Modeling Persuasion Factor of User Decision for Recommendation

Chang Liu
Chen Gao*
Yuan Yuan
Department of Electronic Engineering, Tsinghua University, Beijing, China

Chen Bai
Lingrui Luo
Xiaoyi Du
Xinlei Shi
Hengliang Luo
Meituan Inc., Beijing, China

Depeng Jin
Yong Li
Department of Electronic Engineering, Tsinghua University, Beijing, China

ABSTRACT

In online information systems, users make decisions based on factors of several specific aspects, such as brand, price, etc. Existing recommendation engines ignore the explicit modeling of these factors, leading to sub-optimal recommendation performance. In this paper, we focus on the real-world scenario where these factors can be explicitly captured (the users are exposed with decision factor-based persuasion texts *i.e.*, persuasion factors). Although it allows us for explicit modeling of user-decision process, there are critical challenges including the persuasion factor's representation learning and effect estimation, along with the data-sparsity problem. To address them, in this work, we present our POEM (short for **P**ersuasion fact**O**r **E**ffect **M**odeling) system. We first propose the persuasion-factor graph convolutional layers for encoding and learning representations from the persuasion-aware interaction data. Then we develop a prediction layer that fully considers the user sensitivity to the persuasion factors. Finally, to address the data-sparsity issue, we propose a counterfactual learning-based data augmentation method to enhance the supervision signal. Real-world experiments demonstrate the effectiveness of our proposed framework of modeling the effect of persuasion factors.

CCS CONCEPTS

• **Information systems** → **Information systems applications**.

KEYWORDS

User Persuasion Factor; Recommender Systems; Graph Neural Networks; Counterfactual Learning

ACM Reference Format:

Chang Liu, Chen Gao, Yuan Yuan, Chen Bai, Lingrui Luo, Xiaoyi Du, Xinlei Shi, Hengliang Luo, Depeng Jin, and Yong Li. 2022. Modeling Persuasion Factor of User Decision for Recommendation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534678.3539114

*Chen Gao is the corresponding author (chgao96@gmail.com).

KDD '22, August 14–18, 2022, Washington, DC, USA.

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539114

Figure 1: Example of persuasion factor (shown in red box).

1 INTRODUCTION

Recommendation engines learn from behavioral history (*i.e.*, user-item interaction) and various features, including user profiles, item attributes, and context, which is defined as feature-based recommendation or click-through rate (CTR) prediction [4, 12, 16, 26, 42]. Typical solutions include shallow models such as FM [23], and deep models such as DeepFM [7], AutoInt [25], etc. For example, FM [23] exploits pairwise inner product for extracting second-order feature interactions; AutoInt [25] uses self-attention layers to capture higher-order feature interactions.

Despite their effectiveness, these models usually work in an implicit manner, ignoring the explicit modeling of how positive behavior occurs. One of the main reasons is the lack of real-world data which can be used to understand how users make decisions. In the real world, the decision of a user is driven by several factors [37] and the consumed item also matches several important aspects of user preferences. For example, in restaurant recommendation scenario, the aspects may include the taste of food, the environment of the restaurant, the variety of dishes, the price, etc., which affect the click behavior of users. When it comes to the hotel recommendation, these factors may consist of location, environment, facility, service quality, etc.

In China's most popular local life service provider, Meituan, during the process of recommending item to users, besides showing their regular features like name, picture and rating, there is a text message about the item that is relevant to one of aspects listed above, and it is intended to urge consumers to click. As shown in Figure 1, these texts are generated according to the above-mentioned aspects, which provide us the opportunity of explicit modeling the user decision process. For example, as for *facility*, there is a corresponding *"Full-equipped Facilities"* text. Therefore, we define this kind of text

as *persuasion factor*. In this scenario, we can model the effect of the persuasion factor on user decision, but however, there are still three critical challenges as follows.

- **First, the encoding and representation of persuasion factors are difficult.** Different from users and items of which the representations can be easily learned by many effective ways, persuasion factors reflect more fine-grained user decisions. Besides, they exert effects in a different way compared with attributes of users and items and are not explicitly reflected by click behaviors. Therefore, how to encode persuasion factors and learn their representations are challenging.
- **Second, the effect of persuasion factor on users is complex and heterogeneous.** In fact, the specific user or item determines whether the exposed persuasion factor could have an influence. For example, some users will hardly be influenced by the exposed texts, and they usually make decisions solely by themselves. Therefore, simply using a persuasion factor-based model may not be beneficial for them. In summary, the difference in sensitivity among different users to persuasion factors is an essential problem in modeling the effect of persuasion factors.
- **Third, the historical data is always sparse.** Compared with normal user-item interaction data, the users' behavioral log containing exposed persuasion factors are usually more sparse, which makes it more difficult to learn users' preferences on them, thus limiting the capability of modeling their effect on user decision. Hence, it is essential to address the data sparsity issue.

Considering the above issues and challenges, in this work, we propose a novel solution **POEM** (short for ***P**ersuasion fact**O**r **E**ffect **M**odeling*) to tackle these problems. Specifically, we first organize users' clicks on items under the influence of persuasion factors by employing the form of graph, where users and items are nodes and persuasion factors serve as edges. Then we utilize a graph convolutional network to learn representations of users, items, and persuasion factors from the structured data. Following that, we present a user-sensitivity-based prediction that can adaptively assess how much persuasion factors are influencing users' ultimate behavior. As for the third challenge, we propose a counterfactual learning-based data augmentation method that generates data according to assumptions we make in the counterfactual world, which can significantly address the data-sparsity issue. The working pipeline of POEM is shown in Figure 2, using the features of users and items, and persuasion factors in the dataset to generate recommendations, while collecting new user behavioral data and adding it to the dataset for future training. The main contributions of our work can be summarized as follows.

- We take the pioneering step to approach the problem of modeling the effect of persuasion factors on user decision for large-scale recommendation in Meituan, which is an important problem in real-world applications but has not been well explored.
- We propose a solution that first constructs a graph and deploys graph convolutional layers to learn representations from the graph. We further suggest adaptive user sensitivity-based prediction and counterfactual learning to alleviate the data sparsity problem.

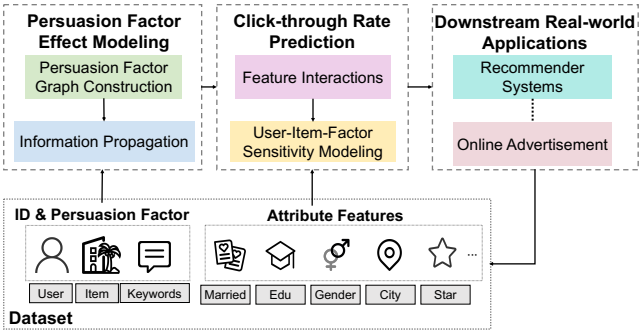


Figure 2: Persuasion factor effect modeling application.

Table 1: Notation Table.

Notations	Description
$\mathbf{x}$	Feature vector.
u, i, t	User, item, persuasion factor.
f, N_f	Graph node, the neighbor of node f.
M, N, W, Q	The number of users, items, persuasion factors, fields.
$\mathbf{E}_u, \mathbf{E}_i, \mathbf{E}_t$	Embedding matrix of users, items and persuasion factors.
$\mathbf{e}_u, \mathbf{e}_i, \mathbf{e}_t$	Embedding of user, item and persuasion factor.
$\mathbf{e}_f^{(i)}$	The i-th layer embedding of graph nodes.
$\mathbf{e}_{N_f}^{(i)}$	The i-th layer information of the node f's neighbors.
$\omega/\mathbf{W}$	The attention weight/transition matrix.
Θ	Trainable parameters.
λ	L_2 regularization hyperparameter.
O_f/O_{cf}	Factual/Counterfactual dataset.
O_{cf_1}/O_{cf_2}	Counterfactual dataset according to assumption 1/2.

- We collect two real-world datasets for evaluation and release them to benefit the community. The extensive experimental results show the superiority of our proposed approach. Compared with existing methods, our method makes significant improvement on several metrics. In addition, we conduct ablation studies to validate the efficacy of designs in our model. We also illustrate the advantages of our model in terms of explainable recommendations by modeling the effect of persuasion factors on user decisions in large-scale recommendation scenarios.

2 PROBLEM FORMULATION

The feature-based recommendation (also known as click-through rate (CTR) prediction) is formulated as predicting a binary target y from a feature vector $\mathbf{X}$, which is composed of the user profile and item attributes. In this work, we innovatively propose to consider the effect of persuasion factors on user decisions. Therefore, we formulate our problem as follows:

- **Input:** The feature vector $\mathbf{x}_{ij} \in \mathbb{R}^n$ of the i-th user and the j-th item, and persuasion factor t of the j-th item shown to the i-th user.
- **Output:** The probability of the i-th user click on the j-th item, i.e., $\hat{y}_{ij}$, under the condition of $\mathbf{x}_{ij}$ and t.

3 METHODOLOGY

We propose an effective method entitled POEM (short for ***P**ersuasion fact**O**r **E**ffect **M**odeling*) to model the effect of persuasion factors on user decisions. Figure 3 illustrates the holistic design of POEM, which consists of three key components:

- **Persuasion-factor Graph Convolutional Layer.** We propose to encode persuasion factors that influence users' decisions, along with users and items, as embeddings. In order to better learn representations from the brand new form of data, which consists of persuasion factors, we first construct a unified graph then propose graph convolutional layers based on information propagation.
- **User-sensitivity Based Prediction.** To tackle the heterogeneity of users' sensitivity to external influence, we propose to model the sensitivity to persuasion factors for each user in the prediction process.
- **Signal-enhanced Counterfactual Learning.** To address the challenge of data sparsity, we propose a counterfactual learning-based technique for data augmentation.

3.1 Persuasion-factor Graph Convolutional Layer

3.1.1 Unified Embedding Layer. Suppose that we have W kinds of disentangled and independent persuasion factors, for the i-th persuasion factor, we define its one-hot vector as t_i. We then project its one-hot vector to latent embeddings, and the embedding matrix of persuasion factors $\mathbf{E}_t$ can be denoted as follows,

$$\mathbf{E}_t = [\mathbf{e}_{t_1}; \mathbf{e}_{t_2}; \cdots; \mathbf{e}_{t_W}], \tag{1}$$

where $\mathbf{E}_t \in \mathbb{R}^{W \times d}$ and d denotes the embedding size. Then the embedding of the i-th persuasion factor can be denoted as $\mathbf{e}_{t_i} = t_i \mathbf{E}_t$.

Similarly, the one-hot vectors are u_j for the j-th user and i_k for the k-th item. Following the paradigm of existing recommendation models [11, 15], we also represent users and items by embeddings, which have the same embedding size d as persuasion factors, formulated as follows,

$$\begin{aligned} \mathbf{E}_u &= [\mathbf{e}_{u_1}; \mathbf{e}_{u_2}; \cdots; \mathbf{e}_{u_M}], \\ \mathbf{E}_i &= [\mathbf{e}_{i_1}; \mathbf{e}_{i_2}; \cdots; \mathbf{e}_{i_N}], \end{aligned} \tag{2}$$

where $\mathbf{e}_{u_j} \in \mathbb{R}^d$ and $\mathbf{e}_{i_k} \in \mathbb{R}^d$ denote the embedding of the j-th user and the k-th item, respectively. Here M/N denotes the number of users/items. Following the similar rule, we compute embeddings of the j-th user and the k-th item as $\mathbf{e}_{u_j} = u_j \mathbf{E}_u$, $\mathbf{e}_{i_k} = i_k \mathbf{E}_i$, respectively.

3.1.2 Graph convolutional layer. Persuasion factors reflect more fine-grained user decisions. Besides, they exert effects in a different way compared with attributes of users and items and are not explicitly reflected by click behaviors. To address this problem, we leverage representation learning [6, 18, 21, 29, 30, 40] based on the graph to model psychological persuasion factors.

However, learning effective embeddings of persuasion factors is not trivial. Here we propose a graph convolutional network (GCN) [5, 9, 14]-based solution to facilitate the embedding learning.

Graph Construction We construct a heterogeneous graph to represent the user-item interaction affected by persuasion factors. The graph contains two kinds of nodes (users and items) and multiple edges (persuasion factors). Specifically, an edge t_k is added between user u_i and item i_j if the i-th user u_i clicks on the j-th item i_j under the k-th persuasion factor t_k. In this way, we are able to utilize GCN to represent user-item interactions influenced by persuasion factors, which is more efficient and explainable than conventional feature interaction models.

Information Propagation As mentioned before, we characterize the interaction between user u_i and item i_j under persuasion factor t_k as a triplet (u_i, t_k, i_j). In the real-world scenario, a user will click on multiple items and an item will also be clicked by many users. Therefore, a node in the graph can be contained in several triplets. Taking $i_1 \xleftrightarrow{t_1} u_1 \xleftrightarrow{t_2} i_2$ and $i_4 \xleftrightarrow{t_5} u_1 \xleftrightarrow{t_2} i_2$ as an example, u_1 combines information from i_1 and i_4 via t_1 and t_5, respectively, and passes them to i_2 as its neighbor, which can be simulated by propagating information from i_1 and i_4 to i_2, enriching its representation. Inspired by existing works [9, 14, 17, 27, 31**?**], we perform information propagation between a node and its neighbors, which is shown in Figure 4.

From the perspective of graph convolutional network, the embeddings of users and items defined in Equation (2) are also layer-0 node embeddings in the graph, which can be reformulated as:

$$\begin{aligned} \mathbf{E}_u^{(0)} &= [\mathbf{e}_{u_1}^{(0)}; \mathbf{e}_{u_2}^{(0)}; \cdots; \mathbf{e}_{u_M}^{(0)}], \\ \mathbf{E}_i^{(0)} &= [\mathbf{e}_{i_1}^{(0)}; \mathbf{e}_{i_2}^{(0)}; \cdots; \mathbf{e}_{i_N}^{(0)}]. \end{aligned} \tag{3}$$

Similarly, the embeddings of persuasion factors defined in Equation (1) are also edge embeddings in the graph. For a node f in the graph, $\mathcal{N}_f = \{(t, b) | (f, t, b) \in \mathcal{G}\}$ denotes the set of triplets where f is involved. To represent the first-order connectivity of the node f, taking layer-0 as an example, the information from all its neighbors can be formulated as

$$\mathbf{e}_{\mathcal{N}_f}^{(0)} = \frac{1}{|\mathcal{N}_f|} \sum_{(t,b) \in \mathcal{N}_f} \mathbf{e}_t \odot \mathbf{e}_b^{(0)}. \tag{4}$$

For simplicity, we omit the subscripts of nodes in Equation (4), where $\mathbf{e}_t$ is the embedding of persuasion factor t, $\mathbf{e}_b^{(0)}$ is the layer-0 embedding of node b (users or items), $\mathbf{e}_{\mathcal{N}_f}^{(0)}$ is the layer-0 information of the neighbors of node f. $|\mathcal{N}_f|$ denotes the cardinality of the neighbor triplet set $\mathcal{N}_f$.

To get the next layer representation of a node in the graph, we aim to aggregate the information from its neighbor and its current embedding. Formally, for the layer-1 embedding of node f, we have

$$\mathbf{e}_f^{(1)} = f(\mathbf{e}_{\mathcal{N}_f}^{(0)}, \mathbf{e}_f^{(0)}), \tag{5}$$

where $f(\cdot)$ is the aggregation function.

For the implementation of function $f(\cdot)$, we have multiple choices, inspired by the existing works [9, 14**?**] and considering the complex effect from neighbors, we have:

$$\begin{aligned} f(\mathbf{e}_{\mathcal{N}_f}^{(0)}, \mathbf{e}_f^{(0)}) = \text{LeakyReLU}(\mathbf{W}_1(\mathbf{e}_{\mathcal{N}_f}^{(0)} + \mathbf{e}_f^{(0)})) \\ + \text{LeakyReLU}(\mathbf{W}_2(\mathbf{e}_{\mathcal{N}_f}^{(0)} || \mathbf{e}_f^{(0)})) \end{aligned} \tag{6}$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ are trainable transformation matrices.

Multi-layer Propagation Following the same rule, we can stack more layers to capture the higher-order connectivity on the graph.

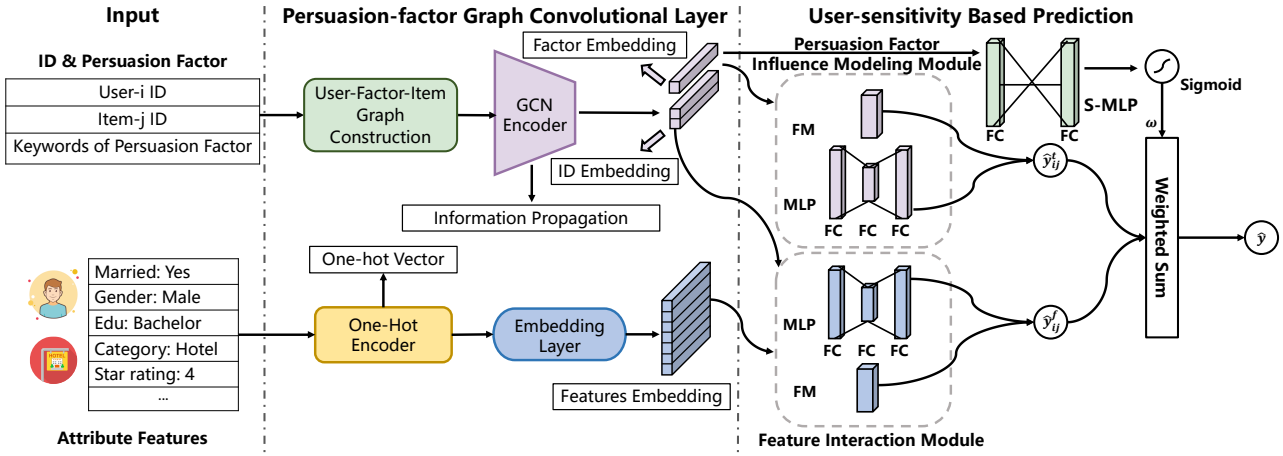


Figure 3: The persuasion-factor graph convolutional layer and user-sensitivity based prediction of our proposed model POEM.

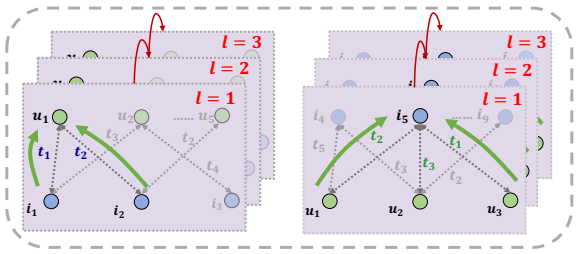


Figure 4: A detailed illustration of information propagation.

For the l-th layer embedding of node f, we can calculate it similarly as follows,

$$\begin{aligned} \mathbf{e}_{\mathcal{N}_f}^{(l-1)} &= \frac{1}{|\mathcal{N}_f|} \sum_{(t,b)\in\mathcal{N}_f} \mathbf{e}_t \odot \mathbf{e}_b^{(l-1)}, \\ \mathbf{e}_f^{(l)} &= f(\mathbf{e}_{\mathcal{N}_f}^{(l-1)}, \mathbf{e}_f^{(l-1)}). \end{aligned} \tag{7}$$

Utilizing multi-hop propagation to capture high-order connections also makes better use of the information from collaborative filtering, which benefits the representation learning of nodes (users and items) and edges (persuasion factors).

After propagating for l-layers, the representation of users, items and persuasion factors can be formulated as

$$\begin{aligned} \mathbf{E}_u^{(l)} &= [\mathbf{e}_{u_1}^{(l)}, \mathbf{e}_{u_2}^{(l)}, \cdots, \mathbf{e}_{u_M}^{(l)}], \\ \mathbf{E}_i^{(l)} &= [\mathbf{e}_{i_1}^{(l)}, \mathbf{e}_{i_2}^{(l)}, \cdots, \mathbf{e}_{i_N}^{(l)}], \\ \mathbf{E}_t &= [\mathbf{e}_{t_1}, \mathbf{e}_{t_2}, \cdots, \mathbf{e}_{t_W}]. \end{aligned} \tag{8}$$

In conclusion, graph convolutional network serves as an encoder for users, items, and persuasion factors in our model, where information propagation significantly contributes to learning their representations.

3.2 User-sensitivity Based Prediction

3.2.1 Base Model. We construct our base model with widely used feature-based recommendation models [1, 2, 7, 11, 25]. It takes input feature vector $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_Q]$ and generates embedding for the i-th field via

$$\mathbf{g}_i = \mathbf{x}_i \mathbf{G}_i, \tag{9}$$

where $\mathbf{G}_i \in \mathbb{R}^{K_i \times d}$ denotes the embedding matrix of the i-th field, K_i denotes the number of features in the i-th field and d denotes the embedding size. Overall embedding matrix of base model can be formulated as

$$\mathbf{G} = [\mathbf{G}_1; \mathbf{G}_2; \cdots; \mathbf{G}_Q]. \tag{10}$$

Then we calculate the prediction score with G and other parameters Θ of the model, formulated as

$$\hat{y} = \psi(\mathbf{x}|\mathbf{G}, \Theta), \tag{11}$$

where $\hat{y}$ is the prediction score, ψ refers to the union of prediction models such as FM [23] and MLP. The base model consists of three modules: feature embedding layer, feature interaction layer and MLP layer.

3.2.2 Prediction Module. We divide the prediction score into two parts, one of which comes from feature interaction module and the other from persuasion factor influence modeling module. Specifically, we treat user profile and item attributes as normal features and put them into the feature interaction module, where IDs of user and item are also involved as features. For prediction, the feature interaction module consists of a base model. However, the user and item ID embeddings are extracted from the graph convolutional layer's output rather than the feature embedding layer of base model. We denote the prediction score between the i-th user and the j-th item from the feature interaction module as $\hat{y}_{ij}^f$.

The persuasion factor influence modeling module also consists of a base model without feature embedding layer independent of the feature interaction module. Its input is the embedding of user ID, item ID, and persuasion factor, extracted from the graph convolutional layer. As mentioned before, persuasion factors reflect more fine-grained user decisions and exert effects in a different way compared with attributes of users and items. Therefore, before using embeddings to perform high-order feature interactions, we

first conduct *embedding projection* for user and item, formulated as:

$$\begin{aligned} \mathbf{e}_u^{'} &= \mathbf{e}_u^{(l)} + \sigma(\mathbf{e}_u^{(l)} \odot \mathbf{e}_t) \odot \mathbf{e}_u^{(l)}, \\ \mathbf{e}_i^{'} &= \mathbf{e}_i^{(l)} + \sigma(\mathbf{e}_i^{(l)} \odot \mathbf{e}_t) \odot \mathbf{e}_i^{(l)}, \end{aligned} \tag{12}$$

then the final representation of the user, item and persuasion factor change to $\mathbf{e}_u^{'}, \mathbf{e}_i^{'}$ and $\mathbf{e}_t$, respectively, and we input them to the feature interaction layer and MLP layer of the persuasion factor influence modeling module. Similarly, we denote the prediction score between the i-th user and the j-th item from the persuasion factor influence modeling module as $\hat{y}_{ij}^t$.

3.2.3 Fusion Strategy. The effect of persuasion factors on user is heterogeneous. Considering this situation, there is an exploiting balance between prediction scores from two parts defined in Section 3.2.2. Formally, the final prediction score between the i-th user and the j-the item, i.e., $\hat{y}_{ij}$, can be formulated as follows:

$$\hat{y}_{ij} = \omega \hat{y}_{ij}^t + (1-\omega)\hat{y}_{ij}^f, \tag{13}$$

where ω is a learnable weight, $\omega \in [0,1]$ and $\omega \in \mathbb{R}$. ω can also be seen as the *sensitivity* to persuasion factor of user u_i when *fixing* the item and the persuasion factor.

To adaptively learn the weight ω, we utilize a fully connected layer and the sigmoid function to limit its scale. Specifically, ω of the user u_i under the j-th item and the k-th persuasion factor can be formulated as:

$$\omega = \sigma(\text{MLP}(\mathbf{e}_{u_i}^{(l)} || \mathbf{e}_{i_j}^{(l)} || \mathbf{e}_{t_k})), \tag{14}$$

where we denote the MLP we use as S-MLP (short for Sensitivity-MLP). The structure of S-MLP is a hyperparameter, $\sigma(\cdot)$ is sigmoid function and $\mathbf{e}_{u_i}^{(l)}$, $\mathbf{e}_{i_j}^{(l)}$, $\mathbf{e}_{t_k}$ denote the embeddings of the i-th user, the j-th item, and the k-th persuasion factor extracted from graph convolutional layer, respectively.

3.3 Signal-enhanced Counterfactual Learning

In real-world scenarios, the sparsity of exposed persuasion factors hinders effective learning of users' corresponding preferences. To tackle this issue, we present a counterfactual learning-based data augmentation technique to address this problem.

We define the treatment r as whether a persuasion factor exists and the outcome y as whether a user will click on the item recommended to him/her. If a persuasion factor exists, the decision of the user to click may be influenced by it. Then we can denote the outcome under $r=0$ as y_{r_0} and the outcome under $r=1$ as y_{r_1}. One of the crucial causes of data sparsity is the absence of counterfactual data. In the factual scenario, the persuasion factor can either exist ($r=1$) or not ($r=0$), so we can only observe the outcome under the treatment that has taken place. Then we denote the outcome we observe in collected data as the *factual* outcome y_f and unobserved outcome in the opposite treatment as the *counterfactual* outcome y_{cf}.

Inspired by in-depth analysis of the effect of persuasion factor on user decision, we propose to solve the challenge from the root by counterfactual data augmentation based on causal knowledge. According to our analysis, we propose the following two reasonable assumptions:

ASSUMPTION 1: If a user clicks on an item ($y_f = 1$) **without** the existence of persuasion factor ($r=0$), we assume that the user will still be likely to click on the item ($y_{cf}=1$) when a persuasion factor matching the item exists ($r=1$).

ASSUMPTION 2: If a user does not click on an item ($y_f = 0$) **with** the existence of persuasion factor ($r=1$), we assume that the user will not click on the item ($y_{cf}=0$) when the persuasion factor does not exist ($r=0$).

The click behavior matching the first assumption is more related to the intrinsic interests of users. If the persuasion factor *were* exposed, it *would* enhance the possibility of clicking (*i.e.*, the counterfactual label is 1). The users matching the second assumption will not click even if there is a persuasion factor, which means they do not like the item. If the persuasion factor *were* removed, it *would* further reduce the possibility of click (*i.e.*, the counterfactual label is 0). Then we can generate counterfactual data based on the observed dataset and assumptions, formulated as follows,

$$\begin{aligned} O_{cf_1} &= \{y_{cf}^{(u_j,i_k)} = y_{r_1}^{(u_j,i_k)} = 1 | y_f^{(u_j,i_k)} = y_{r_0}^{(u_j,i_k)} = 1, (u_j,i_k) \in O_f\}, \\ O_{cf_2} &= \{y_{cf}^{(u_j,i_k)} = y_{r_0}^{(u_j,i_k)} = 0 | y_f^{(u_j,i_k)} = y_{r_1}^{(u_j,i_k)} = 0, (u_j,i_k) \in O_f\}, \\ O_{cf} &= O_{cf_1} \cup O_{cf_2}, \end{aligned} \tag{15}$$

where u_j and i_k represent the j-th user and the k-th item in the dataset. O_{cf_1} and O_{cf_2} represent the counterfactual datasets we construct according to the two assumptions, O_{cf} represents counterfactual data and O_f represents the factual data we observe. In this way, we can obtain counterfactual labeled data, which well addresses the data-sparsity problem and can be used for training to improve model performance.

3.4 Model Optimization

To optimize the prediction of interactions between users and items, our loss function is *Log loss* with L_2 regularization to prevent overfitting, which is defined as follows:

$$\mathcal{L} = -\frac{1}{N}\sum_{j=1}^{N}(y_j log(\hat{y}_j) + (1-y_j) log(1-\hat{y}_j)) + \lambda ||\Theta||_2, \tag{16}$$

where y_j and $\hat{y}_j$ are ground truth and estimated CTR, respectively. Here j is the index of training instance, and N is the number of training samples. λ is the hyperparameter to control the L_2 regularization term. Θ represents all trainable parameters in our model, including embeddings, weights of MLPs, attention weight ω, etc., and $||\cdot||_2$ denotes the L_2-regularization term.

4 EXPERIMENTS

4.1 Experimental Settings

4.1.1 Dataset. We conduct our experiments on two million-scale datasets collected in two time periods, from the real-world local service-providing application Meituan, widely downloaded and used in China. Most of the records in these datasets contain persuasion factors, which is an advantage that existing public datasets do not have. To supply further research, we have released our datasets

Table 2: Statistics of two datasets.

	Dataset-A	Dataset-B
#Instance	5, 571, 683	5, 542, 104
#Users	2, 280, 880	2, 270, 021
#Items	109, 893	112, 335
#Type of persuasion factors	14	14
#Fields	14	14

to benefit the community[1]. The statistics of two utilized datasets are reported in Table 2.

- **Dataset-A.** The dataset is collected from Meituan, which consists of exposure data during the users' browsing for hotels. The labels in the dataset indicate whether the user clicks the provided hotel ("1" for click). Each instance in the dataset also includes user-profiles and item attributes, such as the age group, gender, hotel category and star ratings. There are also persuasion factors in some instances, which is significant for the modeling of their effects.
- **Dataset-B.** Meituan also collects this dataset but in a different time period, consisting of exposure data during the users' browsing for hotels. The feature fields of this dataset are similar to Dataset-A, but it covers different users and items.

4.1.2 Baselines. To illustrate the effectiveness of POEM, we compare the performance with other feature-based models which are feasible to be deployed for real-world application. In baseline models, persuasion factors are treated as common features.

- **LR** is a linear regression model to combine all input features for CTR prediction.
- **NFM [11]** introduces a Bi-Interaction Pooling layer to capture the second-order feature interaction and replace the FM module. It also contains a deep neural network to model high-order feature interactions to enhance prediction performance.
- **PNN [22]** defines a product layer which exploits inner product and outer product to capture high-order feature interactions.
- **DeepFM [7]** ensembles factorization machines and deep neural networks to capture both low and high order feature interactions to better fit the relationship between input features and labels.
- **AutoInt [25]** extends multi-head self-attentive network with residual connections to capture the correlation between different fields.
- **DCN [28]**, i.e., DeepCrossNet, improves the Wide part of Wide & Deep [1] by using Cross Network to perform feature interactions, which avoids manual feature selection.
- **xDeepFM [15]** includes a Compressed Interaction Network, which produce high-order cross features by using outer product of stacked feature matrix. The model combines the Compressed Interaction Network and Deep Neural Networs for CTR prediction.
- **AFN [2]** is capable to automatically learn the specific order of high-order feature interactions, avoiding the huge computational effort and weak correlation that may arise when constructing high-order features.

4.1.3 Metrics. We adopt two widely-used and accepted metrics in CTR prediction tasks, **AUC** (Area Under the ROC curve) and **Logloss** (cross entropy). These two metrics evaluate model performance through different perspectives.

- **AUC** measures the probability that a model will score a randomly chosen positive item higher than a randomly chosen negative item. It concerns more about the order of predicted instances, reflecting the model's capability to distinguish between positive and negative samples.
- **Logloss** measures the distance between the predicted score and true label for each instance. Our proposed method also aims to minimize Logloss in Equation (16), so we also use it as a straightforward metric.

It is worth mentioning that a *slightly* higher **AUC** or lower **Logloss** at *0.001-level* is considered significant for recommendation tasks, which is emphasized by many previous works [1, 7, 25, 28]. In our real-world application scenarios in Meituan, a relatively small improvement in offline datasets will lead to a much more significant increasing on online metrics, which has also been verified by Google and many existing works [1, 7].

4.1.4 Implementation Details. We implement our model in PyTorch. We first shuffle each dataset and select the early 80% as the training set, middle 10% as the validation set, and the last 10% as the test set. For a fair comparison, the number of learnable parameters should be the same in different models, so we empirically fix the feature embedding size to 8 in all embedding-based methods. Considering the relatively large data volume, we set the batch size to 50000. For AutoInt [25], we use three interaction layers, and the number of hidden units is 64. For each interaction layer, there are two attention heads. For xDeepFM [15], we use two interaction layers following the default settings. For AFN [2], we set the number of hidden units of the logarithmic neuron to 128 for its better performance. After all network structures are fixed, we deploy a grid search to achieve optimal performance on our dataset. To prevent models from overfitting, we also apply a grid search for dropout rate in $\{0.0, 0.1, \cdots, 0.8\}$ for two datasets and find that it has a limited effect on final results. We also adopt a careful grid search of learning rate in $\{10^{-4}, 10^{-3}, 10^{-2}\}$ and the coefficient of L_2 normalization in $\{10^{-5}, 10^{-4}, \cdots, 10^{-1}\}$. We use the Adam optimizer and Xavier initialization for model parameters. Moreover, early stopping is used. We stop training when the validation performance does not increase for two successive valid epochs. We conduct several experiments on two dataset using different models and obtained average experimental results with different random seeds.

4.2 Overall Performance

We show the performance of all models in Table 3. From the experimental data, we can conclude as follows.

- **Our proposed model steadily achieves the best performance.** Our proposed model can achieve the highest AUC and the lowest LogLoss on both datasets. Specifically, POEM improves over the best baseline *w.r.t.* AUC by 0.0087, LogLoss by 0.0082 on Dataset-A, and improves over *w.r.t.* AUC by 0.0070, LogLoss by 0.0162 on Dataset-B. However, this is only compared to the numerically optimal value. According to Table 3, no baseline model can achieve the best AUC and the best LogLoss simultaneously. If we consider the model with the highest AUC as the best baseline, our model

[1] https://github.com/tsinghua-fib-lab/POEM.

Table 3: Overall performance comparison.

Model	Dataset-A		Dataset-B	
	AUC	LogLoss	AUC	LogLoss
LR	0.5377	0.3216	0.5238	0.3029
NFM [11]	0.6342	0.3138	0.6327	0.3322
PNN [22]	0.6429	0.2816	0.6407	0.4052
DeepFM [7]	0.6359	0.3389	0.6327	0.3813
AutoInt [25]	0.6356	0.3617	0.6375	0.3583
DCN [28]	0.6435	0.3128	0.6389	0.3176
xDeepFM [15]	0.6414	0.3654	0.6444	0.2995
AFN [2]	0.6362	0.3001	0.6303	0.2931
POEM (Ours)	**0.6522**	**0.2734**	**0.6514**	**0.2769**
Imp.	**+0.0087**	**-0.0082**	**+0.0060**	**-0.0162**

will improve over *w.r.t.* LogLoss by 0.0394 on Dataset-A, where the best baseline is DCN. Our model improves over *w.r.t.* LogLoss by 0.0226 on Dataset-B, where the best baseline is xDeepFM. If we consider the model with the lowest LogLoss as the best baseline, under this perspective, our model can improve over *w.r.t.* AUC by 0.0093 on Dataset-A, where the best baseline is PNN. Our model improves over *w.r.t.* AUC by 0.0211 on Dataset-B, where the best baseline is AFN.

- **Necessity and significance of modeling the effect of persuasion factors on user decision.** Compared with other recommendation models that treat persuasion factors as a common feature, our model considers the specific influence of persuasion factors on user-item interactions. POEM achieves the best performance *w.r.t.* AUC and LogLoss by fine-grained modeling the effect of persuasion factors from different perspectives.
- **Failure to properly model the effect of persuasion factors will have a relatively large impact on model performance.** On other widely-used benchmark datasets like Criteo, Avazu and Movielens, AutoInt [25] and AFN [2] are relatively the best baselines. However, although they have a complex feature interaction structure, they do not perform well on our dataset. On Dataset-A, AFN cannot beat DCN. On Dataset-B, AFN also cannot perform better than xDeepFM, *w.r.t.* AUC. The performance of AutoInt is also not so outstanding on both datasets. These results indicate that special designs are acquired to consider the effect of persuasion factors and utilizing only conventional feature interaction structure may lead to sub-optimal results. Our model adopts a graph convolutional network to explicitly represent the effect of persuasion factors along with user-sensitivity based prediction and counterfactual learning, which steadily achieves the best performance.

4.3 Ablation Study

To further measure the role of each designed component, we perform several ablation experiments. Our proposed method POEM contains four special designs: persuasion-factor graph convolutional network, embedding projection, user-sensitivity based prediction and signal-enhanced counterfactual learning. To study the effectiveness of these designs, we remove these modules in turn to verify their contribution to the model performance.

The experimental results on two datasets are reported in Figure 5. We also present the specific results in Table 4 for visual comparison,

Table 4: Ablation Study for the designs of POEM.

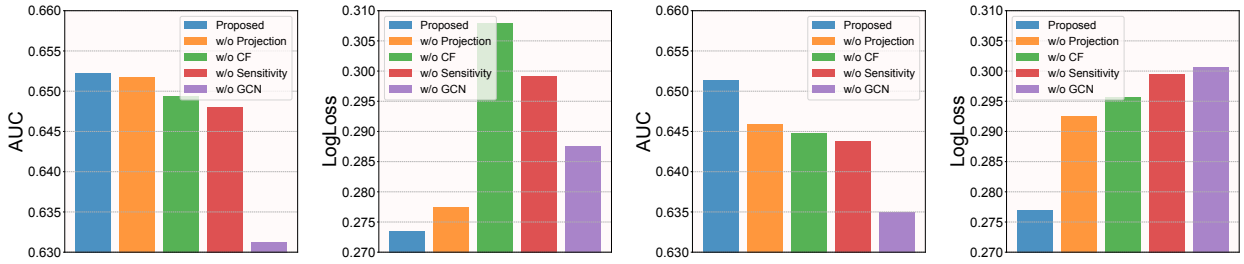
Model	Dataset-A		Dataset-B	
	AUC	LogLoss	AUC	LogLoss
POEM (Ours)	**0.6522**	**0.2734**	**0.6514**	**0.2769**
$\text{GCN}_{w/o}$	0.6313	0.2875	0.6350	0.3007
$\text{Projection}_{w/o}$	0.6518	0.2775	0.6459	0.2925
$\text{CF}_{w/o}$	0.6494	0.3079	0.6448	0.2957
$\text{Sensitivity}_{w/o}$	0.6480	0.2992	0.6438	0.2995

where $\text{GCN}_{w/o}$, $\text{Projection}_{w/o}$, $\text{Sensitivity}_{w/o}$ and $\text{CF}_{w/o}$, denote the removal of the four modules, respectively.

- **Effectiveness of GCN Encoder.** We compare the performance between the model with GCN Encoder for the embedding of users, items and persuasion factors and the model only with Xavier embedding initialization. The results show that the model with GCN Encoder outperforms the other model *w.r.t.* AUC by 0.0209, *w.r.t.* LogLoss by 0.0141 on Dataset-A, *w.r.t.* AUC by 0.0164, *w.r.t.* LogLoss by 0.0238 on Dataset-B, which is a significant performance drop according to existing works [1, 7, 25, 28]. By utilizing GCN Encoder, we explicitly model the effect of persuasion factors in the form of graph, and the idea of collaborative filtering is also applied in the process of information propagation. In general, GCN Encoder is an essential part of our proposed model.
- **Effectiveness of Embedding Projection.** Without embedding projection, the performance on all metrics declines to a certain degree. The results show that the model without embedding projection decreases *w.r.t.* AUC by 0.0004, *w.r.t.* LogLoss by 0.0041 on Dataset-A, *w.r.t.* AUC by 0.0055, *w.r.t.* LogLoss by 0.0156 on Dataset-B, which is relatively significant. It demonstrates the importance of embedding projection in fine-grained modeling the relations between user, item and persuasion factors.
- **Effectiveness of Sensitivity Modeling.** After compare our proposed method with the model which has fixed attention weight, we find that the proposed method performs better *w.r.t.* AUC by 0.0042, *w.r.t.* LogLoss by 0.0258 on Dataset-A, *w.r.t.* AUC by 0.0076, *w.r.t.* LogLoss by 0.0226 on Dataset-B. The experimental results verify our theory that the effect of persuasion factor is complex and heterogeneous. Naive method will lead to significant decline on overall performance. It is essential to assign individual attention weight for each data instance.
- **Effectiveness of Counterfactual Learning.** We compare the performance between the model training with extra counterfactual data and the model training with only original data. From the results, we find that the model with counterfactual data augmentation outperforms the model training only on original data *w.r.t.* AUC by 0.0028 , *w.r.t.* LogLoss by 0.0345 on Dataset-A, *w.r.t.* AUC by 0.0066, *w.r.t.* LogLoss by 0.0188 on Dataset-B. The significant improvement in performance illustrates that counterfactual learning-based data augmentation is a reasonably effective way to address the data sparsity problem, contributing to modeling the effect of persuasion factors.

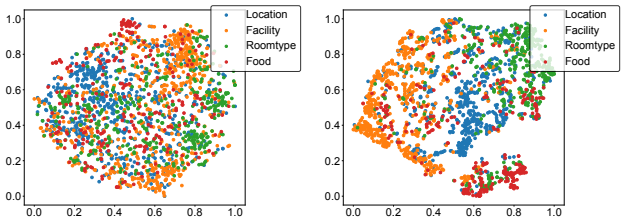
4.4 Explainable Recommendation

Researchers and developers in the industry have been paying more attention to the explainability of deep learning-based recommender



(a) AUC on Dataset-A (b) LogLoss on Dataset-A (c) AUC on Dataset-B (d) LogLoss on Dataset-B

Figure 5: Ablation studies on two datasets.



(a) Layer-0 user embeddings (b) Layer-3 user embeddings

Figure 6: In-depth analysis for user embeddings using t-SNE.

systems in recent years [25], since more explainable models allow for more consistent performance. In this part, we use specific case studies to illustrate that our model is explainable with respect to prediction results. For the sake of simplicity, we will take the performance of model on Dataset-A as the example.

For each user, we filter out his click records from the dataset, while observing the persuasion factor conditions under which these clicks occur. For instance, if the maximum amount of the user's clicks is from the presence of *service* persuasion factor, we regard this user as a *service*-concern user. From this perspective, each user can be regarded as concerned a certain kind of persuasion factor. We want to investigate whether our model can learn the persuasion factor that each user concerns, and distinguish difference preferences on persuasion factors between users. To solve this problem, we conduct experiments by in-depth analysis for user embeddings.

We select four groups of users which concern about four types of persuasion factors, *location*, *facility*, *roomtype* and *food*, in hotel recommendation scenario, respectively. We use t-SNE to shrink the embeddings of these users into 2-D plane. From Figure 6, we find that before information propagation (layer-0 user embeddings), the embeddings of four groups of users is randomly distributed throughout the 2-D plane. However, after using the graph convolutional layer, we can see that the embedding of users (layer-3 user embeddings) who concern about the same type of persuasion factor are almostly clustered together in 2-D spatial regions. We also find that for users who concern about the same persuasion factor, the embeddings are also not too close to each other in the 2-D plane. It is explainable because, in addition to preference about persuasion factors, users still vary in other features, leading to the difference of their embeddings. Overall, our model is capable to express users' preferences on persuasion factors, while also well reflecting the differences among users, which is critical for better recommendation performance.

5 RELATED WORK

Click-through rate prediction. Click-through rate prediction is of great importance for many Internet business companies that provide services to users and it is also a significant research field in recommender systems [13, 19, 23, 24, 38]. Deep neural networks (DNN)-based CTR prediction [3, 33, 41] has become a paradigm in this research area nowadays. A common approach is to embed features to learnable dense vectors then use deep neural networks for feature interactions. Qu et al. [22] introduce product layer before DNN to perform complex and adequate feature interactions. He et al. [11] replace product layer with bi-interaction pooling layer for feature interactions, and Yang et al. [35] use field-aware and attention mechanism to conduct feature interactions. However, a significant disadvantages of using DNN for higher-order feature interactions is implicit, and even the elements within the same field embedding vector will influence each other. Then, other models [15, 25, 28] try to not rely on DNN but explicitly achieve finite-order feature interactions to improve the performance.

In recent years, graph neural network (GNN) has been widely noticed and utilized in the field of CTR prediction. Ying et al. [36] combine random walks and graph convolutions to generate item embeddings. Ouyang et al. [20] focus on cold-start ads and build a graph to connect old and new ads and adaptively distill useful information. Guo et al. [8] utilize dual graph embedding to alleviate feature sparsity and user behavior sparsity problem. He et al. [10] build hypergraphs to yield modal-specific representations of users and micro-videos to better capture user preferences. Besides, we model the effect of persuasion factors on user decision with GCN, which enables better model performance and much more explainability.

Counterfactual learning-based data augmentation. Counterfactual learning is an effective method to implement data augmentation, which is a common way to address data sparsity problem. Wang et al. [32] develop a counterfactual learning-based sampler model to generate new user behavior sequences based on the observed ones. Zhang et al. [39] model counterfactual data distribution to identify noisy and indispensable behaviors and replace dispensable and indispensable concepts within the original concept behaviors. Xiong et al [34] investigate the influence of feature-level

interests on user decisions as well as augment training samples by intervening the feature-level interests of users in a counterfactual manner. In our work, we make reasonable assumptions about user behaviors and utilize counterfactual learning to generate effective new training samples, which contributes a lot to overall performance.

6 CONCLUSION

In this work, we handle recommendation from a brand new perspective, modeling the effect of persuasion factors, which has not been well explored by existing works, and propose an effective method POEM. We first build a graph where users and items are nodes and persuasion factors are edges, then perform information propagation by graph convolutional networks to learn their representations. After that, considering the heterogeneity between users and items, we present user-sensitivity based prediction. We also propose a counterfactual-based data augmentation method to alleviate data-sparsity problem. Extensive experiments verify the effectiveness of our model and the effect of persuasion factors, which is a new field that is worthy of further research.

ACKNOWLEDGMENT

This work is supported in part by National Key Research and Development Program of China under 2020YFA0711403, and by National Natural Science Foundation of China under 61972223, 61971267 and U1936217. This work is also supported by Meituan.

REFERENCES

[1] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.

[2] Weiyu Cheng, Yanyan Shen, and Linpeng Huang. 2020. Adaptive Factorization Network: Learning Adaptive-Order Feature Interactions. arXiv:1909.03276 [cs.LG]

[3] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Recsys*. 191–198.

[4] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. 2019. Deep session interest network for click-through rate prediction. *arXiv preprint arXiv:1905.06482* (2019).

[5] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, et al. 2021. Graph Neural Networks for Recommender Systems: Challenges, Methods, and Directions. *arXiv preprint arXiv:2109.12843* (2021).

[6] Martin Grohe. 2020. word2vec, node2vec, graph2vec, x2vec: Towards a theory of vector embeddings of structured data. In *SIGMOD*. 1–16.

[7] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

[8] Wei Guo, Rong Su, Renhao Tan, Huifeng Guo, Yingxue Zhang, Zhirong Liu, Ruiming Tang, and Xiuqiang He. 2021. Dual Graph enhanced Embedding Neural Network for CTR Prediction. arXiv:2106.00314 [cs.IR]

[9] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*. 1025–1035.

[10] Li He, Hongxu Chen, Dingxian Wang, Shoaib Jameel, Philip Yu, and Guandong Xu. 2021. Click-Through Rate Prediction with Multi-Modal Hypergraphs. In *CIKM*. 690–699.

[11] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. arXiv:1708.05031 [cs.IR]

[12] Tongwen Huang, Zhiqi Zhang, and Junlin Zhang. 2019. FiBiNET: combining feature importance and bilinear feature interaction for click-through rate prediction. In *Recsys*. 169–177.

[13] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware factorization machines for CTR prediction. In *Recsys*. 43–50.

[14] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[15] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *SIGKDD*. 1754–1763.

[16] Bin Liu, Ruiming Tang, Yingzhi Chen, Jinkai Yu, Huifeng Guo, and Yuzhou Zhang. 2019. Feature generation by convolutional neural network for click-through rate prediction. In *WWW*. 1119–1129.

[17] Fan Liu, Zhiyong Cheng, Lei Zhu, Zan Gao, and Liqiang Nie. 2021. Interest-aware message-passing gcn for recommendation. In *WWW*. 1296–1305.

[18] Shang Liu, Wanli Gu, Gao Cong, and Fuzheng Zhang. 2020. Structural relationship representation learning with graph embedding for personalized product search. In *CIKM*. 915–924.

[19] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. 2013. Ad click prediction: a view from the trenches. In *SIGKDD*. 1222–1230.

[20] Wentao Ouyang, Xiuwu Zhang, Shukui Ren, Li Li, Kun Zhang, Jinmei Luo, Zhaojie Liu, and Yanlong Du. 2021. Learning Graph Meta Embeddings for Cold-Start Ads in Click-Through Rate Prediction. In *SIGIR*.

[21] Aditya Pal, Chantat Eksombatchai, Yitong Zhou, Bo Zhao, Charles Rosenberg, and Jure Leskovec. 2020. Pinnersage: Multi-modal user embedding framework for recommendations at pinterest. In *SIGKDD*. 2311–2320.

[22] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-based neural networks for user response prediction. In *ICDM*. IEEE, 1149–1154.

[23] Steffen Rendle. 2010. Factorization machines. In *ICDM*. IEEE, 995–1000.

[24] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: estimating the click-through rate for new ads. In *WWW*. 521–530.

[25] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *CIKM*. 1161–1170.

[26] Zhulin Tao, Xiang Wang, Xiangnan He, Xianglin Huang, and Tat-Seng Chua. 2020. Hoafm: A high-order attentive factorization machine for CTR prediction. *Information Processing & Management* 57, 6 (2020), 102076.

[27] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[28] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *ADKDD*. 1–7.

[29] Weiqing Wang, Hongzhi Yin, Xingzhong Du, Wen Hua, Yongjun Li, and Quoc Viet Hung Nguyen. 2019. Online user representation learning across heterogeneous social networks. In *SIGIR*. 545–554.

[30] Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. 2018. Tem: Tree-enhanced embedding model for explainable recommendation. In *WWW*. 1543–1552.

[31] Xiang Wang, Tinglin Huang, Dingxian Wang, Yancheng Yuan, Zhenguang Liu, Xiangnan He, and Tat-Seng Chua. 2021. Learning Intents behind Interactions with Knowledge Graph for Recommendation. In *WWW*. 878–887.

[32] Zhenlei Wang, Jingsen Zhang, Hongteng Xu, Xu Chen, Yongfeng Zhang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Counterfactual data-augmented sequential recommendation. In *SIGIR*. 347–356.

[33] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *WSDM*. 495–503.

[34] Kun Xiong, Wenwen Ye, Xu Chen, Yongfeng Zhang, Wayne Xin Zhao, Binbin Hu, Zhiqiang Zhang, and Jun Zhou. 2021. Counterfactual Review-based Recommendation. In *CIKM*. 2231–2240.

[35] Yi Yang, Baile Xu, Shaofeng Shen, Furao Shen, and Jian Zhao. 2020. Operation-aware neural networks for user response prediction. *Neural Networks* 121 (2020), 161–168.

[36] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *SIGKDD*. 974–983.

[37] Yuan Yuan, Fengli Xu, Hancheng Cao, Guozhen Zhang, Pan Hui, Yong Li, and Depeng Jin. 2021. Persuade to click: Context-aware persuasion model for online textual advertisement. *TKDE* (2021).

[38] Li Zhang, Weichen Shen, Shijian Li, and Gang Pan. 2019. Field-aware Neural Factorization Machine for Click-Through Rate Prediction. arXiv:1902.09096 [cs.LG]

[39] Shengyu Zhang, Dong Yao, Zhou Zhao, Tat-Seng Chua, and Fei Wu. 2021. Causerec: Counterfactual user sequence synthesis for sequential recommendation. In *SIGIR*. 367–377.

[40] Shijie Zhang, Hongzhi Yin, Tong Chen, Quoc Viet Nguyen Hung, Zi Huang, and Lizhen Cui. 2020. Gcn-based user representation learning for unifying robust recommendation and fraudster detection. In *SIGIR*. 689–698.

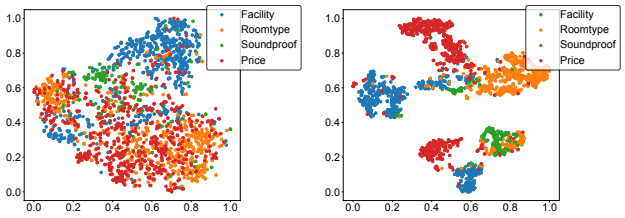
[41] Lei Zheng, Vahid Noroozi, and Philip S Yu. 2017. Joint deep modeling of users and items using reviews for recommendation. In *WSDM*. 425–434.

[42] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *AAAI*, Vol. 33. 5941–5948.

A APPENDIX FOR REPRODUCIBILITY

A.1 Additional Results of Case Study

For items, we can also deploy a similar approach to analyze their embeddings. Assuming that the maximum amount of the clicks belong to the item is under the presence of the *environment* persuasion factor, we regard this item as an *environment*-matching item. We also select four groups of items which match four persuasion factors, *facility*, *roomtype*, *soundproof* and *price*. According to the same analysis as user embeddings, we present the results in Figure 7. From the results, our model also achieves a balance between modeling the item persuasion factor matching relations and its individual attributes.



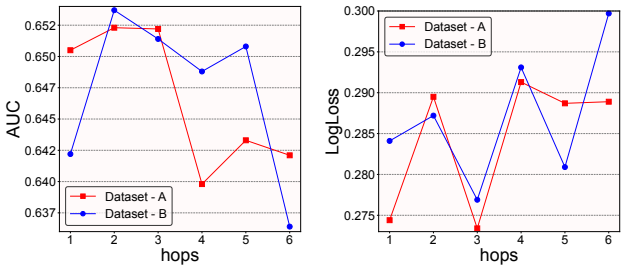
(a) Layer-0 item embeddings (b) Layer-3 item embeddings

Figure 7: In-depth analysis for item embeddings using t-SNE.

A.2 Hyper-parameter Study

In this section, we explore the impact of different hyperparameters on the proposed model for two datasets. The hyperparameters include the number of propagation layers, aggregator and Sensitivity-MLP structure. Other hyperparameters such as node dropout rate and message dropout rate also affect the performance of model based on our further research, but we omit the results due to the space limit.

A.2.1 Effect of the Number of Propagation Layers. We test the performance of different numbers of propagation layers, also called *hops*. Based on the existing works [9, 31**?**], we search for optimal hops in {1, 2, 3, 4, 5, 6} to find the most suitable number of propagation layers. The results are shown in Figure 8.



(a) AUC (b) LogLoss

Figure 8: Performance of different propagation layers.

From the results, we can find that our proposed model can achieve relatively higher performance with three propagation layers *w.r.t.* a higher AUC and a lower LogLoss on Dataset-A, *w.r.t.* a competitive AUC and much lower LogLoss on Dataset-B. Although the model with two propagation layers achieves the highest AUC on Dataset-B, but its performance of LogLoss is much worse. The phenomenon about the effect of number of propagation layers on model performance is understandable and explainable. If we only have few propagation layers, only low-order connectivity, i.e., nearest neighbors, can contribute to information propagation, limiting the amount of information. However, if we stack too many propagation layers, the performance of our model can also be adversely affected. Since the nodes in the graph obtain information from its neighbors, too many propagation layers could make all nodes aggregate information from almost all other nodes in the graph, making embeddings of nodes indistinguishable, and this is the over-smoothing of graph representations. Hence, considering the need for information propagation and prevent from overfitting, we empirically recommend to use three propagation layers.

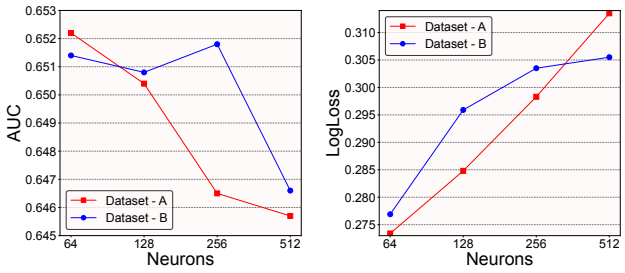
A.2.2 Effect of Aggregator. We also analyze the influence of aggregator on the performance of our model, just using one part of our aggregation function (add or concatenate) and the results are shown in Table 5. We find that *Mixed Aggregator* outperforms other two aggregators significantly. Mixed Aggregator combines the advantages of the other two approaches for information aggregation, so it should be a relatively better aggregator choice.

Table 5: Performance of different aggregators.

Model	Dataset-A		Dataset-B	
	AUC	LogLoss	AUC	LogLoss
Add	0.6491	0.2824	0.6471	0.2848
Concat	0.6495	0.2840	0.6476	0.2838
Mixed	**0.6522**	**0.2734**	**0.6514**	**0.2769**

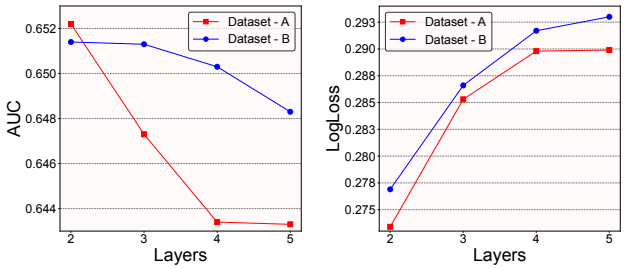
A.2.3 Effect of Sensitivity-MLP Structure. The structure of Sensitivity-MLP contains three aspects: the number of neurons per layer, the depth of hidden layers and the shape of neural networks. They are all hyperparameters that we need to adjust.

- **Effect of the number of neurons per layer.** We search the number of neurons per layer in {64,128,256,512}. Increasing the number of neurons per layer introduces more learnable parameters. We present model performance with different number of neurons in Figure 9. Experimental results show that 64-128 neurons are relatively better choices for two datasets. The reason is that an overly complex model is prone to overfitting the training data, which could explain the decrease of performance when the number of neurons becomes large.
- **Effect of the depth of hidden layers.** We search the depth of hidden layers in {2,3,4,5}, and show the results in Figure 10. We find that our model can reach the best performance when the hidden layer is shallow. As the hidden layers get deeper, the model starts to behave worse. Specifically, two hidden layers are appropriate choice for both datasets. This kind of phenomenon is also due to overfitting.
- **Effect of the shape of neural networks.** Existing works [7] point out that the shape of neural network has an impact on the performance of the model. We test three different shapes of networks: increasing, constant and decreasing network. During



(a) AUC (b) LogLoss

Figure 9: Performance of different neurons per layer.



(a) AUC (b) LogLoss

Figure 10: Performance of different depth of hidden layers.

Table 6: Performance of different neural network shapes.

Model	Dataset-A		Dataset-B	
	AUC	LogLoss	AUC	LogLoss
Constant	**0.6522**	0.2734	0.6514	**0.2769**
Increasing	0.6483	0.2893	**0.6515**	0.2892
Decreasing	0.6505	**0.2706**	0.6497	0.2817

our test, we fix the number of hidden layers as two and the number of total neurons as 128. Then the three kinds of network shape are: increasing (32,96), constant (64,64) and decreasing (96,32). As we can see in Table 6, the constant shape of neural network achieves relatively better performance. However, we can also know from the results that the decreasing shape of neural network gets a lower LogLoss on Dataset-A than the constant shape. Then these is a balance: if you are more concerned about AUC (like ranking tasks), you should choose the constant network shape. If a lower LogLoss is what you need (requires accuracy of CTR prediction), then the decreasing network shape will meet your demand without losing too much AUC.