electronics

Article

RGMeta: Enhancing Cold-Start Recommendations with a Residual Graph Meta-Embedding Model

Fuzhe Zhao [1], Chaoge Huang [1], Han Xu [1], Wen Yang [1] and Wenlin Han [2,*]

[1] School of Computer Science, Central China Normal University, Wuhan 430079, China; zhaofzh@ccnu.edu.cn (F.Z.); hcg@mails.ccnu.edu.cn (C.H.); han_xu77@mails.ccnu.edu.cn (H.X.); mio2736885580@mails.ccnu.edu.cn (W.Y.)
[2] Department of Computer Science, California State University, Fullerton, CA 92834, USA
[*] Correspondence: whan@fullerton.edu

Abstract: Traditional recommendation models grapple with challenges such as the scarcity of similar user or item references and data sparsity, rendering the cold-start problem particularly formidable. Meta-learning has emerged as a promising avenue to address these issues, particularly in solving the item cold-start problem by generating meta-embeddings for new items as their initial ID embeddings. This approach has shown notable success in enhancing the accuracy of click-through rate predictions. However, prevalent meta-embedding models often focus solely on the attribute features of the item, neglecting crucial user information associated with it during the generation of initial ID embeddings for new items. This oversight hinders the exploitation of valuable user-related information to enhance the quality and accuracy of the initial ID embedding. To tackle this limitation, we introduce the residual graph meta-embedding model (RGMeta). RGMeta adopts a comprehensive approach by considering both the attribute features and target users of both old and new items. Through the integration of residual connections, the model effectively combines the representation information of the old neighbor items with the intrinsic features of the new item, resulting in an improved initial ID embedding generation. Experimental results demonstrate that RGMeta significantly enhances the performance of the cold-start phase, showcasing its effectiveness in overcoming challenges associated with sparse data and limited reference points. Our proposed model presents a promising step forward in leveraging both item attributes and user-related information to address cold-start problems in recommendation systems.

Keywords: cold-start; ID embedding; meta-learning; residual connection

check for updates

Citation: Zhao, F.; Huang, C.; Xu, H.; Yang, W.; Han, W. RGMeta: Enhancing Cold-Start Recommendations with a Residual Graph Meta-Embedding Model. *Electronics* **2024**, *13*, 3473. https://doi.org/10.3390/electronics13173473

Academic Editor: George Angelos Papadopoulos

Received: 17 July 2024
Revised: 16 August 2024
Accepted: 29 August 2024
Published: 1 September 2024

1. Introduction

Driven by the booming development of the internet, massive amounts of data and information are generated every day. For individuals with limited information processing ability, it is difficult for individuals to find valuable content from the vast amount of information. In order to solve the problem of information overload, recommendation systems [1] have emerged. These recommendation systems provide personalized services to users and quickly recommend items that meet the characteristics of users from massive amounts of data, thus reducing the search costs of users. Personalized services require accurately identifying user preferences, and click behavior is often seen as an indication of users expressing their preferences. Therefore, click-through rate (CTR) prediction has attracted widespread attention from both academia and industry [2–4].

Utilizing the latest advances in deep learning has become a dominant direction for CTR prediction [5–7]. In recent years, research based on deep learning has identified several models for CTR prediction, such as the deep neural network (DNN) [5], product-based neural network (PNN) [6], DeepFM [7], Wide&Deep [5], and Deep&Cross [8]. These deep models can usually be decomposed into two parts, namely, an embedding layer and multi-layer perceptron (MLP) [9,10]. The embedding layer is used to convert the original input features into a dense real-valued vector representation, thereby capturing richer semantic information and overcoming the limitations of one-hot encoding [11]. The embedding vectors are then input into complex models that can be viewed as different MLP types.

In addition, the application of self-supervised learning and comparative learning in recommendation systems has also made great progress. Traditional self-supervised recommendation systems are limited by manual data augmentation methods and cannot adapt to different data and scenarios to generate high-quality self-supervised learning signals. To fill this gap, AutoCF [12] has adopted a unified method to automatically generate data augmentation. By introducing the masked graph autoencoder architecture, this framework can effectively extract self-supervised signals and perform data augmentation without manually specifying contrast views. Most graph-contrastive learning frameworks use heuristic data augmentation strategies to construct contrastive pairs, which can easily lead to the loss of important information. Recently, a candidate-aware graph-contrastive learning (CGCL) framework [13] was proposed to explore the relationship between users and candidate items in different layers of representation, using similar semantic representations to construct contrastive pairs. Specifically, constructing structural-neighbor contrastive learning objectives, candidate-contrastive learning objectives, and candidate structural-neighbor contrastive learning objectives can help obtain high-quality node representations.

These recent advances provide more effective self-supervised signal extraction and data augmentation methods for recommendation systems. These methods are generally applicable to scenarios with rich data, aiming to improve the overall performance of recommendation systems in these scenarios. However, in practical applications, recommendation systems still face some challenges, especially when dealing with new items, known as the cold-start problem [14,15]. In recommendation systems, CTR prediction is a critical task, typically represented by embedding item IDs into dense vectors. But when a new item is added to the candidate pool, due to its ID feature not appearing in the training data, the new item often lacks available embedding vectors, resulting in a decrease in prediction accuracy. This issue further highlights the necessity of effectively generating and utilizing embedding vectors in cold-start scenarios. Well-learned ID embeddings can greatly improve the prediction accuracy of cold-start items compared to methods that do not use ID inputs [16–18]. In this respect, the application of meta-learning is relatively successful.

DropoutNet [16] uses dropout in model training to improve the robustness of the recommendation model and the effectiveness of item ID embedding; MetaEmb [17] generates an ideal initial ID embedding vector for each new item by training the embedding generator; GME [18] constructs an item graph that utilizes the attribute features of new items and neighboring items to learn the initial ID embeddings required for each new item. However, these methods have some limitations. DropoutNet does not leverage collaborative information and cannot fully train item ID embedding. MetaEmb only considers the new item itself and ignores the information that can help improve predictive performance in existing old items. Although GME considers both new and old items, it ignores the target user information of the item and only focuses on the attribute features, which means the initial ID embedding of the item needs to be improved.

To enhance the initial ID embedding of the new item, we propose a residual graph meta-embedding model (RGMeta). For a new item, RGMeta utilizes the inherent attributes of items to associate old items related to its attributes with the item, calculates the similarity score of attributes and target users of the neighbor items, which in turn calculates the weighted sum of attributes and target user at-tributes of neighbor items. Then, RGMeta processes the attribute-weighted sum matrix and the target user attribute-weighted sum matrix of neighbor items through residual operation to obtain a residual matrix that differs significantly from the original matrix. The residual matrix is overlaid with the original attribute matrix to obtain a more refined attribute embedding representation and target user's attribute embedding for the new item, thereby generating the initial ID embedding for the item. The major contributions of this paper are as follows:

- A meta-learning-based method, RGMeta, is proposed to solve the item cold-start problem by generating initial ID embeddings for new items. RGMeta serves as a meta-embedding model for generating initial ID embeddings and can be applied as a separate module to recommendation models that use ID embeddings.
- RGMeta further strengthens the connection between items by considering both the attribute features and target user attributes of the new and old items. The initial ID embedding of the new item is enhanced by introducing the residual operation to obtain a modified attribute embedding representation and target user embedding representation for the new item.
- Experiments were conducted on public datasets. The experimental results show that compared to the main meta-learning methods on the MovieLens-1M datasets, the AUC values of RGMeta have increased by averages of 1.5%, 1.1%, and 0.26%, respectively, thereby improving the prediction performances of cold-start problems.

2. Related Work

CTR prediction is a binary classification task. For users and products, their characteristics are generally composed of ID features and attribute features (such as the user's age and gender, the product's price and category, etc.). Each instance (x, y) represents an interaction between a user and an item, with x representing item and user features, and y representing user feedback, i.e., whether the user clicked on the item. A typical deep learning recommendation model follows the framework shown in Figure 1 [19].

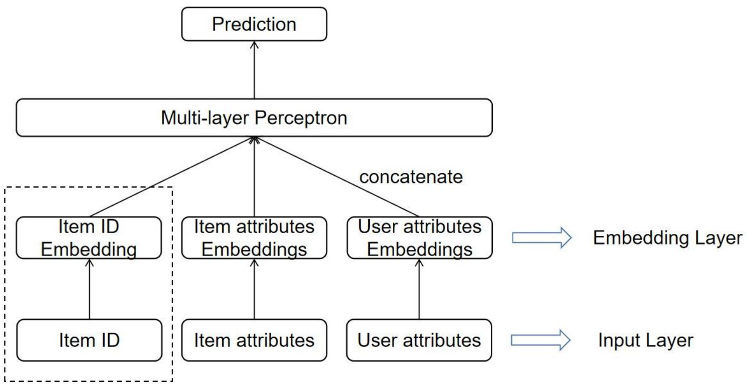

Figure 1. A typical deep learning recommendation model.

In the input layer, the model input samples can be divided into three parts, as follows:

$$x_{\text{input}} = (ID, z_i, u_i) \tag{1}$$

where ID is the unique identifier of the item; z_i represents the attribute features of item i and can have multiple fields; u_i represents the target user attribute features of the item, which can also have multiple fields.

In the embedding layer, embedding techniques can convert the original features into dense vectors, as follows:

$$x_{emb} = \left(i_{emb}, z_{i_{emb}}, u_{i_{emb}}\right) \tag{2}$$

The concatenation operation then concatenates the embeddings of all the input features into a long embedding vector, as follows:

$$e = \left[i_{emb} \| z_{i_{\text{emb}}} \| u_{i_{\text{emb}}}\right] \tag{3}$$

where | | is a connection operator. In the hidden layer that captures nonlinear high-order interactions, typically in multi-layer perceptron (MLP), the long embedding vector e is transformed into a high-level representation vector, s, through multiple fully connected (FC) layers to utilize nonlinear and high-order data interactions.

$$s = f_l(\cdots f_2(f_1(e))) \tag{4}$$

where l is the number of FC layers, and f_j is the jth layer's FC layer. Finally, the prediction layer consists of a linear layer and an active layer, and the prediction result is as follows:

$$\hat{p} = \sigma(Ws + b) \tag{5}$$

where $\hat{p} \in [0, 1]$ denotes the final prediction result, with 0 denoting that the user did not click on the item, 1 denoting that the user clicked on the item, $\sigma(\cdot)$ denoting the sigmoid activation function; and W and b representing the learnable parameters in the linear layer, namely weights and biases.

For model training, this paper learns model parameters through cross-entropy loss on the training datasets. The loss function is as follows:

$$\text{loss} = -\frac{1}{M}\Sigma_{y \in M}[y \log \hat{y} + (1 - y)\log(1 - \hat{y})] \tag{6}$$

where M is the number of training samples, y is the true label, and $\hat{y}$ is a prediction label.

3. Residual Graph Meta-Embedding Model

3.1. Overview

RGMeta only generates initial ID embeddings for new items, so the ID embeddings for new and old items are different. Figure 2 shows the differences between them. When the ID of the item is given, the first step involves searching for the presence of this ID embedding in the trained embedding matrix. If it can be found, the item is considered an old item, and the found ID embedding is directly used in subsequent training. Otherwise, it is a new item, and RGMeta is used to generate the initial ID embedding for the item.

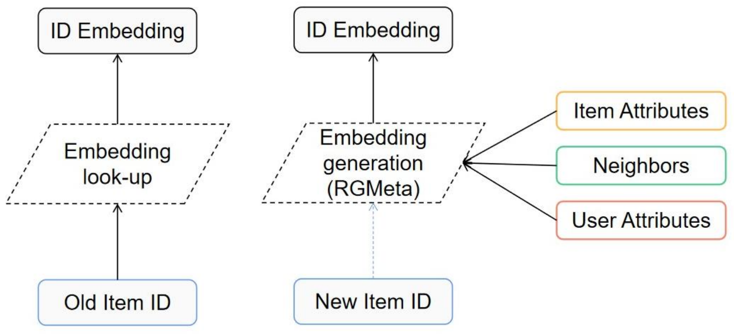

Figure 2. Example of ID embeddings for new and old items.

RGMeta constructs the high-level attribute embedding representation of the new item and the target user's attribute embedding representation by using the weighted sums of the attribute features and the target user attribute features of the new item and neighboring items. Then, it generates the initial ID embedding for the new item based on this foundation.

3.2. Model Design

The structure of RGMeta is shown in Figure 3, which consists of three modules: the first module, which refines the embedding of item attributes, is represented by a black dotted box; the second module, which refines the target user's attribute embedding, is represented by the green dotted box; the third module, which generates the initial embedding of the item ID, is represented by the red dashed box.

The item attribute embedding module calculates the attribute similarity between the new item and neighboring items using the graph attention network (GAT) [20], and generates an improved item attribute embedding. The target user's attribute embedding module integrates the target user's attribute features and generates a weighted summation user attribute embedding by calculating the similarity with the target user attributes of neighboring items. A residual operation then processes a weighted summation matrix to generate a residual matrix, which is overlaid with the original matrix to obtain a more stable and accurate attribute embedding. Finally, the refined attribute embedding is used by the embedding generator to generate ID embedding for the new item. The following is an analysis of each module separately:

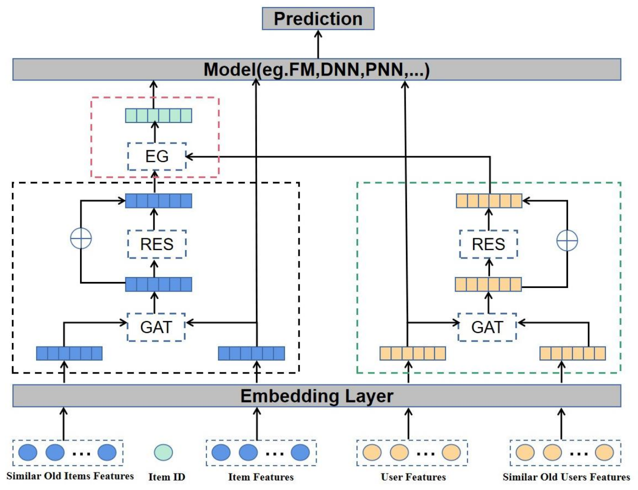

Figure 3. The framework of RGMeta.

3.2.1. Refine Item Attribute Embedding

The input features of the example in this paper are represented as [ID,s,t], where ID is the identification of the item, s denotes the attribute features of the item, such as price, category, etc., and t denotes the target user attribute features of the item, such as age, gender, etc. It is usually possible to observe the associated item attributes (s) and the user attributes (t) in the training data, then find the embedding corresponding to s and obtain a long-connected embedding vector (z). Based on z, the similarity between the attribute features of the new item and neighbor items is calculated.

We use graph attention networks (GATs) [20] to calculate the similarity between the attribute features of new and adjacent items. The attention coefficient between the new item's attribute embedding z_0 and the neighboring item's attribute embedding z_k is calculated as follows:

$$p_{z_{0k}} = \varphi(W_z z_0, W_z z_k) \tag{7}$$

where W_z is a shared weight parameter corresponding to the item attributes, which is used to transform the input into higher-level features and obtain sufficient expressiveness. φ is an attention mechanism function implemented by a single-layer feed-forward neural network, parameterized by the weight vector a_z corresponding to the item attribute, and utilizes the LeakyReLU nonlinear activation function [21]. In order to facilitate the comparison of coefficients between the new item and neighboring item nodes, we use the softmax function to normalize these values and obtain the normalized attention coefficient:

$$\alpha_{z_{0k}} = \frac{\exp\left(\text{LeakyRelu}\left(a_z^\top [W_z z_0 \| W_z z_k]\right)\right)}{\sum_{k\in N} \exp\left(LeakyRelu\left(a_z^\top [W_z z_0 \| W_z z_k]\right)\right)} \tag{8}$$

where N is the number of neighbor items and | | is the connection operation. Then, the weighted sum of the attribute embedded z_0 of the new item and the attribute embedded z_k of neighbor items is calculated as follows:

$$\bar{z}_0' = \sigma\left(\Sigma_{k\in N} \alpha_{z_{0k}} W_z z_k\right) \tag{9}$$

where σ is the activation function; here, we use the exponential linear unit activation function ELU [22], which allows for encoding positive and small negative signals. After obtaining the weighted sum representation of neighbor items, we perform a residual operation on it and then add to the original weighted sum as a refined attribute embedding representation of the new item:

$$\bar{z}_0 = F\left(\bar{z}_0'\right) + \bar{z}_0' \tag{10}$$

where $F(\cdot)$ represents residual operation. The residual operation uses a residual module to process the attribute-weighted sum matrix of neighbor items, resulting in a residual matrix that differs significantly from the original matrix. Fusion $F(\bar{z}_0')$ can take into account the similarities or differences between the features of neighbor items and the new item, and emphasize or suppress certain aspects of the features of the neighbor items to accommodate the individual characteristics of the new item, resulting in a more refined representation of the attribute embedding.

3.2.2. Refine the Target User's Attribute Embedding

Finding the embedding corresponding to the target user attribute and obtaining a long-connected embedding vector, u. We calculate the attention coefficient between the new item's target user's attribute embedding u_0 and the neighbor item's target user's attribute embedding u_k as follows:

$$p_{u_{0k}} = \varphi(W_u u_0, W_u u_k) \tag{11}$$

where W_u is the shared weight parameter corresponding to the target user attribute. Then the weight vector a_u of the target user attribute is parameterized to obtain the normalized attention coefficient of the target user attribute between the new item and neighbor items:

$$\alpha_{u_{0k}} = \frac{\exp\left(\text{LeakyRelu}\left(a_u^\top [W_u u_0 \| W_u u_k]\right)\right)}{\sum_{k\in N} \exp\left(LeakyRelu\left(a_u^\top [W_u u_0 \| W_u u_k]\right)\right)} \tag{12}$$

We calculate the weighted sum representation of the target user's attribute embedding of the new item u_0 and the target user's attribute embedding of neighbor items u_k based on the normalized attention coefficient, as follows:

$$\bar{u}_0' = \sigma\left(\Sigma_{k\in N} \alpha_{u_{0k}} W_u u_k\right) \tag{13}$$

We perform the residual operation on it, and then add it to the original weighted sum, as follows:

$$\bar{u}_0 = F\left(\bar{u}_0'\right) + \bar{u}_0' \tag{14}$$

$F(\bar{u}_0')$ performs feature enhancement on the original target user attribute weighting and representation to obtain more refined target user attribute embeddings for the new item.

3.2.3. Generate Initial ID Embedding

The above two steps yield the refined attribute embedding and the refined target user attribute embedding for the new item. Next, we generate the initial ID embedding for the new item by using the embedding generator (EG):

$$r_0 = \tanh(V[\bar{z}_0 \| \bar{u}_0]) \tag{15}$$

where V is the parameter of the fully connected layer and tanh is the activation function. The resulting initial ID embedding is generated by using item attributes and target user attributes. The combination of attribute features and target user attribute features provides more information about items and users so that the generated embedding can capture key attribute features and demonstrate better predictive ability in the model.

3.3. Model Training

We use the main prediction model to pre-train old items and obtain embedding vectors of the item and user features as well as model parameters θ. Since θ is trained using a large amount of data, its effectiveness can be fully guaranteed. We view CTR prediction as an example of meta-learning, the learning problem for each item ID is considered a unique task and uses gradient-based meta-learning methods [19], which generalize model-independent meta-learning (MAML) [23]. Each task shares the same set of parameters θ in the prediction model. Therefore, when training the RGMeta model, we freeze θ, and only learn specific parameters, $\omega \overset{\Delta}{=} (W,a,V)$, in the model to reduce the cost of repetitive training.

We use pre-trained old items to simulate the cold-start process, randomly selecting two disjoint mini-batches of labeled data, D_1 and D_2, from the given training set of old items. Each data type has K samples. The initial embedding r_0 of the item ID is generated on the first mini-batch D_1 using the RGMeta model. For the ith sample in D_1, the prediction result is $\hat{y}_{1i}$, and the average loss of the sample is calculated as follows:

$$l_1 = -\frac{1}{K}\Sigma_{y_{1i}\in K}[y_{1i}\log \hat{y}_{1i} + (1 - y_{1i})\log(1 - \hat{y}_{1i})] \tag{16}$$

We simulate the learning process during the warm-up phase on the second mini-batch D_2. A new adaptive ID embedding is obtained by updating the meta-parameter ω in the mini-batch through random gradient descent, as follows:

$$r_0' = r_0 - \varepsilon\frac{\partial l_1}{\partial r_0} \tag{17}$$

where $\varepsilon > 0$ is the step size of gradient descent. The average loss is then calculated as follows:

$$l_2 = -\frac{1}{K}\Sigma_{y_{2i}\in K}[y_{2i}\log \hat{y}_{2i} + (1 - y_{2i})\log(1 - \hat{y}_{2i})] \tag{18}$$

In the first mini-batch D_1, l_1 is a natural metric used to evaluate the generator during the cold-start phase since using the generated initial embeddings for prediction. For the second mini-batch D_2, as the embedding has been updated once, l_2 can directly evaluate the sample efficiency in the warm-up phase. In order to unify these two losses, the average losses of l_1 and l_2 are selected as the final loss function:

$$\text{loss} = \gamma l_1 + (1-\gamma) l_2 \tag{19}$$

where $\gamma \in [0, 1]$ is the equilibrium coefficient. Since the warm-up phase usually takes more time than the cold-start phase, the model needs to pay more attention to the warm-up phase to achieve rapid adaptation.

The training algorithm in this paper updates meta-parameters in mini-batches by random gradient descent. The pseudo-code of the algorithm is demonstrated in Algorithm 1.

Algorithm 1: Train RGMeta by SGD.

```
Input: f_θ: the pre-trained base model.
Input: N: the set of old item IDs.
Input: γ: hyperparameter, the coefficient for meta-loss.
Input: ε, τ: step sizes.
1: Randomly initialize $\omega$
2: while not done do
3:     Randomly sample m items from N
4:     for i in range(0,m) do
5:         Use RGMeta to generate the initial ID embedding: $r_i$
6:         Sample mini-batches $D_1$ and $D_2$ each with K samples
7:         Evaluate loss $l_1$ on $D_1$
8:         Compute the adapted embedding: $r_0' = r_0 - \varepsilon\frac{\partial l_1}{\partial r_0}$
9:         Evaluate loss $l_2$ on $D_2$
10:        Compute the final loss: loss = $\gamma\, l_1 + (1-\gamma)\, l_2$
11:    Update $\omega \leftarrow \omega - \tau\sum_{i=1}^{m}\frac{\partial loss}{\partial \omega}$
```

4. Experiments

4.1. Datasets

We evaluate the performance of the proposed RGMeta models on two real-world datasets, whose statistics are listed in Table 1.

The MovieLens-1M dataset (http://www.grouplens.org/datasets/movielens/) (accessed on 6 May 2023): This dataset contains 1 million movie rating instances; each movie has a unique ID and can be seen as an item. The relevant attribute features include the release year, title, and genre, while user features include user ID, gender, age, and occupation. In order to verify the predictive performance of the proposed method at different phases, we preprocess the datasets. The movies are rated from 1 to 5; we convert the ratings of at least 4 to label 1, which means that the item was clicked, and the other ratings are converted to label 0, which means that the item was not clicked.

The Taobao Ad dataset (https://tianchi.aliyun.com/dataset/dataDetail?dataId=408) (accessed on 2 November 2023): It is collected from the traffic logs in Taobao and contains 26 million click records from 1.14 million users within 8 days. Each ad can be considered an item, with features such as ad ID, campaign ID, category ID, brand ID, and price. Each user has 9 features: user ID, CMS group ID, micro group ID, gender, age, shopping depth, consumption level, occupation, and city level.

Table 1. Statistics of experimental datasets.

Dataset	**#Fields**	**#Old Item IDs**	**#Samples to Train the Main Prediction Model**	**#Samples to Train the Cold-Start ID Embedding Model**	**#New Item IDs**	**# Samples for Warm-Up Training**	**#Samples for Testing**
MovieLens-1M	8	1058	765,669	43,320	1127	67,620	123,787
Taobao Ad	23	62,209	3,592,047	1,784,000	531,593	810,000	109,712

4.2. Backbone and Baseline

4.2.1. Backbone

The main prediction models of deep learning can handle high-dimensional and sparse features, which are suitable for recommendation applications in CTR prediction tasks. We conducted experiments on the following backbones:

- DNN: This is a deep neural network that includes an embedding layer, multiple FC layers, and an output layer [5]. Unlike traditional shallow neural networks, DNNs have multiple hidden layers, each of which can learn different levels of abstract features, thus solving some complex tasks better.
- DeepFM: This consists of a factorization machine (FM) and deep neural network (DNN) [7]. The FM part is used to model the second-order interaction between features. It is based on a Factorizer model and can effectively capture sparse interactions between features. The deep part is similar to traditional deep neural networks and is used to learn higher-order representations of features.
- Wide&Deep: This model consists of logistic regression (LR) and DNN, which can model low-order and high-order feature interactions [5]. The wide part is used to learn generalized cross-terms between features, and it can capture the linear relationship between features well. The deep part is used to learn higher-order representations of features, capturing nonlinear relationships between features.
- Deep&Cross: This model is divided into two parts: deep and cross [8]. The deep part is similar to traditional deep neural networks and is used to learn higher-order representations of features. In the cross part, the interactive information between features is mined through the calculation of cross features. It is a combination of deep

learning and generalized linear models, aiming to solve the problem of traditional models when dealing with high-dimensional sparse features.

- PNN: Different from the traditional model based on feature crossing, it introduces a production layer into DNN [6]. The PNN model models the second-order interaction between features by introducing product vectors, and learns the higher-order representation of features through the deep layer, to better capture the relationship between features and further improve the accuracy of prediction.

4.2.2. Baseline

We select four models that generate initial ID embeddings for new items to solve the cold-start problem as baselines.

- NgbEmb [18]: For each selected neighbor item, its pre-trained ID embedding is already available. These embeddings, derived from historical data and model training on old items, effectively capture item features. NgbEmb utilizes the embedding information from these adjacent old items to generate the initial ID embedding for the new item. This generation process typically involves techniques such as weighted averaging, clustering, or other synthesis methods to effectively transfer the embedded features from old items to the new item. NgbEmb is used as a baseline method to evaluate the effectiveness of generating new item embeddings based solely on the old item's information.
- MetaEmb [17]: Before generating ID embeddings for new items, it is necessary to first clarify the attribute characteristics of the new project. These features can include various types of information such as project category, description, price, etc. These attribute features provide rich information for the context of new items, enabling embeddings to better reflect their uniqueness. Using the attribute features of the new item, MetaEmb generates initial embeddings through specific algorithms. This process aims to transform different attribute features into a unified embedding vector, providing a more comprehensive representation of the new item. MetaEmb serves as a baseline for only considering new items.
- GME-A [18]. The GME-A model not only relies on the independent features of new items but also correlates the attribute information of old items, to comprehensively consider more data when generating ID embeddings. The core of this method lies in combining the features of new and old items to provide a more representative and robust initial embedding for new items. This enables subsequent recommendation or classification tasks to be based on richer information. GME-A is therefore regarded as a baseline model that focuses exclusively on item attribute features. It generates preliminary embedding representations by deeply mining the attribute relationships between items, without relying on user behavior data.
- CoMeta [24]: The CoMeta model consists of two submodules, namely B-EG and S-EG, which utilize collaborative information to enhance the generated meta-embeddings. Specifically, for a new item, B-EG computes a base embedding by calculating the weighted sum of the ID embeddings of similar old items. Meanwhile, S-EG generates a shift embedding that incorporates the item's attribute features as well as the average ID embedding of users who have interacted with it. The final meta-embedding is obtained by summing the base embedding and the shift embedding.

4.3. Evaluation Metrics

AUC: The area under the ROC curve is a widely used metric to evaluate the performance of binary classification models. It reflects the probability that the model ranks the randomly selected positive examples higher than the randomly selected negative examples. The larger the AUC, the better the prediction performance; its slight improvement is likely to lead to a significant increase in online CTR [25].

We follow [10] to introduce the RelaImpr metric to measure relative improvement over models. For a random guesser, the value of AUC is 0.5. Hence, RelaImpr is defined as follows:

$$\text{RelaImpr} = \left(\frac{\text{AUC (measured model) }) - 0.5}{\text{AUC (base mode) } - 0.5} - 1 \right) \times 100\% \tag{20}$$

Cross-entropy loss: This is a commonly used loss function [26] that is particularly suitable for classification tasks. Cross-entropy loss calculates the difference between the predicted result and the real result by comparing the predicted click probability of the model with the actual click probability of the label. The smaller the cross-entropy loss, the smaller the difference between the probability distribution predicted by the model and the distribution of the real label.

4.4. Experimental Results

We conducted experiments during the cold-start phase and two rounds of warm-up phases. The results for various ID embedding models based on different CTR prediction models at the cold-start phase are shown in Table 2.

Table 2. Test AUC and loss. Backbone: prediction model. Method: ID embedding generation model. AUC is the larger the better. RelaImpr is the larger the better. Loss is the smaller the better.

Backbone	**Method**	**MovieLens_1M**			**Taobao Ad**		
		AUC	**RelaImpr**	**Loss**	**AUC**	**RelaImpr**	**Loss**
DNN	NgbEmb	0.7132	1.19%	0.6436	0.6081	0.93%	0.2053
	MetaEmb	0.7138	1.47%	0.6436	0.6103	2.99%	0.2017
	GME-A	0.7235	6.07%	0.6321	0.6198	11.86%	0.1967
	CoMeta	0.7217	5.22%	0.6330	0.6144	6.82%	0.1984
	RGMeta	0.7253	6.93%	0.6176	0.6256	17.27%	0.1953
DeepFm	NgbEmb	0.7133	1.23%	0.6435	0.6161	0.87%	0.2066
	MetaEmb	0.7136	1.38%	0.6433	0.6185	2.95%	0.2013
	GME-A	0.7233	5.98%	0.6326	0.6232	7.04%	0.1967
	CoMeta	0.7220	5.36%	0.6320	0.6219	5.91%	0.2016
	RGMeta	0.7242	6.41%	0.6108	0.6280	11.21%	0.1959
Wide&Deep	NgbEmb	0.7166	2.8%	0.6521	0.6166	0.78%	0.4042
	MetaEmb	0.7132	1.19%	0.6457	0.6173	1.38%	0.2292
	GME-A	0.7207	4.75%	0.6382	0.6236	6.83%	0.2012
	CoMeta	0.7187	3.80%	0.6399	0.6215	5.01%	0.1985
	RGMeta	0.7219	5.32%	0.6235	0.6266	9.42%	0.1962
Deep&Cross	NgbEmb	0.7102	0.96%	0.6431	0.6081	0.75%	0.2309
	MetaEmb	0.7146	3.07%	0.6476	0.6081	0.75%	0.2005
	GME-A	0.7171	4.27%	0.6534	0.6122	4.57%	0.1976
	CoMeta	0.7146	3.07%	0.6313	0.6120	4.38%	0.1956
	RGMeta	0.7212	6.24%	0.6258	0.6217	13.42%	0.1952
PNN	NgbEmb	0.7061	1.32%	0.6404	0.6025	0.69%	0.2112
	MetaEmb	0.7131	4.77%	0.6475	0.6051	3.24%	0.2088
	GME-A	0.7154	5.90%	0.6248	0.6080	6.09	0.2042
	CoMeta	0.7152	5.80%	0.6455	0.6072	5.30%	0.2051
	RGMeta	0.7166	6.49%	0.6222	0.6214	19.25%	0.2027

In the first round of the warm-up phase, we provide some training examples related to new items for the main CTR models, but different embedding generation models provide different initial ID embeddings. In the second round of the warm-up phase, additional training examples related to new items are provided for the main CTR model. This training example is different from the first warm-up training because they are trained based on different ID embeddings learned after the first warm-up training.

In Table 1, in the case of MovieLens-1M, it can be observed that MetaEmb increases the AUC value over NgbEmb by an average of 0.39%, which indicates that using attribute features of the new item can alleviate cold-start problems, whereas simply considering the pre-training of neighboring ID embeddings is not effective. GME-A increases the AUC value by an average of 0.89% over MetaEmb as GME-A aggregates useful information from its neighbors directly at the attribute level, which can better capture associations between similar items and, thus, improve the performance in the cold-start phase. CoMeta performs slightly worse than GME-A but better than the other two methods.

Compared to the AUC values of NgbEmb, MetaEmb, GME-A, and CoMeta, RGMeta increases by averages of 1.5%, 1.1%, 0.26%, and 0.47%, respectively. In addition, RGMeta also has the best RelaImpr and loss values. These results indicate that the performance of RGMeta is significantly better than the other three baseline models and can help alleviate the cold-start problem.

Figure 4 presents the performances of various embedding models in the warm-up phase on the DNN prediction model. MetaEmb performs better than Ngb in the warm-up phase although it performs poorly in the cold-start phase, this is because in the warm-up phase, the new item already has some click-through and display rate data, at which time the use of pre-trained neighbor ID embeddings can better establish the embedding representation of the new item, the data of the neighbor items can provide more contextual information. New items in the warm-up phase may still face the challenge of representation learning because they still have relatively few behavioral characteristics. In this case, GME-A can still help new items obtain richer initial embedding representations by aggregating attribute information from their neighbors, thus showing good performance in the warm-up phase. CoMeta performs better than GME-A in the warm-up phases because CoMeta utilizes the collaborative information of the interaction items and does not only consider the attribute information.

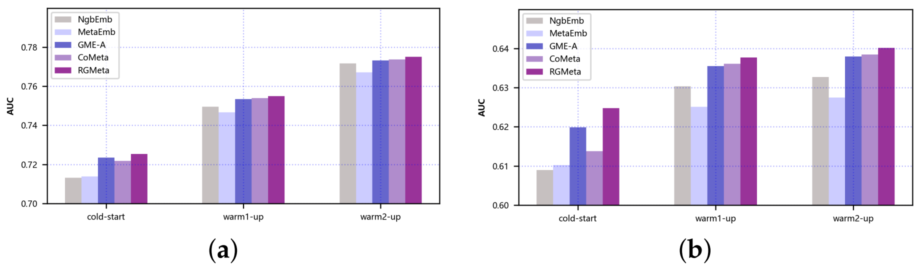

Figure 4. Performance in the warm-up phase on the DNN prediction model. (**a**) MovieLens-1M (**b**) Taobao Ad.

RGMeta not only shows better performance in the cold-start phase but also performs better than other baseline models in the warm-up phase. It has been observed that models that produce good performance in the cold-start phase usually produce good performance in the warm-up phase.

4.5. Ablation Studies

4.5.1. Effect of Equilibrium Coefficient

The equilibrium coefficient γ is used to balance the cold-start loss and the warm-up loss. We perform experiments with values of γ between 0.1 and 0.9 on the MovieLens-1M dataset and the result is shown in Figure 5. It was observed that Ngb was relatively insensitive; CoMeta, GME-A, and RGMeta were relatively sensitive; and the AUC values decreased gradually with increasing values. At a value of 0.1, the AUC of RGMeta reached a maximum of 0.7253, which in turn indicated that the RGMeta had a better performance.

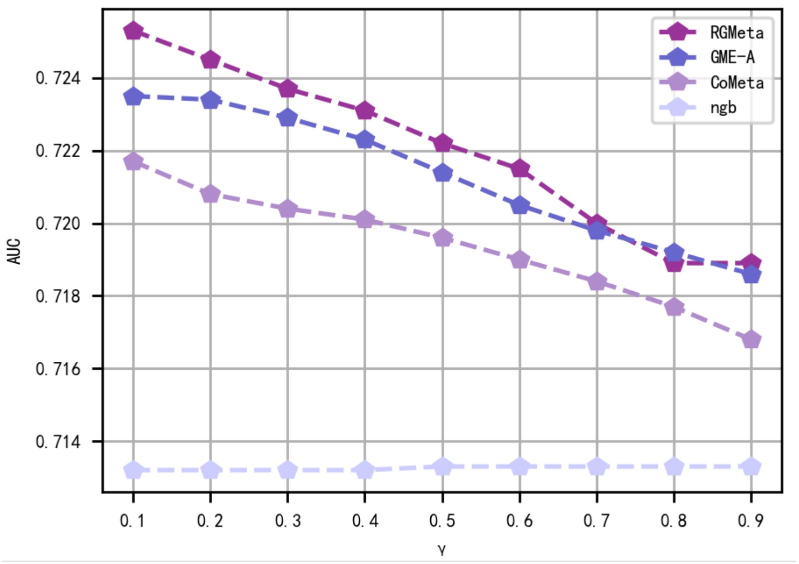

Figure 5. Effect of the equilibrium coefficient (the main prediction model is DNN).

4.5.2. Effect of the Number of Neighborhood Items

We consider the effects of the number of neighboring items on the model. Figure 6 shows the results of the experiments on the MovieLens-1M dataset. The number of neighbors increases from 2 to 10, the value of AUC is also relatively small when the number is small. The number is 10, the value of AUC increases by 1.2% compared to number 2, and RGMeta achieves better performance. This indicates that it is difficult to obtain enough useful information from too few similar items and the performance of the model improves when more neighbors are available.

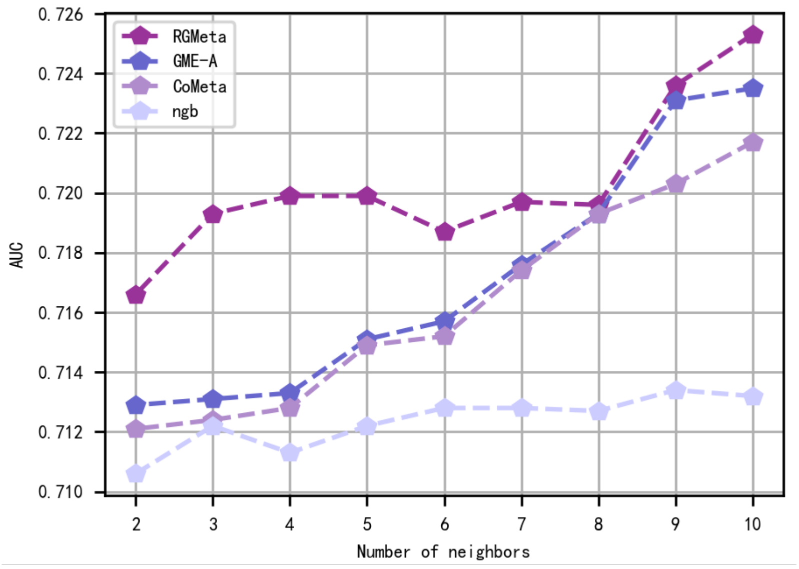

Figure 6. Effect of the number of neighborhood items (main prediction model is DNN).

4.5.3. Effect of Model Components

In order to study the effects of different components in RGMeta, we performed ablation studies on different prediction models: (1) GME-A: baseline. (2) RGMeta/UF: without target user information. (3) RGMeta/Res: without residual operation. (4) RGMeta: general framework. The experimental results of the cold-start phase are shown in Table 3.

Table 3. Results of ablation studies.

Backbone	**Emb.Model**	**MovieLens_1M**		**Taobao Ad**	
		AUC	**Loss**	**AUC**	**Loss**
DNN	GME-A	0.7235	0.6321	0.6198	0.1967
	RGMeta/UF	0.7245	0.6320	0.6215	0.1916
	RGMeta/Res	0.7250	0.6227	0.6224	0.1985
	RGMeta	0.7253	0.6176	0.6256	0.1953
DeepFm	GME-A	0.7233	0.6326	0.6232	0.1967
	RGMeta/UF	0.7234	0.6242	0.6250	0.1971
	RGMeta/Res	0.7237	0.6252	0.6257	0.1968
	RGMeta	0.7242	0.6108	0.6280	0.1959
Wide&Deep	GME-A	0.7207	0.6382	0.6236	0.2012
	RGMeta/UF	0.7217	0.6246	0.6248	0.1982
	RGMeta/Res	0.7214	0.6279	0.6252	0.1980
	RGMeta	0.7219	0.6235	0.6266	0.1962
Deep&Cross	GME-A	0.7171	0.6534	0.6122	0.1976
	RGMeta/UF	0.7188	0.6347	0.6136	0.2035
	RGMeta/Res	0.7198	0.6408	0.6139	0.2003
	RGMeta	0.7212	0.6258	0.6217	0.1952
PNN	GME-A	0.7154	0.6248	0.6080	0.2042
	RGMeta/UF	0.7157	0.6247	0.6111	0.2060
	RGMeta/Res	0.7164	0.6294	0.6106	0.2054
	RGMeta	0.7166	0.6222	0.6214	0.2027

Using the MovieLens-1M dataset as an example, the AUC value of RGMeta/UF increases by 0.11% on average compared to GME-A, which indicates that the residual operation makes full use of the attribute features of neighboring items to generate ID embeddings for new items, which improves the quality and effectiveness of ID embeddings. The AUC value of RGMeta/Res increased by an average of 0.18% compared to that of GME-A, indicating that simultaneously considering target user information can generate better initial ID embeddings for new items than solely considering item attribute features. This improvement is due to a deeper connection between items at the level of target user information, which better utilizes the data from old items. The AUC value of RGMeta increased by 0.26% compared to GME-A, which in turn improved the performance of the RGMeta.

5. Conclusions

In this paper, we propose a residual graph meta-embedding model. Unlike traditional methods, RGMeta introduces the target user attributes when calculating the weighted sum of attributes of neighboring items. Through this process, RGMeta not only takes into account the similarity between neighboring items but also incorporates more contextual features through the target user's information to more accurately predict the characteristics of new items. In addition, RGMeta introduces residual operations. The residual operation uses a residual module to process the weighted sum matrix of attributes and the weighted sum matrix of target user attributes of neighboring items to obtain a residual matrix that differs significantly from the original matrix. By summing the residual matrix with the original feature matrix, information from neighboring items is used, and weighting of different nodes is achieved to distinguish differences between nodes. This allows RGMeta to effectively learn the similarities and differences between different items and generate more distinguishable initial ID embeddings. The experimental results on two datasets demonstrate that our proposed RGMeta has good performance and compatibility.

In the future, we will explore how to extract more representative information from neighbors to further improve prediction performance. In addition, the interpretability of the RGMeta model still needs to be strengthened. We will consider using methods such as SHAP (Shapley additive explanation) and LIME (local interpretable model agnostic explanations) to analyze which attribute features and target user features have the greatest impact on initial ID embedding, and how these features affect the final prediction results of the model.

Author Contributions: Conceptualization, F.Z.; methodology, F.Z.; software, C.H.; validation, F.Z. and C.H.; formal analysis, C.H.; investigation, H.X.; resources, W.Y.; data curation, H.X.; writing—original draft preparation, C.H.; writing—review and editing, F.Z. and W.H.; visualization, H.X. and W.Y.; supervision, W.H. All authors have read and agreed to the published version of the manuscript.

Funding: There is no funding in this paper.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. He, L.; Xia, L.; Zeng, W.; Ma, Z.; Zhao, Y.; Yin, D. Off-policy learning for multiple loggers. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 1184–1193.
2. Ouyang, W.; Zhang, X.; Ren, S.; Li, L.; Liu, Z.; Du, Y. Click-through rate prediction with the user memory network. In Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data, Anchorage, AK, USA, 4–8 August 2019; pp. 1–4.
3. Xia, Y.; Cao, Y.; Hu, S.; Liu, T.; Lu, L. Deep intention-aware network for click-through rate prediction. In Proceedings of the Companion Proceedings of the ACM Web Conference, Austin, TX, USA, 30 April–4 May 2023; Association for Computing Machinery: New York, NY, USA, 2023; pp. 533–537.
4. Kamal, M.; Bablu, TA. Machine learning models for predicting click-through rates on social media: Factors and performance analysis. *Int. J. Appl. Mach. Learn. Comput. Intell.* **2022**, *12*, 1–4.
5. Cheng, H.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; et al. Wide & deep learning for recommender systems. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, Boston, MA, USA, 15–19 September 2016; pp. 7–10.
6. Qu, Y.; Fang, B.; Zhang, W.; Tang, R.; Niu, M.; Guo, H.; Yu, Y.; He, X. Product-based neural networks for user response prediction over multi-field categorical data. ACM Trans. *Inf. Syst. (TOIS)* **2018**, *37*, 1–35.
7. Guo, H.; Tang, R.; Ye, Y.; Li, Z.; He, X. DeepFM: A factorization-machine based neural network for CTR prediction. *arxiv* **2017**, arxiv:1703.04247.
8. Wang, R.; Fu, B.; Fu, G.; Wang, M. Deep & cross network for ad click predictions. In Proceedings of the ADKDD'17, Halifax, NS, Canada, 13–17 August 2017; pp. 1–7.
9. Valanarasu, J.M.J.; Patel, V.M. Unext: Mlp-based rapid medical image segmentation network. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Singapore, 18–22 September 2022; pp. 23–33.
10. Zhou, G.; Zhu, X.; Song, C.; Fan, Y.; Zhu, H.; Ma, X. Deep interest network for click-through rate prediction. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 1059–1068.
11. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. In Proceedings of the 26th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013.
12. Xia, L.; Huang, C.; Huang, C.; Lin, K.; Yu, T.; Kao, B. Automated self-supervised learning for recommendation. In Proceedings of the ACM Web Conference 2023, Austin, TX, USA, 30 April 30–4 May 2023; pp. 992–1002.
13. He, W.; Sun, G.; Lu, J.; Fang, X. Candidate-aware graph contrastive learning for recommendation. In Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, Taipei, Taiwan, 23–27 July 2023; pp. 1670–1679.
14. Wei, C.; Liang, J.; Liu, D.; Dai, Z.; Li, M.; Wang, F. Meta graph learning for long-tail recommendation. In Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Long Beach, CA, USA, 6–10 August 2023; pp. 2512–2522.
15. Wei, Y.; Wang, X.; Li, Q.; Nie, L.; Li, Y.; Li, X.; Chua, T. Contrastive learning for cold-start recommendation. In Proceedings of the 29th ACM International Conference on Multimedia, Chengdu, China, 20–24 October 2021; pp. 5382–5390.
16. Volkovs, M.; Yu, G.; Poutanen, T. Dropoutnet: Addressing cold start in recommender systems. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017.
17. Pan, F.; Li, S.; Ao, X.; Tang, P.; He, Q. Warm up cold-start advertisements: Improving ctr predictions via learning to learn id embeddings. In Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, Paris, France, 21–25 July 2019; pp. 695–704.

18. Ouyang, W.; Zhang, X.; Ren, S.; Li, L.; Zhang, K.; Luo, J.; Liu, Z.; Du, Y. Learning graph meta embeddings for cold-start ads in click-through rate prediction. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Montreal, QC, USA, 11–15 July 2021; pp. 1157–1166.
19. Song, W.; Shi, C.; Xiao, Z.; Duan, Z.; Xu, Y.; Zhang, M.; Tang J. Autoint: Automatic feature interaction learning via self-attentive neural networks. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; pp. 1161–1170.
20. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *arXiv* **2017**, arXiv:1710.10903.
21. Dubey, A.K.; Jain, V. Comparative study of convolution neural network's relu and leaky-relu activation functions. In *Applications of Computing, Automation and Wireless Systems in Electrical Engineering: Proceedings of MARC*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 873–880.
22. Zhao, X.; Ren, Y.; Du, Y.; Zhang, S.; Wang, N. Improving item cold-start recommendation via model-agnostic conditional variational autoencoder. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, 11–15 July 2022; pp. 2595–2600.
23. Finn, C.; Abbeel, P.; Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In Proceedings of the International Conference on Machine Learning, Sydney, Australia, 15–20 July 2017; pp. 1126–1135.
24. Hu, H.; Rong, D.; Chen, J.; He, Q.; Liu, Z. CoMeta: Enhancing meta embeddings with collaborative information in cold-start problem of recommendation. In Proceedings of the International Conference on Knowledge Science, Engineering and Management, Guangzhou, China, 16–18 August 2023; pp. 213–225.
25. Qin, J.; Zhang, W.; Wu, X.; Jin, J.; Fang, Y.; Yu, Y. User behavior retrieval for click-through rate prediction. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Xian, China, 25–30 July 2020; pp. 2347–2356.
26. Ho, Y.; Wookey, S. The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. *IEEE Access* **2019**, *8*, 4806–4813. [CrossRef]