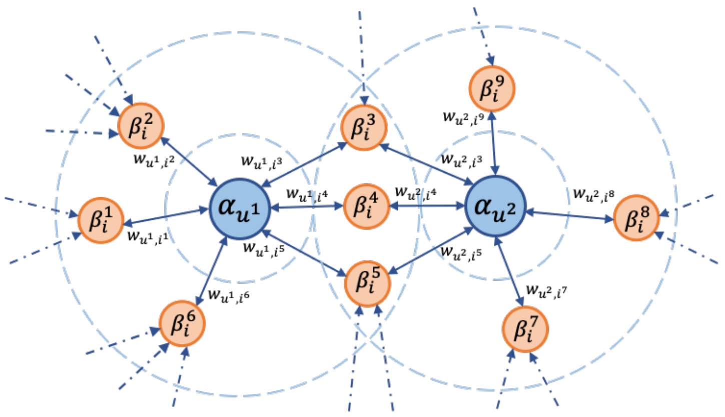


Figure 4: Graph construction in Step 1, where $w_{u,i}$ is the weight of edge (u, i). In this case, user u^1 have three co-clicks with user u^2, where we assume that u^1 may share similar interests with u^2, and such a *second-order* structural information is preserved in our model.

each vertex is assigned with a context vector. Given an edge (i, j), we define the probability of context vertex v_j conditioned on vertex v_i as:

$$p(v_j|v_i) = \frac{\exp(\mathbf{v}'^{\mathrm{T}}_j \cdot \mathbf{v}_i)}{\sum_{k=1}^{|V|} \exp(\mathbf{v}'^{\mathrm{T}}_k \cdot \mathbf{v}_i)}, \tag{3}$$

where $\mathbf{v}'$ is the representation if the vertex is treated as context, and $|V|$ is the number of context vertices. The objective is to minimize the KL-divergence between the conditional distribution of the contexts $p(\cdot|v_i)$ and the empirical distribution $p(\cdot|v_i)$, which is defined as $p(v_j|v_i) = \frac{w_{i,j}}{d_i}$, where d_i is the degree of vertex i, by minimizing the objective function:

$$O_1 = - \sum_{(i,j)\in\mathcal{E}} w_{i,j} \log p(v_j|v_i) \tag{4}$$

However, the computational cost of the loss function in Eqn. 4 is expensive as the summation in the denominator needs to cover the entire set of vertices to compute the conditional probability. To address this issue, a negative sampling method, Noise Contrastive Estimation (NCE) [32], is employed to approximate the loss term. The negative samples are drawn from the same discrete distribution by utilizing the alias table method [33] according to the weights of the edges, which is of $O(1)$ complexity. Given the edge (i, j), the loss term is approximated by

$$O_1 = \log \sigma(\mathbf{v}'^{\mathrm{T}}_j \cdot \mathbf{v}_i) + \sum_{n=1}^{N} \mathbb{E}_{v_n \sim P_n(v)} \left[\log \sigma(-\mathbf{v}'^{\mathrm{T}}_n \cdot \mathbf{v}_i) \right], \tag{5}$$

where σ is a sigmoid function, N is the number of negative samples, and following the suggestion in [32], the noise distribution $P_n(v)$ is set to be the $3/4$rd power of the degree of vertex v_i.

By optimizing the model, we obtain $\boldsymbol{\alpha} \in \mathbb{R}^{n \times K}$ and $\boldsymbol{\beta} \in \mathbb{R}^{n \times K}$ as representations for users and items, respectively. Note that the general representation is obtained by exploiting click frequency information. Thus, no purchase information is included in this pretraining phase, and it is fair to use generated embeddings in the next step prediction task.

The time complexity of transiting through the graph is $O(nm)$, i.e., quadratic time. However, due to the sparsity of the graph, the number of each users interactions can be treated as a constant. Therefore, the complexity of training node embedding is linear time $O(n)$.

4.2. Step 2: Purchase Prediction Model

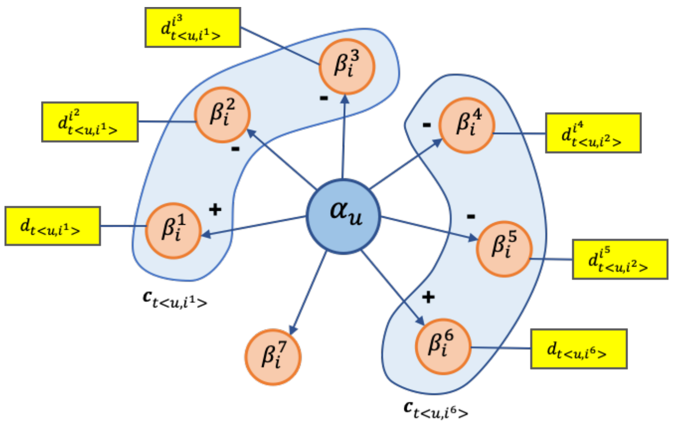
In this subsection, we introduce the proposed model for transaction-based purchase prediction. Figure 5 demonstrates the general framework of this step. In this step, the model is built to extract latent relationship in a transaction by exploiting the *latent first-order* proximity. Compared with *second-order* proximity, the *first-order* proximity implies the similarity of two vertices that are pair-wisely connected. The proposed model measures the *first-order* proximity in a latent space to incorporate price features. The motivation is concluded as follows:

In the first step, users and items are mapped into a shared "interest" space according to their historical interactions. If the similarity between the embedding vectors of a users u and an item i is high, we can assert that the user u is interested in the item i and would click item i if i is recommended. However, we cannot simply deduce that the user u is more likely to buy an item i without other substantial evidence. Intuitively, the decision-making process of purchasing an item involves multiple factors, which include explicit factors, such as expected values and available discount options, and implicit factors, such as usefulness, necessity, and practical applicability. We argue that both explicit factors and implicit factors can contribute to the final decision, together with the user's interest. To capture such dependency relationship in purchase prediction, we subsequently map price features and the embeddings of users and items into another latent "decision" space as an analogue of exploiting implicit factors. By doing this, the representations in the "decision" space can better depict the user's desire to buy a specific item.

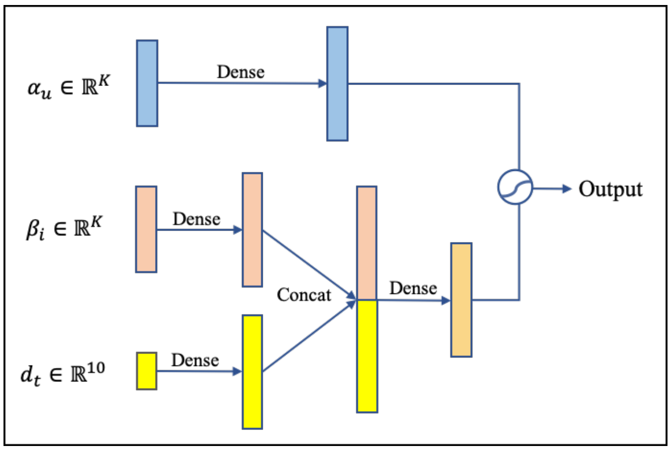
Under the transaction-based setting, as shown in Figure 5a, the model updates weights regarding the inputs relevant to each transaction at each training step. The idea behind is similar to NCE with negative sampling. However, in the proposed approach, we do not need to sample the negative samples because items in $\mathbf{c}^{(-)}_{t<u,i>}$ can work as natural negative samples.

4.2.1. Model Input

Given a transaction record $t_{<u,i>}$, the input of our model consists of the "interest" embedding $\boldsymbol{\alpha}_u \in \mathbb{R}^K$ of user u, the "interest" embedding $\boldsymbol{\beta}_i \in \mathbb{R}^K$, and the price feature $\mathrm{d}_{t<u,i>} \in \mathbb{R}^{10}$ of each



(a) An illustration of transaction-based approach on the subgraph.



$\alpha_u \in \mathbb{R}^K$
Dense
$\beta_i \in \mathbb{R}^K$
Dense
Concat
Dense
$d_t \in \mathbb{R}^{10}$
Dense
Output

(b) The model proposed for purchase prediction incorporating price information.

Figure 5: A generic framework for Step 2.

item $i \in \mathbf{c}_{t<u,i>}$. Since the input items at each step are of the same length, the model training can be parallelized.

4.2.2. Model Architecture

To extract purchase-related features of a user, we map the "interest" embedding of the user to the M-dimension "decision" space through a fully-connected layer as follows:

$$\mathbf{H}_u = f(\mathbf{W}^u \cdot \alpha_u + \mathbf{b}^u), \tag{6}$$

where $\mathbf{H}_\alpha \in \mathbb{R}^M$, and $f(\cdot)$ is the activation function. In this work, we adopt ReLU as the activation function.

Because the price is as an essential attribute of an item, we merge price feature and item information first and then map the combined representation to the "decision" space. By doing this, the "decision" embedding of an item is integrated with explicit factors when simulating the decision making process. Concretely, the "interest" embedding and price vector of an item are projected into a shared space for feature merging,

$$\begin{aligned}\mathbf{H}_{\beta} &= f(\mathbf{W}^{\beta} \cdot \boldsymbol{\beta}_i + \mathbf{b}^{\beta})\\ \mathbf{H}_{d} &= f(\mathbf{W}^{d} \cdot \mathrm{d}_{t<u,i>} + \mathbf{b}^{d})\\ \mathbf{H'}_{i} &= \mathbf{H}_{\beta} || \mathbf{H}_{d},\end{aligned} \tag{7}$$

where $\mathbf{H}_{\beta} \in \mathbb{R}^{M}$, $\mathbf{H}_{d} \in \mathbb{R}^{M}$, and $\mathbf{H'}_{i} \in \mathbb{R}^{2M}$, and $||$ is the concatenate operation. By mapping the low-dimension price vectors to a higher-dimensional space, the model can extract essential and latent features that are decisive to the purchase prediction. Then, we map the concatenated representation $\mathbf{H'}^{i}$ to the "decision" space,

$$\mathbf{H}_{i} = f(\mathbf{W'}^{i} \cdot \mathbf{H'}_{i} + \mathbf{b'}^{i}), \tag{8}$$

where $\mathbf{H}_i \in \mathbb{R}^{M}$. Ultimately, the model outputs the probability of an item being purchased by a user by computing the joint probability between the user and the item as follows:

$$p_{<u,i>} = \sigma(\mathbf{H}_u^{\top} \cdot \mathbf{H}_i), \tag{9}$$

where $\sigma(\cdot)$ is a sigmoid function.

We will predict that the user u is likely to buy the item i if $p_{<u,i>} > \tau$, where τ is the threshold value [1]. Otherwise, user u will not place an order.

Note that weights of $\mathbf{W}^{\beta}$, $\mathbf{W}^{d}$, and $\mathbf{W'}^{i}$ are shared among all items in the same input. Moreover, the "interest" embeddings of both users and items are set as "not trainable", which means only the weights of fully-connected layers are tuned during training so that the learned classifiers can be robust even if a user is not in the training set.

4.2.3. Loss and Optimization

To explicitly model the user-item relationship in the "decision" space, a straightforward way is to minimize the following objective function

$$O_2 = -\sum_{u \in \mathcal{U}} \sum_{i \in \mathbf{p}_u} \log p_{<u,i>}. \tag{10}$$

To optimize the objective in Eqn. 10 and avoid a trivial solution that produces infinite loss term, we can utilize the negative sampling approach similar in Eqn. 5. We regard the clicked-not-purchased items as natural negative samples, and the model can therefore be trained by minimizing the following objective function without extra sampling procedure:

$$O_2 = \sum_{t \in \mathcal{T}} \left[\sum_{i^{(+)} \in \mathbf{c}_t^{(+)}} \log \sigma(p_{<u,i^{(+)}>}) + \sum_{i^{(-)} \in \mathbf{c}_t^{(-)}} \log \sigma(-p_{<u,i^{(-)}>}) \right], \tag{11}$$

where $\mathbf{c}_t^{(+)}$ and $\mathbf{c}_t^{(-)}$ (with $< u, i >$ omitted for concise representation) represent the purchased items, which are treated as positive samples, and clicked-not-purchased items, which are treated as negative samples, within a transaction $t_{<u,i>}$, respectively. The model is optimized by Adam optimizer. Both the training and testing processes are parallelized by feeding data in batches.

The time complexity of model training is linear time and can be further reduced because of the parallelizable input.

[1] In our experiments, we found the model produces best results when $\tau = 0.4$

5. Experiment

5.1. Dataset

Since our research question is about purchase prediction exploiting price information, it is necessary to obtain both transaction data and the price records associated with each transaction from the dataset. To the best of our knowledge, only a newly available dataset provided by JD.com [34] [2] can fulfill our requirements.

JD.com is China's largest e-commerce retailer, with over 320 million annual active customers. The whole dataset contains click records of over 2.5 million customers ($457,298$ of among them placed orders) on $30,000$ items (from one product category) during March in 2018. Though key identification information such as user ID and item ID is anonymized to ensure confidentiality, each user and item has a unique identifier as shown in Table 2, so as the interaction history of a user, including browsing and purchasing, and sales record of an item, such as pricing strategies and promotion activities, are trackable.

5.2. Data Processing

5.2.1. For Step 1

We present the statistics of the raw data in Figure 3. To keep the size of the click data manageable, we first remove click records of users who have never placed an order, and in total, we have $1,200,000$ user-item pairs. Then we assign value to each user-item click pair via Eqn. 1 and Eqn. 2, which forms the click graph used for generating embeddings in Step 1.

Raw Dataset	
Characteristic	Value
Users	$2,500,000$
Items	$31,868$
Clicks	$20,214,515$
Purchases	$455,383$

Table 3: Characteristics of raw dataset.

5.2.2. For Step 2

In this work, we focus on the effect of price information on purchase prediction instead of addressing the sparsity issue. Therefore, we select active users by removing less-active users who have no more than 20 click records and have not purchased any item. By doing this, we have $54,756$ transaction records involving $49,142$ users and $8,800$ items. For each transaction, the data that used for model training and evaluation is generated by sorting relevant items according to timestamp and retrieving promotion information for each item respectively through the process stated in Section 4.0. Specifically, in this work, we regard each purchase as an independent event, and the data samples in the generated transaction dataset are also independent with each other. Thus, we can feed data to the model in parallel, and can conduct 10-fold cross validation. We present statistics of processed data in Table 4. An example of the final format of the transaction data after the pre-processing is shown in Table 5.

[2]More details via `https://shen.ieor.berkeley.edu/MSOMData.pdf`

Processed Dataset	
Characteristic	Value
Users[†]	49, 142
Items	8, 800
Purchase record	54, 756
# of purchased items	62, 939
# of clicked-not-purchased items	484, 621

[†] active users w/ at least 20 clicks and 1 purchase

Table 4: Characteristics of processed dataset.

user c50ae46635

	ID	Promotion	Label
items	2f038132e8	[188.9, 78.0, 30.9, 80.0, 0.0, 0.0]	**1**
	3418d59e22	[28.0, 15.0, 0.0, 13.0, 0.0, 0.0]	**0**
	904fbf8b97	[69.8, 35.4, 4.0, 32.0, 0.0, 0.0]	**0**
	416aed740e	[162.9, 11.6, 153.0, 0.0, 0.0, 6.0]	**0**
	5a745fb2ca	[109.0, 57.3, 0.0, 50.0, 0.0, 0.0]	**0**
	3c79df1d80	[59.7, 44.0, 15.0, 0.0, 0.0, 0.0]	**0**
	38d636d2a6	[127.3, 106.32, 21.0, 0.0, 0.0, 0.0]	**0**
	5f58bfd286	[79.8, 41.7, 4.0, 37.0, 0.0, 0.0]	**0**
	ac61f4e10e	[139.8, 72.6, 7.0, 66.0, 0.0, 0.0]	**0**
	3a64d9667a	[94.8, 53.5, 0.0, 47.0, 0.0, 0.0]	**0**

Table 5: An example of generated transaction data.

5.3. Baselines

To verify the effectiveness of the proposed model, we mainly compare different node embedding learning methods and different variants of the proposed model for ablation study due to very limited deep learning models for purchase prediction.

We compare the proposed graph-based model with traditional **Matrix Factorization** [35], which constructs latent representations for users and items based on a click matrix, and the following graph node embedding methods:

DeepWalk [36] samples a sequence of nodes using random walk via depth-first searching, and applies skip-gram model to learn the node representations.

LINE [31] embeds nodes into representation with preserving *first-order* and/or *second-order* proximity. Specifically, **LINE**-1^{st} with *first-order* proximity, and **LINE**-*all* with both *first-order* and *second-order* proximities are chosen form comparison. Also, we also compare our model with the above two LINE models under the binary edge setting and weighted edge setting.

Node2Vec [37] is an extension of DeepWalk with the biased random walk, which can control walking between depth-first strategy and breadth-first strategy using two hyperparameters.

LightGCN [38] is a simplified GCN-based recommender model preserving the neighbourhood aggregation component of GCN only. The LightGCN is a ranking-based collaborative filtering model and is optimized by *Bayesian Personalized Ranking* loss.

We implement the baseline methods described above to obtain different user/item embeddings and test these embeddings in the prediction model, respectively.

To examine the effectiveness of different components of the proposed model, we compare the following variants:

Ours (w/o price info) is a variant with price information removed. This variant computes purchase probability based on projected embeddings of users and items, which is explicitly trained with pairwise user-item interaction.

Ours (w/ general price info) is a variant that neglects the price fluctuation of each item during the whole period, which simulates the situation where we can only access the general pricing information of each item, instead of each transaction. The general price feature of an item is obtained by calculating the average price values of all transactions that the item is involved. This variant can explain why the transaction-based price information is essential.

Ours (random initialization) is a variant that replaces Step 1 with random embedding initialization.

Ours (embedding trainable) is a variant with embedding vectors trainable during training.

Ours (w/o discount-aware) and **Ours (w/o price-aware)** are variants exploiting only price-aware attributes and only discount-aware attributes of price features, respectively.

5.4. Evaluation

In this section, we introduce the evaluation method for the proposed purchase prediction model.

5.4.1. Evaluation Metrics

We evaluate the model performance by using the following metrics.

- **F1 score** assesses classification performance in a comprehensive manner as it measures both precision and recall as a whole:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}. \tag{12}$$

  We report *Macro*-average results in this paper.

- **Accuracy** measures how many instances are correctly classified among all instances.

- **T-test** reveals how significant the improvements are against the second-best baseline. We report *p*-value of the proposed model compared with baselines for each trial.

5.4.2. Classification Results on Positive Label

Even though we have limited the number of negative samples by truncating the length of item sequence, the positive samples are still too sparse in the dataset. As a consequence, there is a trivial situation that a failed model can achieve a considerable overall accuracy by predicting every instance as negative. Therefore, we also report accuracy and F1 results of the positive label.

5.4.3. Cross Validation

The JD.com datasets have no standard train/test split. Thus, we employ ten-fold cross-validation as one trial to test the model performance. We run five trials and report average scores. Based on the five-trial results, we conduct t-test and report p-values.

Model	**Hyperparam.**	**Value**
DeepWalk	Walk length	10
Node2Vec	Walk length	10
	Return parameter p	0.8
	In-out parameter q	0.4
LINE-based models	Modulating factor ϵ	4

Table 6: Hyperparameter settings

5.5. Grid Search and Hyperparameter Setting

In this work, we set the dimension of embeddings $K, M = 100$. The prediction model is trained with a batch size of 100. For models with hyperparameters, such as DeepWalk (walk length), Node2Vec (walk length, return parameter p, and in-out parameter q), and LINE-based models (modulating factor ϵ), we conduct grid search on a validation set, which is a 20% subset of the training set. We find the settings in Table 6 achieve the best performance on the validation set.

5.6. Experiment Results

Results of comparing different pretrain methods and results of the ablation study are listed in Table 7 and Table 8, respectively. In general, our proposed model achieves the best accuracy and F1 score on the purchase prediction task.

Especially, embeddings generated by the proposed method achieves the best results. Therefore, we conduct ablation experiments based on this pretrain model. Comprehensive discussions are given in Section 6.

Pretrain Model	Overall		Label **1**	
	Accu.(%)	F1 (%)	Accu. (%)	F1 (%)
Random init.	89.74	64.05	27.54	37.45
MF	93.22	83.56	70.61	70.28
DeepWalk	93.03	83.39	70.81	70.42
Node2Vec	93.72	83.00	69.42	68.83
Light GCN	92.86	81.03	67.38	68.31
LINE-1st (bin)	93.09	82.27	70.45	70.02
LINE-1st (weighted)	93.16	83.15	71.94	70.28
LINE-*all* (bin)	93.11	83.39	72.83	70.85
LINE-*all* (weighted)	94.04	84.24	74.49	71.43
Ours (bin)	94.14	84.38	74.68	72.83
Ours (weighted)	**94.33**	**84.81**	**76.43**‡	**72.54**†

†: $p < .01$, ‡: $p < .005$.

Table 7: Experiment results of the proposed model with different pretrain methods. Reported results are obtained with a threshold $\tau = 0.4$.

Models	Overall		Label **1**	
	Accu.(%)	F1 (%)	Accu. (%)	F1 (%)
Always negative predictor	88.60	48.59	0.00	0.00
Highest discount predictor	82.41	53.97	16.70	17.79
w/o price info	88.62	48.92	26.28	32.21
w/ general price info	88.63	49.63	49.51	55.84
w/o discount-aware	93.47	82.69	68.32	68.99
w/o price-aware	92.84	78.80	52.98	61.65
Embedding trainable	93.45	82.16	68.24	68.05
Ours	**94.33**	**84.81**	**76.43**	**72.54**

Table 8: Experiment results of ablation study

6. Discussion

6.1. Effect of Different Pretrain Methods

As shown in Table 7, we compare the prediction performances of the proposed model using different pretrain methods. The first finding is that, due to label imbalance, the differences in overall evaluation between models are very marginal, indicating the necessity of evaluating the model based on each label. Compared with depth-first searching-based methods (DeepWalk and Node2Vec), the breadth-first searching-based methods, including MF and LINE-based models, achieve better performance, which validates the hypothesis that the neighbourhood information is more useful to the click graph.

Moreover, the LINE-based models preserving *second-order* proximity (Ours and LINE-*all*) show improvement when compared with LINE-1^{st}, which focuses on *first-order* proximity only. We conclude that this is because the neighbourhood feature, such as co-click activity, is more informative than the pairwise interaction.

The results of training classifier with embedding vectors generated by non-weighted graph and weighted graph demonstrate the effectiveness of the frequency feature of a user's click-through behavior. We also train the classifier on the randomly initialized embedding vectors. The results show that the model performance is severely degraded, which suggests that pretraining using click information is essential to the user-item modelling for purchase prediction task due to the sparsity issue of the dataset.

In this work, we mainly focus on the feasibility of incorporating discount information in purchase prediction. Mapping user/item to the embedding vector is an intermediate step, and the embedding size is a fixed empirical value for all pretrain methods, which could be a potential limitation of this work.

6.2. Importance of Pricing Information

We present the results regarding the utilization of price information in Table 8. The model without pricing information has only 26.28% of accuracy and 32.21% of F1 score on the purchased items, which validates our argument on the importance of pricing information towards the purchase prediction task. In our experiments, the model without pricing information only produces reasonable results at the first epoch and deteriorates quickly, becoming steady as a low level (3% to 5% of accuracy). We discuss the cause as following.

When we train the node embedding, the loss function encourages embedding vectors of interacted user/item pair to share more similarity (positive inner product) and those of never interacted user/item pair to show more divergence (negative inner product). In Step 2, items within a group all interact with the same user. Therefore, the inner products between user/item pairs are all positive when training starts. The loss function tends to encourage the classifier to return negative results for all input, which is similar to always negative predictor and quickly overfits. Moreover, we conducted undersampling on negative samples and found the difference is not significant.

However, the model still fails to achieve good performance when item-based price information is added, which is a significant observation because it indicates only the item-based price information is not helpful to the transaction-based prediction. We conclude the reason behind as each item has a unique item-based price vector, so as the price vector cannot provide complement information to the item feature representation in the "decision" space. As a consequence, the classifier overfits on the training set and fails to handle any new cases because it cannot capture additional knowledge that is independent of the user-item representation learned from click data.

This experiment reveals an intriguing fact that the price information of each transaction is a key to the purchase-relevant tasks, which can also explain the difficulty and limitation of traditional recommendation tasks that are solely based on user-item interaction.

We also compare the effects of different feature engineering approaches on the price vector. Notable improvements are observed as both price-aware attributes and discount-aware attributes are included. When employing two types of attributes separately, the model without discount-attributes outperforms that without price-attributes, which implies that the price-aware attributes are more dominant than discount-aware attributes in the prediction task.

To further explore the effect of discount percentage in purchase prediction, we added a simple baseline, the highest discount predictor (as shown in the second line of Table 8), which returns the highest discounted item in a sequence as the output. Note that the results are for reference only and are not comparable with other baselines because such a method cannot properly handle the multiple-purchase situation in a sequence. Though lower than other methods, the results are still reasonable for a nave approach and show that the discount plays a non-trivial role in users decision making.

6.3. Trainable parameter

As a commonly adopted approach in both recommendation and natural language processing, the input embedding vectors are set to be trainable during the training stage. Therefore, the weights in embeddings can be continuously tuned to capture more domain-dependent or class-related features and make the feature extraction and decision making of the classifier less challenging, which can effectively boost the model performance. However, a shortcoming of this approach is that the model can hardly make a correct prediction on unseen instances due to the same problem stated in Subsection 6.2. To avoid such a problem, a popular solution adopted by traditional recommendation models is splitting the dataset into train/test sets by dividing the records of each user, so as the model can predict the user's future actions based on his/her past behavior. However, this strategy is under an unrealistic scenario and cannot address the cold-starting issue.

In contrast, the transaction-based dataset utilized for purchase prediction task is generated by sorting transactions grouped by users. In most cases, users in the test set will not appear in the training set. Therefore, the model will be less robust on test set because users in the training set and users in the test set are following different distributions if embeddings are trainable.

Pretrain Model	Overall		Label **1**	
	Accu.(%)	F1 (%)	Accu. (%)	F1 (%)
MF	93.36	82.24	67.06	68.32
DeepWalk	93.18	82.95	67.28	66.13
Node2Vec	93.58	82.26	67.22	66.25
Light GCN	92.86	80.61	64.32	65.52
LINE-1st (bin)	93.10	81.39	64.32	65.53
LINE-1st (weighted)	93.20	81.58	65.17	66.84
LINE-*all* (bin)	93.42	81.21	66.00	66.34
LINE-*all* (weighted)	93.66	82.74	68.57	69.14
Ours (bin)	93.45	82.25	68.66	68.35
Ours (weighted)	93.68	82.38	67.58	68.64

Table 9: Results of different pretrain models with embeddings trainable

In Table 9, we present the prediction performances of models with trainable embeddings. The results indicate that both accuracy and F1 score plunge compared with models with embedding constant.

6.4. Comparison with Ranking-based Method

In this section, we further clarify the difference between ranking-based approach and the proposed framework.

As described in [39, 20], given a set of all recommendable items, ranking-based methods compute an ordered list for each user and recommend Top-*N* items as output. The recommendable candidates can be all unseen items to increase the coverage of recommendation results. They can also be restricted to a subset of recently interacted items in a session-based purchase recommendation task, as discussed in [28]. Similar to [28], the proposed framework targets to correctly identify the purchased items from a particular group of items that are clicked by the user. The items in such a group have a temporal relationship, which can be easily applied to real-time session-based purchase prediction.

Furthermore, each data sample may contain multiple purchased items (with label 1). By predicting the binary label, the proposed method can properly handle such a situation. However, a ranking-based method cannot explicitly predict the positive label or negative label by setting a constant threshold.

7. Conclusion & Future Work

In this paper, we proposed a new form of transaction-based purchase prediction task, which aims to identify the purchased item(s) from recently viewed items. Such a framework can evaluate the model performance under a more realistic scenario compared with traditional ranking-based approach. Based on a real life dataset with detailed price and promotion information, we discussed the relationship between different pricing strategies and customer's purchasing will from *price-aware* aspect and *discount-aware* aspect, which previous works did not address. Moreover, we proposed a two-step graph-based purchase prediction model incorporating price and promotion information of each transaction. Extensive experiments conducted on a

JD.com dataset demonstrated the effectiveness of the graph-based model and the importance of transaction-based price information in purchase prediction task.

For future work, we plan to focus on (1) processing and utilization of price features, and (2) addressing the prediction task by using a ranking-based approach. The normalized price vectors employed in this work may overlook price rage information, which is also a pivotal factor in customer's decision making. Complicated models like attention mechanism can be added to distinguish the effects of different promotion strategies. Besides, there exist similar datasets providing price information, some of which also provide discount information [3]. However, the prices and discounts from those dataset are not varying over time. Therefore, we will focus on the model design to well incorporate these static information. Moreover, we will also attempt to generate outputs in a contrastive manner, so as we can address the purchase prediction task from a ranking-based perspective.

8. Acknowledgment

The research described in this article has been supported by the Lam Woo Research Grant (LWI20011), Faculty Research Grant (102041), Direct Grant (101138) and the Research Seed Fund (102367) at Lingnan University, Hong Kong, and a General Research Fund from the Hong Kong Research Grants Council (project no. PolyU 11204919).

References

[1] R. Jia, R. Li, M. Yu, S. Wang, E-commerce purchase prediction approach by user behavior data, in: 2017 International Conference on Computer, Information and Telecommunication Systems (CITS), 2017, pp. 1–5.

[2] R. Winer, A framework for customer relationship management, California management review 43 (4) (2001) 89–105.

[3] M. Platzer, T. Reutterer, Ticking away the moments: Timing regularity helps to better predict customer activity, Mark. Sci. 35 (2016) 779–799.

[4] C. Park, D. Kim, M. Yang, J. Lee, H. Yu, Click-aware purchase prediction with push at the top, Information Sciences 521 (2020) 350–364.

[5] Z. Li, H. Xie, Y. Zhao, Q. Li, Incorporating latent space correlation coefficients to collaborative filtering, in: 2019 IEEE 35th International Conference on Data Engineering Workshops (ICDEW), 2019, pp. 155–160.

[6] Y. Zhang, G. Liu, A. Liu, Y. Zhang, Z. Li, X. Zhang, Q. Li, Personalized geographical influence modeling for poi recommendation, IEEE Intelligent Systems 35 (5) (2020) 18–27. doi:10.1109/MIS.2020.2998040.

[7] X. Fan, M. Gong, Y. Xie, F. Jiang, H. Li, Structured self-attention architecture for graph-level representation learning, Pattern Recognition 100 (2020) 107084.

[8] L. Bai, L. Cui, Y. Jiao, L. Rossi, E. Hancock, Learning backtrackless aligned-spatial graph convolutional networks for graph classification, IEEE Transactions on Pattern Analysis and Machine Intelligence (2020) 1–15doi:10.1109/TPAMI.2020.3011866.

[9] Z. Zhang, D. Chen, J. Wang, L. Bai, E. R. Hancock, Quantum-based subgraph convolutional neural networks, Pattern Recognition 88 (2019) 38 – 49. doi:https://doi.org/10.1016/j.patcog.2018.11.002.

[10] Y. Bin, Z.-M. Chen, X.-S. Wei, X. Chen, C. Gao, N. Sang, Structure-aware human pose estimation with graph convolutional networks, Pattern Recognition 106 (2020) 107410.

[11] Y. Jing, J. Wang, W. Wang, L. Wang, T. Tan, Relational graph neural network for situation recognition, Pattern Recognition 108 (2020) 107544.

[12] J. Wu, S. hua Zhong, Y. Liu, Dynamic graph convolutional network for multi-video summarization, Pattern Recognition 107 (2020) 107382.

[13] Q. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, D. Yin, Graph neural networks for social recommendation, in: The World Wide Web Conference, WWW 19, Association for Computing Machinery, 2019, pp. 417–426.

[3]http://archive.ics.uci.edu/ml/datasets/online+retail

[14] H. Xie, Q. Li, X. Mao, X. Li, Y. Cai, Y. Rao, Community-aware user profile enrichment in folksonomy, Neural Networks 58 (2014) 111–121.

[15] H. Xie, Q. Li, X. Mao, X. Li, Y. Cai, Q. Zheng, Mining latent user community for tag-based and content-based search in social media, The Computer Journal 57 (9) (2014) 1415–1430.

[16] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, Bpr: Bayesian personalized ranking from implicit feedback, in: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 09, AUAI Press, 2009, pp. 452–461.

[17] J. Cheng, C. Lo, J. J. Leskovec, Predicting intent using activity logs: How goal specificity and temporal range affect user behavior, in: Proceedings of the 26th International Conference on World Wide Web Companion, WWW 17 Companion, International World Wide Web Conferences Steering Committee, 2017, pp. 593–601.

[18] D. Jannach, B. Mobasher, S. Berkovsky, Research directions in session-based and sequential recommendation, User Model. User Adapt. Interact. 30 (4) (2020) 609–616.

[19] D. Jannach, M. Ludewig, When recurrent neural networks meet the neighborhood for session-based recommendation, in: Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys '17, Association for Computing Machinery, 2017, pp. 306–310.

[20] M. Quadrana, P. Cremonesi, D. Jannach, Sequence-aware recommender systems, ACM Comput. Surv. 51 (4).

[21] D. Jannach, M. Zanker, Collaborative filtering: Matrix completion and session-based recommendation tasks, in: S. Berkovsky, I. Cantador, D. Tikk (Eds.), Collaborative Recommendations - Algorithms, Practical Challenges and Applications, WorldScientific, 2018, pp. 1–34.

[22] H. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Daniel, et al., Ad click prediction: a view from the trenches, in: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, 2013, pp. 1222–1230.

[23] J. Cheng, C. Lo, J. Leskovec, Predicting intent using activity logs: How goal specificity and temporal range affect user behavior, in: Proceedings of the 26th International Conference on World Wide Web Companion, WWW 17 Companion, 2017, pp. 593–601.

[24] T. Iwata, S. Watanabe, T. Yamada, N. Ueda, Topic tracking model for analyzing consumer purchase behavior, in: Twenty-First International Joint Conference on Artificial Intelligence, 2009.

[25] G. Liu, T. Nguyen, G. Zhao, W. Zha, J. Yang, J. Cao, M. Wu, P. Zhao, W. Chen, Repeat buyer prediction for e-commerce, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 16, Association for Computing Machinery, 2016, pp. 155–164.

[26] Y. Zhang, M. Pennacchiotti, Predicting purchase behaviors from social media, in: Proceedings of the 22nd International Conference on World Wide Web, WWW 13, Association for Computing Machinery, 2013, pp. 1521–1532.

[27] Y. Zhao, L. Yao, Y. Zhang, Purchase prediction using tmall-specific features, Concurrency and Computation: Practice and Experience 28 (14) (2016) 3879–3894.

[28] D. Jannach, M. Ludewig, L. Lerche, Session-based item recommendation in e-commerce: on short-term intents, reminders, trends and discounts, User Modeling and User-Adapted Interaction 27 (3-5) (2017) 351–392.

[29] W.-C. Chen, C.-Y. Wang, S.-C. Lin, A. Ou, T.-C. Liou, Psac: Context-based purchase prediction framework via user's sequential actions, in: eCOM@SIGIR, 2019.

[30] Q. Zhao, M. C. Willemsen, G. Adomavicius, F. M. Harper, J. A. Konstan, Interpreting user inaction in recommender systems, in: Proceedings of the 12th ACM Conference on Recommender Systems, RecSys '18, Association for Computing Machinery, 2018, pp. 40–48.

[31] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, Line: Large-scale information network embedding, in: Proceedings of the 24th International Conference on World Wide Web, WWW 15, International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

[32] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, NIPS13, Curran Associates Inc., 2013, pp. 3111–3119.

[33] A. Q. Li, A. Ahmed, S. Ravi, A. J. Smola, Reducing the sampling complexity of topic models, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 14, Association for Computing Machinery, 2014, pp. 891–900.

[34] Z. M. Shen, C. S. Tang, D. Wu, R. Yuan, W. Zhou, Jd. com: Transaction level data for the 2020 msom data driven research challenge, Available at SSRN 3511861.

[35] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009) 30–37.

[36] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 14, Association for Computing Machinery, 2014, pp. 701–710.

[37] A. Grover, J. Leskovec, Node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 16, Association for Comput-

ing Machinery, 2016, pp. 855–864.

[38] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, M. Wang, Lightgcn: Simplifying and powering graph convolution network for recommendation, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 20, Association for Computing Machinery, 2020, pp. 639–648.

[39] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions, IEEE Transactions on Knowledge and Data Engineering 17 (6) (2005) 734–749. doi:10.1109/TKDE.2005.99.