Dynamic Graph Evolution Learning for Recommendation

Haoran Tang
The Hong Kong Polytechnic University
Hong Kong, China
haoran.tang@connect.polyu.hk

Shiqing Wu*
University of Technology Sydney
Sydney, Australia
shiqing.wu@uts.edu.au

Guandong Xu†
University of Technology Sydney
Sydney, Australia
guandong.xu@uts.edu.au

Qing Li†
The Hong Kong Polytechnic University
Hong Kong, China
qing-prof.li@polyu.edu.hk

ABSTRACT

Graph neural network (GNN) based algorithms have achieved superior performance in recommendation tasks due to their advanced capability of exploiting high-order connectivity between users and items. However, most existing GNN-based recommendation models ignore the dynamic evolution of nodes, where users will continuously interact with items over time, resulting in rapid changes in the environment (e.g., neighbor and structure). Moreover, the heuristic normalization of embeddings in dynamic recommendation is de-coupled with the model learning process, making the whole system suboptimal. In this paper, we propose a novel framework for generating satisfying recommendations in dynamic environments, called **D**ynamic **G**raph **E**volution **L**earning (DGEL). First, we design three efficient real-time update learning methods for nodes from the perspectives of inherent interaction potential, time-decay neighbor augmentation, and symbiotic local structure learning. Second, we construct the re-scaling enhancement networks for dynamic embeddings to adaptively and automatically bridge the normalization process with model learning. Third, we leverage the interaction matching task and the future prediction task together for joint training to further improve performance. Extensive experiments on three real-world datasets demonstrate the effectiveness and improvements of our proposed DGEL. The code is available at https://github.com/henrictang/DGEL.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Dynamic Recommendation, Node Evolution, Representation Learning, Graph Neural Network

*equal contribution with the first author
†the corresponding author

SIGIR '23, July 23–27, 2023, Taipei, Taiwan

ACM ISBN 978-1-4503-9408-6/23/07…$15.00
https://doi.org/10.1145/3539618.3591674

ACM Reference Format:
Haoran Tang, Shiqing Wu, Guandong Xu, and Qing Li. 2023. Dynamic Graph Evolution Learning for Recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23), July 23–27, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3539618.3591674

1 INTRODUCTION

Personalized recommendation systems that help users discover items they are likely to consume alleviate the information overload in many applications, such as e-commerce (e.g., Amazon and Taobao) and social network platforms (e.g., Facebook and Instagram) [2, 4, 23, 42]. In recent years, graph neural networks (GNN), which model user-item interactions as a graph, have achieved phenomenal success in recommendation tasks [41]. The domination of GNN in various fields profits from its advanced ability to preserve graph topological structure information [21, 38] and node intrinsic features aggregated from neighbors and itself [26, 29]. A significant superiority of such GNN-based recommendation models is that they exploit the possibility of linking users and items in the interaction graph and the high-order connectivity therein by explicitly propagating the node embeddings over the graph [6, 16]. Hence, from LightGCN [9] to HCCF [36], we can observe that the improvements for recommendation lie at the heart of the neighboring aggregation function for deep representation learning, which determines the performance of the recommendation services.

However, most existing GNN-based recommendation models are consistent with traditional one-round recommendation patterns [5, 6, 13, 36], which generate a fixed recommendation list for the target user during the prediction stage and lack the real-time update strategy when the user preferences change [14, 33]. In other words, it significantly ignores the dynamic evolution of nodes in the interaction graph over time. For example, in real-world recommendation scenarios, users tend to interact with items continuously at distinct times, resulting in rapid evolution and changes in the environment, such as neighbors and structure [34]. The discrepancy between multi-round and one-round recommendations is illustrated in Figure 1. Although some models have considered temporal context to correct representation from the graph [15, 24, 27, 35], the ultimate return lists are still designed under traditional one-time recommendation strategies. In addition, the user preferences are dynamic and diverse over time [7, 11], making it necessary to update and modify the real-time recommendation list to satisfy the user requirements by capturing the current user dynamic preferences. Another tough

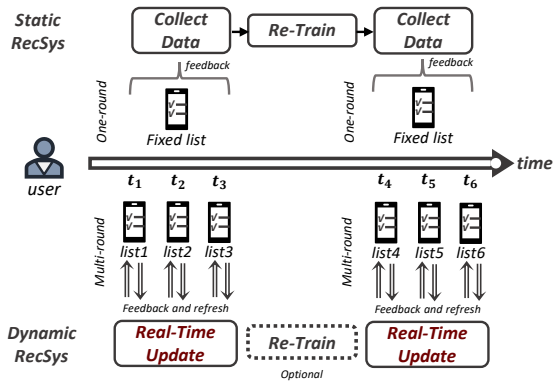

Figure 1: Dynamic multi-round recommendation vs. traditional one-round recommendation. For dynamic recommendation, the core is to precisely capture user and item dynamics and update the recommendation list in real time.

challenge is that the heuristic and intuitive normalization methods in GNN (e.g., L2 normalization) decoupled from model learning can lead to the inflexible limitation to the value range of each node embedding and cause undesirable effects, such as scale distortion issues between different embeddings [25]. Namely, the performance of the whole system is likely to become sub-optimal when applying such static pre-defined normalization. Unfortunately, despite the urgent desire for adaptive and automatic normalization design in the dynamic recommendation, it is frustrating that such an issue does not receive enough attention.

To tackle the abovementioned challenges, we propose a novel framework for the multi-round dynamic recommendation, named **D**ynamic **G**raph **E**volution **L**earning (DGEL). Specifically, to learn user and item dynamics effectively and efficiently, we design three learning methods from different perspectives to accurately represent dynamic sub-embeddings in DGEL, i.e., inherent interaction potential (IIP), time-decay neighbor augmentation (TNA), and symbiotic local structure learning (SLS). Furthermore, user-item interactions follow a sequential order and bring temporal evolution [31], forcing our DGEL to update and modify recommendations at each distinct time when a new interaction is formed. Meanwhile, the heuristic and intuitive normalization methods, which are independent of dynamic update and model learning, impose limitations on these dynamic embeddings, restricting the performance of the whole dynamic recommendation system. Considering we design individual dynamic embeddings from three perspectives, it is essential to bridge normalization with the learning process of our framework to adaptively and automatically modify these dynamic embeddings. Hence, we construct the re-scaling enhancement networks (RENet) to make embeddings comparable and to fit differences between inputs. Moreover, inspired by the idea of multi-tasks to compensate for the absence of rich information in the sparse graph [32], we leverage joint training for the interaction matching task and the future prediction task to ensure that the rapid evolution of nodes is consistent with the temporal-linear graph pattern. We summarize the contributions of this work as follows:

- To investigate user and item dynamics precisely, we carefully devise inherent interaction potential, time-decay neighbor augmentation, and symbiotic local learning to update node dynamic embeddings with rich graph information comprehensively.
- We propose the re-scaling enhancement networks to bridge the normalization with the dynamic embedding learning process adaptively, which can keep the consistency and comparability of dynamic embeddings and avoid the model becoming suboptimal.
- To further improve the performance and ensure the temporal-linear evolution of nodes, we leverage two tasks, the interaction matching task and the future prediction task together for joint training.
- We utilize real-world datasets to evaluate the effectiveness and superiority of our proposed model. Furthermore, we conduct ablation studies and variant research to confirm the importance of each proposed component in our model.

2 RELATED WORK

In the new era of dynamic recommendation, re-training the models based on whole data becomes time-consuming with the rapidly increasing volume of user-item interactions over time. To tackle this issue, many studies have been devoted to novel model updating methods to minimize computational costs and maintain prediction accuracy in the future. For example, SPMF [28] utilizes a probabilistic matrix factorization model under the guide of flexible extendability of the streaming setting to extend itself to the dynamic environment. DeepCoevolve [3] trains two mutually-recursive networks to generate embedding trajectories while maintaining the same embedding of a user between two consecutive interactions. Jodie [12] further uses a novel project function to predict the future trajectory of users, achieving a milestone for dynamic embedding learning. IncCTR [30] jointly extracts knowledge from the historical data and new incoming data to achieve incremental recommendations. SML [40] employs a neural transfer learning component to transform the old model to a new one and then optimizes the performance evaluated in the next time period to adjust the transfer learning. However, updating the model may produce the same expense as the re-training procedure. Therefore, inspired by graph signal processing, FIRE [33] adopts a fast incremental non-parametric recommendation method that does not suffer from the time-consuming back-propagation as in previous learning-based methods.

Meanwhile, inspired by the success of a large body of graph representation learning methods [8–10, 38], many studies consider utilizing graph snapshots to capture graph evolution over time. For example, TGN [22] is a framework consisting of a memory module and an embedding module, which can learn continuous-time dynamic graphs to capture the rapid changes of node features over time. Instead of focusing on updating dynamic embeddings, EvolveGCN [19] provides a novel paradigm to evolve the parameters of GCN. DGCF [14] extends Jodie [12] to the graph through customized updating mechanisms. In order to overcome the catastrophic forgetting issue by previous models, IGCN [34] presents a novel incremental temporal convolutional network that fuses information from the last period and the current period. Moreover,

TGSRec [7] applies a temporal kernel to map continuous timestamps on edges to vectors and then infers temporal embeddings of nodes. TREND [31] points out that the event and node dynamics investigate the individual and collective interaction characteristics, driving more precise modeling of the link formation process.

3 PRELIMINARIES

In this section, we first briefly introduce the graph neural network for recommendation and then define the dynamic recommendation with the evolving graph over time.

Recap Graph-Based Paradigm for Recommendation. Suppose the recommendation scenario involves m users and n items with the user set $\mathcal{U} = \{u_1, u_2, ..., u_m\}$ and item set $\mathcal{V} = \{v_1, v_2, ..., v_n\}$. Edges in the user-item interaction graph $\mathcal{G}$ are built if user u_i has interacted with item v_j (e.g., click action in e-commerce). To begin with, we initialize users and items via a d-dimensional latent space to encode their representation under interaction patterns. More precisely, $\mathbf{h}_i \in \mathbb{R}^d$ and $\mathbf{h}_j \in \mathbb{R}^d$ are generated for user u_i and item v_j, respectively. In addition, all the users and items compose the embedding matrices $\mathbf{H}^{\mathcal{U}} \in \mathbb{R}^{m \times d}$ and $\mathbf{H}^{\mathcal{V}} \in \mathbb{R}^{n \times d}$. Upon the constructed interaction graph, the core idea of the graph-based paradigm lies in the information aggregation functio (e.g., average aggregation) [37]. Inspired by LightGCN [9], an efficient aggregation for both users and items could be indicated as follows:

$$\mathbf{h}_i^{(k+1)} = \sum_{j \in N_i} \frac{1}{\sqrt{|N_i|}\sqrt{|N_j|}} \mathbf{h}_i^{(k)}, \mathbf{h}_j^{(k+1)} = \sum_{i \in N_j} \frac{1}{\sqrt{|N_i|}\sqrt{|N_j|}} \mathbf{h}_j^{(k)}, \tag{1}$$

where $\mathbf{h}_i^{(k+1)}$ and $\mathbf{h}_j^{(k+1)}$ denote the refined embedding of user u_i and item v_j after k convolutional layers, respectively . N_i denotes the set of items that are interacted with user u_i while N_j denotes the set of users that interact with item v_j. With the support of such aggregations, capturing local graph structure information to learn better node representation for making link prediction becomes a reality.

Evolving Graph for Dynamic Recommendation. In real-world scenarios, users tend to interact with items continuously, leading to the rapid evolution of the graph. Now we define evolving graph for dynamic recommendation according to work [12, 14, 34], that is $\mathcal{G}^t$, the snapshot at time t. It is formed by $\mathcal{S}$ which represents an ordered sequence of temporal user-item interactions in the graph. The r-th interaction happened between user u_i and item v_j is denoted as $s_r^{i,j} = (u_i, v_j, t, f)$, where t and f represent the timestamp and feature vector of the interaction, respectively (note that not all the interactions have features). Hence, as interactions are observed in temporal order, the graph will evolve over time. As a consequence, the recommendation system needs to modify its recommendation results. That is also consistent with real-world applications where users will not only receive a fixed recommendation list.

The core component of dynamic recommendation based on the interaction graph is how to update dynamic embedding efficiently since the graph evolves all the time. Here, we devise two embeddings for each node in the graph, i.e., static and dynamic embeddings, to encode both the long-term stationary properties of the entities and their dynamic properties. Static embeddings (e.g., one-hot vectors or side-information vector) of user u_i and item v_j, denoted as $\bar{\mathbf{h}}_i \in \mathbb{R}^{\bar{d}}$ and $\bar{\mathbf{h}}_j \in \mathbb{R}^{\bar{d}}$, will keep unchanged. While dynamic embeddings, denoted as $\tilde{\mathbf{h}}_i \in \mathbb{R}^{\tilde{d}}$ and $\tilde{\mathbf{h}}_j \in \mathbb{R}^{\tilde{d}}$, evolve over time to model their time-varying representations. Therefore, our goal is to update node dynamic embeddings accurately after observing current interaction at time t and generate a modified recommendation list for future $t+1$ based on that:

$$\tilde{\mathbf{h}}_i^t, \tilde{\mathbf{h}}_j^t = F_\Theta\left(\tilde{\mathbf{h}}_i^{t-1}, \tilde{\mathbf{h}}_j^{t-1}, f\right), \tag{2}$$

$$Rec_i^{t+1} = \{v_j \mid v_j \in \mathcal{V}, [\tilde{\mathbf{h}}_j^t \parallel \bar{\mathbf{h}}_j] \simeq [\tilde{\mathbf{h}}_i^{t+1} \parallel \bar{\mathbf{h}}_i]\}_k, \tag{3}$$

where F_Θ is our framework with parameter set Θ that investigates node dynamics from various generative perspectives and f is the feature of current interaction. Rec_i^{t+1} is the new recommendation list for the user u_i at future time $t+1$ when he/she uses our recommendation service again and k is the length of list. $[x \parallel y]$ represents the concatenation of vector x and y. Moreover, $\simeq$ indicates that we aim to discover the target items nearest to user embeddings at time $t+1$ (see section 4.5).

4 METHODOLOGY

In this section, we present the overall architecture of DGEL in Figure 2. First, we devise three dynamic learning methods, i.e., inherent interaction potential, time-decay neighbor augmentation, and symbiotic local structure learning, to comprehensively capture node dynamics over time. Second, we construct the re-scaling enhancement network to bridge the normalization with the dynamic embedding learning process to seek consistency and comparability of dynamic embeddings and avoid scale distortion issues. Third, we leverage the future prediction task and interaction matching task together for joint training to improve the overall performance of the proposed DGEL framework. Technical details are discussed in the following sub-sections.

4.1 Dynamic Representation Learning

4.1.1 ***Inherent Interaction Potential***. Inspired by [7, 12], we assume that the current interaction between user u_i and item v_j at time t plays a key role in reflecting the user and item current states, leading to more meaningful dynamic embeddings. Also, the interaction itself indicates the explicit attraction for user u_i and item v_j to each other, revealing the inherent potential of linking user u_i with item v_j under dynamic circumstances in the graph. We call this Inherent Interaction Potential (IIP).

Since our scenario belongs to interaction-based multi-round dynamic recommendation, the first and foremost objective is to investigate IIP for user u_i and item v_j with the following form:

$$\mathbf{h}_i^t(IIP) = \sigma(\mathbf{W}_{11}^u \tilde{\mathbf{h}}_i^{t-1} + \mathbf{W}_{12}^u \tilde{\mathbf{h}}_j^{t-1} + \mathbf{W}_{13}^u f + \mathbf{W}_{14}^u \Delta t_i), \tag{4}$$

$$\mathbf{h}_j^t(IIP) = \sigma(\mathbf{W}_{11}^v \tilde{\mathbf{h}}_j^{t-1} + \mathbf{W}_{12}^v \tilde{\mathbf{h}}_i^{t-1} + \mathbf{W}_{13}^v f + \mathbf{W}_{14}^v \Delta t_j), \tag{5}$$

where $\mathbf{h}_i^t(IIP), \mathbf{h}_j^t(IIP) \in \mathbb{R}^{\tilde{d}}$ represent the updated embeddings for user u_i and item v_j at time t from the perspective of inherent interaction potential. $\tilde{\mathbf{h}}_i^{t-1}$ and $\tilde{\mathbf{h}}_j^{t-1}$ are the dynamic embeddings at previous time. In addition to nodes, we also consider the current interaction feature f. Δt_i denotes the time interval between the current interaction and the previous interaction of user

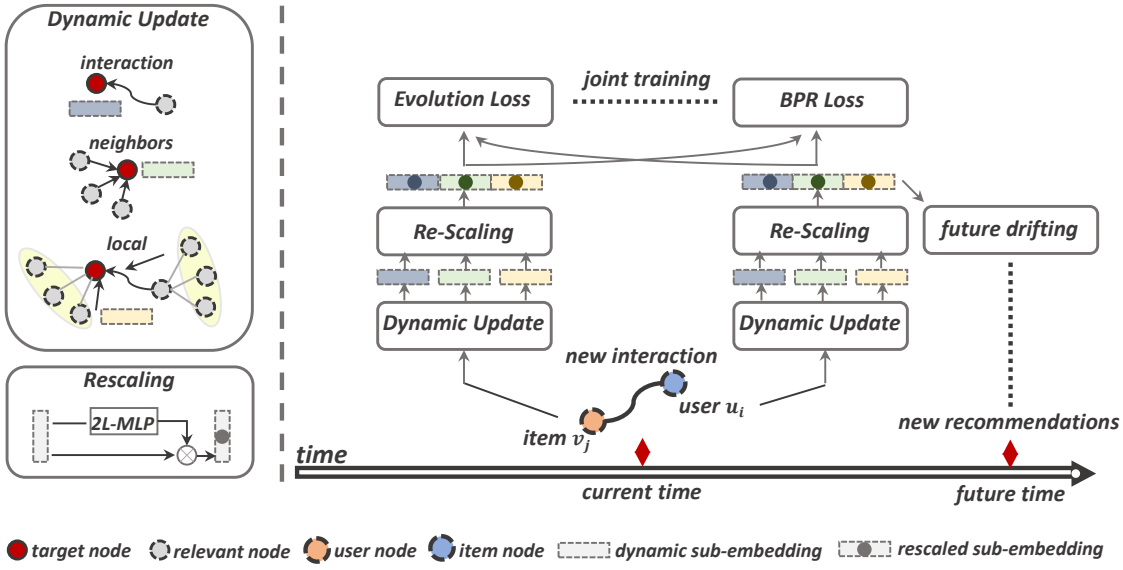

Figure 2: The illustration of DGEL's overall architecture. We capture user and item dynamics using three learning methods from the perspectives of interaction, neighbors, and local structure. Then we construct the re-scaling enhancement network to bridge the normalization of dynamic embeddings with model learning. Last, we leverage two tasks, the interaction matching task and the future prediction task together for joint training.

u_i (with any item) while Δt_j is the time interval between current interaction and previous interaction of item v_j (with any user). $\mathbf{W}^u_{11-13} \in \mathbb{R}^{\tilde{d}\times\tilde{d}}, \mathbf{W}^u_{14} \in \mathbb{R}^{\tilde{d}}$, and $\mathbf{W}^v_{11-13} \in \mathbb{R}^{\tilde{d}\times\tilde{d}}, \mathbf{W}^v_{14} \in \mathbb{R}^{\tilde{d}}$ are learnable weight matrices for users and items, respectively. σ is the activation function throughout this paper (e.g., Tanh and LeakyReLU). The reason that we devise two channels $\mathbf{W}^u_{11-14}$ and $\mathbf{W}^v_{11-14}$ is the explicit updating trend from the current interaction exhibits different dynamic levels for users and items. In general, user dynamics are more sensitive to interaction than item dynamics.

4.1.2 ***Time-Decay Neighbor Augmentation.*** Having investigated user-item interaction behaviors, we seek to explore the influence information from historical records of nodes (both users and items) that reveals the long-term dynamic attributes. In section 3.1, we recap the graph convolutional paradigm by neighbors. However, adopting static attributes (e.g., degrees of nodes) makes such methods (e.g., LightGCN [9]) impotent to freeze the snapshots at distinct times.

Moreover, simple mean or graph attention [14] in dynamic neighbor aggregation will lose the temporal information severely. Hence, following the work in [31, 34], to construct efficient neighbor aggregation that maintains the influence derived by time interval, we propose the single-convolution time-decay neighbor augmentation (TNA) aggregation as formulated below:

$$\mathbf{h}^t_i(TNA) = \sigma(\mathbf{W}^u_{21}\tilde{\mathbf{h}}^{t-1}_i + \sum_{j\in N_i} \mathbf{W}^u_{22}\kappa^u\tilde{\mathbf{h}}^{t-1}_j), \tag{6}$$

$$\mathbf{h}^t_j(TNA) = \sigma(\mathbf{W}^v_{21}\tilde{\mathbf{h}}^{t-1}_j + \sum_{i\in N_j} \mathbf{W}^v_{22}\kappa^v\tilde{\mathbf{h}}^{t-1}_i), \tag{7}$$

where $\mathbf{h}^t_i(TNA), \mathbf{h}^t_j(TNA) \in \mathbb{R}^{\tilde{d}}$ represent the updated embeddings for user u_i and item v_j at time t from the view of time-decay neighbor augmentation. N_i denotes the set of items that are interacted with user u_i, and N_j denotes the set of users that interact with item v_j. $\mathbf{W}^u_{21,22} \in \mathbb{R}^{\tilde{d}\times\tilde{d}}$ and $\mathbf{W}^v_{21,22} \in \mathbb{R}^{\tilde{d}\times\tilde{d}}$ are learnable weight matrices for users and items, respectively. Our function κ outputs the time-weight of each neighbor that is based on Softmax: $\kappa^u(t_i - t_j) = exp(\tau_{ij})/\sum_{j^-\in N(i)} exp(\tau_{ij^-}), \tau_{ij} = -(t_i - t_j)/max(t_i - t_{j^-|j^-\in N(i)})$. Instead of the complex time-aware attention score that requires additional training overhead, we adopt the negative time-interval normalization for Softmax, which is fast to compute without other hyperparameters. Note that, the neighbor sets for both user u_i and item v_j are only the snapshots at current time t and will change over time as new interactions between users and items will form. Consequently, our time-decay neighbor augmentation keeps consistent with real-world scenarios.

4.1.3 ***Symbiotic Local Structure Learning.*** Although the aforementioned inherent interaction potential and time-decay neighbor augmentation endow our DGEL with the capability of exploiting the node evolution from interactions and relevant neighbors, they are still decoupled from each other and work in their own way. Hence, we further supercharge our DGEL with Symbiotic Local Structure Learning (SLS), which absorbs the impact of interaction and the excitements from the neighbors. The excitements measure the tendency of current interaction happening under the environment (or context) composed of neighbors. In our assumption, forming a new interaction between user u_i and item v_j depends on their inherent features and mutual neighbors simultaneously. It is reasonable to

encode the interaction and neighbors into a local structure. Towards this end, we apply pooling operation [38] to realize our SLS for conserving computational cost. Formally, the SLS for user u_i and item v_j is derived as follows:

$$\mathbf{h}_i^t(SLS) = \sigma(\mathbf{W}_{31}^u(\tilde{\mathbf{h}}_i^{t-1} - \tilde{\mathbf{h}}_j^{t-1})^p + \mathbf{W}_{32}^u \text{avg}(\{\tilde{\mathbf{h}}_z^{t-1}, \forall z \in N_i\}) + \mathbf{W}_{33}^u \text{avg}(\{\tilde{\mathbf{h}}_x^{t-1}, \forall x \in N_j\})), \tag{8}$$

$$\mathbf{h}_j^t(SLS) = \sigma(\mathbf{W}_{31}^v(\tilde{\mathbf{h}}_j^{t-1} - \tilde{\mathbf{h}}_i^{t-1})^p + \mathbf{W}_{32}^v \text{avg}(\{\tilde{\mathbf{h}}_x^{t-1}, \forall x \in N_j\}) + \mathbf{W}_{33}^v \text{avg}(\{\tilde{\mathbf{h}}_z^{t-1}, \forall z \in N_i\})), \tag{9}$$

where $\mathbf{h}_i^t(SLS), \mathbf{h}_j^t(SLS) \in \mathbb{R}^{\tilde{d}}$ denote the updated embeddings for user u_i and item v_j at time t from the perspective of symbiotic local structure learning. $(x-y)^p$ represents the element-wise power function between x and y where $(x-y)^p = (y-x)^p$ when $p = 2$. Otherwise, the outputs are opposite for the user and item to focus on their own learning injection when $p = 1$. $\mathbf{W}_{31-33}^u \in \mathbb{R}^{\tilde{d}\times\tilde{d}}$ and $\mathbf{W}_{31-33}^v \in \mathbb{R}^{\tilde{d}\times\tilde{d}}$ are learnable weight matrices for users and items, respectively. To model the discrepancy between users and items on dynamic sensitivity, similar to Section 4.1.1, we construct two channels to process the same pooling operation on user neighbors and item neighbors. The SLS helps us to understand the implicit correlation between two local groups under the light paradigm of structure learning. So far, we have obtained three categories of dynamic embeddings. Next, we will introduce how we re-scale these embeddings to form the final dynamic embedding.

4.2 Re-scaling Enhancement Network

4.2.1 ***Bridge the Normalization with Model Learning.*** Even though the superiority of GNNs over shallow graph embeddings comes from various aggregations and encodings, there is still one common weakness in such aggregation and encoding mechanism of many dynamic recommendation models [7, 12, 14, 31, 34]. Heuristic and intuitive normalization methods (e.g., L2-Norm) on representations of nodes are independent of the model learning, causing inflexible limitations to the value range of node embedding and undesirable effects (e.g., the scale distortion issue between different embeddings). Since we derive dynamic sub-embeddings from three novel generative perspectives (see Section 4.1), it is essential to link the normalization coefficients to the learning process of our model to endow it with an automatic and adaptive ability to determine the appropriate scaling degree.

Motivated by [25], we propose the Re-scaling Enhancement Network (RENet) to further scale the dynamic embeddings. To the best of our knowledge, it is the first time applying the neural re-scaling strategy on dynamic recommendations. Without losing generality, here we use $\tilde{h}$ to represent dynamic embeddings derived from any perspectives (in Section 4.1) and define our RENet with a 2-layer MLP as follows:

$$G(\tilde{h}) = \sigma_2(\mathbf{w}_2\sigma_1(\mathbf{W}_1\tilde{h} + \mathbf{b}_1) + b_2), \tag{10}$$

$$\tilde{h}^\circ = G(\tilde{h}) \cdot \tilde{h}, \tag{11}$$

where $\mathbf{W}_1 \in \mathbb{R}^{\tilde{d}/2\times\tilde{d}}$, $\mathbf{b}_1 \in \mathbb{R}^{\tilde{d}/2}$, $\mathbf{w}_2 \in \mathbb{R}^{\tilde{d}/2}$, and $b_2 \in \mathbb{R}$ are the learnable parameters of $G(\cdot)$ that outputs a re-scaling factor. σ_1 is the activation of the first layer, which could be any option (e.g., Tanh and LeakyReLU) while σ_2 is the activation of the second layer and it determines the value of re-scaling factor. In [25], σ_2 is the *Sigmoid* function to force the output value in $(0, 1)$. However, we observe that the value range of each dimension in node embeddings is too small to bring obvious numerical variation in the second layer before *Sigmoid*, which means the output of *Sigmoid* always approaches $Sigmoid(0) = 0.5$ (e.g., 0.48 and 0.52). Thus, we select LeakyReLU as the activation of the second layer to allow more valuable space for the factor. $G(\cdot)$ could adaptively control the scaling intensity according to the dynamic embedding. Meanwhile, we separately assign independent RENets for the distinct-perspective dynamic embeddings, such that our model can distinguish the impact of them. There are some other re-scaling variants (e.g., Feed-Forward Network), and we will compare them with our RENet.

4.2.2 ***Final Dynamic Representation.*** To fulfill the final dynamic embedding of user u_i and achieve evolution learning for user, we adopt concatenation of the re-scaled $\mathbf{h}_i^t(IIP)$, $\mathbf{h}_i^t(TNA)$ and $\mathbf{h}_i^t(SLS)$ (same operation on item) for two reasons: 1) maintain their own dynamic information from the various perspectives of interaction, neighbors, and local structure; 2) keep the consistency with RENet since RENet corrects these sub-embeddings individually and enhances the expression of them. Moreover, we adopt a learnable weight sum of dynamic states from the current time and previous time because remembering the essential information from the previous embedding at time $t-1$ is a common idea in the sequential recommendation. Therefore, the final dynamic embeddings for user u_i and item v_j at time t are constructed as follows:

$$\tilde{\mathbf{h}}_i^t = \sigma(\mathbf{W}_3^u[\mathbf{h}_i^t(IIP)\|\mathbf{h}_i^t(TNA)\|\mathbf{h}_i^t(SLS)] + \mathbf{W}_4^u\tilde{\mathbf{h}}_i^{t-1} + \mathbf{b}_3), \tag{12}$$

$$\tilde{\mathbf{h}}_j^t = \sigma(\mathbf{W}_5^v[\mathbf{h}_j^t(IIP)\|\mathbf{h}_j^t(TNA)\|\mathbf{h}_j^t(SLS)] + \mathbf{W}_6^v\tilde{\mathbf{h}}_j^{t-1} + \mathbf{b}_4), \tag{13}$$

where $\mathbf{W}_3^u \in \mathbb{R}^{\tilde{d}\times 3\tilde{d}}$, $\mathbf{W}_4^u \in \mathbb{R}^{\tilde{d}\times\tilde{d}}$, $\mathbf{b}_3 \in \mathbb{R}^{\tilde{d}}$ and $\mathbf{W}_5^v \in \mathbb{R}^{\tilde{d}\times 3\tilde{d}}$, $\mathbf{W}_6^v \in \mathbb{R}^{\tilde{d}\times\tilde{d}}$, $\mathbf{b}_4 \in \mathbb{R}^{\tilde{d}}$ are learnable parameters for users and items, respectively. Here, we finally obtain the user and item dynamic embeddings after dynamic evolution learning and re-scaling enhancement networks. Note that our dynamic model DGEL only updates the corresponding user u_i and item v_j of the current interaction instead of updating all nodes in the interaction graph. In the following sections, we will present the details about leveraging multi-tasks in the dynamic recommendation for joint training.

4.3 Multi-Task Joint Training

4.3.1 ***Match the Current User and Item.*** Having established the evolution of user u_i and item v_j based on the current interaction at t, we treat them as the positive pair to measure the level of matching the user and item under our DGEL framework. The auxiliary supervision of positive pairs ensures the accuracy of updated dynamic embeddings according to the fact they have linked with each other at the current time. For simplicity, we adopt the BPR loss function to calculate the matching level of all the positive pairs from the ordered interaction sequence:

$$\mathcal{L}_{bpr} = \frac{1}{|\mathcal{S}|}\sum_{t=1}^{|\mathcal{S}|} Log(\sigma(s_t^{i,j} - s_t^{i,j^-})), s_t^{i,j} \in \mathcal{S}, s_t^{i,j^-} \notin \mathcal{S}, \tag{14}$$

where inner-product $s_t^{i,j} = \tilde{\mathbf{h}}_i^t \times \tilde{\mathbf{h}}_j^t$ is positive pair belonging to interaction set $\mathcal{S}$ and s_t^{i,j^-} is sampled negative pair for user u_i. σ is

Sigmoid function. In our assumption, the positive pairs brought by real interactions will amplify the distinction from fake interactions via this BPR loss.

4.3.2 ***Future Drifting and Evolution Loss.*** Dynamic recommendation methods usually generate embeddings only when users take actions on items and do not explicitly model the future trajectory of the user in the embeddings space. Thus, motivated by [12, 14], we introduce a future drifting component to predict the future embeddings of user preference and target item at any future time, which encourages the flexibility and extendibility of our framework:

$$\tilde{\mathbf{h}}_i^{*(t+1)} = (1 + \triangle(t+1)\mathbf{w}_1) \cdot \tilde{\mathbf{h}}_i^{t}, \tag{15}$$

$$\tilde{\mathbf{v}}_j^{*(t+1)} = \mathbf{W}_2\tilde{\mathbf{h}}_i^{*(t+1)} + \mathbf{W}_3\bar{\mathbf{h}}_i + \mathbf{W}_4\tilde{\mathbf{h}}_j^{t} + \mathbf{W}_5\bar{\mathbf{h}}_j + \mathbf{b}, \tag{16}$$

where $\tilde{\mathbf{h}}_i^{*(t+1)}$ is the predicted future dynamic representation of u_i at time $t+1$ and $\triangle$ calculates the interval between time t and time $t+1$. $\bar{\mathbf{h}}_i$ and $\bar{\mathbf{h}}_j$ are static embeddings of user u_i and item v_j. $\mathbf{W}_1 \in \mathbb{R}^{\tilde{d}}, \mathbf{W}_{2,4} \in \mathbb{R}^{(\tilde{d}+\bar{d})\times\tilde{d}}, \mathbf{W}_{3,5} \in \mathbb{R}^{(\tilde{d}+\bar{d})\times\bar{d}}, \mathbf{b} \in \mathbb{R}^{\tilde{d}+\bar{d}}$ are learnable parameters to obtain the predicted item embedding $\tilde{\mathbf{v}}_j^{*(t+1)}$ that user will interact with in future. Note that, in Equation 3 (see Section 3.2), we update recommendation results and generate a new list according to the combination of two parts: dynamic embedding and static embedding. Therefore, $\tilde{\mathbf{v}}_j^{*(t+1)}$ is the $(\tilde{d}+\bar{d})$-dimensional embedding and once user u_i access our system at future time $t+1$, we will return top-k items whose embeddings are nearest to $\tilde{\mathbf{v}}_j^{*(t+1)}$.

In practice, we propose the evolution loss to minimize the distance between the predicted item embeddings and ground truth item embedding at every interaction. We compute the evolution loss below:

$$\mathcal{L}_{evol} = \sum_{i,j,t\in\mathcal{S}} \|\tilde{\mathbf{v}}_j^{*(t+1)} - \mathbf{h}_j^{(t+1)}\|^2 + \|\tilde{\mathbf{h}}_i^{t} - \tilde{\mathbf{h}}_i^{t-1}\|^2 + \|\tilde{\mathbf{h}}_j^{t} - \tilde{\mathbf{h}}_j^{t-1}\|^2, \tag{17}$$

where the first term minimizes the predicted embeddings error and $\mathbf{h}_j^{(t+1)} = [\tilde{\mathbf{h}}_j^{(t+1)}, \bar{\mathbf{h}}_j]$ is the ground truth that user actually interact with at time $t+1$. $\|\cdot\|^2$ is the L2 regularization. The last two terms are added to regularize the evolution loss and protect the consecutive dynamic evolution of users and items in order to avoid varying too much.

4.3.3 ***Joint Training.*** To improve the overall performance of the proposed DGEL framework for generating recommendations in dynamic environments, we leverage a multi-task training strategy to jointly optimize the interaction matching task and dynamic evolution task as follows:

$$\mathcal{L} = \mathcal{L}_{evol} + \alpha\mathcal{L}_{bpr} + \delta\|\Theta\|^2, \tag{18}$$

where Θ is the set of model parameters. α controls the BPR loss value to balance it with evolution loss and δ controls the strength of L_2 regularization. We fully take the benefits of multi-task joint training in our scenario: 1) investigate the cohesion of current interaction by the corresponding updated user node and item node; 2) make accurate predictions for the future that allows user preference to drift without limitation of time.

4.4 Model Complexity

Since our framework only updates the corresponding user and item of current interaction to modify recommendation results, the model complexity totally depends on the number of interactions. The inherent interaction potential takes $O(|\mathcal{S}| \times \tilde{d})$ complexity where $|\mathcal{S}|$ is the number of interactions and $\tilde{d}$ is the dimension of dynamic embeddings. Additionally, time-decay neighbor augmentation and symbiotic local structure learning both take $O(|\mathcal{S}| \times |N| \times \tilde{d})$ where $|N|$ is the number of neighbors. The re-scaling network takes $O(L \times |\mathcal{S}| \times \tilde{d})$ cost where L denotes the number of layers (which is set to 2 in our DGEL). In the future drifting module for prediction, owing to the combination of dynamic and static embeddings, the cost is $O(|\mathcal{S}| \times (\tilde{d}+\bar{d}))$ where $\bar{d}$ is the dimension of static embedding. The computation cost for the multi-task joint loss function is $O(|\mathcal{S}| \times (\tilde{d}+\bar{d}))$. Last, we can complete the top-k list in a small linear time based on the predicted future item embedding by data structure algorithms instead of calculating the scores of all items.

5 EXPERIMENTS

Extensive experiments are conducted to evaluate the performance of our proposed model DGEL by answering the following research questions:

- **RQ1**: How dose our DGEL perform when comparing with the state-of-art dynamic recommendation methods?
- **RQ2**: How do different modules (e.g., time-decay neighbor augmentation, re-scaling enhancement network, etc.) contribute to the overall performance of DGEL?
- **RQ3**: What are the effects of other optional variants (e.g., sum-pooling local learning, feed-forward network re-scaling, etc.) in our DGEL?
- **RQ4**: How do the key hyper-parameters (e.g., length of the historical neighbor set, dimension of dynamic embedding) influence the performance of our DGEL?

5.1 Experimental Settings

5.1.1 ***Datasets.*** We conduct experiments on three public datasets collected from different real-world platforms. **Reddit**[1]: this dataset consists of one month of user posts on subreddits. We select the 1,000 most active subreddits as items and the 10,000 most active users, resulting in 672,447 interactions. Besides, the text of each post is converted into a feature vector representing their LIWC categories [20]. **Wikipedia**[2]: this dataset contains one month of edits on Wikipedia. 157,474 interactions are filtered out for recommendation by the editors who made at least 5 edits and the 1,000 most edited pages. Also, the edited texts are converted into LIWC vectors as features. **Foursquare**[3]: this dataset contains check-ins in Tokyo collected for about 10 months [39]. We choose the 1,000 most active users and the 1,000 most visited locations for the recommendation task, totaling 209,730 check-ins. In this dataset, interactions do not have features. Table 1 presents the statistical information of our experimented datasets.

[1] http://files.pushshift.io/reddit/
[2] https://meta.wikimedia.org/wiki/Data_dumps
[3] https://sites.google.com/site/yangdingqi/home/foursquare-dataset

Table 1: Statistics of the experimental datasets.

Dataset	User	Item	Interaction	Average	Feature
Reddit	10,000	984	672,447	67	✓
Wikipedia	8,227	1,000	157,474	19	✓
Foursquare	1,000	950	209,730	209	✗

5.1.2 ***Baselines for Comparison.*** We compare our DGEL with various outstanding dynamic recommendation methods for performance evaluation.

- **Time-LSTM** [43]: A LSTM variant to model user sequential actions, equipping LSTM with time gates to model time intervals.
- **DeepCoevolve** [3]: A deep co-evolutionary network to define and capture complex mutual influence between users and items.
- **LatentCross** [1]: An easy-to-use technique to incorporate contextual data by embedding the context feature and performing an element-wise product of them.
- **CTDNE** [18]: A general framework that gives rise to methods for learning time-respecting embeddings from continuous-time dynamic networks.
- **DGCF** [14]: A novel framework leveraging dynamic graphs to capture collaborative and sequential relations of both items and users at the same time.
- **FIRE** [33]: A fast-incremental non-parametric recommendation method from a graph signal processing perspective that does not suffer from the time-consuming back-propagation.
- **TREND** [31]: A novel framework for temporal graph representation learning driven by temporal events and node dynamics and built upon a Hawkes process-based graph neural network.

5.1.3 ***Evaluation Protocols and Metrics.*** For a fair comparison, we employ the interaction-based all-ranking strategy to be consistent with the same setting in [12]. Specifically, we use the first 80% data to train, the subsequent 10% to validate, and the final 10% to test. We evaluate the performance of models in terms of the Mean Reciprocal Rank (MRR) and Recall@10. Since we consider the multi-round dynamic recommendation scenarios, we measure each test interaction in chronological order, which is totally different from static GNN-based recommendation evaluation. For each interaction, the ranking of the ground truth item is computed with respect to all the items in the dataset by their L_2 distance from predicted item embedding (see Equation 16). We train the proposed DGEL by adopting temporal batch algorithm [12, 14], which follows three criteria: 1) it should process the interactions in each batch simultaneously; 2) the batching process should maintain the original sequential ordering; 3) each batch should not have duplicate nodes therein (it will lead to conflict when updating the same node). Note that we do not re-train all the dynamic models on the testing data to avoid touching future information and leading to unfair predictions during the test stage.

5.1.4 ***Hyperparameter Settings.*** We adopt AdamW [17] with the learning rate of $1e-3$ as our optimizer for training. We only use a single propagation layer for both time-decay neighbor augmentation and symbiotic local structure learning. In the re-scaling enhancement network, the dimension of the first layer is the size of input dynamic embedding and the dimension of the second layer is half of the first layer. The dimension of dynamic embedding and the length of the historical neighbor set are selected from $\{16, 32, 64, 128\}$ and $\{50, 100, 150, 200, 250, 300\}$, respectively. The coefficient α that controls the BPR loss is selected from $\{0.001, 0.0005\}$ because evolution loss is much smaller, and the coefficient δ of regularization is set as $1e-2$. Last, we implement all the experiments with Pytorch in an NVIDIA 3090 GPU.

5.2 Performance Comparison (RQ1)

Now we report the performance evaluation of all dynamic methods to demonstrate the superiority of DGEL in Table 2. From the results, we summarize the following observations.

- **Overall Performance**. DGEL consistently outperforms all dynamic baseline methods in all datasets, especially yielding significant improvement on Reddit. This result verifies the accuracy and effectiveness of our proposed DGEL that endows the graph with the capability of evolving from three novel and appropriate perspectives with the re-scaling network. The average of interactions in Foursquare is the largest, and this dataset lacks features of interactions. In this case, it is hard to distinguish valuable interactions for dynamic evolution, and the evaluation metrics across all methods (including DGEL) are inferior to the other two datasets. Except for Reddit, the improvements on Recall are greater than MRR in Wikipedia and Foursquare. It may result from our joint loss function design that focuses on investigating positive user preferences and discovering high-relevant future items. To sum up, DGEL gains performance improvements for multi-round dynamic recommendations.
- **Generality and Flexibility**. The diversity of evaluation datasets varying in different sparsity degrees, availabilities of features, and various recommendation scenarios justifies the generality and flexibility of DGEL. Several factors determine that of DGEL: 1) The time-decay neighbor augmentation and symbiotic local structure learning, which capture the collaborative signals independently, only depends on the temporal interaction sequence. 2) All components in dynamic representation learning are adopted with single-layer convolution to avoid over-fitting and over-smoothing caused by multiple aggregations, which also speeds up the updating process. 3) Our re-scaling enhancement network can adaptively and automatically explore the impact of various dynamic sub-embeddings derived from different datasets.
- **Dynamic Graph Superiority**. In most cases from the comparison result, the dynamic recommendation models that apply dynamic graph networks (e.g., DGCF, TREND, and our proposed DGEL) achieve superiority over non-graph dynamic recommendation. We contribute this to the natural investigation of the graph on the implicit connectivity between users and items, which reveals the possibility of users linking with other items. Although the incremental dynamic recommendation FIRE brings the fast updating strategy on interval-based sessions, it is inadequate in real-time preference update leading to comparable results in terms of Recall ratio only. By contrast, our DGEL performs better as it focuses on fast-updating methods for the current interaction and accurately explores the users' rapid-changing preferences.

Table 2: Performance comparison of all models on Reddit, Wikipedia, and Foursquare in terms of MRR and Recall@10. The bold and underlined numbers are the best and second-best performances. The last row is the improvement compared with the second-best performance.

Models	Reddit		Wikipedia		Foursquare	
	MRR	Recall@10	MRR	Recall@10	MRR	Recall@10
Time-LSTM	0.387	0.573	0.247	0.342	0.135	0.279
DeepCoevolve	0.171	0.275	0.515	0.563	0.217	0.326
LatentCross	0.421	0.588	0.424	0.481	0.193	0.289
CTDNE	0.165	0.257	0.035	0.056	0.018	0.041
DGCF	0.629	0.795	<u>0.646</u>	<u>0.713</u>	0.368	<u>0.672</u>
FIRE	0.332	<u>0.814</u>	0.259	0.648	0.154	0.524
TREND	<u>0.647</u>	0.809	0.588	0.691	<u>0.370</u>	0.663
Our Proposed DGEL	**0.701**	**0.849**	**0.674**	**0.756**	**0.381**	**0.697**
Improvement	**8.3%**	**4.2%**	**4.3%**	**6.0%**	**2.9%**	**3.7%**

Table 3: Ablation study on core modules of DGEL: Inherent Interaction Potential (IIP), Time-decay Neighbor Augmentation (TNA), Symbiotic Local Structure Learning (SLS), Re-scaling Enhancement Network (RENet) and Future Drifting (Drift). 'w/o' means 'without'.

Models	Reddit		Wikipedia		Foursquare	
	MRR	Recall	MRR	Recall	MRR	Recall
DGEL w/o IIP	0.675	0.824	0.637	0.723	0.354	0.683
DGEL w/o TNA	0.682	0.830	0.659	0.736	0.361	0.690
DGEL w/o SLS	0.689	0.827	0.654	0.730	0.346	0.675
DGEL w/o RENet	0.661	0.820	0.648	0.709	0.336	0.671
DGEL w/o Drfit	0.692	0.833	0.652	0.728	0.365	0.684
DGEL	**0.701**	**0.849**	**0.674**	**0.756**	**0.381**	**0.697**

5.3 Ablation Study of DGEL (RQ2)

We examine the effects of the core modules in DGEL from 1) the three dynamic updating methods for node evolution; 2) the re-scaling enhancement network; and 3) the future drifting for generating recommendations. Table 3 lists the results.

- **Effect of Dynamic Updating Methods for Node Evolution**. We investigate the effect derived from our inherent interaction potential module, time-decay neighbor augmentation module, and symbiotic local structure learning module by removing each of them separately. We observe that lacking exploration of the interaction itself (e.g., corresponding user and item, interaction feature, etc.) severely degrades the performance of DGEL on Reddit and Wikipedia datasets, where both have rich information describing the interaction. Compared with the inherent interaction potential module, equipping with time-decay neighbor augmentation and symbiotic local structure learning modules improves the performance by capturing the collaborative signals, indicating the necessity of fast and efficient graph convolution for dynamic node evolution.
- **Effect of Re-scaling Enhancement Network**. As we can observe from the results, the performance of DGEL without the re-scaling strategy is almost the worst in all the ablation experiments, which verifies our assumption: the different scales of various embeddings that are conducted from their own space and learning method will harm the performance of graph-based recommendation, not to mention the scale distortion issues existing in both static and dynamic recommendation. Moreover, in our scenario, the three dynamic updating methods will repeat multi times for target users (or items) since users will interact with items continuously. Hence, DGEL requires the re-scaling network to adjust different dynamic embeddings to ensure the accuracy of node evolution.
- **Effect of Future Drifting for Predicting Recommendations**. We also investigate the effectiveness of applying future drifting in the recommendation system, which aims to drift user preferences to any future time and then discover the potential items that the user will consume. Specifically, we replace the drifted user preference embedding with the updated user embedding to predict future items directly. From the results, it is clear that on the basis of the other complete modules, introducing the future drifting component could bring a rise in the overall performance through its ability to exploit future preference. We can further conclude that additional future drifting will benefit our whole framework in predicting recommendations.

5.4 Research on Feasible Variants (RQ3)

In this section, we conduct experiments to analyze different feasible variants of our proposed DGEL: 1) sum and max pooling strategies for symbiotic local structure learning; 2) Feed-Forward Network, Additive Network, and basic L2 normalization for re-scaling enhancement network. and 3) linear weighted strategy for combining dynamic embeddings. The results are shown in Table 4.

- **Pooling Selection for Local Structure Learning**. For our symbiotic local structure learning, we replace the average pooling strategy with two options, i.e., max pooling and sum pooling, which are widely used to encode graph structure. From the results, we can observe that sum pooling is not competitive with max pooling and our adopted average pooling as the data sparsity is over-low or over-high (e.g., Foursquare and Wikipedia). We contribute this to the continuous multi-time updating characteristic that leads to a fast increase of embedding value in the dynamic recommendation. On the other hand, compared with

Table 4: Research on feasible variants of DGEL: pooling variants for symbiotic local structure learning, normalization variants for re-scaling enhancement network, fusion strategy for combining dynamic embeddings. We also report their corresponding ablation results (if they have).

Variants	Reddit		Wikipedia		Foursquare	
	MRR	Recall	MRR	Recall	MRR	Recall
Max	0.692	0.831	0.667	0.732	0.373	0.695
Sum	0.698	0.843	0.661	0.715	0.372	0.680
w/o SLS	0.689	0.827	0.654	0.730	0.346	0.675
Feed-Forward	0.696	0.840	0.668	0.731	0.351	0.687
Additive	0.682	0.830	0.670	0.738	0.364	0.679
L2	0.667	0.825	0.665	0.729	0.362	0.673
w/o RENet	0.661	0.820	0.648	0.709	0.336	0.671
Linear Weighted	0.688	0.835	0.669	0.734	0.377	0.692
DGEL	**0.701**	**0.849**	**0.674**	**0.756**	**0.381**	**0.697**

the ablation of local structure learning, any pooling selection for that brings positive performance improvement.

- **Various Re-scaling Network Forms**. The proposed re-scaling enhancement network (RENet) is one of the core components in our DGEL because it bridges the normalization with model learning adaptively and automatically. The observations are two-fold. First, the design of the re-scaling factor by a 2-layer MLP executes efficiently and scales the embeddings simply without complex learning paradigms such as Feed-Forward Network and Additive Network, gaining more generality for the whole framework. Second, adopting neural network based re-scaling methods can outperform the decoupled heuristic L2 normalization, not to mention that remove normalization for maintaining original embeddings. This further indicates the importance of bridging the normalization process with model learning.
- **Weighted Fusion of Dynamic Embeddings**. From the results, it is obvious that the linear weighted fusion of dynamic embeddings derived from different perspectives is still competitive across all the datasets, proving the power of the feed-forward layer to extract deep information. However, the concatenation of embeddings speeds up the learning process. Moreover, under the effect of our re-scaling enhancement network, the concatenation of embeddings maintains its own dynamic information from different perspectives and keeps consistency with RENet.

5.5 Hyperparameter Sensitivity Analysis (RQ4)

We conduct sensitivity analysis of two hyper-parameters on the performance of DGEL in Figure 3:

- **Dimension of Dynamic Embedding**. This determines the complexity of all the components and the length of concatenating dynamic embeddings derived from three perspectives. From the results, we have two significant observations. First, compared with the Wikipedia and Foursquare datasets, Reddit consists of a large amount of interactions and requires more dimensions to extract user and item features. Second, across all the datasets, overdimensioning (e.g., 128) severely degrades the performance, and no further improvements are observed due to saturation.

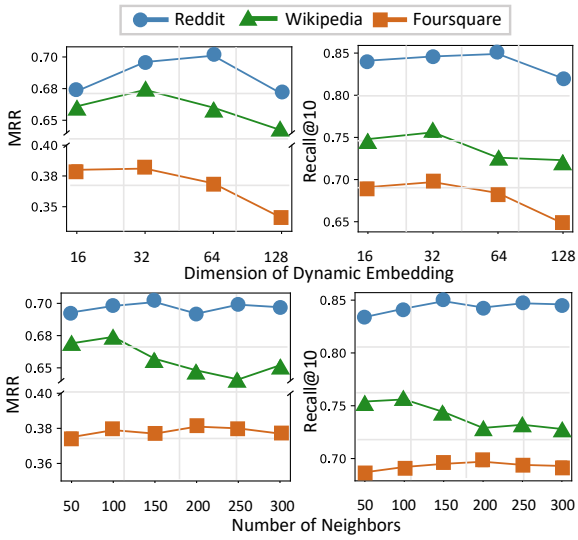

Figure 3: Sensitivity of hyperparameters: The dimension of dynamic embedding and the number of neighbors.

- **Number of Neighbors**. Both time-decay neighbor augmentation and symbiotic local structure learning are influenced by this parameter. Similarly, we still have two observations. On the one hand, the best number of neighbors for all the datasets is strongly related to their sparsity (e.g., 100 produces the best performance for the Wikipedia dataset because it has the least average interactions). On the other hand, the Foursquare dataset seems less sensitive to the increasing number of neighbors. We contribute this phenomenon to the data characteristic.

6 CONCLUSION

In this work, we propose a novel framework, named Dynamic Graph Evolution Learning (DGEL), to realize the satisfying performance in the multi-round dynamic recommendation. The proposed DGEL can comprehensively and efficiently capture user and item dynamics over time from the perspectives of inherent interaction potential, time-decay argumentation, and symbiotic local structure learning. Meanwhile, DGEL performs the initial attempts to adaptively and automatically bridge the normalization of dynamic embeddings with the model learning process for the dynamic recommendation. Moreover, DGEL is designed to update user preferences in real time and adjust the recommendation results to satisfy users' new requirements as users continuously interact with the system. Extensive experiments on three real-world datasets demonstrate the superiority and effectiveness of our DGEL.

ACKNOWLEDGMENTS

The research of this work has been supported by the Hong Kong Research Grants Council under the General Research Fund (project number 15200021), the Australian Research Council (ARC) under Grants number DP22010371 and LE220100078, and the National Natural Science Foundation of China under Grants number 62072257.

REFERENCES

[1] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H. Chi. 2018. Latent Cross: Making Use of Context in Recurrent Recommender Systems. In *WSDM*. 46–54.

[2] Zheng Chai, Zhihong Chen, Chenliang Li, Rong Xiao, Houyi Li, Jiawei Wu, Jingxu Chen, and Haihong Tang. 2022. User-Aware Multi-Interest Learning for Candidate Matching in Recommenders. In *SIGIR*. 1326–1335.

[3] Hanjun Dai, Yichen Wang, Rakshit Trivedi, and Le Song. 2016. Deep coevolutionary network: Embedding user and item features for recommendation. *arXiv preprint arXiv:1609.03675* (2016).

[4] Jing Du, Zesheng Ye, Lina Yao, Bin Guo, and Zhiwen Yu. 2022. Socially-Aware Dual Contrastive Learning for Cold-Start Recommendation. In *SIGIR*. 1927–1932.

[5] Yuntao Du, Xinjun Zhu, Lu Chen, Baihua Zheng, and Yunjun Gao. 2022. HAKG: Hierarchy-Aware Knowledge Gated Network for Recommendation. In *SIGIR*. 1390–1400.

[6] Wenqi Fan, Xiaorui Liu, Wei Jin, Xiangyu Zhao, Jiliang Tang, and Qing Li. 2022. Graph Trend Filtering Networks for Recommendation. In *SIGIR*. 112–121.

[7] Ziwei Fan, Zhiwei Liu, Jiawei Zhang, Yun Xiong, Lei Zheng, and Philip S. Yu. 2021. Continuous-Time Sequential Recommendation with Temporal Graph Collaborative Transformer. In *CIKM*. 433–442.

[8] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[9] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *SIGIR*. 639–648.

[10] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. 2022. GraphMAE: Self-Supervised Masked Graph Autoencoders. In *KDD*. 594–604.

[11] Cheng Hsu and Cheng-Te Li. 2021. RetaGNN: Relational Temporal Attentive Graph Neural Networks for Holistic Sequential Recommendation. In *WWW*. 2968–2979.

[12] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. In *KDD*. 1269–1278.

[13] Siqi Lai, Erli Meng, Fan Zhang, Chenliang Li, Bin Wang, and Aixin Sun. 2022. An Attribute-Driven Mirror Graph Network for Session-based Recommendation. In *SIGIR*. 1674–1683.

[14] Xiaohan Li, Mengqi Zhang, Shu Wu, Zheng Liu, Liang Wang, and Philip S. Yu. 2020. Dynamic Graph Collaborative Filtering. In *ICDM*. 322–331.

[15] Nicholas Lim, Bryan Hooi, See-Kiong Ng, Xueou Wang, Yong Liang Goh, Renrong Weng, and Jagannadan Varadarajan. 2020. STP-UDGAT: Spatial-Temporal-Preference User Dimensional Graph Attention Network for Next POI Recommendation. In *CIKM*. 845–854.

[16] Haoxin Liu. 2022. LightSGCN: Powering Signed Graph Convolution Network for Link Sign Prediction with Simplified Architecture Design. In *SIGIR*. 2680–2685.

[17] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).

[18] Giang Hoang Nguyen, John Boaz Lee, Ryan A. Rossi, Nesreen K. Ahmed, Eunyee Koh, and Sungchul Kim. 2018. Continuous-Time Dynamic Network Embeddings. In *Companion Proceedings of the The Web Conference 2018*. 969–976.

[19] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. 2020. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *AAAI*, Vol. 34. 5363–5370.

[20] James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic inquiry and word count: LIWC 2001. *Mahway: Lawrence Erlbaum Associates* 71 (2001), 2001.

[21] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. In *KDD*. 1150–1160.

[22] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. 2020. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637* (2020).

[23] Javier Sanz-Cruzado and Pablo Castells. 2022. RELISON: A Framework for Link Recommendation in Social Networks. In *SIGIR*. 2992–3002.

[24] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-Based Social Recommendation via Dynamic Graph Attention Networks. In *WSDM*. 555–563.

[25] Xiran Song, Jianxun Lian, Hong Huang, Mingqi Wu, Hai Jin, and Xing Xie. 2022. Friend Recommendations with Self-Rescaling Graph Neural Networks. In *KDD*. 3909–3919.

[26] Yixin Su, Rui Zhang, Sarah M. Erfani, and Junhao Gan. 2021. Neural Graph Matching Based Collaborative Filtering. In *SIGIR*. 849–858.

[27] Yu Tian, Yuhao Yang, Xudong Ren, Pengfei Wang, Fangzhao Wu, Qian Wang, and Chenliang Li. 2021. Joint Knowledge Pruning and Recurrent Graph Convolution for News Recommendation. In *SIGIR*. 51–60.

[28] Weiqing Wang, Hongzhi Yin, Zi Huang, Qinyong Wang, Xingzhong Du, and Quoc Viet Hung Nguyen. 2018. Streaming Ranking Based Recommender Systems. In *SIGIR*. 525–534.

[29] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *SIGIR*. 165–174.

[30] Yichao Wang, Huifeng Guo, Ruiming Tang, Zhirong Liu, and Xiuqiang He. 2020. A practical incremental method to train deep ctr models. *arXiv preprint arXiv:2009.02147* (2020).

[31] Zhihao Wen and Yuan Fang. 2022. TREND: TempoRal Event and Node Dynamics for Graph Representation Learning. In *WWW*. 1159–1169.

[32] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-Supervised Graph Learning for Recommendation. In *SIGIR*. 726–735.

[33] Jiafeng Xia, Dongsheng Li, Hansu Gu, Jiahao Liu, Tun Lu, and Ning Gu. 2022. FIRE: Fast Incremental Recommendation with Graph Signal Processing. In *WWW*. 2360–2369.

[34] Jiafeng Xia, Dongsheng Li, Hansu Gu, Tun Lu, Peng Zhang, and Ning Gu. 2021. Incremental Graph Convolutional Network for Collaborative Filtering. In *CIKM*. 2170–2179.

[35] Lianghao Xia, Chao Huang, Yong Xu, Peng Dai, Xiyue Zhang, Hongsheng Yang, Jian Pei, and Liefeng Bo. 2021. Knowledge-Enhanced Hierarchical Graph Transformer Network for Multi-Behavior Recommendation. In *AAAI*, Vol. 35. 4486–4493.

[36] Lianghao Xia, Chao Huang, Yong Xu, Jiashu Zhao, Dawei Yin, and Jimmy Huang. 2022. Hypergraph Contrastive Collaborative Filtering. In *SIGIR*. 70–79.

[37] Lianghao Xia, Chao Huang, and Chuxu Zhang. 2022. Self-Supervised Hypergraph Transformer for Recommender Systems. In *KDD*. 2100–2109.

[38] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).

[39] Dingqi Yang, Daqing Zhang, Vincent W Zheng, and Zhiyong Yu. 2014. Modeling user activity preference by leveraging user spatial temporal characteristics in LBSNs. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45, 1 (2014), 129–142.

[40] Yang Zhang, Fuli Feng, Chenxu Wang, Xiangnan He, Meng Wang, Yan Li, and Yongdong Zhang. 2020. How to Retrain Recommender System? A Sequential Meta-Learning Method. In *SIGIR*. 1479–1488.

[41] Minghao Zhao, Le Wu, Yile Liang, Lei Chen, Jian Zhang, Qilin Deng, Kai Wang, Xudong Shen, Tangjie Lv, and Runze Wu. 2022. Investigating Accuracy-Novelty Performance for Graph-Based Collaborative Filtering. In *SIGIR*. 50–59.

[42] Qihang Zhao. 2022. RESETBERT4Rec: A Pre-Training Model Integrating Time And User Historical Behavior for Sequential Recommendation. In *SIGIR*. 1812–1816.

[43] Yu Zhu, Hao Li, Yikang Liao, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai. 2017. What to Do Next: Modeling User Behaviors by Time-LSTM.. In *IJCAI*, Vol. 17. 3602–3608.