Gemini: A Novel and Universal Heterogeneous Graph Information Fusing Framework for Online Recommendations

Jixing Xu*
DiDi BizTech Dept.
Beijing, China
xujixing@didiglobal.com

Zhenlong Zhu
DiDi BizTech Dept.
Beijing, China
zhuzhenlong@didiglobal.com

Jianxin Zhao
DiDi BizTech Dept.
Beijing, China
zhaojianxin_i@didichuxing.com

Xuanye Liu
DiDi BizTech Dept.
Beijing, China
liuxuanye@didiglobal.com

Minghui Shan
DiDi BizTech Dept.
Beijing, China
shanminghui@didiglobal.com

Jiecheng Guo
DiDi BizTech Dept.
Beijing, China
jasonguo@didiglobal.com

ABSTRACT

Recently, network embedding has been successfully used in recommendation systems. Researchers have made efforts to utilize additional auxiliary information (e.g., social relations of users) to improve performance. However, such auxiliary information lacks compatibility for all recommendation scenarios, thus it is difficult to apply in some industrial scenarios where generality is required. Moreover, the heterogeneous nature between users and items aggravates the difficulty in network information fusion. Many works tried to transform user-item heterogeneous network to two homogeneous graphs (i.e., user-user and item-item), and then fuse information separately. This may limit the representation power of learned embedding due to ignoring the adjacent relationship in the original graph. In addition, the sparsity of user-item interactions is an urgent problem need to be solved.

To solve the above problems, we propose a universal and effective framework named Gemini, which only relies on the common interaction logs, avoiding the dependence on auxiliary information and ensuring a better generality. For the purpose of keeping original adjacent relationship, Gemini transforms the original user-item heterogeneous graph into two semi homogeneous graphs from the perspective of users and items respectively. The transformed graphs consist of two types of nodes: network nodes coming from homogeneous nodes and attribute nodes coming from heterogeneous node. Then, the node representation is learned in a homogeneous way, with considering edge embedding at the same time. Simultaneously, the interaction sparsity problem is solved to some extent as the transformed graphs contain the original second-order neighbors. For training efficiently, we also propose an iterative training algorithm to reduce computational complexity. Experimental results on the five datasets and online A/B tests in recommendations of DiDiChuXing show that Gemini outperforms state-of-the-art algorithms.

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403388

CCS CONCEPTS

• **Mathematics of computing → Graph algorithms**; • **Computing methodologies → Learning latent representations**.

KEYWORDS

network embedding; heterogeneous graph; recommendation

ACM Reference Format:
Jixing Xu*, Zhenlong Zhu, Jianxin Zhao, Xuanye Liu, Minghui Shan, and Jiecheng Guo. 2020. Gemini: A Novel and Universal Heterogeneous Graph Information Fusing Framework for Online Recommendations. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3394486.3403388

1 INTRODUCTION

With the rapid development of the Mobile Internet, recommendation system has become an effective way to help user to get personal interest (e.g., news, videos, products) from massive information resources, and also the key solution to help industry to achieve greater business growth (e.g., e-commerce order, advertising revenue). As one of the most popular online serving platforms of immediate ride-hailing in the world, DiDiChuxing serves hundreds of millions of active passengers and millions of active drivers. For different requirements of daily operation, DiDiChuxing builds a dozen different types of recommendation scenarios, such as coupon recommendation for passenger growth, product recommendation for driver incentives and advertising recommendation for order growth. In different recommendation scenarios, the recommended items are completely different and the information available is very different. So how to build a unified recommendation system to address the recommendation requirements of all these scenarios is a huge challenge, especially when facing the sparsity problem of user-item interactions. With the development of deep learning, many neural models[4, 11] have been proposed to directly learn low dimensional representations of features (e.g. age, gender, category) through huge amounts of click-through log data. These representations can capture memorization and generalization of features and can be used to match user's interest. However, these representations do not directly capture the global topology relations between users and users and between items and items (e.g., neighbor relations). In recent years, network embedding has attracted

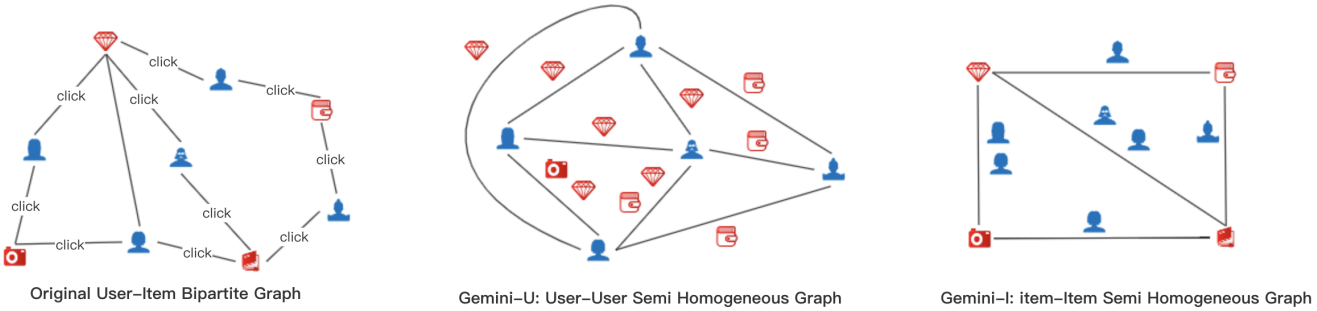

Figure 1: Examples of transformation from user-item bipartite graph to Gemini-U and Gemini-I

extensive attentions from researchers and has been successfully applied in recommendation system. Network embedding can fuse information based on network topology and capture the relations between nodes (e.g., first-order or second-order neighbors) by distributing a low dimensional representation for each node. As the vectors learned in the same space, they can be used to measure the similarity of the entities, which is crucial in recommendation systems.

Several works[18, 34] have demonstrated that introducing additional auxiliary information (e.g., comment text, query words) can further improve effectiveness. However, these information is a double-edged sword, improving performance while reducing the model's versatility. As mentioned above, the various type of users (e.g., passenger, driver) and items (e.g., discount coupons, marketing ads) make it difficult to implement such algorithms in a unified recommendation system. Therefore, we urgently need a much more universal network embedding that can be applied to all scenarios. In a recommendation scenario, the most common information is user-item interaction logs, which can be used to build a typical user-item bipartite network. So in this paper, we try to propose a more effective and universal network embedding based on the most common user-item interactions without relying on any auxiliary information.

There are two major challenges need to be solved. One is how to deal with heterogeneous user and item nodes in network. Algorithms [12, 21] that deal with homogeneous graph may not work well. Research[28] converts user-item network to an item-item homogeneous network first, and then adopts random walk to capture the co-occurrences between items. Its downside is that it cannot preserve the topology relations between users. There are also a few researches based on heterogeneous networks. To list a few, researches[6, 7, 15] rely on meta-path (e.g., user-user-item-item) that requires expert domain knowledge. Research[34] utilizes additional auxiliary information to transform the user-item heterogeneous network to two separate homogeneous networks (i.e., user-user and item-item). Its downside is that it may limit the representation power of learned embedding due to missing the user-item neighbor relations in original network topology. Its network information fusions are done separately in the two transformed graphs, so the fusion processes of users and items do not directly affect each other. However, we found that combining the two network fusion processes can further improve performance. BiNE[8] transforms user-item heterogeneous network to user-user and item-item homogeneous networks without relying on auxiliary information. It preserves the first-order neighborhood relations between users and items, but still separately does the two network information fusions. Another challenge is how to solve the sparsity of user-item interactions. In a real recommendation scenario of industry, a user usually interacts with only a small number of items while an item can only be exposed to a small number of users, which results in a very sparse user-item network and limits the effectiveness of embedding representation.

To solve the above problems, we propose a Graph Convolution[17] based heterogeneous graph information fusing framework named Gemini. Firstly, in order to be applied in various types of recommended scenarios in DiDiChuxing, Gemini only utilizes the most common user-item click logs. Secondly, instead of working directly on the user-item heterogeneous network, Gemini transforms it into two semi homogeneous graphs (i.e., Gemini-U and Gemini-I in Figure 1) from the perspective of users and items respectively. Gemini-U and Gemini-I have similar structures that consist of network nodes and attribute nodes, just as twins, which is the origin of its name. From the perspective of users, if two users both click the same some items, then they have some common interests, thus adding an edge in Gemini-U. These items are the attribute nodes of the edge, referred to as Att-U for simplicity. In addition to providing item embeddings to represent the common interests of the user pair, Att-U can also characterizes the importance of the edge in two ways. One way is the quantity of items, the more items there are, the more important the edge is. The other way is the importance of the individual item, and the more important the item is, the more important the edge is. Some previous[34] works transform heterogeneous user-item graph to user-user homogeneous graph, but losing the original topology relations between user and item. By contrast, the transformation from user-item graph to Gemini-U does not lose any original topology information, because all user nodes, item nodes and the relations between them can still be found in Gemini-U, but just from a different perspective. The advantage of this transformation is homogeneous Graph Convolution Network (GCN) based algorithm can be applied on Gemini-U, just additionally taking into account the item embeddings in Att-U. We extend GCN based algorithm with a novel aggregate stage, in which we

compute attention weights of local neighborhoods of nodes by taking into the quantity and quality of the items in Att-U account, and then apply a weighted sum pooling. The detailed algorithm is shown in Section 4. Similarly, from the perspective of items, we can get Gemini-I and Att-I (i.e., user nodes of edge), and apply same GCN algorithm. Thirdly, in training process, we share the embeddings of users and items between Gemini-U and Gemini-I, then we can correlate the two network information fusing processes very closely through the edge attributes Att-U and Att-I. This brings two benefits: one is that the representations of the two types of nodes are distinct but still in the same low dimensional space; the other is that as network information fusing goes on, item embedding combines the information of its multi-order neighbors, then the neighbor relation information between items can be introduced into user′s information fusing on Gemini-U. Similarly, such information of users is also introduced into Gemini-I. In order to reduce the time complexity of the joint training of the two graphs, we also propose a Gemini-Collaboration algorithm to alternately train them.

The main contributions of this work are summarized in the following:

1)We propose a new heterogeneous graph fusing framework, Gemini, which does not rely on any auxiliary information, and handles heterogeneous graph more effectively through a novel and effective network transformation. Thus, Gemini can be applied to all kinds of recommended scenario and achieve satisfactory results. To our best knowledge, this is the first work to transform heterogeneous graph to two semi homogeneous graphs that does not miss any key topology information.

2)We propose a GCN based algorithm which effectively processes graph edge consisting of heterogeneous nodes by capturing the global importance and local importance of these nodes. Simultaneously, through an attention function, the algorithm focuses on more important homogeneous neighbors in aggregation stage. In addition, adding edge information while aggregating information from neighbor nodes can exchange heterogeneous topology information between Gemini-U and Gemini-I. Thus, the information fusion processes on the two graphs are interdependent. To our best knowledge, this is also the first work to take into account the above optimizations.

3)To some extent, Gemini solves the sparsity problem of user-item interactions. Because, in addition to the first-order neighbor relations of user-item, the second-order neighbor relations of user-user and item-item are introduced to Gemini-U and Gemini-I.

4)We design a training algorithm, Gemini-Collaboration, that enables the Gemini framework to run on a large-scale dataset.

5)We conduct extensive offline experiments and deploy an online A/B tests at DiDiChuxing. Experimental results show the superiority of our Gemini over state-of-the-art algorithms.

2 RELATED WORK

Recently, network embedding[32] has attracted extensive attention. It aims to learn latent low dimensional representations of network nodes based on topology structure. According to the types of nodes in the network, there are two types of network: homogeneous and heterogeneous. For homogeneous network,matrix decomposition based works[1, 3, 33] try to decompose the representations of the nodes from the adjacency matrix of network; random walk based works[10, 21, 24] convert network into articles with nodes as words, and then use SkipGram[20] to get the representations of nodes; SNDE[26] and GraphGAN[27] use deep neural networks to learn network embedding. In addition to the mentioned heterogeneous network algorithms in Introduction section, HEER[23] adds edge representation to bridge the semantic gap between heterogeneous nodes; JUST[16] embeds heterogeneous graphs using random walks with jumps and stops without involving meta-paths.

Nowadays, more attention is focused on GCNs[2, 12, 17, 31]. Graphsage[12] is a typical GCN model that instead of training individual embeddings for each node, it learns a function that generates embeddings by sampling and aggregating local neighbors. GAT[25] believes that different neighbor nodes have different importance, so it adopts attention mechanism to learn the weight of different neighbor nodes.

In a recommendation system, network can be constructed according to user-item interactions. So many works[18, 19, 22, 29, 30, 34, 35] try to introduce network embedding into recommendation system to retrieve the closest items in the embedding space. To list a few, GeoSAGE[30] extends Line[24] to multiple bipartite graphs to obtain the embedding of poi. PGE[19] captures the sequential impact of items by converting historical purchase records into item network, and then adopts the SkipGram to obtain item embedding. The HERec[22] proposes meta-path based random walk strategy to generate meaningful node sequences for network embedding. DiffNet[29] is a typical GCN based algorithm that uses a layer-wise influence propagation structure to model how user embeddings evolve as the social diffusion process continues.

3 PROBLEM DEFINITION

In this section, we introduce some related concepts and notions, and then define the problem.

Definition1. (Interaction Record). An interaction record indicates a single interaction between an certain user and an item, denoted by $< u, v >$, where u denotes the user and v denotes the item.

Definition2. (Bipartite Graph). A bipartite graph $G = (U, V, E)$, where U is the set of users, V is the set of items and $E \subseteq U * V$ is the set of edges between U and V which is based on interaction records. Note that a bipartite graph is composed of two types of nodes: users and items, so it is a heterogeneous graph.

Definition3. (Semi homogeneous Graph). A semi homogeneous graph $G = (X, E, Att)$, where X is the set of node and $E \subseteq X * X$ is the set of edges. Att represents the set of attributes nodes on E, and the type of attribute nodes is different from X. We use $Attr_{i,j}$ to represent the set of attribute nodes on the edges of node x_i and node x_j. If we only consider the topology of G, then G is a homogeneous Graph. If we also consider edge attributes, then G is not a homogeneous Graph. In this case, we call G semi homogeneous.

Definition4. (User-Item Recommendation). Given a user u, the task of user-item recommendation is to recommend a list of items that u would be interested based on the interaction records. A bipartite graph $G = (U, V, E)$ can be built based on interaction records so that we can transform the user-item recommendation problem into a link prediction problem on graph.

4 RESEARCH METHODS

In this section, we detail the proposed framework Gemini, which has four key characteristics: 1) Network Transformation, which transforms user-item heterogeneous graph into two semi homogeneous graphs, Gemini-U and Gemini-I. 2) Edge embedding, which is represented with attribute information. 3) Information Convolution, by which we could learning the user and item representation from the graph efficiently. 4) Gemini-Collaboration, an interactive(alternate) training procedure, which speeds up the training without hurting performance.

4.1 Network Transformation

As shown in Figure 1, the most common network in recommendation system is the bipartite graph that describes the interactions (e.g., click) between users and items. To utilize the bipartite, most works try to transform such a bipartite to user-user graph or/and item-item graph, with some assumed auxiliary information, such as social relations between users. However, the major downside is the reliance on specific auxiliary information may limit the scope of its application and the original user-item topology relationships. Approaches directly model the heterogeneous graph usually encounter sparsity problem as the user-item interaction is sparse. Compared with 1-hop neighbors, nodes have more 2-hop neighbors, which can be used to alleviate the problem of graph sparsity. And compared to the different types of 1-hop neighbors, the 2-hop neighbors have the same node type. So the biggest challenge is to learn network embedding by using second-order neighbor relationships, while taking first-order neighbor relationship into account and not relying on any additional auxiliary information. To conquer this, we transforms user-item heterogeneous graph into two semi homogeneous graphs, Gemini-U and Gemini-I, from the perspective of users and items respectively.

Specifically, if two users both link to the same items, then they have some common interested items, thus adding an edge between them in Gemini-U. These items are the attributes of the new edge, referred to as Att-U as mentioned above. Similarly, if two items both link to the same some users, then they share the same target user group, thus adding an edge between them in Gemini-I. These users are the attributes of the new edge, referred to as Att-I. Att-U and Att-I describe the reason why the edge is built and then the original user-item neighbor relationships are all kept in edges. Then, the nodes in Gemini-U/Gemini-I are all same type and the 1-hop neighbors describe the original seconded-order neighbor relationships. Finally, we can easily extend a GCN based algorithm to learn node representations, not relying on any additional auxiliary information and not missing any topology relationship.

4.2 Edge Embedding

In Gemini framework, edge information is crucial. First of all, the edge attributes (i.e., nodes in Att-U and Att-I) describe the original first-order neighbor relationship and can be used to measure the strength of neighbor relationships of nodes in Gemini-U or Gemini-I. Second, through sharing node embeddings, the edge attributes can provide information of heterogeneous nodes in another graph. This brings two advantages: one is that the topology information of Gemini-U/Gemini-I, especially the high order neighbor relationship, is exchanged to each other; the other is that the separate graphs, Gemini-U and Gemini-I, are closely related to each other and their network information fusion affects each other.

Sum Pooling. An intuitive idea is that the more attribute nodes on edge, the more information the edge contains. Based on this, instead of mean pooling, we use a sum pooling to aggregate edge information as follows:

$$\mathrm{h_e} = \sum_{\mathrm{f} \in \mathrm{Att-U_{i,j}\ or\ Att-I_{i,j}}} \mathrm{h_f} \tag{1}$$

Where $e = Att - U_{i,j}$ *or* $Att - I_{i,j}$ is an edge between node i and node j, f is an attribute nodes of the e, h_e and h_f are the embedding vectors. The downside of this approach is that it ignores the importance of the different attribute node. Usually, different node plays different roles, especially in different edges.

Local & Global Information. Different attribute nodes usually appear different times on the edge due to that, for example, an item may be clicked by two user neighbors many time. The more times a node appears on the edge, the more important it becomes. So the number of times a node appears on an edge describes the importance of the node to the edge, which we call Local Information because that this is from the perspective of a single edge. Conversely, the more edges a node appears on, the less important it is. We call the IDF of node Global Information because that this is from the perspective of all edges.

TF-IDF Pooling. To consider the quantity and quality of the attributes on edge simultaneously, we combine Sum Pooling (i.e., quantity) and TF-IDF (i.e., quality) to propose TF-IDF Pooling as follows:

$$\mathrm{h_e} = \sum_{\mathrm{f} \in \mathrm{Att-U_{i,j}\ or\ Att-I_{i,j}}} \mathrm{h_f} \odot \mathrm{h_{tf-idf_f}} \tag{2}$$

where $\odot$ denotes the element-wise product of two vectors. We divided the TF-IDF values of all nodes from low to high into K slots. For each slot, we learn a parameter vector (i.e., h_{tf-idf_f}) to represent its weight vector.

4.3 Information Convolution

For a central node, the core idea of GCNs is to iteratively use the embeddings of its neighbor nodes to update its own embedding. Here, we extend the typical GCNs for working on Gemini-U and Gemini-I. The key issue is to effectively handle edge embedding when aggregating information from neighbors.

Attention based Aggregating. Our aggregator function is an attention-layer that combines edge embeddings and node embeddings, which can be formulated as follows:

$$\mathrm{h^{k-1}_{N_k(u)}} = \mathrm{AGG}(\mathrm{h^{k-1}_u}, \mathrm{h^{k-1}_d}, \mathrm{h^{k-1}_e}, \forall \mathrm{d} \in \mathrm{N_k(u)}) \tag{3}$$

Where $N_k(u)$ is a fixed-size set of neighbors of u, sampling from the set $\{(u, d) \in E\}$. $h_u^{k-1}, h_d^{k-1} \in \mathbb{R}^{d_{k-1}}$ is the embedding of node u and d after the (k-1)-th convolution-layer. h_e^{k-1} donates the embedding of the edge between node u and node d after (k-1)-th convolution-layer.

When we calculate the attention weights, we need not only information about the neighbor nodes, but also about the edges, so

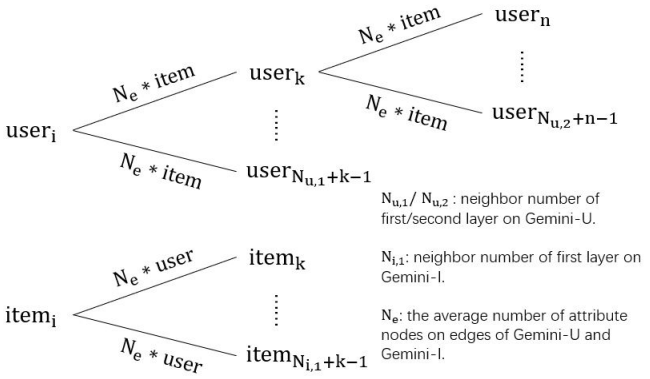

Figure 2: Example of calculation process of Gemini

we need to get the edge vectors first:

$$h_e^{k-1} = \sum_{f \in Att-U_{i,j} or Att-I_{i,j}} h_f^{k-1} \odot h_{tf-idf_f} \tag{4}$$

Where h_f^{k-1} denotes the embedding of attribute node f after the(k-1)-th convolution-layer. The attention aggregator can be calculated as follow:

$$h_{ed}^{k-1} = CONCAT(h_e^{k-1}, h_d^{k-1}) \tag{5}$$

$$q_d = V^T Tanh(W h_u^{k-1} + U h_{ed}^{k-1}) \quad \forall d \in N_k(u) \tag{6}$$

$$q_d = \frac{e^{q_d}}{\sum_{a \in N_k(u)} e^{q_a}} \tag{7}$$

$$AGG(h_u^{k-1}, h_d^{k-1}, h_e^{k-1}) = \sum_{a \in N_k(u)} (q_a * h_{ed}^{k-1}) \tag{8}$$

Where $W \in \mathbb{R}^{d_{k-1} * d_{k-1}}$, $U \in \mathbb{R}^{d_{k-1} * 2d_{k-1}}$ and $V \in \mathbb{R}^{d_{k-1} * 1}$ are parameter matrices.

Edge CONV. After the aggregation, We pass the neighbor information to self node by the following convolution function:

$$h_u^k = \sigma(W^k CONCAT(h_u^{k-1}, h_{N_k(u)}^{k-1})) \tag{9}$$

Where h_u^k is the embedding of node u after the(k)-th convolution-layer. This function can be expressed as $CONV$.

4.4 Gemini Framework

We now summarize our proposed Gemini framework, as illustrated in Algorithm 1. Line 2-12 is the sampling stage of Gemini. Each set U^k contains the nodes that are needed to compute the representations of nodes $u \in U^{k+1}$, the same for each set V^k. Lines 15-20 and 21-27 correspond to the aggregation stage of the user nodes and the item nodes, respectively. Given a user u and an item v, we can get their final representation h_u^k and h_v^k by their respective aggregation and convolution functions. We use two full connection-layers to predict whether the user is interacted (eg.click) in item, and the calculation process is shown in equation 10.

$$p = CLASSIFIER(CONCAT(h_u^k, h_v^k)) \tag{10}$$

p is the probability of the user interacting with the item. In our work, we take the cross entropy as the loss function, and the negative sample is the item exposed by the user but not clicked.

Algorithm 1: Gemini

Input: graphs Gemini-U and Gemini-I; number of-layers K; non-linearity σ; dataset: (u_i, v_i), $\forall i \in \{1, ...n\}$

Output: Hidden states of the K-th layer, include the hidden states of users: $z_u, \forall u \in U$ and the hidden states of items: $z_v, \forall v \in V$

1 Initialize all parameters: user and item embedding matrix H_U^0 and H_V^0; differentiable aggregator functions: AGG_{uk}, AGG_{vk}, $\forall k \in \{1, ...K\}$; differentiable convolution functions: $CONV_{uk}$, $CONV_{vk}$, $\forall k \in \{1, ...K\}$; classifier: $CLASSIFIER$
2 $U^K = U, V^K = V$
3 **for** k = *K-1...1* **do**
4 $U^k = U^{k+1}$
5 **for** $node \in U^k$ **do**
6 $U^k = U^k \cup N_k(node)$
7 **end**
8 $V^k = V^{k+1}$
9 **for** $node \in V^k$ **do**
10 $V^k = V^k \cup N_k(node)$
11 **end**
12 **end**
13 **for** b = *1...batchnum* **do**
14 Sampling batch of tuples $< u, v >$
15 **for** k = *1...K* **do**
16 **for** $u \in U_u^k$ **do**
17 $h_{N_k(u)}^{k-1} = AGG_{uk}(h_u^{k-1}, h_d^{k-1}, h_e^{k-1}, \forall d \in N_k(u))$
18 $h_u^k = CONV_{uk}(CONCAT(h_u^{k-1}, h_{N_k(u)}^{k-1}))$
19 **end**
20 **end**
21 **for** k = *1...K* **do**
22 **for** $v \in V_v^k$ **do**
23 $h_{N_k(v)}^{k-1} = AGG_{vk}(h_v^{k-1}, h_d^{k-1}, h_e^{k-1}, \forall d \in N_k(v))$
24 $h_v^k = CONV_{vk}(CONCAT(h_v^{k-1}, h_{N_k(v)}^{k-1}))$
25 **end**
26 **end**
27 $p = CLASSIFIER(CONCAT(h_u^k, h_v^k))$
28 **end**
29 $z_u = h_u^k, \forall u \in U, z_v = h_v^k, \forall v \in V$

Joint training. When updating the embedding h_u^k of network node in Gemini-U or Gemini-I, the attribute node embedding h_f^{k-1} is also a required input which needs to be computed by the same information convolution on another graph. So Gemini-U and Gemini-I are interdependent and they are jointly trained.

4.5 Gemini-Collaboration Framework

Gemini Complexity. Intuitively, the time complexity of joint training can be very high. For the sake of simplicity, we approximate the number of nodes introduced in the training to the time complexity. Assume that: 1) the neighbor sampling number for K convolution layer on Gemini-U and Gemini-I are $N_{u,z}$ and $N_{i,z}$,

$z \in \{1,...,K\}$; 2) the average number of attribute nodes on edge are N_e.

As shown in Figure 2, for a user, the calculation of its 1-layer embedding requires $N_{u,1}$ 1-hop neighbors and $N_{u,1} * N_e$ attribute nodes (i.e., item nodes). Similarly, the calculation of 1-layer embedding of an item requires $N_{i,1}$ 1-hop neighbors and $N_{i,1} * N_e$ attribute nodes (i.e., user nodes). In the case of K = 2, for a user, its 2-layer embedding requires the 1-layer embeddings of $N_{u,1}$ 1-hop neighbors and $N_{u,1} * N_e$ attribute nodes, which requires $N_{u,1} * N_{u,2} + N_{u,1} * N_e * N_{i,1} * N_e$ user nodes and $N_{u,1} * N_{u,2} * N_e + N_{u,1} * N_e * N_{i,1}$ item nodes, a total of $N_{u,1} * N_{u,2} + N_{u,1} * N_e * N_{i,1} * N_e + N_{u,1} * N_{u,2} * N_e + N_{u,1} * N_e * N_{i,1}$ nodes. $N_{i,1} * N_{i,2} + N_{i,1} * N_e * N_{u,1} * N_e + N_{i,1} * N_{i,2} * N_e + N_{i,1} * N_e * N_{u,1}$ nodes are required to compute the 2-layer embedding of item. As K increases, the amount of computation becomes too much for us to handle.

Therefore, we propose an iterative training method named Gemini-Collaboration to reduce computational complexity. As shown in Algorithm 2, the core idea of Gemini-Collaboration is that in one iteration, when the z-th layer $z \in \{1, \cdots, K\}$ embedding is calculated, the embedding of its attribute nodes is the calculated value of this or last iteration and the attribute embedding is not updated in this lteration. For example, in the first iteration, we first calculate the Gemini-U graph, items as the attribute node, we use $H_i^0(1)$ which denotes the 0-layer embedding of item in the first iteration, to participate in the calculation, and we get $H_u^k(1)$ which is the K-layer embedding of user, then calculate the Gemini-I graph, in this case the user is the attribute nodes, we use $H_u^K(1)$ to participate in the calculation, we get $H_i^K(1)$. In the second iteration, when calculating the Gemini-U graph, item is the attribute node, and $H_i^K(1)$ is used to participate in the calculation to get $H_u^K(2)$ and so on.

Gemini-Collaboration Complexity. In each iteration of Gemini-Collaboration, for a user, $N_{u,1} + N_{u,1} * N_{u,2}$ user nodes and $N_{u,1} * N_e + N_{u,1} * N_{u,2} * N_e$ item nodes participate in the calculation and only $N_{u,1} + N_{u,1} * N_{u,2}$ user nodes are updated. For an item, $N_{i,1} + N_{i,1} * N_{i,2}$ item nodes and $N_{i,1} * N_e + N_{i,1} * N_{i,2} * N_e$ user nodes participate in the calculation and only $N_{i,1} + N_{i,1} * N_{i,2}$ item nodes are updated.

Algorithm 2: Gemini-Collaboration

Input: graphs Gemini-U and Gemini-I; number of layers K; non-linearity σ; dataset: (u_i, v_i), $\forall i \in \{1, ...n\}$

Output: Hidden states of the K-th layer, include the hidden states of users: $z_u, \forall u \in U$ and the hidden states of items: $z_v, \forall v \in V$

Initialize all parameters: user and item embedding matrix $H_u^0(1)$ and $H_v^0(1)$; differentiable aggregator functions: $AGG_{uk}, AGG_{ik}, \forall k \in \{1, ...K\}$; differentiable convolution functions: $CONV_{uk}, CONV_{ik}, \forall k \in \{1, ...K\}$; classifier: $CLASSIFIER$; Neighbor nodes of different layers: $U^K, V^K, \forall k \in \{1, ...K\}$

for b = 1...$iterationnum$ **do**
  $H_u^0(b) = H_u^0(1)$
  $H_v^0(b) = H_v^0(1)$
  while *not convergence* **do**
    Sampling batch of tuples $< u, v >$
    for $u \in U_u^k$ **do**
      $h_e = (b == 1\ ?\ h_e^0(1) : h_e^K(b-1))$
      for k = 1...K **do**
        $h_{N_k(u)}^{k-1}(b) = AGG_{uk}(h_u^{k-1}(b), h_d^{k-1}(b), h_e, \forall d \in N_k(u))$
        $h_u^k(b) = CONV_{uk}(CONCAT(h_u^{k-1}(b), h_{N_k(u)}^{k-1}(b)))$
      end
    end
    $h_v = (b == 1\ ?\ h_v^0(1) : h_v^K(b-1))$
    $p = CLASSIFIER(CONCAT(h_u^K(b), h_v))$
  end
  while *not convergence* **do**
    Sampling batch of tuples $< u, v >$
    for $v \in V_v^k$ **do**
      for k = 1...K **do**
        $h_{N_k(v)}^{k-1}(b) = AGG_{vk}(h_v^{k-1}(b), h_d^{k-1}(b), h_e^K(b), \forall d \in N_k(u))$
        $h_v^k(b) = CONV_{vk}(CONCAT(h_v^{k-1}(b), h_{N_k(v)}^{k-1}(b)))$
      end
    end
    $p = CLASSIFIER(CONCAT(h_u^K(b), h_v^K(b)))$
  end
end
$z_u = h_u^K(iterationnum), \forall u \in U$
$z_v = h_v^K(iterationnum), \forall v \in V$

5 EXPERIMENTS

In this section, we evaluate our framework on five offline datasets and perform online A/B tests on four recommendation scenarios of DiDiChuxing. The launched system architecture is shown in Figure 6 and Appendix A.1. For the baselines, we choose four state-of-the-art methods:

DeepWalk[21]: It involves language model to analyze truncated random walks on a graph. We apply it to user-item recommendation.

Graphsage[12]: It is a state-of-the-art method of GCN that aggregators neighborhood information into center nodes. We used the same classifier as Gemini in Graphsage and implement it for user-item recommendation. We set the number of GCN layers to 2 and 4, respectively.

Metapath2vec++[6]: It captures the relationships between heterogeneous nodes through artificially defined meta-paths. We apply it to the original bipartite user-item graph and to the Gemini-U, Gemini-I graph, and set the meta-path to user-user-item-item.

BiNE[8]: It preserves both explicit preference and implicit relation through joint optimization. We apply it to user-item recommendation.

Gemini-Collaboration(1): It is Gemini-Collaboration with the 1 GCN layer (The formal Gemini-Collaboration has 2 GCN layers).

In offline evaluations, we employ the common metric: area under the ROC curve (AUC). And the final presented results are the average of five experiments. In online A/B tests, we employ the most common industry metric: click through rate (CTR).

5.1 Offline Datasets

In order to verify the universality and effectiveness of Gemini, we choose five different types of recommendation datasets. The first four datasets are extracted from the following recommendation scenarios of DiDiChuxing which is shown in Figure 3: the product recommendation in Integral Mall (DiDi-Product), the content recommendation in DiDiChezhu app (DiDi-Content), the music recommendation in DiDiChuxing FM (DiDi-Music) and the coupon recommendation in DiDiChuxing app (DiDi-Coupon). DiDi-Product, DiDi-Content and DiDi-Music are all the recommendation scenarios that serve tens of millions of drivers, and DiDi-Coupon is a recommendation scenario that serves hundreds of millions of passengers. For the above recommendations, we separately extracted the online logs for 12 days, the first 11 days for the training set, and the twelfth day for the test set. Specifically, DiDi-Product dataset has 8976117 interaction records, DiDi-Content dataset has 7934319 interaction records, DiDi-Music dataset has 3321930 interaction records and DiDi-Coupon dataset has 82666658 interaction records. The last dataset is a common public data, MovieLens[13], which is extracted from a movie recommendation scenario and has about 2006859 records. MovieLens is different from the above four types of recommendations, so we select it as the additional dataset to evaluate performance under different type of recommendations.

For the first four real datasets, we take the items that the user has interacted with positive instance, and the items that have been exposed but not interacted with as negative instance. MovieLens dataset contains positive instances only (i.e., all instances have target value 1), we randomly select twice as many movies as the number of positive instances for each user as negative instances.

5.2 Offline Evaluation

The evaluation results of all the methods are presented in Table 1. In most datasets, BINE works better than Deepwalk, illustrating the need to consider the type of nodes. Graphsage is better than Deepwalk overall, demonstrating the effectiveness of the way the GCN aggregators information. Compared to the best baselines in each scenario, Gemini-Collaboration increases the AUC by 3.7%, 10.8%, 1.1%, 3% and 1%. These results prove the effectiveness and universality of the method.

5.3 Number of GCN layers Evaluation

Gemini-Collaboration(1) aggregators information about original second-order neighbors and contains the same amount of information as Graphsage(2). The Gemini-Collaboration aggregators information about original fourth-order neighbors and contains the same amount of information as Graphsage(4). From table 1, we can see that Graphsage(2) is better than Graphsage(4), but that Gemini-Collaboration (1) is less effective than Gemini-Collaboration. This suggests that Gemini-Collaboration can aggregator neighborhood information more efficiently, and automatically eliminate noise. In most datasets, Gemini-Collaboration(1) is better than Graphsage(2), further proving that we can make better use of the information we already have.

5.4 Superiority of Algorithm Design.

To demonstrate the superiority of Gemini's designs (i.e., Network Transformation, Edge Embedding, TF-IDF pooling, Attention based Aggregating), we conduct several additional experiments on DiDi-Product offline dataset. The results are shown in Table 2. The result of Proposal_1 is better than the best baseline, which proves the correctness of our Gemini framework. Proposal_2 increases the value of AUC to Proposal_1, indicating that attribute nodes have different importance. The AUC of Proposal_3 is higher than that of Proposal_1, proving that the contribution of each neighbor node is different.

5.5 Gemini-Collaboration Evaluation.

In this section, we compare the performance and training time of methods Gemini and Gemini-Collaboration on the same configured machines. The Figure5 shows Gemini-Collaboration get a better performance. The reason is, when using Gemini-Collaboration to train model, we train h_u to convergent firstly and then train h_v to convergent per iteration. It is similar to train discriminator and generator in GAN[9]. On the contrary, Gemini jointly trains Gemini-U and Gemini-I, which means that the embeddings of network nodes are updated based on the not well trained embeddings of attribute nodes. Moreover, Figure5 also demonstrates that Gemini-Collaboration is able to greatly reduce the time complexity in the actual training process.

5.6 Online Evaluation

To evaluate the real online performance of our Gemini, we perform four online A/B tests on the real recommendation scenarios of DiDiChuxing. Because of the preciousness of online experiments, we only compare to the best baseline according to the offline experiments. The evaluation results are presented in Figure 4. In all four scenarios, Gemini achieves the better CTR values in each day of the test week. For the four real scenarios, Gemini increases the average CTR value by 4.13%, 31.03%, 15.45% and 49.13% separately.

6 CONCLUSIONS

In this paper, we first propose a novel heterogeneous graph fusing framework, Gemini, which does not relay on any auxiliary information, but handles heterogeneous graph more efficiently through a novel and effective network transformation. Empirical experiments demonstrate that Gemini can be applied to various types of recommended scenarios and achieve satisfactory results. We then designed a training algorithm, Gemini-Collaboration, that enables the Gemini framework to run on a large-scale dataset.

REFERENCES

[1] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J. Smola. 2013. Distributed Large-scale Natural Graph Factorization. In *Proceedings of the 22nd international conference on World Wide Web*.

[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. [n.d.]. Spectral Networks and Locally Connected Networks on Graphs. ([n. d.]).

[3] Shaosheng Cao, Lu Wei, and Qiongkai Xu. 2015. GraRep: Learning Graph Representations with Global Structural Information.

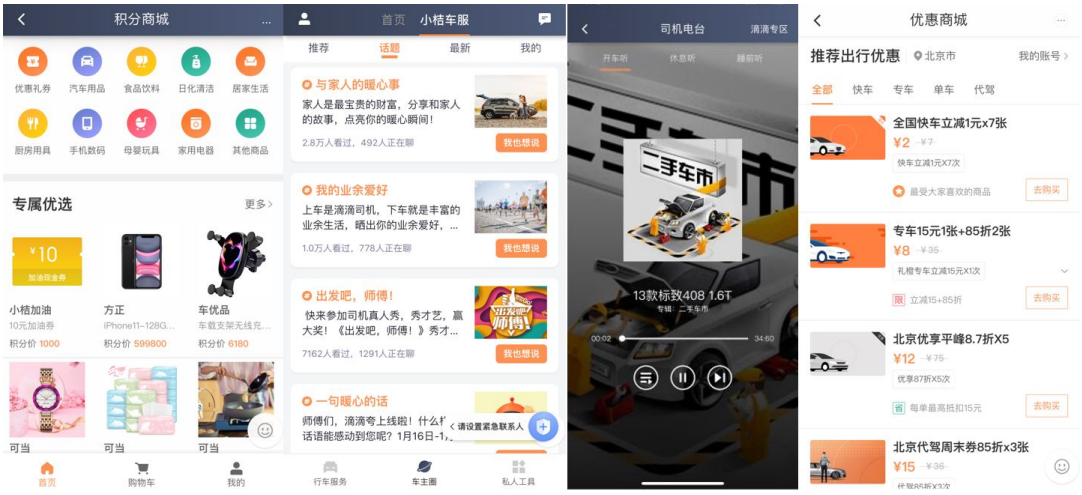

Figure 3: Different recommendation scenarios (product, content, music and coupon) of DiDiChuxing

Table 1: Offline performance of compared methods

Algorithm	DiDi-Product	DiDi-Coupon	DiDi-Content	DiDi-Music	MovieLens
	AUC	AUC	AUC	AUC	AUC
DeepWalk	0.6418	0.6534	0.6205	0.5547	0.7266
Metapath2vec++	0.6385	0.6513	0.6128	0.5661	0.7778
BiNE	0.6523	0.6325	0.6793	0.5729	0.8416
GraphSage(2)	0.6616	0.6358	0.6989	0.6058	0.8636
GraphSage(4)	0.6596	0.6064	0.6958	0.6004	0.8654
Gemini-Collaboration(1)	0.7255	0.6468	0.7003	0.6083	0.8512
Gemini-Collaboration	0.7333	0.6774	0.7069	0.6240	0.8735

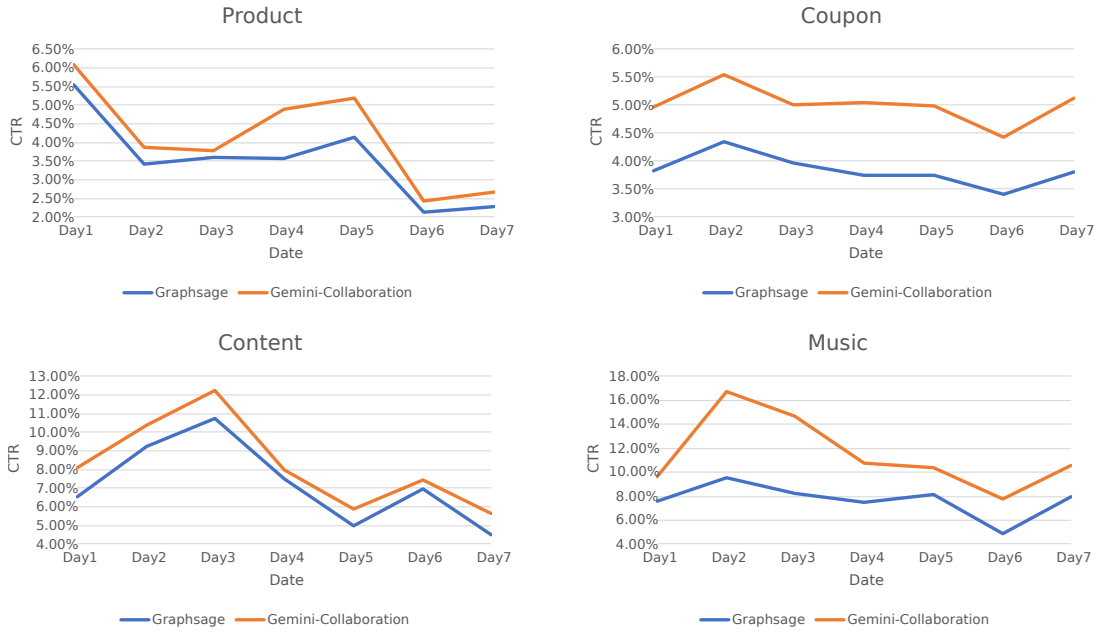

Figure 4: Online performance of compared methods

Table 2: Performance of different designs on DiDi-Product dataset

Proposal with different designs	AUC
Proposal_1 (evaluate Network Transformation): we implement Gemini-Collaboration without using edge information.	0.6783
Proposal_2 (evaluate Edge Embedding): we implement Gemini-Collaboration by using sum pooling instead of TF-IDF pooling and using mean function instead of attention based function to remove the affect of of **TF-IDF pooling** and **Attention based Aggregating**.	0.7145
Proposal_3 (evaluate TF-IDF pooling): we add TF-IDF pooling to Proposal_2.	0.7226
Proposal_4 (evaluate Attention based Aggregating): we add attention based function to Proposal_2.	0.7272

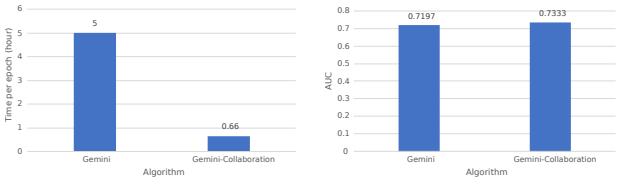

Figure 5: Gemini VS Gemini-Collaboration

[4] 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*. ACM, 7–10.

[5] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 191–198.

[6] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[7] Tao-yang Fu, Wang-Chien Lee, and Zhen Lei. 2017. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1797–1806.

[8] Ming Gao, Leihui Chen, Xiangnan He, and Aoying Zhou. 2018. Bine: Bipartite network embedding. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 715–724.

[9] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. *Advances in Neural Information Processing Systems* 3 (2014), 2672–2680.

[10] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.

[11] 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).

[12] William L. Hamilton, Rex Ying, and Jure Leskovec. [n.d.]. Inductive Representation Learning on Large Graphs. ([n. d.]).

[13] F. Maxwell Harper and Joseph A. Konstan. 2015. *The MovieLens Datasets: History and Context*.

[14] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2333–2338.

[15] Zhipeng Huang and Nikos Mamoulis. 2017. Heterogeneous information network embedding for meta path based proximity. *arXiv preprint arXiv:1701.05291* (2017).

[16] Rana Hussein, Dingqi Yang, and Philippe Cudré-Mauroux. 2018. Are Meta-Paths Necessary? Revisiting Heterogeneous Graph Embeddings. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 437–446.

[17] Thomas N. Kipf and Max Welling. [n.d.]. Semi-Supervised Classification with Graph Convolutional Networks. ([n. d.]).

[18] Ao Li, Zhou Qin, Runshi Liu, Yiqun Yang, and Dong Li. 2019. Spam Review Detection with Graph Convolutional Networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2703–2711.

[19] Yuqi Li, Weizheng Chen, and Hongfei Yan. 2017. Learning graph-based embedding for time-aware product recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 2163–2166.

[20] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *Computer Science* (2013).

[21] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[22] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and S Yu Philip. 2018. Heterogeneous information network embedding for recommendation. *IEEE Transactions on Knowledge and Data Engineering* 31, 2 (2018), 357–370.

[23] Yu Shi, Qi Zhu, Fang Guo, Chao Zhang, and Jiawei Han. 2018. Easing embedding learning by comprehensive transcription of heterogeneous information networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2190–2199.

[24] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1067–1077.

[25] Petar Veli?kovi?, Guillem Cucurull, Arantxa Casanova, Adriana Romero, and Yoshua Bengio. [n.d.]. Graph Attention Networks. ([n. d.]).

[26] Daixin Wang, Cui Peng, and Wenwu Zhu. 2016. Structural Deep Network Embedding. In *the 22nd ACM SIGKDD International Conference*.

[27] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. [n.d.]. GraphGAN: Graph Representation Learning with Generative Adversarial Nets. ([n. d.]).

[28] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. [n.d.]. Billion-scale Commodity Embedding for E-commerce Recommendation in Alibaba. ([n. d.]).

[29] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A neural influence diffusion model for social recommendation. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*. 235–244.

[30] Min Xie, Hongzhi Yin, Hao Wang, Fanjiang Xu, Weitong Chen, and Sen Wang. 2016. Learning graph-based poi embedding for location-based recommendation. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 15–24.

[31] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J. Kim. [n.d.]. Graph Transformer Networks. ([n. d.]).

[32] Daokun Zhang, Yin Jie, Zhu Xingquan, and Zhang Chengqi. [n.d.]. Network Representation Learning: A Survey. *IEEE Transactions on Big Data* ([n. d.]), 1–1.

[33] Ziwei Zhang, Peng Cui, Xiao Wang, Jian Pei, Xuanrong Yao, and Wenwu Zhu. 2018. Arbitrary-order proximity preserved network embedding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2778–2786.

[34] Jun Zhao, Zhou Zhou, Ziyu Guan, Wei Zhao, Wei Ning, Guang Qiu, and Xiaofei He. 2019. IntentGC: a Scalable Graph Convolution Framework Fusing Heterogeneous Information for Recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2347–2357.

[35] Zhenlong Zhu, Ruixuan Li, Minghui Shan, Yuhua Li, Lu Gao, Fei Wang, Jixing Xu, and Xiwu Gu. 2019. TDP: Personalized Taxi Demand Prediction Based on Heterogeneous Graph Embedding. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1177–1180.

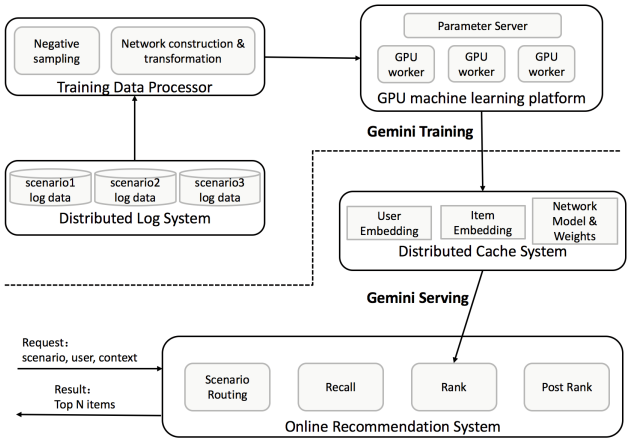

Figure 6: Launched System Architecture

Table 3: Hyper-parameter settings hyper

hyper parameter	setting
learning rate	0.001
optimizer	adam
mini-batch size	512
number of GCN layers	2
number of neighbors sampled in Gemini-U	5,5
number of neighbors sampled in Gemini-I	5,5
number of the slots of Gemini-U	100
number of the slots of Gemini-I	100
dimension of embedding	16

A REPRODUCIBILITY

A.1 Online Launch

Gemini-Collaboration has been launched to production environment of DiDiChuxing. The system architecture is depicted in Figure6. The upper left portion shows the offline training process, and the bottom right portion shows the online serving process. For solving recommendation diversity and cold start issues, we use hot item recall and some common recall algorithms[5, 14] in the recall stage instead of using Gemini-Collaboration. In the rank stage, we use Gemini-Collaboration. Specifically, we use Gemini-Collaboration to predict the interaction probability for each candidate item, and then rank them based on their probabilities.

A.2 Hyper-parameter Settings

The table 3 is the hyper-parameter setting of our off-line experiment, and we use the same hyper-parameter for all datasets to ensure the fairness of the comparison. During the experiment, we explored each of the hyper-parameters, and the values in the table got the best results.