Efficient Sphere-Effect Based Information Diffusion Prediction on Large-scale Social Networks

Zihan Feng
Tianjin University
Tianjin, China
zihanfeng@tju.edu.cn

Yajun Yang*
Tianjin University
Tianjin, China
yjyang@tju.edu.cn

Xin Huang
Hong Kong Baptist University
Hong Kong, China
xinhuang@comp.hkbu.edu.hk

Hong Gao
Zhejiang Normal University
Jinhua, Zhejiang, China
honggao@zjnu.edu.cn

Liping Jing
Beijing Jiaotong University
Beijing, China
lpjing@bjtu.edu.cn

Qinghua Hu
Tianjin University
Tianjin, China
huqinghua@tju.edu.cn

Abstract

Information diffusion prediction is fundamental for forecasting user participation in information sharing on social networks, such as retweets on Twitter. Existing methods typically extract user relationships from social networks and historical interactions, while further capturing contextual information within the specific diffusion process. However, these methods have several limitations: (1) They often utilize *sequential diffusion process* for prediction and simplify *differentiated influences* among participants; (2) They capture user relationships on the entire graph for all users, in which *most information is not necessary* for a specific diffusion process and is too *inefficient* for real-world large-scale networks. To tackle these limitations, we propose a novel and scalable model SILN, for sphere-based information diffusion prediction on large social networks. Specifically, SILN features three components. First, we integrate two kinds of sphere effects in terms of structural and temporal views, which learn an enhanced cascade representation. Second, SILN designs an efficient learning scheme based on the cascade-specific subgraph, which significantly reduces the entire graph computation to smaller subgraphs. Third, to facilitate subgraph extraction, we develop an optimized graph storage technique to allow constant-time neighbor access and reduce the storage cost by about 30% in practice. Extensive experiments on six real-world datasets validate that SILN consistently outperforms seven state-of-the-art competitors in prediction performance while exhibiting exceptional time and space efficiency on million-node social networks.

CCS Concepts

• **Computing methodologies** → Knowledge representation and reasoning; • **Information systems** → **Social networks**.

Keywords

Social Network Analysis; Social Sphere; Large Graph Learning

ACM Reference Format:
Zihan Feng, Yajun Yang, Xin Huang, Hong Gao, Liping Jing, and Qinghua Hu. 2025. Efficient Sphere-Effect Based Information Diffusion Prediction on Large-scale Social Networks. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3711896.3736925

KDD Availability Link:
The source code of this paper has been made publicly available at https://doi.org/10.5281/zenodo.15546402.

*Corresponding author.

KDD '25, Toronto, ON, Canada

ACM ISBN 979-8-4007-1454-2/2025/08
https://doi.org/10.1145/3711896.3736925

1 Introduction

Information diffusion prediction aims to identify potential users who are likely to participate in information sharing (e.g., retweets on Twitter), which is crucial for understanding topic evolution and has many important applications, such as public opinion analysis [24], fake news control [41], and online marketing [11]. Existing information diffusion prediction methods can be broadly classified into three categories. 1) *Probabilistic methods* use probabilistic distribution models [2, 10, 29, 40] (e.g., SIR models) to calculate diffusion probabilities across all social links and then predict future participants based on initial seed users. However, these methods are often constrained by predefined and rigid parameters. 2) *Cascade-learning methods* [14, 37, 42] consider each diffusion process as a "cascade". As shown at the top of Fig. 1, each cascade is a sequence of tuples (v_x, t_x), where v_x is a participant and t_x is the participation time. These methods exploit deep sequence models like RNNs and Attention mechanisms to extract data-driven diffusion patterns for prediction. 3) *Graph-learning methods* [30, 33, 44] design various GNN-based models (e.g., heterogeneous graphs and hypergraphs) to learn user relationships within global social networks, which are often used to supplement the local cascade learning. Recently, most models [4, 18, 28, 48] adopt a two-stage framework, which integrates both graph learning and cascade learning to collectively capture global user relationships and local diffusion patterns.

Despite the promising improvements, there remain two major limitations: ***(1) Simplistic Utilization of Sequential Cascades.*** Real-world information diffusion processes exhibit complex structural and temporal patterns, in which participants have different

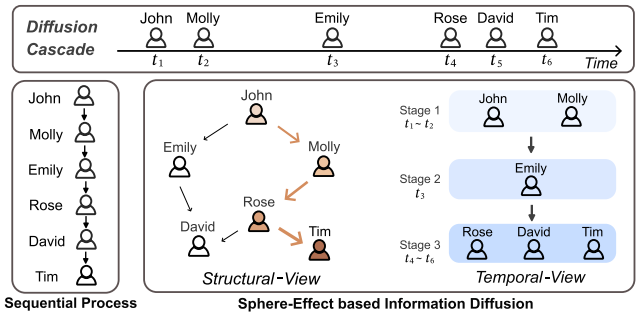

Figure 1: Example of the sphere effect in information diffusion. A sphere-based cascade indicates who is structurally or temporally relevant to *Tim* within the diffusion process.

influences on the diffusion process. However, most existing models [18, 28, 35, 48] consider the diffusion process as a sequential cascade and treat all participants equally in diffusion prediction, which limits their ability to capture the differentiated influences among participants. For instance, a new participant in a diffusion process is likely to be influenced only by those participants who are socially or temporally relevant, while unrelated participants may exert little to no influence. This sequential simplification fails to differentiate the complex and unequal influences among participants. Therefore, a key challenge is how to identify diverse influences of participants from temporal and structural view for improving the prediction. ***(2) Limited Efficiency and Scalability.*** Existing models [4, 18, 28, 33, 44] need to compute embeddings for all users on the entire graph and then retrieve corresponding embeddings of participants when predicting a specific cascade. This workflow incurs expensive costs to process the entire graph, posing challenges for large-scale networks. In fact, the entire graph embedding is unnecessary for a specific cascade prediction because the l-hop neighborhood contains sufficient information for a general l-layer GNN model [12, 21, 34]. In addition, social network data often contains noisy edges [6, 27]. Considering that each cascade involves many participants and the cascade-specific subgraph needs to be extracted online, the online time cost of subgraph extraction becomes a bottleneck. Moreover, the diffusion range is often unknown and potentially widespread, and thus distributed processing for large-scale graphs causes heavy cross-GPU transfer. A crucial challenge is how to efficiently extract a well-selected cascade-specific subgraph and compute embeddings without compromising performance.

In this paper, we propose a novel and efficient prediction framework named SILN. ***For limitation (1)***, we utilize *Sphere Effect* under structural-temporal dual view to enhance the cascade representation for distinguishing participant influences in the diffusion process for the first time. The sphere effect is an important phenomenon in media and communications [5, 8, 22, 23], which indicates users are always involved in information diffusion influenced by various social spheres. Each sphere represents a cluster of users with similar influence characteristics on the diffusion process. This paper considers two representative observations from structural and temporal views, respectively. In the structural view, different social proximities typically have different influences and form distinct spheres [15, 17]. Users are more likely to be influenced by those spheres which are *"closer"* to them. In Fig. 1, Rose, as a one-hop neighbor of Tim, typically has a more direct influence on Tim than those users farther away. In the temporal view, participants with *similar characteristics* tend to engage in the diffusion process at the same temporal-stage, naturally forming spheres based on time proximity [1, 3, 20]. In Fig. 1, John and Molly are the earliest participants, suggesting that they are more likely to be opinion leaders, while Tim is more likely to be an opinion follower, who joins the diffusion at the last stage. By integrating both structural and temporal views, SILN achieves a more nuanced understanding of the influence factors and significantly improves the prediction. ***For limitation (2)***, we propose an efficient user embedding scheme based on the well-selected cascade-specific subgraph, which reduces computation from the entire graph to smaller sampled subgraphs (e.g., for the *Weibo* dataset with 1.04M nodes, fewer than 1% of nodes need to be processed online). Specifically, a multiplex relational sampler and a relation-aware learning module are employed to generate high-quality participant embeddings without sacrificing performance. Furthermore, an optimized graph storage ensures constant-time subgraph extraction (speedup of three orders of magnitude) and reduces storage costs by approximately 30% in practice.

In summary, the main contributions of this paper are as follows:

- We formulate the sphere-effect-based information diffusion under two representative views: structural and temporal. This is the first attempt to integrate the real-world sphere effect into prediction. By leveraging this phenomenon, SILN can effectively identify the differentiated influences within diffusion processes.
- We design an efficient subgraph learning scheme with an optimized graph storage, which significantly reduces the entire graph computation to small cascade-specific subgraphs, and exhibits excellent time and space efficiency on million-scale graphs.
- Extensive experiments on six real-world datasets validate that SILN consistently outperforms seven state-of-the-art baselines. For million-node social networks, all baselines cannot work (e.g., out of memory), whereas SILN still shows excellent performance.

2 Preliminaries

2.1 Concepts and Definitions

A social network can be represented as a directed graph $G = (V, E)$, where V is the set of nodes and $E \in V \times V$ is the set of directed edges. Let $N = |V|$ and $M = |E|$ denote the number of nodes and edges, respectively. In this paper, the concepts of "user" and "node" are interchangeable for social networks.

Definition 1. (**Information Diffusion Cascade**). An information diffusion cascade C_k with k users is a chronological sequence of k tuples, i.e., $C_k = ((v_1, t_1), (v_2, t_2), \ldots, (v_k, t_k))$, where $v_x \neq v_y$ for all $x \neq y$, and $t_x \leq t_{x+1}$ for all $1 \leq x < k$. Each tuple (v_x, t_x) represents a distinct *participation event*, indicating that user v_x is the x-th participant, with the participation time denoted as t_x.

Information diffusion prediction typically uses the sequential cascade C_k as input to predict the next participant v_{k+1}. However, these approaches do not fully account for the differentiated influence among participants. To address this, we introduce the concept of a *sphere-based cascade* to better capture influence patterns.

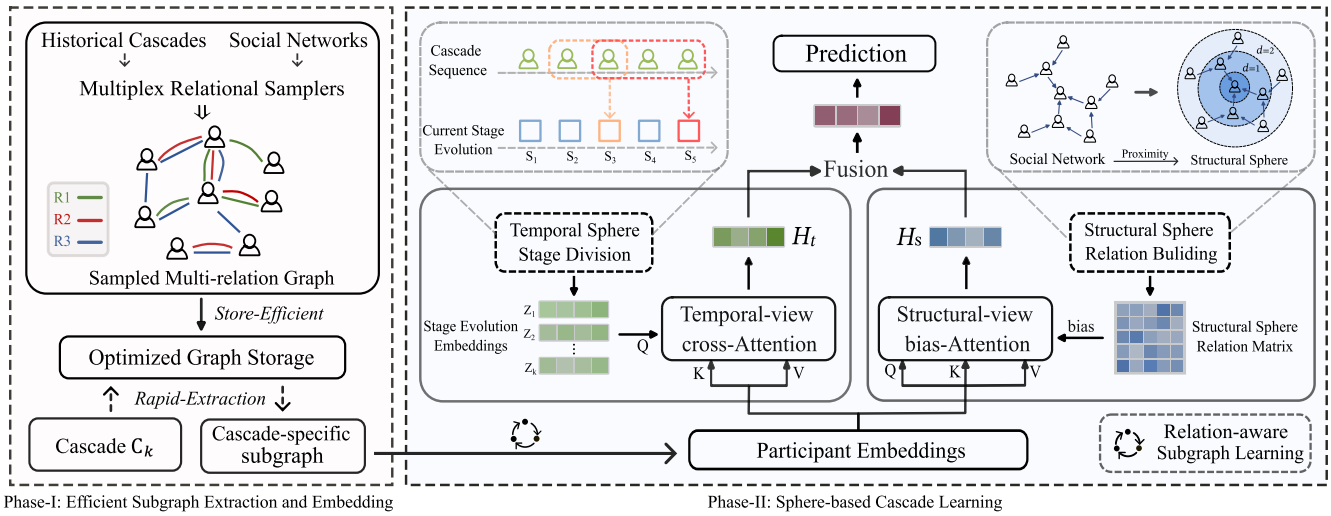

Figure 2: The overall framework of the proposed SILN.

Definition 2. (**Sphere-based Cascade**). A sphere-based cascade $\mathcal{S}_k$ is a partition of the cascade C_k into distinct subsets (spheres), i.e., $\mathcal{S}_k = \{S_1, \cdots, S_{|\mathcal{S}_k|}\}$. Each sphere S_i consists of users who share a specific relationship π_i with the participant v_k. Formally, $S_i = \{(v_x, t_x) | (v_x, t_x) \in C_k, ||v_x, v_k|| \models \pi_i\}$, where $||v_x, v_k|| \models \pi_i$ denotes a relation paradigm, indicating that v_x and v_k satisfy the specified relationship π_i.

Each sphere S_i contains participants who exert similar influence on v_k within the cascade C_k, characterized by the specified relationship π_i. This partitioning allows for a clearer understanding of why v_k is involved in the diffusion process, with each sphere potentially revealing different roles or degrees of influence to the participant v_k. The relation paradigm $||v_x, v_k|| \models \pi_i$ can be formulated from various views to quantify the influence of each sphere on v_k.

2.2 Observation of Sphere Effect

Given a user v_k, we investigate the sphere effect on v_k by introducing specific forms of the relation paradigm. This is based on two representative observations in structural and temporal views.

Structural View. The relation paradigm $||v_x, v_k|| \models \pi_i^s$ in structural view is defined as $d(v_x, v_k) \in \pi_i^s$, where $d(v_x, v_k)$ is a structural proximity metric such as the shortest path distance. In this case, π_i^s *denotes a specified distance range*. For example, in an unweighted social network, if $\pi_i^s = (0, 2]$, the sphere S_i includes participants v_x who are within two-hop neighborhood of v_k.

Temporal View. The relation paradigm $||v_x, v_k|| \models \pi_i^t$ in temporal view is defined by the time proximity to v_k, i.e., $t_k - t_x \in \pi_i^t$, where π_i^t *denotes a specified time range*. For example, if $\pi_i^t = [0, 12)$ hours, the sphere S_i includes participants v_x whose participation time t_x occurs within 12 hours of t_k, i.e., the participation time of v_k.

In this paper, we integrate two representative observations from the structural and temporal views, which highlight participants who are socially or temporally relevant. Moreover, Definition 2 is agnostic to a specific view; more validated views can seamlessly extend to our framework based on the relation paradigm π_i.

2.3 Problem Formulation

Information diffusion prediction aims to identify the next participant v_{k+1} given the current cascade C_k. As discussed above, the sequential cascade C_k overlooks the *sphere effect*—the localized influence patterns inherent in the diffusion process. To address this, we utilize the *sphere-based cascade* $\mathcal{S}_k$ to capture the influence dynamics surrounding v_k. For the entire cascade history, we construct a collection of sphere-based cascades, denoted as $\Gamma_k = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k\}$, which includes the evolving sphere effects for all participants in C_k. Formally, the prediction task is to select an optimal v_x from $V \setminus V_k$, to maximize the following conditional likelihood, as the next participant v_{k+1}:

$$\hat{v}_x = \arg \max_{v_x \in V \setminus V_k} P(v_x \mid G, \Gamma_k^s, \Gamma_k^t) \tag{1}$$

where V_k denotes the set of users already involved in the cascade C_k, and Γ_k^s, Γ_k^t represent the collections of sphere-based cascades under the structural and temporal views, respectively. In the appendix, Table 8 provides a notation table for sphere-related concepts.

3 Methodology

Figure 2 depicts the overall framework of our proposed SILN. To overcome the two major limitations of existing models over large-scale networks, SILN adopts an efficient two-phase framework:

- **Phase-I: Efficient subgraph extraction and embedding.** We first develop an efficient subgraph learning scheme that obtains more expressive participant embeddings from a well-selected cascade-specific subgraph rather than the entire graph. An optimized graph storage is also proposed to reduce storage space overhead and the I/O cost of subgraph extraction.
- **Phase-II: Sphere-based cascade learning.** We then propose the Structural-view bias-Attention network and the Temporal-view cross-Attention network to enhance the sphere effect among cascade participants, which can identify the differentiated influences among participants. Finally, we employ an adaptive fusion module to integrate the various views for diffusion prediction.

3.1 Subgraph Extraction and Embedding

As discussed in Section 1, a key limitation is the need to process the entire social network, much of which is irrelevant to a specific cascade. Despite these promising observations, cascade subgraph extraction poses technical challenges. **(1) Efficiency:** Predicting a cascade often requires frequent online access to multi-hop neighbors for multiple users, leading to substantial I/O overhead. Access-efficient graph storage is critical to addressing the subgraph extraction bottleneck. **(2) Effectiveness:** To utilize various relationship types among users, such as structural connections and interaction preferences, existing methods [4, 18, 33, 44] often store separate graphs for each relationship type and employ multiple learning modules, resulting in redundant storage and high computational costs. To address these challenges, we propose a three-pronged scheme consisting of: (i) a multiplex relational sampler to focus on high-importance nodes, (ii) efficient subgraph extraction using optimized graph storage, and (iii) a lightweight embedding module tailored to multi-relation graphs.

3.1.1 Multiplex Relational Sampler. In social networks, users often exhibit various types of heterogeneous relationships. For example, one type of edge may represent follower-followee relationships on social platforms, while another may indicate shared interests in trending topics. Accurately capturing these relationships is essential for understanding information diffusion. To this end, we propose two samplers tailored to different types of user relationships. First, ***Follower-Followee Sampler:*** This sampler captures follower and followee relationships in the original social network. It performs multiple l-hop random walks starting from each node in parallel, recording the landing probabilities for each visited node. The top-s neighbors with the highest probabilities are then selected. Since graph convolution is sensitive to edge direction in directed graphs, follower and followee relationships are treated as distinct edge types [25, 38]. For follower relationships, we transpose the directed graph (i.e., reverse the edge directions) and apply the same random walk-based sampling. Second, ***Interaction Sampler:*** This sampler identifies interaction relationships based on historical diffusion cascades, capturing user behavioral preferences [28, 33, 44, 48]. Given historical cascades C and a threshold ρ, it calculates the frequency of co-occurrences for each user pair (v_i, v_j) within historical cascades C. An edge (v_i, v_j) is created if the co-occurrence frequency exceeds the threshold ρ. These edges provide valuable information about users' interaction patterns and preferences.

In general, we maintain three types of relations (i.e., *follower*, *followee*, and *interaction*). Previous methods [18, 28, 33, 44, 48] maintain and learn separate graphs for each relation type, which incurs high storage and computation costs. Notably, *multiple edges of different types may exist simultaneously between a node pair*, which presents an optimization opportunity.

3.1.2 Optimized Graph Storage. In this part, we optimize a compact and efficient graph storage method that supports multiplex relations and enables time-efficient subgraph extraction. It uses three arrays:

- ***User Pointers***: This array maintains the starting position δ_u of each user u's neighbors in the Neighbor Indices array.
- ***Neighbor Indices***: This array contains the indices of the neighbors for each user, each entry represents a neighbor of a user.

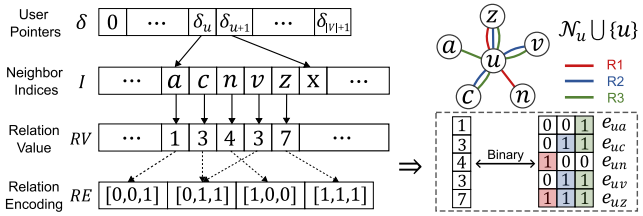

Figure 3: Optimized multiplex relation graph storage.

- ***Relation Values***: This array stores the encoded relation values indicating multiplex relations for every user and its neighbors.

Example. Figure 3 illustrates the optimized multiplex relation graph storage for three types of relations between node u and its neighbors. In *User Pointers* array, δ_u and δ_{u+1} point to the position of a and x respectively, and it means that the segment from a to z in *Neighbor Indices* array maintains the neighbors of u. For a neighbor c of u, its *relation value* is 3, corresponding to binary code 011, which indicates u and c have relations R_2 and R_3 because the second and third digits are 1 in 011. This design allows storing a single entry $\{(u, c) : 3\}$ to represent two relational edges $\{(u, R_2, c), (u, R_3, c)\}$.

Experiments in Table 6 show that this design reduces space usage by approximately 30%. Moreover, as neighbors are stored contiguously, the start and end positions from the User Pointers array enable constant-time ($O(1)$) access to all neighbors of any user. In summary, let r be the number of relation types, s be the sampling number per node, and l be the sampling hops. Compared to baselines [28, 35, 48], the space cost of graph storage is reduced from $O(r \times |E|)$ to $O(|E|)$, and the subgraph extraction time is reduced from $O(k \times s^l)$ to $O(k)$, for a cascade C_k with k participants.

3.1.3 Relation-aware Embedding. After subgraph extraction, we learn embeddings for cascade participants from the cascade-specific subgraph. To balance performance and efficiency for multiplex relations, we design a lightweight relation-aware subgraph learning module. Specifically, for each participant v_i in a cascade C_k, we efficiently extract their neighbor set $\mathcal{N}_i$ of v_i and associated relation types. We then calculate a relation weight ω_{ij} for each neighbor $v_j \in \mathcal{N}_i$ of v_i as follows:

$$\omega_{ij} = \boldsymbol{\beta} \cdot \boldsymbol{e_{ij}} \tag{2}$$

where $\boldsymbol{e_{ij}} \in \mathbb{R}^{|R|}$ is a binary encoding of relation type in RE and $\boldsymbol{\beta} \in \mathbb{R}^{|R|}$ is a learnable parameter, where each element β_r corresponds to the weight of a specific relation type. We apply the *softmax* function to normalize $\boldsymbol{\beta}$, ensuring $\sum_{r=1}^{|R|} \beta_r = 1$. In practice, the number of relation types $|R| = 3$ means that the only parameter $\boldsymbol{\beta}$ is very lightweight. After that, the relation-aware embedding of user v_i can be calculated by relation-weighted neighbor aggregation:

$$\boldsymbol{h_i} = \text{Aggregate}(\{\omega_{ij}\boldsymbol{h_j} : v_j \in \mathcal{N}_i\}) \tag{3}$$

where $\mathcal{N}_i$ is the sampled neighbor set of v_i, $\boldsymbol{h_i} \in \mathbb{R}^d$ denotes the embedding of v_i. We use a simple *Sum* function for aggregation. For a specific cascade C_k with k participants, we obtain the user embeddings of all participants $\boldsymbol{H_k} \in \mathbb{R}^{k \times d}$ in similar way, which are subsequently used for sphere-based cascade learning.

3.2 Sphere-based Cascade Learning

In this section, we enhance the structural-view and temporal-view *sphere effect*. Specifically, we design two variants of the attention mechanism and an adaptive fusion module, which transform participant embeddings $\boldsymbol{H_k}$ into a sphere-based cascade representation.

3.2.1 Structural-view bias-Attention. From the structural view, users are more likely to be influenced by those who are "*closer*" to them [15, 17]. Various efficient structural proximity metrics [15, 31, 36] can be used as relation paradigms. As an example, we adopt the shortest path distance [26] to divide the structural-view sphere-based cascade $\mathcal{S}_k^s = \{S_1, \cdots, S_{|\mathcal{S}_k^s|}\}$ from C_k. Each sphere S_i contains participants at a specified distance from v_k. For each participant $v_x \in S_i$, an influence score $\psi_{k,x}$ is assigned, which also reflects the overall influence of sphere S_i on v_k. The influence score $\psi_{k,x}$ can be calculated as follows:

$$\psi_{k,x} = \frac{1}{\log(\mathrm{e} + d(v_x, v_k))} \tag{4}$$

where the mathematical constant $\mathrm{e} \approx 2.718$, and $d(v_x, v_k)$ is the shortest path distance from v_x to v_k in the social network. In this way, $\boldsymbol{\psi_k} = [\psi_{k,1}, \psi_{k,2}, \ldots, \psi_{k,k}]$ can represent the structural influence between v_k and its previous participants in the cascade C_k.

For the whole diffusion process, we calculate $\boldsymbol{\psi}$ for the collection $\Gamma_k^s = \{\mathcal{S}_1^s, \mathcal{S}_2^s, \cdots, \mathcal{S}_k^s\}$ and can further obtain the structural sphere relation matrix $\boldsymbol{\Psi}$:

$$\boldsymbol{\Psi} = [\boldsymbol{\psi_1}, \boldsymbol{\psi_2}, \ldots, \boldsymbol{\psi_k}]^\top \tag{5}$$

where $\boldsymbol{\Psi} \in \mathbb{R}^{k \times k}$ is a lower triangular matrix. We transform $\boldsymbol{\Psi}$ for model integration as:

$$\tilde{\boldsymbol{\Psi}} = \sigma\left(\boldsymbol{\Psi} \boldsymbol{W_s^1}\right) \boldsymbol{W_s^2} \tag{6}$$

where $\boldsymbol{W_s^1}, \boldsymbol{W_s^2} \in \mathbb{R}^{d \times d}$ are the learnable transformation matrices and $\sigma(\cdot)$ is the ReLU activation. To differentiate the influence of different spheres, we incorporate $\tilde{\boldsymbol{\Psi}}$ as a bias term into the *Structural-view bias-Attention Network* (SSAN), which modulates attention weights to emphasize structurally closer users. The i-th attention head of SSAN is defined as:

$$\boldsymbol{Q_i} = \boldsymbol{H_k'} \boldsymbol{W_i^Q}, \boldsymbol{K_i} = \boldsymbol{H_k'} \boldsymbol{W_i^K}, \boldsymbol{V_i} = \boldsymbol{H_k'} \boldsymbol{W_i^V} \tag{7}$$

$$\mathrm{SSAN}_i(\boldsymbol{Q_i}, \boldsymbol{K_i}, \boldsymbol{V_i}) = \mathrm{softmax}\left(\frac{\boldsymbol{Q_i} \boldsymbol{K_i}^\top}{\sqrt{d'}} + \boldsymbol{M} + \tilde{\boldsymbol{\Psi}}\right) \boldsymbol{V_i} \tag{8}$$

where d is the embedding dimension, B is the number of heads, $d' = d/B$, $\boldsymbol{W_i^Q}$, $\boldsymbol{W_i^K}$, and $\boldsymbol{W_i^V}$ are learnable weight matrices. The mask matrix $\boldsymbol{M} \in \mathbb{R}^{n \times n}$ is used to block out future participants.

Finally, we concatenate the output of B attention heads and use a position-wise Feed-Forward Network (FFN) to capture non-linear dependencies within the cascade:

$$\boldsymbol{T} = [\mathrm{SSAN}_1 || \ldots || \mathrm{SSAN}_B] \boldsymbol{W^o} \tag{9}$$

$$\boldsymbol{H_s} = (\mathrm{ReLU}(\boldsymbol{T} \boldsymbol{W_s^1} + \boldsymbol{b_s^1})) \boldsymbol{W_s^2} + \boldsymbol{b_s^2} \tag{10}$$

where $\boldsymbol{W^o}$, $\boldsymbol{W_s^1}$, $\boldsymbol{W_s^2}$, $\boldsymbol{b_s^1}$, and $\boldsymbol{b_s^2}$ are learnable parameters, $||$ is the concatenation operation, and the representation $\boldsymbol{H_s} \in \mathbb{R}^{k \times d}$ is the sphere-based cascade representation in structural view. Our formulation supports the seamless substitution of alternative proximity metrics via $\psi_{k,x}$, allowing flexible hot-swapping; we leave this exploration for future work.

3.2.2 Temporal-view cross-Attention. As introduced in Section 2.2, participants with *similar characteristics* tend to engage in the diffusion process during the same temporal stage [1, 3, 19, 45]. To capture this behavior, we identify each participant's temporal stage and enhance the shared characteristics within that stage.

For a participant v_k, the current stage S_k^c is defined based on temporal proximity. Specifically, S_k^c includes participants whose interactions occurred within a certain time range preceding (v_k, t_k). The time range of current stage S_k^c is identified as:

$$\min\{q \times \overline{\Delta t}, t_k - t_{k-q}\}, \quad \text{where } \overline{\Delta t} = \frac{1}{k} \sum_{x=1}^{k} (t_x - t_{x-1}) \tag{11}$$

where parameter q controls the time range of current stage. If timestamps are unavailable, we fallback to selecting the most recent q participants as the current stage. The earlier stage S_k^e includes all participants not in S_k^c. Hence, the sphere-based cascade for v_k is represented as $\mathcal{S}_k^t = \{S_k^c, S_k^e\}$. Across the whole diffusion process, the collection $\Gamma_k^t = \{\mathcal{S}_1^t, \mathcal{S}_2^t, \cdots, \mathcal{S}_k^t\}$. However, since each earlier stage S_k^e is implicitly captured in $\{S_1^c, \ldots, S_{k-1}^c\}$, we only need to focus on the evolution of the current stage set $\{S_1^c, S_2^c, \ldots, S_k^c\}$. Then we compute the embedding of each stage, such as S_k^c as follows:

$$\boldsymbol{z_k} = \boldsymbol{o_k} \odot \left(\frac{1}{|S_k^c|} \sum_{v_x \in S_k^c} \boldsymbol{h_x}\right) + (\boldsymbol{1} - \boldsymbol{o_k}) \odot \boldsymbol{h_k} \tag{12}$$

$$\boldsymbol{o_k} = \sigma\left(\boldsymbol{W_g^1} \left(\frac{1}{|S_k^c|} \sum_{v_x \in S_k^c} \boldsymbol{h_x}\right) + \boldsymbol{W_g^2} \boldsymbol{h_k}\right) \tag{13}$$

where $\boldsymbol{h_x}$ is the embedding of participant v_x, $\boldsymbol{W_g^1}, \boldsymbol{W_g^2} \in \mathbb{R}^{d \times d}$ are learnable parameters, $\sigma(\cdot)$ is the sigmoid activation, and $\odot$ is the Hadamard product. The collection of stage embeddings is $\boldsymbol{Z_k} = \{\boldsymbol{z_1}, \boldsymbol{z_2}, \ldots, \boldsymbol{z_k}\}$, representing stage evolution patterns.

We introduce the *Temporal-view cross-Attention Network* (TCAN) to capture stage-wise influence across the cascade. TCAN computes cross-attention between stage embeddings $\boldsymbol{Z_k}$ and participant embeddings $\boldsymbol{H_k} = \{\boldsymbol{h_x} | v_x \in C_k\}$. For the i-th attention head:

$$\boldsymbol{Q_i} = \boldsymbol{Z_k} \boldsymbol{W_i^Q}, \quad \boldsymbol{K_i} = \boldsymbol{H_k} \boldsymbol{W_i^K}, \quad \boldsymbol{V_i} = \boldsymbol{H_k} \boldsymbol{W_i^V} \tag{14}$$

$$\mathrm{TCAN}_i(\boldsymbol{Q_i}, \boldsymbol{K_i}, \boldsymbol{V_i}) = \mathrm{softmax}\left(\frac{\boldsymbol{Q_i} \boldsymbol{K_i}^\top}{\sqrt{d'}} + \boldsymbol{M}\right) \boldsymbol{V_i} \tag{15}$$

Finally, the Multi-Head mechanism and Feed-Forward Network (FFN) are applied as in Eqs. (9) and (10), resulting in the temporal-view sphere-based cascade embedding $\boldsymbol{H_t} \in \mathbb{R}^{k \times d}$. Eq. (15) allows each stage to attend over all prior participants, assigning higher weights to those most relevant to the current stage S_k^c, including influential users from earlier stages.

Overall, both SSAN and TCAN assign higher attention weights to sphere-important participants. In Section 4.5, we empirically verify that such participants exert greater influence on the diffusion process and significantly improve prediction accuracy.

3.2.3 Dual-View Gated Fusion. Both structural and temporal views offer valuable insights into information diffusion, and we employ gated fusion to adaptively integrate them:

$$\boldsymbol{\alpha} = \sigma(\boldsymbol{W_f^1} \boldsymbol{H_s} + \boldsymbol{W_f^2} \boldsymbol{H_t} + \boldsymbol{b_f}) \tag{16}$$

$$\boldsymbol{H} = \boldsymbol{\alpha} \odot \boldsymbol{H_s} + (\boldsymbol{1} - \boldsymbol{\alpha}) \odot \boldsymbol{H_t} \tag{17}$$

Algorithm 1: Learning Procedure of SILN

Input: Social network G_o, Training set of cascades C
Output: Trained parameters θ

1 $G \leftarrow$ MultiplexRelationalSamplers(G_o, C);
2 Offline store G in optimized graph storage;
3 Initialize all parameters in θ;
4 **while** $\mathcal{L}$ *has not converged* **do**
5 **for** $C_k \in C$ **do**
  // Phase-I: Subgraph Extraction and Embedding
6 **for** $v_i \in C_k$ **do**
7 Extract neighbors $\mathcal{N}_i$ in $O(1)$ time from G;
8 $\boldsymbol{h_i} \leftarrow$ Aggregate$(\{\omega_{ij}\boldsymbol{h_j} : v_j \in \mathcal{N}_i\})$;
9 Participant embeddings $\boldsymbol{H_k} = \{\boldsymbol{h_1}, \ldots, \boldsymbol{h_k}\}$ for C_k;
  // Phase-II: Sphere-based Cascade Learning
10 Structural relation matrix $\tilde{\boldsymbol{\Psi}}$ in Eq. (6);
11 $\boldsymbol{H_s} \leftarrow \text{SSAN}(\boldsymbol{H_k'}, \boldsymbol{H_k'}, \boldsymbol{H_k'}, \tilde{\boldsymbol{\Psi}})$;
12 Stage evolution embeddings $\boldsymbol{Z_k}$ in Eq. (12);
13 $\boldsymbol{H_t} \leftarrow \text{TCAN}(\boldsymbol{Z_k}, \boldsymbol{H_k}, \boldsymbol{H_k})$;
14 Cascade representation $\boldsymbol{H} \leftarrow \text{Fusion}(\boldsymbol{H_t}, \boldsymbol{H_s})$;
15 Predict next participant $\hat{y}$ in Eq. (18);
16 Update parameters by minimizing the loss $\mathcal{L}(\theta)$;
17 **return** *All parameters* θ;

where $\boldsymbol{W_f}^1, \boldsymbol{W_f}^2 \in \mathbb{R}^{d\times d}$, and $\boldsymbol{b_f}$ are learnable parameters, $\sigma(\cdot)$ is the *sigmoid* function, $\odot$ denotes the Hadamard product, $\boldsymbol{H} \in \mathbb{R}^{k\times d}$ is the final cascade representation for diffusion prediction.

3.3 Prediction & Training

3.3.1 Diffusion Prediction. We can calculate the probability $\hat{y} \in \mathbb{R}^{|C_k|\times|V|}$ of the next participant using a softmax function:

$$\hat{y} = \text{softmax}(\boldsymbol{W_p}\boldsymbol{H} + Mask) \tag{18}$$

where $\boldsymbol{W_p}$ maps the cascade representation $\boldsymbol{H}$ to user-specific space, $Mask$ is applied to exclude already participating users.

3.3.2 Training Objective. We use the cross-entropy function to minimize the predicted $\hat{y}_{ij}$ and the ground-truth y_{ij} as the objective:

$$\mathcal{L}(\theta) = -\sum_{j=2}^{|C_k|}\sum_{i=1}^{|V|} y_{ij}\log(\hat{y}_{ij}) \tag{19}$$

in which θ denotes all parameters in the model, if the user v_i participates in cascade C_k at the step j, $y_{ij} = 1$, otherwise $y_{ij} = 0$. This objective encourages the model to assign higher probability to the correct participant at each diffusion step.

3.3.3 Algorithm Pseudocode. Algorithm 1 implements the SILN training process. Lines 1-2 selectively sample and store neighbors with multiplex relations. For each cascade C_k, SILN first conducts the *Subgraph Extraction and Embedding* module to rapidly extract cascade-specific subgraph and encode user embeddings (Lines 5–9). Next, SILN processes *Sphere-based Cascade Learning* module on structural and temporal views, followed by adaptive fusion to optimize the prediction objective (lines 10-16).

Table 1: Statistics for evaluation datasets.

Datasets	Benchmark				Large-scale	
	Twitter	Douban	Android	Christianity	Weibo-M	Weibo-L
# Users	12,627	12,232	2,927	1,651	331,321	1,046,140
# Links	309,631	396,580	48,573	35,624	14,534,265	63,487,436
Density	24.52	30.21	16.59	21.58	43.87	60.69
# Cascades	3442	3475	679	589	6828	31,521
Avg. Len.	32.60	21.76	33.30	22.90	163.76	184.87
Density	8.89	6.18	7.72	8.17	3.36	5.57

3.3.4 Complexity Analysis. Let M be the number of edges in G, k be the length of cascade C_k, s denote the number of sampled neighbors per user, and d be the embedding dimension. For graph learning, SILN extracts and learns the sampled subgraph for C_k with time complexity $O(skd)$. In contrast, existing models learns on the entire graph with different relation types, leading to at least $O(Md)$ complexity. The neighbor sampler ensures that $\boldsymbol{sk} \ll \boldsymbol{M}$. For sphere-based cascade learning, the additional complexity for modeling sphere effect is $O(kd)$, and thus the complexity is still $O(k^2d)$, similar to Self-Attention. Overall, the total time complexity of SILN is $O((sk+k^2)d)$, while other models require $O((M+k^2)d)$.

4 Performance Evaluation

4.1 Experiment Setup

4.1.1 Datasets. We conduct experiments on four widely-used benchmark datasets, including *Twitter* [13], *Douban* [47], *Android* [30], and *Christianity* [30]. Further, we evaluate the large-scale social network dataset Weibo [46] for the first time, where we preprocess the raw data and extract two sub-datasets of different sizes, denoted as *Weibo-M* and *Weibo-L*. Each dataset includes a static social network and a set of diffusion cascades. We follow prior work by splitting cascades chronologically into training, validation, and test sets in an 8:1:1 ratio. To avoid information leakage, we construct a multiplex relation graph using only the original social network and training cascades. Detailed statistics are shown in Table 1.

4.1.2 Evaluation Metrics. Following prior studies [18, 28, 33], we use *Hits rate* (Hits@K) and *Mean Average Precision* (MAP@K) to evaluate the performance on benchmark datasets. Let u_i and $\hat{U}_i$ be the ground-truth user and the predicted top-K ranked users (i.e., $|\hat{U}_i| = K$), the Hits@K and MAP@K is defined as:

$$\text{Hits@K} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(u_i \in \hat{U}_i) \qquad \text{MAP@K} = \frac{1}{N}\sum_{i=1}^{N}\frac{\mathbb{I}(u_i \in \hat{U}_i)}{\gamma_{u_i}}$$

where N is the total number of cascades and γ_{u_i} is the predicted rank of user u_i. The indicator $\mathbb{I}(x) = 1$ if x is true, and 0 otherwise.

For large-scale Weibo datasets, identifying a single next participant from million users is exceedingly demanding. Therefore, we define a more relaxed metric on large-scale Weibo datasets, named the *Group Hit Rate* on top-K (GHR@K) as follows:

$$GHR@K = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(|U_i \cap \hat{U}_i| \geq \epsilon)$$

where U_i denotes a group of ground-truth users for the i-th cascade, ϵ is the hit threshold. In our experiments, we select multiple future participants as a ground-truth group $U_i = \{v_{k+1}, \ldots, v_{k+|U_i|}\}$ for a cascade C_k. Note that GHR reduces to Hits when $|U_i| = \epsilon = 1$.

Table 2: Hits@K on four benchmark datasets (%), scores are the higher the better.

Datasets	Twitter			Douban			Android			Christianity		
Metrics	H@10	H@50	H@100	H@10	H@50	H@100	H@10	H@50	H@100	H@10	H@50	H@100
DyHGCN	31.88	45.05	52.19	18.71	32.33	39.71	9.10	16.38	23.09	26.62	42.80	52.47
MSHGAT	33.50	49.59	58.91	21.33	35.25	42.75	10.41	20.31	27.55	28.80	47.14	55.62
DisenIDP	34.01	52.21	60.39	21.46	35.81	43.01	9.88	18.64	26.06	29.04	47.92	57.05
RotDiff	35.90	52.46	61.21	21.50	35.92	42.96	10.73	21.58	28.47	30.71	51.44	59.97
MINDS	34.26	52.83	59.54	21.38	35.44	43.06	10.86	20.47	27.92	31.32	50.63	61.04
GODEN	35.61	53.08	61.87	21.71	35.89	42.81	10.69	21.23	28.05	29.39	49.21	58.02
CARE	36.32	53.29	63.12	21.83	36.04	43.14	10.98	22.74	29.13	30.79	52.07	62.44
SILN (ours)	**39.72**	**55.67**	**65.03**	**21.95**	**36.23**	**43.48**	**11.10**	**23.14**	**30.43**	**32.14**	**53.13**	**63.39**

Table 3: MAP@K on four benchmark datasets (%), scores are the higher the better.

Datasets	Twitter			Douban			Android			Christianity		
Metrics	M@10	M@50	M@100	M@10	M@50	M@100	M@10	M@50	M@100	M@10	M@50	M@100
DyHGCN	20.87	21.48	21.58	10.61	11.26	11.36	6.09	6.40	6.50	15.64	16.30	16.44
MSHGAT	22.49	23.17	23.30	11.72	12.52	12.60	6.39	6.87	6.96	17.44	18.27	18.40
DisenIDP	23.04	23.61	23.94	11.69	12.55	12.64	5.93	6.31	6.40	18.21	18.96	19.22
RotDiff	24.06	24.82	24.95	12.02	12.83	12.97	6.81	7.20	7.36	18.73	19.32	19.65
MINDS	22.90	23.28	23.46	11.75	12.61	12.72	6.79	7.19	7.33	19.13	19.96	20.01
GODEN	24.80	25.62	25.75	12.06	12.53	12.64	6.88	7.25	7.58	19.02	19.51	19.96
CARE	25.17	25.86	25.93	12.42	13.01	13.29	6.91	7.44	7.63	19.20	19.88	20.02
SILN (ours)	**28.23**	**28.79**	**29.09**	**12.87**	**13.41**	**13.58**	**7.17**	**7.69**	**7.79**	**19.65**	**20.52**	**20.66**

4.1.3 Compared Methods. We consider seven recent models as competitors and briefly introduce them as follows: (1) **DyHGCN** [44] utilizes GCN to learn user embeddings on both social and diffusion graphs. (2) **MSHGAT** [33] introduces a Hypergraph Network to capture diffusion relationships, then uses Self-Attention to model the sequential cascade. (3) **DisenIDP** [4] disentangles user embeddings on two intent-specific hypergraphs, and then captures LS-term sequential cascade. (4) **RotDiff** [28] develops a hyperbolic rotation model to encode user embeddings and uses rotation positional encoding for cascade learning. (5) **MINDS** [18] employs adversarial training and orthogonality constraints to mitigate user feature redundancy. After that, it uses the LSTM for sequential cascade modeling. (6) **GODEN** [35] introduces ODE-based GNN to learn the dynamic user relations and also uses timestamps positional encoding to improve local dynamics. (7) **CARE** [48] retrieves similar historical cascades to enhance current cascade learning, which can provide more relevant historical interactions.

4.1.4 Implementation Details. The experiments are conducted on NVIDIA RTX 4090 (24GB) GPU. For baselines, we follow the settings in the original papers and rerun them to report the results accordingly. Cascades are split with a ratio of 8:1:1 for training, validation, and testing. To prevent information leakage, only training cascade data is used as historical interaction data. The maximum cascade length is fixed at 200 across all methods. For hyperparameters, the follower-followee sampler uses l = {2, 3, 4} and s = {30, 50, 100}, while the interaction sampler uses ρ = {2, 3}. The size of the temporal-view current stage is q = {2, 3}. We use the AdamW optimizer with a learning rate of 1e-3. The batch size is 16 or 32, the embedding dimension is {64, 128}. For GHR@K, the ground-truth group size $|U_i|$ is set to 50, and the hit threshold ϵ=1. In addition, we analyze three key hyperparameters {l, ρ, q} in Appendix A.2.

4.2 Effectiveness Analysis

4.2.1 Performance on benchmark datasets. Table 2 and Table 3 summarize the benchmarked performance comparison of SILN with seven recent competitors. Specifically: (1) Compared to competitors, SILN incorporates the sphere effect to capture the differentiated influence among participants in both structural and temporal views, which goes beyond a simple sequential cascade and results in a more accurate prediction. (2) Unlike previous multi-relation methods that emphasize nuanced user embeddings and perform global graph learning by constructing complex structures, SILN efficiently extracts well-selected cascade-specific subgraphs. This avoids using complex modules and enables efficient learning of multiplex relations, while maintaining excellent performance. (3) The results also indicate the need to include various relevant factors, such as user relations, temporal patterns, and structural dependencies. SILN effectively exploits multiplex user relations and captures the sphere effect. With its efficiency and effectiveness, SILN provides a comprehensive solution for diffusion prediction.

4.2.2 Performance on large-scale datasets. We compare the performance of SILN with the strongest competitors CARE and GODEN. However, these models cannot work on large-scale social networks, while all other competitors also cannot work (not shown in the Table 4). Therefore, we extend the analysis to include the classic GCN [21], GAT [34], and GraphSAGE [12] models, which are typically used as base modules in complex competitors and thus *not included in all experiments*. As shown in Table 4, simple GCN and

Table 4: GHR@K on two large-scale Weibo datasets with ϵ=1. "–" indicates methods that are out-of-memory (OOM).

Models	Weibo-M			Weibo-L		
	G@100	G@500	G@1000	G@100	G@500	G@1000
GCN	6.45	21.18	33.42	–	–	–
GAT	–	–	–	–	–	–
SAGE	5.34	18.84	30.23	–	–	–
GODEN	–	–	–	–	–	–
CARE	–	–	–	–	–	–
SILN	**9.78**	**28.39**	**42.14**	**2.17**	**7.49**	**12.83**

Table 5: Computational efficiency on three datasets. Only methods without out-of-memory (OOM) errors are included.

Datasets		Twitter			Weibo-M			Weibo-L
Models		GODEN	CARE	SILN	GCN	SAGE	SILN	SILN
Runtime	**Train**	42.73	48.68	17.67	219.16	192.79	222.58	3169.52
(seconds.)	**Inf.**	7.14	7.91	2.89	24.17	22.17	25.32	297.41
Memory	**RAM**	5.68	6.79	2.25	5.02	5.02	4.94	9.27
(GB)	**GPU**	7.73	8.42	**2.01**	19.47	15.78	**10.34**	**22.58**

Table 6: Efficiency optimization of graph storage.

Efficiency	**Storage Space** (MB)			**Access Time** (sec.)		
	Twitter	Weibo-M	Weibo-L	Twitter	Weibo-M	Weibo-L
Before	10.57	742.65	2074.60	6.0546	58.5473	>100
After	**7.49**	**548.83**	**1380.72**	**~0.002**	**~0.002**	**~0.002**

GraphSAGE can handle the WeiboM dataset, while GAT needs to calculate the attention weights for every node pair, causing expensive memory and computational overheads. On million-node graphs, the full-graph training of GCN and GraphSAGE are still out-of-memory. In contrast, SILN shows excellent performance by focusing on smaller subgraphs. Although GraphSAGE has a sampling mechanism, it still needs to load the graph data to the GPU-side; if sampling is done on the CPU-side, frequent neighbor sampling and data transfer become an efficiency bottleneck.

4.3 Efficiency Analysis

4.3.1 Computational Efficiency. As shown in Table 5, we analyze the efficiency on the Weibo and Twitter datasets, other smaller datasets have similar performance to Twitter. Specifically: **(1) Runtime:** Our offline sampler combined with storage optimization can avoid the complex online graph construction, which significantly saves runtime. Although SILN is more complex compared to simple base models GCN and GraphSAGE, it does not consume much more runtime for trade-off performance. **(2) Memory:** Compared with CARE and GODEN, our SILN avoids maintaining multiple graphs for learning user multiplex relations. Furthermore, the subgraph learning of SILN does not need to load the entire graph into GPU memory, which exhibits superior GPU memory efficiency compared to others with directly learning on the entire graph.

4.3.2 Efficiency of Graph Storage. For space efficiency, we compare the memory cost before and after optimizing the duplication of multiplex relations. For time efficiency, we compare the time cost of subgraph extraction with the widely used COO format [7], specifically simulating the batch size at 64 and the cascade length at 100, resulting in 6400 neighbor accesses. As shown in Table 6, memory cost of graph storage is reduced by approximately 30%.

Table 7: Ablation performance of SILN and its variants.

Models	Twitter		Android	
	Hits@100	MAP@100	Hits@100	MAP@100
SILN	**65.03**	**29.09**	**30.43**	**7.79**
w/o MR	62.38	28.16	28.64	7.37
w/o RW	63.96	28.72	29.11	7.58
w/o TA	63.46	27.76	29.32	7.67
w/o SA	63.91	26.42	29.50	7.26
Sequential	62.27	24.71	28.96	7.01
w/o GF	64.44	27.87	29.81	7.51

Subgraph extraction consumes merely 0.002 seconds in different datasets, almost independent of the entire graph size. In contrast, the COO format would cause an efficiency bottleneck.

4.4 Ablation Study

To investigate the impact of each component on the SILN, we compare different variants with the original model, as shown in Table 7. **(1) Graph Learning Module:** We obtain the **w/o MR** variant using only the original social network and **w/o RW** with average aggregation of multiplex relations. SILN outperforms both variants on the both datasets, which effectively captures the complementary features inherent in multiple-type relationships. **(2) Cascade Learning Module:** Both variants show performance degradation when not considering temporal-view cross-attention (**w/o TA**) or structural-view bias-attention (**w/o SA**). Moreover, the worst performance is observed with the basic self-attention (**Sequential**), which verifies the plausibility of our sphere-based information diffusion and the limitations of the sequential cascade. **(3) Dual-view Fusion Module:** We use a fixed scalar of 0.5 to fuse different views (**w/o GF**), replacing the learnable feature filter α in Eq. (17). As we observed the performance degradation, selective integration of information is crucial for prediction. It validates the necessity of the adaptive dual-view fusion without manual selection of α.

4.5 Analysis on Sphere Effect

4.5.1 Influence Visualization. We randomly select a cascade with 46 participants from the Twitter dataset to study SILN's ability to capture the sphere effect. We compare our *Temporal-view cross-Attention* (TCAN) and *Structural-view bias-Attention* (SSAN) with the *Self-Attention mechanism*, which is widely used for *Sequential Cascade* [4, 28, 33, 44, 48]. Figure 4 shows the attention weights between earlier participants and the next participant. **(1) Variance of Weights:** The attention variance of Self-Attention is 3e-5, while our proposed TCAN and SSAN modules show higher variances at 3e-4 and 1e-3, respectively. As discussed in Section 1, sequential cascades almost treat all participants equally. By contrast, sphere-based cascades better distinguish influences. **(2) Maximum Weight:** The maximum weight of Self-Attention is 0.032, while our TCAN and SSAN reach 0.069 and 0.171, respectively. Moreover, the participant with the maximum weight indicated by both TCAN and SSAN is the 38th participant. We found that *User 38 is a direct neighbor of User 46, and these two users frequently co-occur in other cascades.* As captured by SILN, User 38 is likely to directly influence User 46. However, Self-Attention fails to identify who influences whom, and assigns nearly equal weights to earlier participants.

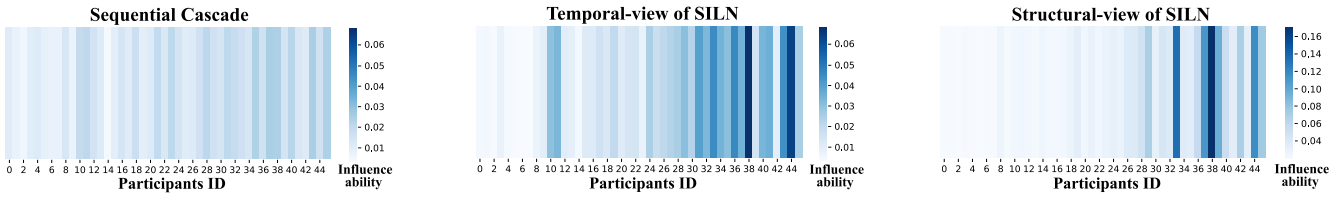

Figure 4: Influence Visualization: Color intensity indicates the influence on the last participant. The sequential cascade considers all participants almost equally. Our proposed SILN captures the differentiated influences in structural and temporal views.

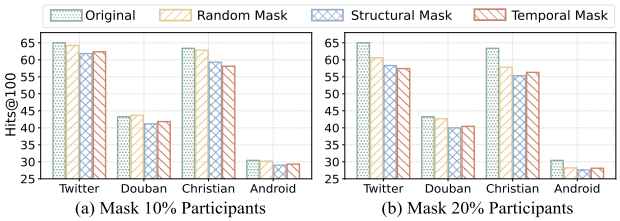

Figure 5: Counterfactual evaluation of sphere effect.

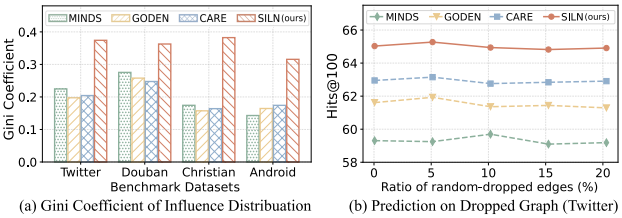

Figure 6: Empirical analysis of motivations. The higher Gini Coefficient indicates more differentiated user influences.

4.5.2 Counterfactual Evaluation. We conduct a counterfactual evaluation to determine *whether sphere-important participants are indeed more important for prediction*. Specifically, we deliberately ignore the influence among some participants by masking attention weights (set to 0) in Eqs. (8) and (15). ① **Structural Mask:** Mask the influence between participants who are structural neighbors. ② **Temporal Mask:** Mask the influence between participants in the same temporal stage. As shown in Figure 5, masking sphere-important participants in both structural and temporal views leads to significantly larger performance degradation compared to random masking. This highlights the critical role of these participants, consistent with the observations in Section 2.2. Notably, random masking of 10% of participants yields performance close to the original results, and even improves performance in the Douban dataset. This suggests that not all participants are influential or useful, further supporting our motivation.

4.5.3 Empirical Analysis of Motivation. We conduct an empirical analysis of the two limitations discussed in Section 1. To quantify *Limitation 1*, we use the common statistical *Gini Coefficient* [9] to measure the inequality in influence distribution (i.e., correlation weights). As shown in Fig. 6(a), three baselines exhibit lower Gini coefficients, indicating they tend to assign nearly equal influence to all participants. For *Limitation 2*, Fig. 6(b) shows that randomly dropping some edges can preserve or even enhance performance, which indicates the inherent noise in the entire graph data.

5 Related Work

Information diffusion prediction aims to identify future participants in social networks, with significant applications across various domains [16, 24, 32, 41]. Recent learning-based methods can be summarized as follows: **(1) Cascade-focused Methods:** Earlier works like NDM [42] and SNIDSA [39] use CNNs and RNNs to model sequential diffusion dependencies, which overlook global social relationships among users. **(2) Social Network-based Methods:** Graph Neural Networks (GNNs) extend this by learning user relationships on social networks [30, 43], enabling the integration of social and sequential patterns. For example, Inf-VAE [30] uses GraphVAE and a co-attentive module to enhance user embeddings on social networks. **(3) Interaction-enhanced Methods:** Historical interaction data can reveal preference relationships between users. DyHGCN [44] utilizes both the social network and diffusion graph to learn user embeddings. Recent advancements in hypergraph learning enable more nuanced modeling of user interactions. Methods such as DisenIDP [4], MSHGAT [33], and MINDS [18] employ hypergraphs to capture dynamic user preferences. RotDiff [28] uses hyperbolic space instead of Euclidean space for more refined embeddings. GODEN [35] designs an ODE-based GNN to model dynamic relationships, while CARE [48] retrieves similar historical cascades to enhance current cascade learning by integrating more interaction data. While these methods successfully learn detailed user embeddings, they often face scalability challenges, limiting their real-world practicality. Moreover, these methods struggle to identify differentiated influence patterns among participants.

6 Conclusion and Future Work

In this paper, we propose an efficient and effective framework for information diffusion prediction, called SILN. SILN employs a subgraph-centric learning scheme with optimized graph storage, reducing computation from the entire graph to smaller subgraphs. It also captures the sphere effect through both structural and temporal views, enhancing cascade representations for improved prediction. Extensive experiments demonstrate the superiority of SILN in both effectiveness and efficiency. In future work, we plan to integrate LLMs for modeling emerging topics and users with sparse activity, especially considering rich multimodal data on social platforms.

Acknowledgments

This work is partially supported by National Natural Science Foundation of China (No. 62472304, No. 62436001, and No. U22A2025). All opinions in this paper are those of the authors and do not necessarily reflect the views of the funding agencies. We would like to thank all the anonymous reviewers for their helpful feedback.

References

[1] A.-L. Barabási. 2005. The Origin of Bursts and Heavy Tails in Human Dynamics. *Nature* 435 (2005), 207–211.

[2] Jan B Broekaert, Davide La Torre, and Faizal Hafiz. 2022. Competing control scenarios in probabilistic SIR epidemics on social-contact networks. *Annals of Operations Research* (2022), 1–24.

[3] Qi Cao, Huawei Shen, Keting Cen, Wentao Ouyang, and Xueqi Cheng. 2017. Deephawkes: Bridging the gap between prediction and understanding of information cascades. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1149–1158.

[4] Zhangtao Cheng, Wenxue Ye, Leyuan Liu, Wenxin Tai, and Fan Zhou. 2023. Enhancing Information Diffusion Prediction with Self-Supervised Disentangled User and Cascade Representations. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 3808–3812.

[5] Mark Eisenegger and Mike S Schäfer. 2023. Editorial: Reconceptualizing public sphere(s) in the digital age? On the role and future of public sphere theory. *Communication Theory* 33, 2-3 (2023), 61–69. doi:10.1093/ct/qtad011

[6] Zihan Feng, Rui Wu, Yajun Yang, Hong Gao, Xin Wang, Xueli Liu, and Qinghua Hu. 2024. Multi-level Contrastive Learning on Weak Social Networks for Information Diffusion Prediction. In *Database Systems for Advanced Applications - 29th International Conference, DASFAA 2024*, Vol. 14855. Springer, 84–100.

[7] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

[8] Mirta Galesic, Wändi Bruine de Bruin, Marion Dumas, A Kapteyn, JE Darling, and Erik Meijer. 2018. Asking about social circles improves election predictions. *Nature Human Behaviour* 2, 3 (2018), 187–193.

[9] Gini Coefficient [n. d.]. https://en.wikipedia.org/wiki/Gini_coefficient

[10] Amit Goyal, Francesco Bonchi, and Laks VS Lakshmanan. 2010. Learning influence probabilities in social networks. In *Proceedings of the third ACM international conference on Web search and data mining*. 241–250.

[11] Jiewen Guan, Xin Huang, and Bilian Chen. 2023. Community-Aware Social Recommendation: A Unified SCSVD Framework. *IEEE Trans. Knowl. Data Eng.* 35, 3 (2023), 2379–2393.

[12] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Annual Conference on Neural Information Processing Systems 2017(NeurIPS)*. 1024–1034.

[13] Nathan Oken Hodas and Kristina Lerman. 2013. The Simple Rules of Social Contagion. *CoRR* abs/1308.5015 (2013). http://arxiv.org/abs/1308.5015

[14] Mohammad Raihanul Islam, Sathappan Muthiah, Bijaya Adhikari, B. Aditya Prakash, and Naren Ramakrishnan. 2018. DeepDiffuse: Predicting the 'Who' and 'When' in Cascades. In *IEEE International Conference on Data Mining, ICDM 2018*. 1055–1060.

[15] Glen Jeh and Jennifer Widom. 2003. Scaling personalized web search. In *Proceedings of the Twelfth International World Wide Web Conference, WWW 2003, May 20-24, 2003*. ACM, 271–279.

[16] Xueqi Jia, Jiaxing Shang, Dajiang Liu, Haidong Zhang, and Wancheng Ni. 2022. HeDAN: Heterogeneous diffusion attention network for popularity prediction of online content. *Knowl. Based Syst.* 254 (2022), 109659.

[17] Yuli Jiang, Yu Rong, Hong Cheng, Xin Huang, Kangfei Zhao, and Junzhou Huang. 2022. Query Driven-Graph Neural Networks for Community Search: From Non-Attributed, Attributed, to Interactive Attributed. *Proc. VLDB Endow.* 15, 6 (2022), 1243–1255.

[18] Pengfei Jiao, Hongqian Chen, Qing Bao, Wang Zhang, and Huaming Wu. 2024. Enhancing Multi-Scale Diffusion Prediction via Sequential Hypergraphs and Adversarial Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 8571–8581.

[19] Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus A. Brubaker. 2019. Time2Vec: Learning a Vector Representation of Time. *CoRR* abs/1907.05321 (2019).

[20] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. 137–146.

[21] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017*.

[22] Adam Kucharski. 2016. Study epidemiology of fake news. *Nature* 540, 7634 (2016), 525–525.

[23] Laurens Lauer. 2024. *Public Spheres, Media, and Democracy*. Springer Fachmedien Wiesbaden, Wiesbaden, 9–25. doi:10.1007/978-3-658-43527-1_2

[24] Junliang Li, Yajun Yang, Qinghua Hu, Xin Wang, and Hong Gao. 2023. Public Opinion Field Effect Fusion in Representation Learning for Trending Topics Diffusion. In *Thirty-seventh Conference on Neural Information Processing Systems*.

[25] Baichuan Liu, Deqing Yang, Yuchen Shi, and Yueyi Wang. 2022. Improving information cascade modeling by social topology and dual role user dependency. In *DASFAA 2022*. 425–440.

[26] Kaixin Liu, Sibo Wang, Yong Zhang, and Chunxiao Xing. 2023. An Efficient Algorithm for Distance-based Structural Graph Clustering. *Proc. ACM Manag. Data* (2023), 45:1–45:25.

[27] Yixin Liu, Kaize Ding, Jianling Wang, Vincent C. S. Lee, Huan Liu, and Shirui Pan. 2023. Learning Strong Graph Neural Networks with Weak Information. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*. ACM, 1559–1571.

[28] Hongliang Qiao, Shanshan Feng, Xutao Li, Huiwei Lin, Han Hu, Wei Wei, and Yunming Ye. 2023. RotDiff: A Hyperbolic Rotation Representation Model for Information Diffusion Prediction. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 2065–2074.

[29] Kazumi Saito, Ryohei Nakano, and Masahiro Kimura. 2008. Prediction of information diffusion probabilities for independent cascade model. In *Conference on knowledge-based and intelligent information and engineering systems*. 67–75.

[30] Aravind Sankar, Xinyang Zhang, Adit Krishnan, and Jiawei Han. 2020. Inf-VAE: A Variational Autoencoder Framework to Integrate Homophily and Influence in Diffusion Prediction. In *WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining, 2020*. ACM, 510–518.

[31] Wenbo Shang, Xuliang Zhu, and Xin Huang. 2024. Path-LLM: A Shortest-Path-based LLM Learning for Unified Graph Representation. *arXiv preprint arXiv:2408.05456* (2024).

[32] Longxu Sun, Xin Huang, Zheng Wu, and Jianliang Xu. 2024. Efficient Cross-layer Community Search in Large Multilayer Graphs. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 2959–2971.

[33] Ling Sun, Yuan Rao, Xiangbo Zhang, Yuqian Lan, and Shuanghe Yu. 2022. MS-HGAT: Memory-Enhanced Sequential Hypergraph Attention Network for Information Diffusion Prediction. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022*. AAAI Press, 4156–4164.

[34] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR 2018*.

[35] Ding Wang, Wei Zhou, and Songlin Hu. 2024. Information Diffusion Prediction with Graph Neural Ordinary Differential Equation Network. In *Proceedings of the 32nd ACM International Conference on Multimedia, MM 2024, Melbourne, VIC, Australia, 28 October 2024 - 1 November 2024*. ACM, 9699–9708.

[36] Jianwei Wang, Kai Wang, Xuemin Lin, Wenjie Zhang, and Ying Zhang. 2024. Neural Attributed Community Search at Billion Scale. *Proc. ACM Manag. Data* 1, 4, Article 251 (apr 2024), 25 pages.

[37] Jia Wang, Vincent W. Zheng, Zemin Liu, and Kevin Chen-Chuan Chang. 2017. Topological Recurrent Neural Network for Diffusion Prediction. In *2017 IEEE International Conference on Data Mining, ICDM 2017*. IEEE Computer Society, 475–484.

[38] Ruijie Wang, Zijie Huang, Shengzhong Liu, and etc. 2021. DyDiff-VAE: A Dynamic Variational Framework for Information Diffusion Prediction. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*. ACM, 163–172.

[39] Zhitao Wang, Chengyao Chen, and Wenjie Li. 2018. A Sequential Neural Information Diffusion Model with Structure Attention. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*. ACM, 1795–1798.

[40] Jiyoung Woo, Jaebong Son, and Hsinchun Chen. 2011. An SIR model for violent topic diffusion in social media. In *2011 IEEE International Conference on Intelligence and Security Informatics, ISI 2011, Beijing, China, 10-12 July, 2011*. IEEE, 15–19.

[41] Jiadong Xie, Fan Zhang, Kai Wang, Xuemin Lin, and Wenjie Zhang. 2023. Minimizing the Influence of Misinformation via Vertex Blocking. In *39th IEEE International Conference on Data Engineering, ICDE 2023*. IEEE, 789–801.

[42] Cheng Yang, Maosong Sun, Haoran Liu, Shiyi Han, Zhiyuan Liu, and Huanbo Luan. 2021. Neural Diffusion Model for Microscopic Cascade Study. *IEEE Trans. Knowl. Data Eng.* 33, 3 (2021), 1128–1139.

[43] Cheng Yang, Jian Tang, Maosong Sun, Ganqu Cui, and Zhiyuan Liu. 2019. Multi-scale Information Diffusion Prediction with Reinforced Recurrent Networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. 4033–4039.

[44] Chunyuan Yuan, Jiacheng Li, Wei Zhou, Yijun Lu, Xiaodan Zhang, and Songlin Hu. 2020. DyHGCN: A Dynamic Heterogeneous Graph Convolutional Network to Learn Users' Dynamic Preferences for Information Diffusion Prediction. In *ECML PKDD 2020*, Vol. 12459. Springer, 347–363.

[45] Yuan Yuan, Jingtao Ding, Chenyang Shao, Depeng Jin, and Yong Li. 2023. Spatio-temporal Diffusion Point Processes *(KDD '23)*. Association for Computing Machinery, New York, NY, USA, 3173–3184.

[46] Jing Zhang, Biao Liu, Jie Tang, Ting Chen, and Juanzi Li. 2013. Social Influence Locality for Modeling Retweeting Behaviors. In *IJCAI 2013*. 2761–2767.

[47] Erheng Zhong, Wei Fan, Junwei Wang, Lei Xiao, and Yong Li. 2012. ComSoc: adaptive transfer of user behaviors over composite social network. In *International Conference on Knowledge Discovery and Data Mining, KDD '12*. ACM, 696–704.

[48] Ting Zhong, Jienan Zhang, Zhangtao Cheng, Fan Zhou, and Xueqin Chen. 2024. Information Diffusion Prediction via Cascade-Retrieved In-Context Learning. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2472–2476.

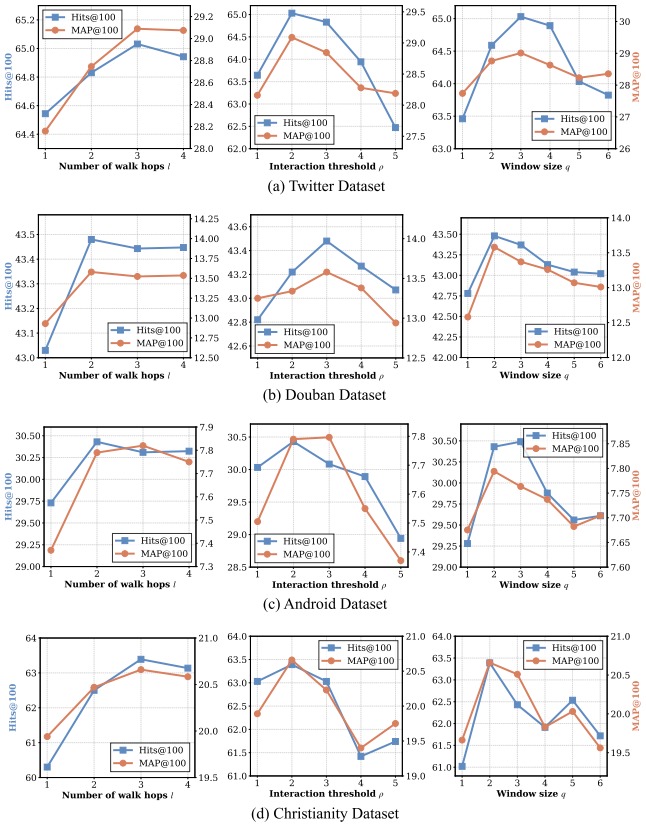

Figure 7: Hyperparameters $\{l, \rho, q\}$ on benchmark datasets.

Table 8: Key notations and concepts used in this paper.

Notation	Description
G_o	The original social network
G	A sampled multiplex relational social network
v_i	A user (node) in the network G
$\mathcal{N}_i$	The set of neighbors of user v_i in G
C_k	A cascade involving k participants, ordered by time
S	A sphere $S = \{(v_i, t_i)\} \subseteq C_k$ with similar influence
$\mathcal{S}$	A sphere-based cascade: a set of spheres, $\mathcal{S} = \{S_1, S_2, \ldots\}$
Γ	A collection of sphere-based cascades: $\Gamma = \{\mathcal{S}_1, \ldots, \mathcal{S}_k\}$
π_i	A relation paradigm used to partition C_k into $\mathcal{S}_i$

Table 9: The settings of key hyper-parameters.

Parameter	Value	Description
l	{2, 3, 4}	Walk-based sampling hop.
ρ	{2, 3}	Interaction threshold.
q	{2, 3}	Factor of time range in Eq.(11).
s	{30, 50, 100}	Total number of sampled neighbors.
d	{64, 128}	The dimension of embedding.
B	{4, 6, 8, 10}	Number of attention heads.
ϵ	1	Threshold of GHR@K.

A Appendix

A.1 Notations

Table 8 summarizes the key notations and concepts introduced in this work, with particular emphasis on the notion of a *sphere*. Given a cascade C_k, which is defined as a sequence of participants (v_i, t_i), each *sphere* $S \subseteq C_k$ represents a group of participants who exert similar influence on a target user v_k. Based on the influence dynamics observed for a specific user v_a, the cascade C_k can be partitioned into a set of disjoint spheres, forming a *sphere-based cascade* denoted as $\mathcal{S}_a = \{S_1, S_2, \ldots\}$. Analogously, for another user v_b, the same cascade may be decomposed into a different sphere-based cascade, denoted as $\mathcal{S}_b$. Consequently, for a given cascade C_k and a set of target users, we derive a collection of sphere-based cascades, expressed as $\Gamma = \{\mathcal{S}_1, \ldots, \mathcal{S}_k\}$, where each $\mathcal{S}_i$ corresponds to the sphere-based decomposition relative to user v_i. The sphere-based cascades enable a structured analysis of the heterogeneous influence patterns within cascades.

A.2 Hyper-parameter Details

Table 9 details the parameter configurations used in all experiments. To evaluate model robustness and inform practical deployment, we analyze the impact of three key hyperparameters across four benchmark datasets, as illustrated in Fig. 7. These hyperparameters exhibit consistent trends across datasets. Below, we summarize their functional roles and provide recommended tuning strategies. **Depth of walk-based sampler (l):** This parameter defines the maximum walk depth used in the walk-based sampling process on the social network. Increasing l enables the model to aggregate information from multi-hop neighbors, which generally improves representational capacity and downstream performance. However, excessively large values may lead to increased yet unnecessary computational overhead. Empirically, setting l in the range of 2 to 3 achieves a favorable trade-off between expressiveness and efficiency. **Interaction threshold (ρ):** The threshold ρ governs the retention of edges based on the frequency of historical interactions. A high threshold may prune weak but informative interactions (especially for users with sparse activities), leading to under-connected structures and suboptimal learning. Conversely, a low threshold introduces excessive noise. We recommend using a moderate threshold that retains a proportion of top-ranked interactions. Notably, the current strategy may cause imbalance—active users tend to retain many interaction edges, while inactive users retain few or none. A potential improvement is to retain the top-k% interactions per user (e.g., top 30%) to ensure equitable edge distribution and connectivity. **Stage time window (q):** The parameter q determines the temporal window used to identify current stage. If q is too small (e.g., $q = 1$), temporally relevant participants may be excluded, resulting in fragmented temporal contexts. In contrast, an excessively large size renders the stage meaningless and weakens performance. In addition, the selection of parameters s, d, and B follows a principled balance. ϵ serves as a constraint on the GHR metric. Specifically, a larger ϵ imposes stricter requirements on the model's capability, thereby leading to a lower observed GHR under fixed conditions. This paper initially mitigates the efficiency challenges, and more practical prediction on large-scale graphs remains an open challenge for future works.