A Deep Prediction Framework for Multi-Source Information via Heterogeneous GNN

Zhen Wu
School of Computer Science and Technology
Soochow University
Suzhou, Jiangsu, China
School of Computer Science and Engineering
Southeast University
Nanjing, Jiangsu, China
zwu1024@stu.suda.edu.cn

Jingya Zhou*
School of Computer Science and Technology
Engineering Lab. of Big Data and Intelligence of Jiangsu Province
Soochow University
Suzhou, Jiangsu, China
State Key Lab. for Novel Software Technology, Nanjing University
Nanjing, Jiangsu, China
jy_zhou@suda.edu.cn

Jinghui Zhang*
School of Computer Science and Engineering
Southeast University
Nanjing, Jiangsu, China
jhzhang@seu.edu.cn

Ling Liu
School of Computer Science
Georgia Institute of Technology
Atlanta, GA, USA
lingliu@cc.gatech.edu

Chizhou Huang
School of Computer Science and Technology
Soochow University
Suzhou, Jiangsu, China
czhuang1016@stu.suda.edu.cn

Abstract

Predicting information diffusion is a fundamental task in online social networks (OSNs). Recent studies mainly focus on the popularity prediction of specific content but ignore the correlation between multiple pieces of information. The topic is often used to correlate such information and can correspond to multi-source information. The popularity of a topic relies not only on information diffusion time but also on users' followership. Current solutions concentrate on hard time partition, lacking versatility. Meanwhile, the hop-based sampling adopted in state-of-the-art (SOTA) methods encounters redundant user followership. Moreover, many SOTA methods are not designed with good modularity and lack evaluation for each functional module and enlightening discussion. This paper presents a novel extensible framework, coined as HIF, for effective popularity prediction in OSNs with four original contributions. First, HIF adopts a soft partition of users and time intervals to better learn users' behavioral preferences over time. Second, HIF utilizes weighted sampling to optimize the construction of heterogeneous graphs and reduce redundancy. Furthermore, HIF supports multi-task collaborative optimization to improve its learning capability. Finally, as an extensible framework, HIF provides generic module slots to combine different submodules (e.g., RNNs, Transformer encoders). Experiments show that HIF significantly improves performance and interpretability compared to SOTAs.

CCS Concepts

• **Information systems** → **Collaborative and social computing systems and tools**; **Data mining**.

Keywords

Social networks, deep Learning, popularity prediction, multi-source information, heterogeneous graph

ACM Reference Format:
Zhen Wu, Jingya Zhou, Jinghui Zhang, Ling Liu, and Chizhou Huang. 2024. A Deep Prediction Framework for Multi-Source Information via Heterogeneous GNN. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3671966

1 Introduction

Users often post content regarding their life and work in online social networks (OSNs). After a user posts content, some other users see and repost the content through platform recommendation or their followership, arousing information diffusion. Information diffuses among users in a cascading manner, called information cascade, and its diffusion topological structure forms a directed acyclic graph (DAG), called cascade graph, as the observed cascade shown in Figure 1, where the original post is the source information and the user sending it is the source user. Users may also mark their post content with topic tags (i.e., hashtags), and the topic is a macro form of information. There might be more than one source under the same topic, i.e., the multi-source information. Posts without topic tags can be simply treated as single-source information.

*Corresponding author

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08
https://doi.org/10.1145/3637528.3671966

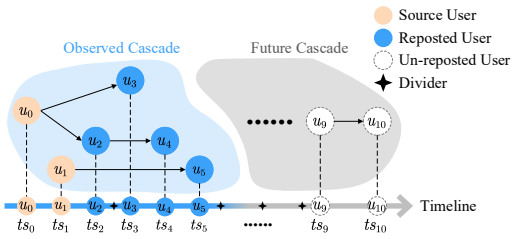


Figure 1: A multi-source cascade graph. u_0, u_1 / u_2-u_5 are source / reposted users, the others are un-reposted users and some of them (e.g., u_9, u_{10}) may repost in the future.

In this work, we take the number of users (e.g., users from u_0 to u_{10} in Figure 1) involved in information diffusion to quantify the popularity, which also measures how influential and widespread the information is. Popularity prediction has been widely studied as a fundamental task that can be applied to broad real-world scenarios, such as online advertisement [11], social recommendation [30], rumor detection [3], epidemiology [42], etc.

We categorize recent works into the following taxonomies: **Diffusion pattern-based models** such as the independent cascade (IC) model and the linear threshold (LT) model (e.g., [22, 31]), estimate the diffusion result with the help of Monte Carlo Simulation; **Feature-based approaches** feed hand-crafted spatial-temporal features (e.g., [8]), user features (e.g., [1]), and content features (e.g., [45]) into machine learning models to make predictions; **Generation-based approaches** suppose that the information diffusion can be characterized by a generative probabilistic model (e.g., [25]); **Deep learning-based approaches** are mainly based on RNNs (e.g., [30]) and homogeneous/heterogeneous GNNs (e.g, [3, 6]) to automatically learn the cascades' spatial-temporal features. Though there have been many related efforts in recent years, we have noticed that most of them are based on less-expressive data structures, such as sequence or homogeneous graph. This is especially true for works proposed in the early stage, and only a few recent works [36, 38] have begun to take advantage of the better expressive capacity of heterogeneous graphs. Currently, existing works still suffer from the following common challenges:

Hard time partition: An information cascade with length $|c|$ can generate $|TE|$ time interval(s) to learn temporal features, and there are three mainstream partition and processing granularities:

1) Fine-grained methods (e.g., [5, 7]) map each (re)posted user on the cascade to a unique time interval, i.e., $|c| = |TE|$, and then process them (e.g. feeding into RNN) separately, which is tricky on sequence's unequal length when using mini-batch training, and is easy to suffer from bottlenecks if the cascade sequence is too long;

2) coarse-grained methods (e.g., [6, 38]) treat all timestamps as one or even ignore temporal features, i.e., $|c| \leq 1$, which may be efficient but is hard to take full advantage of temporal features;

3) middle-grained methods (e.g., [36]) make a good trade-off on efficiency and learning ability, where the timeline is partitioned into several equal time intervals, i.e., $1 < |c| < |TE|$. For example, there are only 5 time intervals in Figure 1, so that temporal features can be captured without incurring more efficiency concerns.

The middle-grained partition simply maps users into different individual intervals, neglecting whether the partition is reasonable. For example, consider a scene where a piece of information is originally posted at 23:25 and the 1 hour observation window is partitioned into 6 time intervals (i.e., 23:25-23:34, 23:35-23:44, ..., 00:15-00:24), and a user is used to browse social networks for 15 minutes before sleep at 23:40, and this user reposts the information at 23:35. In the case of hard-time partition, the user is mapped to the second interval, but actually, the first interval is covered by the users' behavior preferences, and the second is only half overlapped, indicating that the first interval represents the user behavior preferences better than the second interval. It would be great if the different matching degrees of these two intervals could be considered at one time.

Redundant followership network: The followership is necessary for many social networks to construct a heterogeneous graph based on a cascade graph, existing heterogeneous graph construction methods (e.g., [36]) mainly utilize the hop-based sampling. For a sampled followership network with N_f total followers and N_p potential followers ($N_p \ll N_f$ in most cases), we denote its redundancy as $p_r = N_p/N_f$. In practice, p_r's value is usually very low, e.g., as illustrated in our case study, the p_r of hop-based sampling is 0.059, demonstrating that there are only about 1/20 of sampled followers are beneficial.

Lacking of expansive discussion and enlightenment: Various approaches have been proposed in recent works, and each of them has almost completely different strategies or components, which contributes to the prosperity of popularity prediction research. For example, HERI-GCN and CoupledGNN [6] employ dynamic features and hand-crafted features as initialization node features, respectively. Upon reading these works, researchers might be curious about the following questions: what motivated the authors to select such distinct initialization strategies? How will it perform if replaced with another strategy, and why? It is imperative to discuss the above questions as the discussion can provide valuable insights for future studies.

In this paper, we propose a novel **H**eterogeneous GNN-driven **I**nformation prediction **F**ramework (HIF) to solve the above challenges. The major contributions of our work are summarized below:

Soft time partition: We inventively propose a more elastic time interval partition method named Soft-Partition, which connects users and time intervals by multiple links with learnable weights, improving the flexibility, rationality, and accuracy of time partition.

Weighted dynamic sampling: We design weighted dynamic sampling, which is a more efficient followership sampling method than current hop-based sampling. Our sampling method samples followers step by step. At each step, we select a batch of the most meaningful followers via an elaborate weighted strategy until the number of sampled followers reaches the threshold.

Collaborative optimization: We present the Timeline Error, which constrains the variability and continuity of temporal embedding. We minimize the Timeline Error as an additional optimization task to collaboratively optimize temporal feature learning.

Extensive experiments and discussions: We evaluate HIF via extensive experiments on three real-world datasets and present comprehensive discussions. The results show excellent performance and provide enlightening significance.

2 Related Work

We summarize recent research from the following four categories:

Diffusion pattern-based models: The famous models in this category are the IC-based model [17, 20, 26, 28, 29, 31] and the LT-based model [17, 20, 22], and they both assume that a user will be influenced to involve in the information diffusion process when the predefined conditions are satisfied. Specifically, the IC-based models assume that each user has a probability to be influenced once at least one of his/her neighbors has been influenced, while the LT-based models suppose that a user will be influenced once the accumulated weights from his/her influenced neighbors reach the threshold. Both IC-based models and LT-based models heavily rely on predefined diffusion patterns (e.g., the conditions for users to be influenced), which require prior knowledge and are easy to get extremely complicated as pattern complexity increases. These models also have defects in time handling, such as hardly simulating users being influenced at different times and the time decay effect.

Feature-based approaches: The following hand-crafted features were once widely used by feature-based approaches: *content features* such as the TF-IDF and LDA [45]; *temporal features* such as the observation time and the time delta [8]; *spatial features* such as the page ranking [6, 18], edge density [2, 43]; *user features* such as the number of followers [1, 16], profiles [39]; These approaches heavily rely on the quality of features and lack generality across different domains, such as academic networks.

Generation-based approaches: These approaches assume that all data are "generated" by the underlying model, and the point process is one of these models and is often used to model real-world phenomena and user preferences. The most representative generation-based approaches rely on the point process. DeepHawkes [5] is a typical approach that employs the Hawkes point process to model self-activation and time decay. Literature [25, 33] model the influence diffusion via the Poison point process.

Deep learning-based approaches: Most approaches in this category are designed in an end-to-end way, and they automatically learn user embeddings or information embeddings. We further categorize them into three groups: *1) RNN-based approaches*: DeepCas [19] is an end-to-end model that firstly leverages DeepWalk [23] to sample paths from the cascade graph and feed them into a Bi-directional Gated Recurrent Unit (Bi-GRU) for prediction. Topo-LSTM [34] adapts LSTM [14] cell to the topological structure by pooling multiple inputs for each cell. TempCas [30] combines attention mechanism with Path Sampling, RNNs, and CNNs to learn spatial-temporal embeddings. TCAN [27] explicitly encodes the linear, the nonlinear, and the periodic features of time, and uses them as initial features of user nodes for cascade graph and cascade sequence feature learning. Nowadays, it is still strenuous for RNN-based approaches to handle graphs. *2) Homogeneous GNN-based approaches*: DeepInf [24] utilizes GNN and attention mechanism to predict the center user through a sampled ego network. CoupledGNN [6] combines two GNNs to learn the states of information and user alternatively, reinforcing the importance of structure in information diffusion. BiGCN [3] considers the bidirectional influence in OSNs and models it by means of a top-down GNN and a bottom-up GNN. CasCN [7] conducts convolution on each cascade snapshot and then learns temporal features by feeding the convoluted spatial features into an RNN. It is the first GNN-based model incorporated with RNN to learn the temporal feature. CasFlow [37] not only utilizes GNNs and RNNs to learn spatial-temporal embeddings but also takes advantage of variational auto-encoders (VAE) to model the uncertainty of information diffusion. *3) Heterogeneous GNN-based approaches*: DyHGCN [38], and HERI-GCN [36] firstly apply heterogeneous GNNs to learn cascade embeddings to predict the next (re)post user and the popularity, respectively. Specifically, DyHGCN simply treats the heterogeneous graph learning as multiple homogeneous graph learning and aggregates the learned embeddings via a heuristic method; HERI-GCN designs a heterogeneous convolution kernel and integrates RNNs into GNNs to learn temporal embeddings.

In summary, the temporal and spatial features are both critical for popularity prediction, but only a few works, including HERI-GCN, CasFlow, and CasCN, have conducted preliminary exploration. In addition, compared to single-source information, multi-source information diffusion is a more general way of information diffusion, especially in social networks, which is rarely explored by current works except HERI-GCN.

3 Preliminaries

In this section, we present the definitions of fundamental concepts and formally define the popularity prediction problem.

We use N_c^s to denote the number of source users in information cascade c. For multi-source cascades, we have $N_c^s > 1$, e.g., there are two source users (u_0 and u_1) in Figure 1. A single-source cascade is a special case of a multi-source cascade such that $N_c^s = 1$.

Multi-source cascade is a general form of information diffusion. As illustrated in Figure 1, the example cascade is a 2-source cascade with two source users (u_0 and u_1). A single-source cascade can be considered a special case of a multi-source cascade. For each posted information, we record the (re)post actions in a cascading format, i.e., a sequence of triples.

Definition 1 (Information Cascade). Given an information c, the cascade sequentially describes the information's diffusion process. We use c directly to represent the message and its cascade, formulated by $c = [(u_0, v_0, ts_0), (u_1, v_1, ts_1), \dots]$, where triple (u_i, v_i, ts_i) indicates user v_i reposts from user u_i at timestamp ts_i.

We connect all users involved in a cascade c in terms of the reposting relationship and obtain the cascade graph $G^{r,c} = (V_{user}^c, E^{r,c})$, where V_{user}^c is the node set and $E^{r,c}$ is the edge set. Superscript r indicates the cascade is formed by repost actions. In the cascade shown in Figure 1, we have u_0 to u_5 in V_{user}^c and four edges in $E^{r,c}$.

Besides the reposting relationship, there are multiple relationships between users. For example, followership is widely adopted in many OSNs (e.g., Twitter, Sina Weibo), The followership network is denoted by a directed graph $G^f = (V_{user}, E^f)$, where V_{user} contains all users, and E^f is the set of all followership edges, and we have $V_{user}^c \subset V_{user}$. Followership network provides a global graph that connects all users. This work is based on the heterogeneous graph, and we adopt the conception of meta-edge and heterogeneous graph in [36]:

Definition 2 (Meta-Edge). The *meta-edge* is a simplification of a meta-path whose length is exactly 1, and it is denoted by a triple containing both node types and edge type, i.e., *(head node type, edge type, tail node type)*, abbreviated as (T_u, T_e, T_v).

Each meta-edge illustrates the topological relationship of the same species by giving the types of both head/tail nodes and edge. It is the abstraction of a kind of edge in the heterogeneous graph.

Definition 3 (Heterogeneous Graph). A heterogeneous graph is denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the collection of different types of node sets and $\mathcal{E}$ is the edge set. For each meta-edge (T_u, T_e, T_v) of $\mathcal{G}$, we have $T_u, T_v \in \mathcal{V}$ and $T_e \in \mathcal{E}$. For example, for a graph that contains *repost* edges and *follow* edges between users, it has $\mathcal{V} = \{V_{user}\}$ and $\mathcal{E} = \{E_{repost}, E_{follow}\}$, and its meta-edges are $(user, repost, user)$ and $(user, follow, user)$.

Finally, we define the popularity prediction problem as follows:

Definition 4 (Popularity Prediction). For a certain observation time window $\mathcal{T}$ (e.g., 1 hour), our target is to anticipate the final popularity $|c|$ (the size of the cascade) through the cascade observed during $\mathcal{T}$, $c^{\mathcal{T}} = \{(u, v, ts) | \forall ts \leq \mathcal{T}\}$. For example, in Figure 1, we observe the cascade in the light blue area, which contains 5 users, and the final popularity is 11, i.e., the total quantity of involved users from u_0 to u_{10}.

Our goal is to learn a mapping function Θ that maps the input to a predicted value $\hat{y}$ such that $\hat{y}$ is as close as possible to the ground-truth popularity y, i.e., minimize the metric $\mathcal{M}$:

$$\arg\min_{\Theta} \mathcal{M}(\hat{y}, y) = \arg\min_{\Theta} \mathcal{M}\left(\Theta\left(c^{\mathcal{T}}, G^{f}\right), |c|\right) \tag{1}$$

4 Proposed Framework

HIF consists of three phases: heterogeneous graph construction, heterogeneous GNN-driven inference, and optimization. The overview of HIF is shown in Figure 2.

4.1 Heterogeneous Graph Construction

In this paper, we denote each type of heterogeneous edge in an adjacent matrix form.

4.1.1 Meta-Edges Between Users. The meta-edge *(user, repost, user)* means the users and edges in $G^{r,c}$, i.e., $E_{repost}(u, v) = 1$ if $(u, v) \in c$, otherwise, $E_{repost}(u, v) = 0$.

The followership may contain the potential paths for information diffusion [36]. We use meta-edge *(user, follow, user)* to represent these relationships. For each cascade c, instead of using the global followership network G^f, which is liable to lead to massive computation, we sample a subgraph whose edge set is $E^{f,c}$ selected from G^f. The meta-edge *(user, follow, user)* is formulated as: $E_{follow}(u, v) = 1$ if $(u, v) \in E^{f,c}$, otherwise, $E_{follow}(u, v) = 0$.

The sampling method used in [36] is hop-based, i.e., sampling H-hop followerships for all users in a cascade where H is a hyperparameter. However, this sampling method suffers from the efficiency problem as mentioned in Sec. 1, and more details are discussed in Sec. 5.

In this paper, we propose a new method to efficiently sample more potential followers, i.e., Dynamic Weighted Sampling. Note that nodes in different positions in $G^{r,c}$ may have different contributions to information diffusion:

- The source nodes have only outgoing edges (no incoming edges) and act as the most influential nodes that contribute most to the diffusion of information in the future;
- The leaf nodes have only incoming edges and are no longer reposted in observation, so they might have tremendous potential to contribute to future information diffusion;
- The intermediate nodes repost some nodes and have also been reposted by others in observation, i.e., they have both incoming and outgoing edges. These intermediate nodes in common sense are observed to be reposted for diffusion by consuming a portion of their influence so that they might have relatively low contributions compared to nodes in other positions.

As sampling initialization, we assign different weights to sub-cascades, and users on the above positions, then begin multistep sampling. In each step, for all unsampled followers, we first accumulate the weights from their followed users and correlated sub-cascades, and then we select a batch of followers with the highest weight. We repeat the step until all followers are sampled, or the number of sampled followers reaches the given threshold. The pseudo-code and detailed descriptions are presented in A.2.

4.1.2 Meta-Edges With Time. We partition the time series in cascade c into T time intervals equally and treat them as a group of time nodes $V^c_{time} = \{t_0, \ldots, t_{T-1}\}$. Then, we connect the adjacent time nodes (corresponding to the adjacent intervals) to get meta-edge *(time, past to, time)*: $E_{pastto}(t_i, t_j) = 1$ if $i = j - 1$, else 0.

Compared with a single connection for each user towards a time node (corresponds to the time interval that the user (re)posts in) [36], the first step of the innovative soft partition is additionally considering at most $2S$ links to adjacent time nodes (S time nodes earlier than it and S time nodes later than it), which can be regarded as a subsequent operation after hard partition (the second step is the attentive GCN mentioned in Sec. 4.2.2). These additional links connect each user with multiple candidate time intervals and make the time interval partition learnable by adding learnable weights to these links. The meta-edge *(user, post at, time)* is formalized by: $E_{postat}(u, t_j) = 1$ if $0 \leq j \leq T$ and $|j - k| \leq S$, else 0. Afterwards, the meta-edge *(time, contain, user)* can be obtained by $E_{contain} = E^{\top}_{postat}$, where $\top$ indicates the matrix transpose.

In this way, we construct a heterogeneous graph $\mathcal{G}^c$ for cascade c that contains 2 types of nodes (i.e., user and time) and 5 types of edges (i.e., *repost*, *follow*, *past to*, *post at* and *contains*), as shown in Figure 3.

4.2 Heterogeneous GNN Driven Inference

As shown in Figure 2, the heterogeneous GNN-driven inference phase consists of multiple procedures: Firstly, we obtain initial embeddings of the input; Then, we feed these embeddings into the stacked GNNs and time embedding layers; Finally, we read out the predicted value from the learned embeddings. Here, we introduce the forward inference under mini-batch training.

4.2.1 Embedding Initialization. The first phase is to acquire initial embeddings for items in $\mathcal{G}^c$, and this phase invokes at the beginning of each forward inference (i.e., after feeding a batch of data to

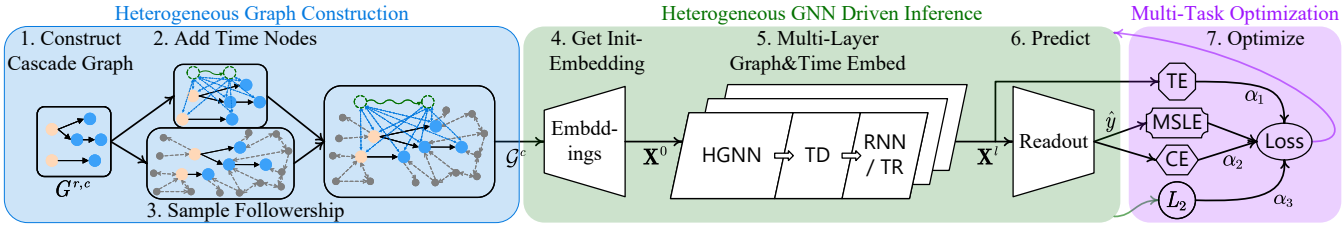


Figure 2: The overview of HIF that includes three phases (total 7 procedures): the heterogeneous graph construction, the heterogeneous GNN-driven inference, and the multi-task optimization. In procedures 2 and 3, the green and gray nodes are time nodes and sampled followers, respectively.

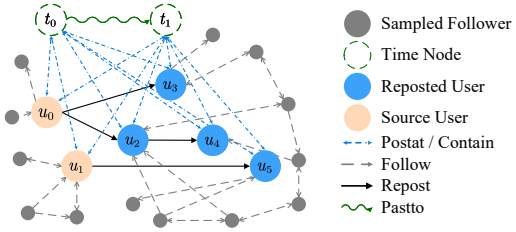


Figure 3: The constructed heterogeneous graph $\mathcal{G}^c$.

the model in the training, validation, or test stage). The acquired initial embeddings can be formulated as $\mathbf{X}_{item} = \mathbf{IE}(\mathcal{G}^c)$ (e.g., $\mathbf{X}_{user}$ for user nodes or $\mathbf{X}_{time}$ for time nodes), where $\mathbf{IE}(\cdot)$ refers to initializing item embeddings. We design two different strategies to implement $\mathbf{IE}(\cdot)$. Firstly, for any type of item (user node or time node), we assume that there are n items in a batch and N items in the entire dataset. Then, the initial d-dimensional embeddings are generated optionally by:

- **Learnable embedding strategy** (Ln) randomly generates a global embedding matrix $\mathbf{E} \in \mathbb{R}^{N\times d}$ and saves it. It will return the corresponding embedding rows $\mathbf{X} \in \mathbb{R}^{n\times d}$ from $\mathbf{E}$ for each item in the batch when invoked. In this strategy, the embedding matrix $\mathbf{E}$ is learnable.
- **Dynamic embedding strategy** (Dy) generates a random matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$ as the initial embedding when invoked, which allows an item to get a different embedding at each invoking.

Based on the above strategies, we acquire the node embedding set $\mathbf{X}_{node} = \{\mathbf{X}_{user}, \mathbf{X}_{time}\}$, and compare the performance for different node embedding initialization strategies in Sec. 5.4. We utilize the dynamic embedding strategy for edges for the purpose of memory saving and uncertainty modeling, then we have $\mathbf{X}_{edge} = \{\mathbf{X}_{repost}, \mathbf{X}_{follow}, \mathbf{X}_{pastto}, \mathbf{X}_{postat}, \mathbf{X}_{contain}\}$.

4.2.2 Heterogeneous Graph Convolution. The forward inference of l-th GCN layer can be formulated in a message-passing manner for each *(head node, edge, tail node)* triple (u, e, v). We use $\mathbf{x}$ and $\mathbf{X}$ to denote the embeddings of a single node/edge and all nodes/edges (distinguished by upper and lower case), respectively, and we use $(\cdot)$ to represent the embedding triples of (u, e, v) (e.g., $(\mathbf{x}_u^{l-1}, \mathbf{x}_e^{l-1}, \mathbf{x}_v^{l-1})$ or $(\mathbf{X}_{T_u}^{l-1}, \mathbf{X}_{T_e}^{l-1}, \mathbf{X}_{T_v}^{l-1})$) in corresponding GCN layer for simplicity.

Firstly, we generate a message based on (u, e, v), i.e., $\mathbf{MSG}(\cdot)$, and then pass it by edge e; Subsequently, the tail node v aggregates all messages from its in-coming neighbor nodes, and this process can be formatted as $\mathbf{AGG}(\{\mathbf{MSG}(\cdot)|\forall(u,e) \in \mathcal{N}_v\})$, where $\mathcal{N}_v$ is the set of all in-coming node-edge pairs for v; Afterwards, each tail node v's embedding will be updated by $\mathbf{X}_v^l = \mathbf{UPD}_N(\cdot)$; Finally, we also update edges' embeddings, i.e., $\mathbf{X}_e^l = \mathbf{UPD}_E(\cdot)$.

Specifically, the message is generated by:

$$\begin{aligned} \mathbf{MSG}(\cdot) &= \alpha_{T_e} \mathbf{X}_{T_u}^{l-1}, \\ \alpha_{T_e} &= \left(W_{T_u}^{M,l} \mathbf{X}_{T_u}^{M,l-1} + W_{T_v}^{M,l} \mathbf{X}_{T_v}^{l-1}\right) W_{T_e}^{l} \mathbf{X}_{T_e}^{l-1}, \end{aligned} \tag{2}$$

where α_{T_e} is the message weight that combines the correlation of u and v, and scales by e, and $W_{T_u}^{M,l}$, $W_{T_e}^{M,l}$ and $W_{T_v}^{M,l}$ are parameters.

Especially for meta-edges containing both time node and user node, we apply a softmax for α_{T_e} to normalize the message weights to help GCN learn the soft partition of time intervals.

Due to the graph's heterogeneity, we firstly collect messages on meta-edges for each node, then utilize sum pooling to aggregate messages passed through meta-edge (T_u, T_e, T_v), i.e.,

$$\begin{aligned} \mathbf{AGG}_{T_e}(\cdot) &= \mathbf{SUM}\left(M_v^{T_e}\right), \\ M_v^{T_e} &= \left\{\mathbf{MSG}(\cdot)|\forall(u,e) \in \mathcal{N}_v^{T_e}\right\}, \end{aligned} \tag{3}$$

where $\mathbf{AGG}_{T_e}$ is the aggregation on T_e, $M_v^{\mathcal{T}_e}$ refers to the set of all messages, and $\mathcal{N}_v^{T_e}$ refers to the in-coming node-edge pairs for v in meta-edge (T_u, T_e, T_v). After aggregation, we use the attention mechanism to update node embeddings:

$$\begin{aligned} \mathbf{UPD}_N(\cdot) &= \sum_{T_{\mathcal{N}_v}}^{T_e} \beta_{T_e} \mathbf{AGG}_{T_e}(\cdot), \\ \beta_{T_e} &= \frac{\exp\left(a_{T_e}\right)}{\sum_{T_{\mathcal{N}_v}}^{T_e} \exp\left(a_{T_e}\right)}, \; a_{T_e} = W_{T_e}^{N_1} \tanh\left(W_{T_e}^{N_2} \mathbf{AGG}_{T_e}(\cdot)\right), \end{aligned} \tag{4}$$

where $W_{T_e}^{N_1}$ and $W_{T_e}^{N_2}$ are learnable parameters, a_{T_e} is the attention score and β_{T_e} is the attended weight after softmax. For edges' update, we simply concatenate the learned embeddings of (u, e, v) in l-th layer: $\mathbf{UPD}_E = W_{T_e}^{E,l}(\mathbf{X}_u^l || \mathbf{X}_e^l || \mathbf{X}_v^l)$, where $W_{T_e}^{E,l} \in \mathbb{R}^{3d,d}$ is a learnable parameter, and $||$ refers to concatenation.

4.2.3 Integrated Time Embedding. We add slots between GCN layers to integrate the time embedding module for temporal feature learning via time nodes. We have the following time embedding modules to integrate:

Time Decay (TD) models the decay effect of source information's influence with the increase of time. The formulation of parameterized time decay is given by $\mathbf{TD}(\mathbf{X}_{time}^l) = W_{TD}^l \mathbf{X}_{time}^l$, where W_{TD}^l is a learnable time decay factor.

RNN and **Transformer** (TR) [32] are devoted to sequential data learning (e.g., word sequences) and can also be applied to time nodes in the heterogeneous graph to enhance temporal feature learning. Bi-GRU [9] is used as the instance of RNN in HIF. For Transformer, HIF mainly stacks its encoder layers [32] as the transformer instance. Each Transformer encoder layer learns embeddings via the multi-head attention and then adds the attended embeddings with positional encoding to make up for the shortcoming that the attention mechanism in Transformer ignores the order in input.

4.2.4 Readout. Motivated by [46], after L layers of GNN and time embedding module, we combine the classification task with the prediction task to handle the long-tail cascade distribution. Specifically, for a cascade c, we classify its popularity into maximum Lv levels by the power of 2, and the level of c is $Lv^c = \min(Lv, \lfloor \log_2 |c| \rfloor)$. Then, we utilize the learned embeddings to train a classifier Cls to predict the level of c, i.e., $p = \text{softmax}(\text{Cls}(\cdot))$ and $\hat{Lv} = \arg\max_i(p_i)$, where p is the logits output by Cls, and $\hat{Lv}$ is the predicted level. We also train Lv regressors that output the predicted popularity at each level and combine them by the sum pooling weighted by p: $\hat{y} = \sum_i^{Lv} p_i \text{Reg}_i(\cdot)$, where Reg_i is the regressor for level i with a reduced value space for easier prediction. In particular, the Cls and all Reg_i are constructed by MLPs with the same number of layers.

In this way, our readout module combines and solves a multi-class classification task and a multi-regression task, which discretizes the long-tailed distribution into several intervals (i.e., the popularity levels) and then predicts the popularity by multiple regressors in the corresponding intervals. As a result, the model is capable of learning a more extensive output value range and thus better handles the problem of data imbalance under the long-tailed distribution, which is hard to fit by traditional regressors.

4.3 Multi-Task Optimization

We use the mean squared logarithmic error (MSLE) as the metric to evaluate performance for the same reasons stated in [36] (B is the batch size): $MSLE = \frac{1}{B}\sum_i^B (\log(y+1) - \log(\hat{y}+1))^2$.

To better optimize the readout layer that consists of both regression and classification tasks, we take the cross entropy as the loss of popularity level classification task for each sample in batch: $CE = -\sum_i^{Lv} Lv_i \log p_i + (1 - Lv_i)\log(1 - p_i)$, where Lv_i can be regarded as the i-th value of the one-hot encoded label of the actual level, e.g., we set Lv to 3 (starts from 0) and Lv^c to 1, then we have the one-hot encoded label $[0, 1, 0]$, and in this case, the Lv_0, Lv_1, Lv_2 are 0, 1 and 0, respectively.

Besides, we also present a novel **Timeline Error** (TE) to constrain the learned time embeddings for each sample in batch. For TE, we mainly use cosine similarity (CS) to measure the difference between two specific time nodes. For T time nodes, the goal of TE is to maximize the difference between the first and the last time nodes and minimize the difference between adjacent time nodes. For optimization purposes, we rewrite it as a minimization problem:

$$TE = \sum_{i=0}^{T-2} \text{CS}(\mathbf{X}^L_{time}[i], \mathbf{X}^L_{time}[i{+}1]) - \text{CS}(\mathbf{X}^L_{time}[0], \mathbf{X}^L_{time}[T{-}1]). \tag{5}$$

Finally, we combine metrics and errors from multiple tasks (e.g., the TE, CE, and L2 regularization) in a weighted manner as our loss to optimize our framework:

$$Loss = MSLE + \sum_i^B (\alpha_1 TE_i + \alpha_2 CE_i) + \alpha_3 L_2, \tag{6}$$

where α_1, α_2, and α_3 are hyperparameters that weight the metrics of other tasks, and L_2 is the widely used L2 parameter regularization to avoid overfitting.

5 Experiments

5.1 Datasets

We conduct experiments on three real-world datasets: SSC, MSC, and Twitter, and the code is open at https://github.com/Les1ie/HIF. Dataset Twitter [10, 15, 21, 40] collects tweets, and followership from some famous users on Twitter, and it contains single-source cascades. Datasets SSC and MSC [36, 44] are collected from Sina Weibo, where SSC is a single-source dataset that ignores the topic, while MSC is a multi-source dataset. Table 1 shows the statistics of datasets. We can observe: 1) Both MSC and SSC contain more users than Twitter; 2) The followership network and cascades in MSC and SSC exhibit a broader scale compared to those observed in Twitter, making the task on datasets SSC and MSC more meaningful and challenging. Furthermore, we analyze the cascade size of each dataset in A.3. For the experiment, we select 0.5 h (hour) and 1 h (hour) as the observation time, respectively.

Table 1: Statistics of datasets. (minimum | mean | maximum)

Properties	MSC	SSC	Twitter
# Users	2,977,573	887,608	8,510
# Followers	9,071,666	3,693,057	80,070
# Cascades	19,688	10,420	365,576
# Sources per Cascade	2 | 8 | 316	1 | 1 | 1	1 | 1 | 1
# Cascade Size	10 | 275 | 9,276	50 | 249 | 63,599	20 | 37 | 193
# Cascades per User	1 | 3 | 1,847	1 | 1 | 462	1 | 2 | 412
# Followers per User	1 | 5 | 15,654	1 | 4 | 5,719	1 | 5 | 94

5.2 Baselines

We select eight state-of-the-art methods as baselines:

FeatureBased simply feeds hand-crafted temporal and spatial features into a multilayer perceptron (MLP) to make a prediction. We extract effective features including: *Temporal features* such as the average time delta of the first half and the last half of the observed cascade [8]; *Spatial features* such as the number of leaf nodes [13], triangles, nodes and edges [35] in the cascade graph, the edge density [43] and clustering coefficient [35].

Node2Vec [12] is a typical node embedding approach, both walking probabilities p and q are set to 1.0. We feed the learned node embeddings to MLP to make predictions.

DeepCas [19] and **DeepHawkes** [5] are set to sample 100 paths with length 10. The number of time intervals is 6, and probabilities p and q keep the same as Node2Vec.

TopoLSTM [34] innovatively extends the standard LSTM cell to a multiple inputs case through mean pooling. We adapt it to the popularity prediction task by replacing the output layer with MLP.

CoupledGNN [6] models the information state and user state by two GNNs that are coupled with each other. The number of GNN layers is 3, and the original loss function is replaced by MSLE.

CasCN [7] presents a dynamic GCN kernel combined with RNN to handle snapshot graphs. The order of the Chebyshev kernel is 2, and the number of time intervals is 6.

HERI-GCN [36] integrates RNN into GNN and learns spatial-temporal features from a heterogeneous graph that contains time nodes. We set the number of heterogeneous GCN layers to 3, and set the number of time nodes to 5.

TCAN [27] is built with a CGAT module with 3 layers and the CSAT block with 2 layers of Bi-LSTM.

5.3 Performance Comparison

The comparison results between our proposed HIF and baselines are shown in Table 2. We obtain the following performance-related observations (POs):

Table 2: Prediction performance (MSLE) of HIF and baselines.

Method	SSC		MSC		Twitter	
	0.5 h	1 h	0.5 h	1 h	0.5 h	1 h
FeatureBased	0.737	0.686	1.147	0.942	0.973	0.848
Node2Vec	1.469	1.126	1.952	1.453	0.172	0.164
DeepCas	3.493	3.763	3.768	3.901	2.876	3.014
DeepHawkes	1.374	0.496	3.117	3.013	0.226	0.076
TopoLSTM	0.859	0.727	3.211	2.398	0.201	0.194
CoupledGNN	0.878	0.746	1.726	1.547	0.070	0.065
CasCN	1.213	1.148	3.084	2.555	0.138	0.100
HERI-GCN	0.464	0.398	1.291	0.881	0.096	0.085
TCAN	0.891	0.852	2.141	2.051	0.126	0.101
HIF	**0.419**	**0.317**	**0.555**	**0.537**	**0.033**	**0.023**

(PO 1): HIF outperforms all baselines, and meanwhile, HERI-GCN also performs better than other baselines in most cases because they not only model both temporal and spatial features but also utilize followership to discover the potential diffusion paths.

(PO 2): DeepCas performs worst in most cases and even performs worse under 1 hour of observation compared to 0.5 hours of observation. The reason is twofold: 1) From the perspective of the model: the performance of DeepCas depends on the quality of the sampled paths, but the structural features contained in the sampled paths are weaker than that in the entire cascade graph (e.g., CoupledGNN). Furthermore, the sampled paths with backtracking contain chaos temporal information compared to using the complete time series directly (e.g., DeepHawkes); 2) From the perspective of data: the observation times are set to 0.5 hours and 1 hour, and are much shorter than the observation time settings in DeepCas's experiments [19] (1, 3, and 5 days for Twitter). It is challenging to extract helpful information from the paths sampled in short-term observations to represent the trend of information propagation.

(PO 3): Interestingly, the performance of all methods on the MSC dataset is somewhat attenuated compared to the SSC dataset. As evaluated in [36], the message propagation prediction in multi-source cascade considers more potential factors (e.g., potential followers who have followed users in several sub-cascades at the same time) and obtains better results compared to prediction on the SSC dataset. We conclude that different observation time settings cause a difference in results. Concretely, the settings of short-term observation time (e.g., 0.5 h and 1 h) often indicate fewer data available, and it is hard for models to discover these potential factors from insufficient information. As a result, making predictions for multi-source cascades with short-term observation time is challenging.

5.4 Ablation Study

In the ablation study, we compare the variants of HIF to explore the contribution of different modules:

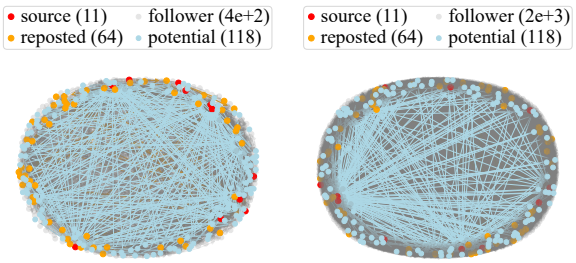
- For initial embedding strategies (IE), we have two options, i.e., the *Dynamic strategy* (Dy) and the *Learnable strategy* (Ln) as mentioned in Sec. 4.
- For followership sampling (FS), we also have two options, i.e., the *Hop-based sampling* (Hp) [36] and the proposed *Weighted dynamic sampling* (WD) as mentioned in Sec. 4.
- For time embedding (TE), we have two options of sequential-input models to further combine with *Time Decay*, i.e., the *Bi-GRU* (RNN) or the *encoder of Transformer* (TR).

From the results reported in Table 3, we have the following ablation observations (AOs):

Table 3: Performance of HIF's variants. IE: initial embedding, FS: followership sampling, TE: time embedding, and '-' means no corresponding module is used.

Modules			SSC		MSC		Twitter	
IE	FS	TE	0.5 h	1 h	0.5 h	1 h	0.5 h	1 h
Ln	Hp	-	1.056	0.742	1.031	0.795	0.056	0.048
Ln	Hp	RNN	0.785	0.613	0.982	0.753	0.047	0.040
Ln	Hp	TR	0.684	0.645	0.923	0.680	0.046	0.034
Ln	WD	-	0.617	0.476	0.887	0.813	0.086	0.069
Ln	WD	RNN	0.606	0.440	0.760	0.636	0.062	0.028
Ln	WD	TR	0.588	0.464	**0.741**	**0.621**	0.045	0.024
Dy	Hp	-	0.870	0.530	1.538	1.192	0.094	0.060
Dy	Hp	RNN	0.648	0.356	0.996	1.086	0.061	0.028
Dy	Hp	TR	0.644	0.519	0.885	0.957	0.087	0.058
Dy	WD	-	0.601	0.467	0.974	0.803	0.052	0.053
Dy	WD	RNN	**0.419**	**0.317**	0.879	0.756	0.048	**0.023**
Dy	WD	TR	0.442	0.452	0.809	0.632	**0.033**	0.032

(AO 1): For initial embedding strategies, dynamic initialization achieves better results than the learnable initialization in some cases (e.g., the results on SSC). We speculate the reason lies in that the learnable initialization suffers from the unbalanced user distribution, i.e., some users are in the validation set or test set only, and their embeddings are not optimized during the training. Specifically, as Table 1 shows, the mean number of cascades involved per user is 1, 3, and 2 for SSC, MSC, and Twitter, respectively, indicating that SSC is more likely to suffer from the unbalanced user distribution than the other two datasets. Besides, the performance is not determined by the IE module only, even though we fix the other modules. For example, for a hub-like cascade graph, the source user is probably the only user that dominates the popularity, and then the embeddings of reposted users are not crucial for prediction,



(a) Weighted Dynamic Sampling (b) 1-Hop Sampling

Figure 4: Heterogeneous user graphs obtained by conducting two sampling methods on MSC. The legend shows the number of users in each type.

ultimately leading to a limited impact of IE strategy on performance. This also explains why the correlation between performance and IE on MSC and Twitter is less pronounced than that on SSC.

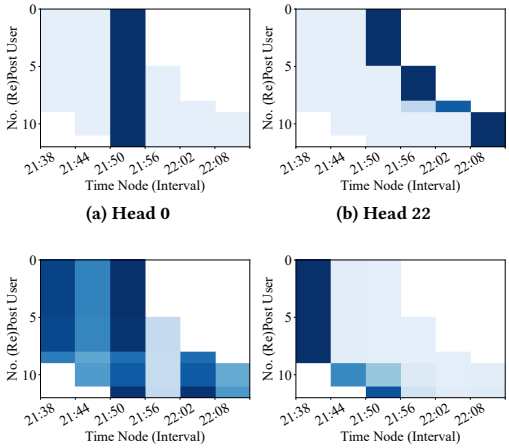
(AO 2): For the time embedding module, both RNN and Transformer improve the model performance for most cases. In other words, it is still tricky to learn the temporal features comprehensively by using only heterogeneous GNN even though the constructed heterogeneous graph already contains temporal information, so we have to integrate the models such as RNN or Transformer to strengthen the learning effect. Considering the short-term observation time (0.5 h and 1 h), the number of time intervals that can be divided is limited; otherwise, too many time intervals will lead to sparse connections between time nodes and user nodes. As a result, given the short input sequence of the temporal embedding module, it is difficult to directly assert which is a better choice between Transformer and RNN for all datasets and other modules. Nevertheless, for a total of 24 cases with the same IE, FS, observation, dataset settings, and different TE modules (except for the cases without TE module), there are 16 cases in which HIF with Transformer achieves better performance than RNN, so that Transformer is more likely to deliver higher performance improvements than RNN.

(AO 3): For followership sampling, the proposed weighted dynamic sampling is more likely to achieve a better performance in most cases, because the sampled followership network keeps meaningful nodes and edges and filters out more redundant nodes and edges that may disturb the learning, see Sec. 5.5 for more details.

5.5 Case Study

5.5.1 Case Study for Weighted Dynamic Sampling. We compare the sampled followership network by our proposed weighted dynamic sampling and 1-hop sampling and draw a specific case in Figure 4. Figure 4 contains source (red), follower (gray), reposted (orange), and potential follower (light blue) nodes, and also corresponding edges (gray for followerships, light blue for followerships by potential users and orange for reposting). These potential followers follow users in different sub-cascades, so they are simultaneously influenced by multiple sources, indicating that they are easily influenced to be involved in information diffusion in the future [36].

Comparing subfigures in Figure 4, subfigure 4a has fewer followers but the same number of potential nodes This is because we accrue weights from different sub-cascades for each follower in addition to considering weights from users in different positions



(a) Head 0 (b) Head 22

(c) Head 26 (d) Head 31

Figure 5: The learned weights of soft-partition edges. X-axis is the timestamp of time node and Y-axis is user ID.

in the cascade graph when sampling. Therefore, it is easier to discover followers who have followed multiple users in the cascade graph at the same time, especially when these followed users are in different sub-cascades (i.e., these followers are potential followers). This result demonstrates that the weighted dynamic sampling can reduce redundancy without losing valuable data.

5.5.2 Case Study for Soft-partition. In Figure 5, we plot the learned weights of soft-partition edges on the latest GCN layer on SSC dataset with 0.5-hour observation. Each sub-figure shows a heatmap of a head's weights in the multi-head attention for *(user, postat, time)* meta-edge. Each pixel represents the connection state of the user on the Y-axis and the time node on the X-axis. A blank pixel means no connection, and for a colored pixel, the darker the color, the higher the attention weight. To better distinguish colored pixels from blank pixels, we slightly darken the color of the pixels whose weights are actually very close to zero.

In this case, the number of time nodes is 6, i.e., t_0-t_5. In hard-partition, users u_0-u_4 (re)post at 18:29-18:33 (the 1st time interval), and connect to time node t_0; users u_5-u_7 (re)post at the 2nd interval and connect to t_1; u_8-u_{11} (re)post at 3rd, 4th, 4th and 6th intervals, and connect to t_2, t_3, t_3 and t_5, respectively. Here, the soft-partition is set to 2, and thus each user in the cascade connects to at most 5 time nodes and at least 3 time nodes. As illustrated in Figure 5, for each user in the cascade, head 0 pays more attention to the time node(s) connecting most users; head 26 focuses on almost all time nodes except one that has low weights; head 22 pays more attention to the latest connected time nodes, while head 31 pays more attention to the earliest time node. We interpret that the soft-partition provides a broader user-wise interval to aggregate features on the timeline. Specifically, we interpret the results on head 0 as learning the mean (re)posting time interval of all observed users that corresponds to the temporal feature of information. Head 26 focuses on filtering out unimportant time node(s) (e.g., t_3 in this case) in a global view. Furthermore, the results on heads 22 and 31 can be interpreted as learning the two ends of time intervals that correspond to the observed users' action preferences, reflecting

users' temporal features. The observation indicates that the soft-partition enables HIF to learn more useful temporal features.

6 Conclusions

In this work, we proposed a novel framework HIF, to predict the popularity of multi-source cascade in OSNs. HIF utilizes soft-partition to learn users' behavioral preferences, and samples followers through the efficient weighted dynamic sampling. Combining with multi-task optimization further enhances HIF's learning ability on both long-tail distribution and spatial-temporal features. Through extensive experiments on real-world datasets, HIF shows excellent performance against the state-of-the-art methods. In the future, we intend to extend HIF by NLP models and graph databases to exploit tweet content and improve efficiency.

Acknowledgments

This work is supported by the NSFC under grant 61972272, the NSF of the Jiangsu Higher Education Institutions of China under grant 21KJA520008, Qinlan Project of Jiangsu Province of China, Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions, the National Key R&D Program for the 14th-Five-Year Plan of China (2023YFC3804104 in 2023YFC3804100), and the Fundamental Research Funds for the Central Universities. Ling Liu acknowledges a partial support from NSF CISE under grants 2302720, 2312758, 2038029, an IBM faculty award, and a grant from CISCO Edge AI program.

References

[1] E. Bakshy, J. M. Hofman, W. A. Mason, and D. J. Watts. Everyone's an influencer: quantifying influence on twitter. In *WSDM*, 2011.
[2] P. Bao, H.-W. Shen, J. Huang, and X.-Q. Cheng. Popularity prediction in microblogging network: a case study on sina weibo. In *WWW*, 2013.
[3] T. Bian, X. Xiao, T. Xu, P. Zhao, W. Huang, Y. Rong, and J. Huang. Rumor detection on social media with bi-directional graph convolutional networks. In *AAAI*, 2020.
[4] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *JSTAT*, 2008.
[5] Q. Cao, H. Shen, K. Cen, W. Ouyang, and X. Cheng. DeepHawkes: Bridging the Gap between Prediction and Understanding of Information Cascades. In *CIKM*, 2017.
[6] Q. Cao, H. Shen, J. Gao, B. Wei, and X. Cheng. Popularity prediction on social platforms with coupled graph neural networks. In *WSDM*, 2020.
[7] X. Chen, F. Zhou, K. Zhang, G. Trajcevski, T. Zhong, and F. Zhang. Information diffusion prediction via recurrent cascades convolution. In *ICDE*, 2019.
[8] J. Cheng, L. Adamic, P. A. Dow, J. M. Kleinberg, and J. Leskovec. Can cascades be predicted? In *WWW*, 2014.
[9] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
[10] Y. Dong, J. Tang, S. Wu, J. Tian, N. V. Chawla, J. Rao, and H. Cao. Link prediction and recommendation across heterogeneous social networks. In *ICDM*, 2012.
[11] H. Gao, D. Kong, M. Lu, X. Bai, and J. Yang. Attention convolutional neural network for advertiser-level click-through rate forecasting. In *WWW*, 2018.
[12] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *SIGKDD*, 2016.
[13] A. Guille and H. Hacid. A predictive model for the temporal dynamics of information diffusion in online social networks. In *WWW*, 2012.
[14] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 1997.
[15] J. Hopcroft, T. Lou, and J. Tang. Who will follow you back? reciprocal relationship prediction. In *CIKM*, 2011.
[16] M. Jenders, G. Kasneci, and F. Naumann. Analyzing and predicting viral tweets. In *WWW*, 2013.
[17] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *SIGKDD*, 2003.
[18] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *WWW*, 2010.
[19] C. Li, J. Ma, X. Guo, and Q. Mei. DeepCas: An End-to-end Predictor of Information Cascades. In *WWW*, 2017.
[20] Y. Li, J. Fan, Y. Wang, and K.-L. Tan. Influence maximization on social graphs: A survey. *TKDE*, 2018.
[21] T. Lou, J. Tang, J. Hopcroft, Z. Fang, and X. Ding. Learning to predict reciprocity and triadic closure in social networks. *TKDD*, 2013.
[22] H. T. Nguyen, T. N. Dinh, and M. T. Thaip. Cost-aware Targeted Viral Marketing in billion-scale networks. In *INFOCOM*, 2016.
[23] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *SIGKDD*, 2014.
[24] J. Qiu, J. Tang, H. Ma, Y. Dong, K. Wang, and J. Tang. Deepinf: Social influence prediction with deep learning. In *SIGKDD*, 2018.
[25] H. Shen, D. Wang, C. Song, and A.-L. Barabási. Modeling and predicting popularity dynamics via reinforced poisson processes. In *AAAI*, 2014.
[26] L. Sun, A. Chen, P. S. Yu, and W. Chen. Influence maximization with spontaneous user adoption. In *WSDM*, 2020.
[27] X. Sun, J. Zhou, L. Liu, and W. Wei. Explicit time embedding based cascade attention network for information popularity prediction. *IPM*, 2023.
[28] J. Tang, X. Tang, and J. Yuan. Profit maximization for viral marketing in Online Social Networks. In *ICNP*, 2016.
[29] J. Tang, X. Tang, and J. Yuan. Towards profit maximization for online social network providers. In *INFOCOM*, 2018.
[30] X. Tang, D. Liao, W. Huang, J. Xu, L. Zhu, and M. Shen. Fully exploiting cascade graphs for real-time forwarding prediction. In *AAAI*, 2021.
[31] G. Tong, W. Wu, S. Tang, and D.-Z. Du. Adaptive Influence Maximization in Dynamic Social Networks. *TON*, 2017.
[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *NeuIPS*, 2017.
[33] D. Wang, C. Song, and A.-L. Barabási. Quantifying long-term scientific impact. *Science*, 2013.
[34] J. Wang, V. W. Zheng, Z. Liu, and K. C.-C. Chang. Topological recurrent neural network for diffusion prediction. In *ICDM*. IEEE, 2017.
[35] L. Weng, F. Menczer, and Y.-Y. Ahn. Predicting successful memes using network and community structure. In *AAAI*, 2014.
[36] Z. Wu, J. Zhou, L. Liu, C. Li, and F. Gu. Deep popularity prediction in multi-source cascade with heri-gcn. In *ICDE*, 2022.
[37] X. Xu, F. Zhou, K. Zhang, S. Liu, and G. Trajcevski. Casflow: Exploring hierarchical structures and propagation uncertainty for cascade prediction. *TKDE*, 2021.
[38] C. Yuan, J. Li, W. Zhou, Y. Lu, X. Zhang, and S. Hu. Dyhgcn: A dynamic heterogeneous graph convolutional network to learn users' dynamic preferences for information diffusion prediction. *arXiv preprint arXiv:2006.05169*, 2020.
[39] N. J. Yuan, Y. Zhong, F. Zhang, X. Xie, C.-Y. Lin, and Y. Rui. Who will reply to/retweet this tweet? the dynamics of intimacy from online social interactions. In *WSDM*, 2016.
[40] J. Zhang, Z. Fang, W. Chen, and J. Tang. Diffusion of "following" links in microblogging networks. *TKDE*, 2015.
[41] X. Zhang, A. Aravamudan, and G. C. Anagnostopoulos. Anytime information cascade popularity prediction via self-exciting processes. In *ICML*, 2022.
[42] L. Zhao, J. Chen, F. Chen, F. Jin, W. Wang, C.-T. Lu, and N. Ramakrishnan. Online flu epidemiological deep modeling on disease contact network. *GeoInformatica*, 2020.
[43] Q. Zhao, M. A. Erdogdu, H. Y. He, A. Rajaraman, and J. Leskovec. Seismic: A self-exciting point process model for predicting tweet popularity. In *SIGKDD*, 2015.
[44] W. Zhen, Z. Jingya, W. Jie, and S. Xigang. Wb-msf: A large-scale multi-source information diffusion dataset for social information diffusion prediction. In *CBD*, 2022.
[45] F. Zhou, X. Xu, G. Trajcevski, and K. Zhang. A Survey of Information Cascade Analysis: Models, Predictions and Recent Advances. *arXiv:2005.11041 [cs]*, 2020.
[46] F. Zhou, L. Yu, X. Xu, and G. Trajcevski. Decoupling representation and regressor for long-tailed information cascade prediction. In *SIGIR*, 2021.

A Appendix

A.1 Performance on R^2 and MAPE

We further measure the performance of HIF and baselines on the R^2 and MAPE, which are widely used in regression tasks, and the results are shown in Figure 6. The R^2 of some methods are negative, which are not plotted. Since the extreme values contribute differently to different metrics, some order changes in the performance of the baselines are observed, but the overall performance of HIF is still the best. Moreover, we can see that the performances are consistent with the MSLE that we analyzed in PO1, PO2 and PO3.

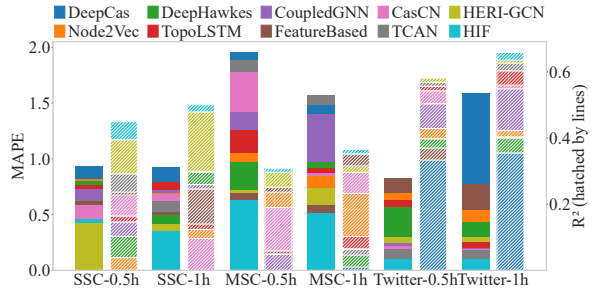


Figure 6: Performance of HIF and baselines on R^2 (right Y-axis, hatched by lines) and MAPE (left Y-axis) metrics, each bar is stacked by difference.

A.2 Details of Weighted Dynamic Sampling

The weighted dynamic sampling is designed to take advantage of differentiated contributions to sample followership. Accordingly, we firstly assign different weight bases ϕ_s, ϕ_r, ϕ_l for nodes in different positions in a cascade graph and assign a weight ϕ_c to all sub-cascades. Let α and β be the scale factors that limit the number Γ of sampled users, i.e., $\Gamma = \alpha|V_{user}^{c^{\mathcal{T}}}| + \beta$. In Algorithm 1, lines 1-6 are set for initialization, where $\mathcal{S}$ is the collection of node sets for sub-cascades, $\mathcal{H}$ is the set of candidate followers, and $\mathcal{N}_i$ is the follower set for each user i in cascade c.

We sample step by step until no more followers in $\mathcal{H}$ to sample or $|E^{f,c}|$ reaches Γ. We select only a batch of followers in each sampling step. At the beginning of each step, we first update the weights for users in the cascade (lines 8-12) according to their positions and the number of unsampled followers. Then, we update the weights of the users' followers in each sub-cascade s (lines 13-18) and build a set f_c to collect all followers for users in s (line 14). Line 16 shows two weight accumulations: one is to accumulate the weight for every user in s to his/her followers' weights (the summation term in line 16); the other is to further add weight ϕ_c to the weights for all followers of s. Let Π denote the maximum number of sampled followers in the current step, and we then select Π followers according to their weights to set $\mathcal{B}$. Finally, we remove $\mathcal{B}$ from $\mathcal{H}$ and merge $\mathcal{B}$ into $E^{f,c}$.

We provide an example in Figure 7 to illustrate the weighted dynamic sampling step by step. This cascade contains 2 sub-cascades, and we set $\phi_s = 3, \phi_r = 1, \phi_l = 2, \phi_c = 3, \pi = 2, \Gamma = 5$ for simplicity, i.e., at most two followers are sampled at each step, with a total of five followers. Specifically, at step 0, we get the weights for users and followers via lines 8-18 in Algorithm 1, and select the f_2 and f_0 with the largest weights. For instance, the intermediate user u_2 has 3 followers, and its weight is $w_2 = \phi_r + \ln(1+3) - \ln(1+0) \approx 2.4$; f_2 follows u_2 and u_5 from two unioque sub-cascades, so f_2's weight is $w_{f_2} = w_2 + w_5 + 2w_c = 11.1$, which is the largest and results the selection of f_2 at step 0. Similarly, we update the weights and select f_1 and f_6 at step 1. Following the update of weights and the selection of f_3 at step 2, the value of $|E^{f,c}|$ reaches Γ, so we stop sampling with the final state shown at step 3.

A.3 Dataset Analysis

The observed cascade sizes are shown in Figure 8a. The longer line on X-axis indicates that the corresponding dataset has more cascades, and the solid lines illustrate the final cascade size (the ground-truth), which follows a long-tail distribution.

Furthermore, we analyze the cascade spread size (in percentage) in relation to the time of spread (in hours) in the MSC and SSC. The results are depicted in Figure 9. Notably, we can observe a restricted scale of information diffusion over short durations. For instance, the average cascade sizes after 0.1 hours are a mere 0.017% for SSC and 0.022% for MSC, indicating a limited utility of model predictions during this early stage. The average time required for information to reach 10% of its final size is found to be 38.2 hours for SSC and 29.9 hours for MSC.

It is worth noting that previous studies, including [5, 19, 36, 41], have consistently demonstrated that longer observation periods contribute to more accurate predictions. Consequently, in Sec. 5, we deliberately opt for the more challenging observation times of 0.5 and 1 hour to evaluate our proposed framework.

Algorithm 1 Weighted Dynamic Sampling

Input: Position weights $\phi_s, \phi_r, \phi_l, \phi_c$, scale factors α, β, sampling-batch size π, cascade graph $G^{r,c} = (V_{user}^{c}, E^{r,c})$, and followership graph $G^{f} = (V_{user}, E^{f})$.

1: $E^{f,c} \leftarrow \emptyset, \Gamma \leftarrow \alpha|V_{user}^{c^{\mathcal{T}}}| + \beta$
2: $\mathcal{S} \leftarrow$ collection of node sets for sub-cascades
3: $\mathcal{H} \leftarrow$ set of all followers for users in c
4: **for** each user i in $V_{user}^{c^{\mathcal{T}}}$ **do**
5: $\quad \mathcal{N}_i \leftarrow$ follower set of i in G^f
6: **end for**
7: **while** $|E^{f,c}| < \Gamma$ and $|H| > 0$ **do**
8: $\quad$ **for** each user i in $V_{user}^{c^{\mathcal{T}}}$ **do**
9: $\quad\quad p_i \leftarrow$ position of i in $G^{r,c}$
10: $\quad\quad f_i \leftarrow$ number of sampled followers in $\mathcal{N}_i$
11: $\quad\quad w_i \leftarrow \phi_{p_i} + \lg(1 + |\mathcal{N}_i|) - \lg(1 + f_i)$
12: $\quad$ **end for**
13: $\quad$ **for** each sub-cascade s in $\mathcal{S}$ **do**
14: $\quad\quad f_c = \bigcup\{\mathcal{N}_i | \forall$ user $i \in s\}$
15: $\quad\quad$ **for** each follower j in f_c **do**
16: $\quad\quad\quad w_j \leftarrow \sum\{w_i | \forall (j,i) \in E^f\} + \phi_c$
17: $\quad\quad$ **end for**
18: $\quad$ **end for**
19: $\quad \Pi \leftarrow \min(|\mathcal{H}|, \pi, \Gamma - |E^{f,c}|)$
20: $\quad \mathcal{B} \leftarrow$ select at most Π followers by w_j from $\mathcal{H}$
21: $\quad \mathcal{H} \leftarrow \mathcal{H} \setminus \mathcal{B}, E^{f,c} \leftarrow E^{f,c} \bigcup \mathcal{B}$
22: **end while**

Output: Sampled *follow* edge set $E^{f,c}$.

A.4 Efficiency Analysis

We further compare the number of followerships sampled by different methods and the total time cost consumed by HIF and baselines.

A.4.1 Sampled Followership. We compare the followerships sampled by hop-based sampling and weighted dynamic sampling, and the results are shown in Figure 8b. In this case, the settings for weighted dynamic sampling are the same as the settings in Sec. 5.5.1. For 1-hop sampling, it samples relatively few followerships for short cascades on the Twitter dataset and samples a large followership network for long cascades on datasets SSC and MSC, which leads to either insufficiency or redundancy of followership network for Twitter or SSC/MSC.

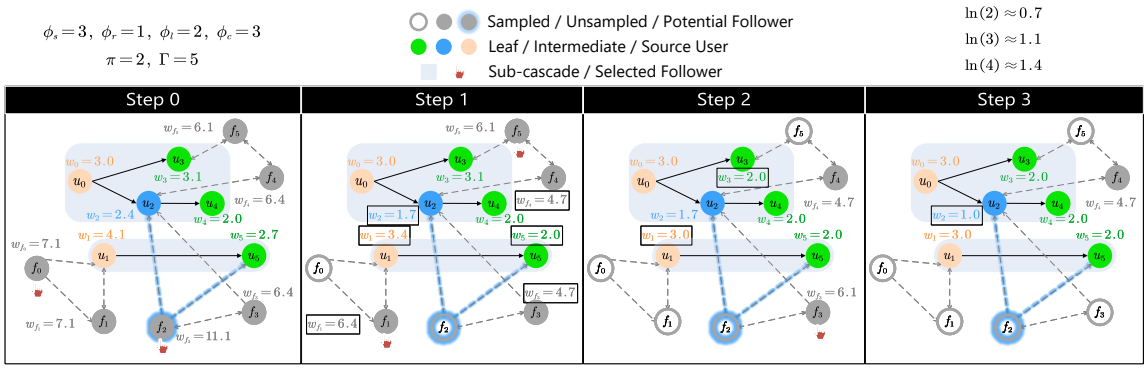
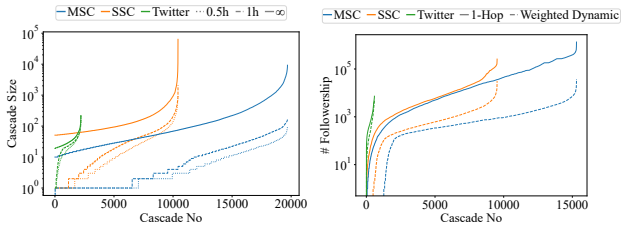
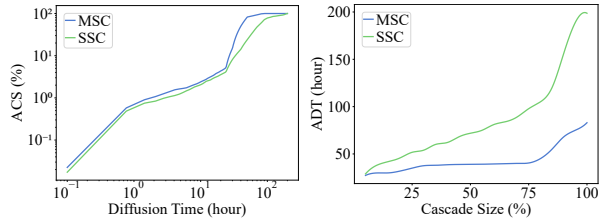


Figure 7: A step-wise example of weighted dynamic sampling. The altered weights are encased in boxes, users in disparate cascade positions are distinguished by different colors, and the potential followers and its followerships are highlighted in blue.



(a) Observed cascade size **(b) Sampled followerships**

Figure 8: The observed and final cascade size of datasets, and the followership sampled by different methods.



(a) ACS **(b) ADT**

Figure 9: The average cascade size (ACS) after a specific diffusion time and average diffusion time (ADT) to reach a specific cascade size on the MSC and SSC.

In comparison, the proposed weighted dynamic sampling samples more followerships for short cascades to provide sufficient beneficial information and also has a tighter upper bound for large cascades to avoid the low efficiency of the framework's inference.

A.4.2 Time Cost. We analyze the time cost of HIF and baselines and plot the results of 1-hour observation in Figure 10. The time cost for each model is mainly determined by both the time of the training step (i.e., the time of forward inference and backward propagation, depending on the model's complexity) and the convergence capability (which determines the total number of training steps). It implies that the simplest model is not necessarily the fastest one, e.g., the FeatureBased and the Node2Vec are slower than DeepCas on datasets SSC and MSC. It is worth mentioning that HIF is more efficient than CasCN and HERI-GCN in most cases, though it is

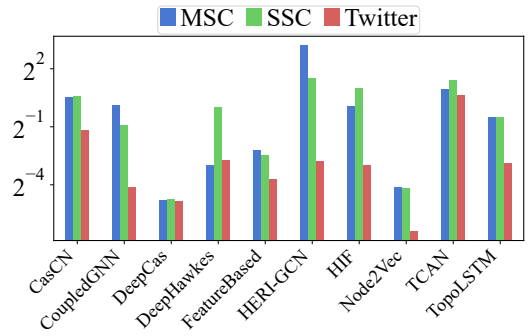


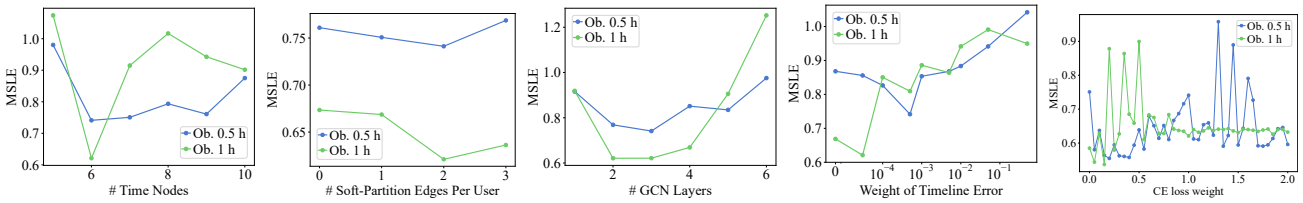
Figure 10: The training time cost on each dataset for all approaches with 1-hour observation.

not the fastest one. Overall, HIF achieves a good trade-off between efficiency and performance.

A.5 Sensitivity Analysis

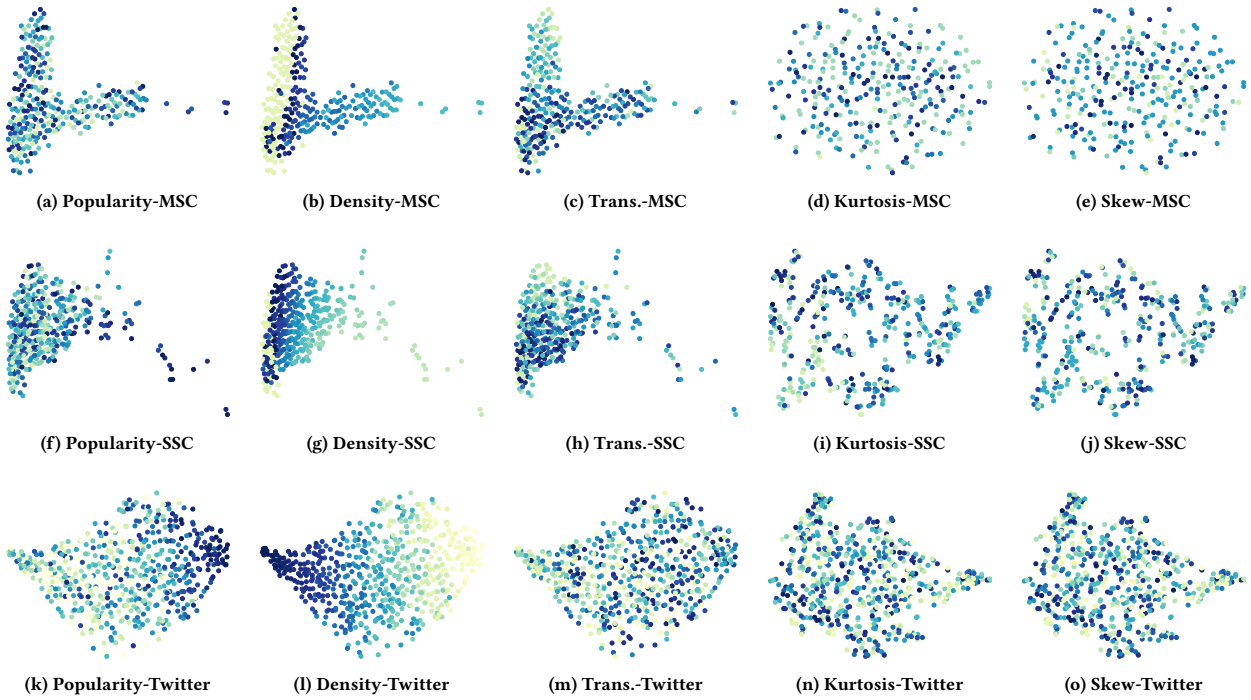
Many notable hyperparameters might have impacts on the model inference and data structure. We use the MSC dataset to investigate the effect of these hyperparameters, including the number S of soft-partition edges per user, the number T of time nodes, the number L of GCN layers, the timeline error weight α_1 and the CE loss weight α_2. As illustrated in Figure 11, we explain the observations below:

- The number of time nodes determines the private receptive field of each time node, and the number of soft-partition edges determines the shared receptive field of each user. These two hyperparameters are complementary to each other, and determine the total receptive field to users for each time node commonly.
- As the number of GCN layers increases from 1 up to 6, the performance first improves and then falls since the shallow model has limited learning ability, while the deep model is easy to suffers from over-fitting.
- We select multiple representative values for the weight of timeline loss, and the results demonstrate that timeline loss with a suitable weight (e.g., 10^{-3} for 0.5-hour observation and 5×10^{-5} for 1-hour observation) can benefit the prediction performance.
- With regard to the CE loss weight α_2, the results show that the best performance is achieved when α_2 is set to 0.2 for 0.5-hour observation and 0.15 for 1-hour observation.



(a) number of time nodes T (b) soft-partition per user S (c) number of GCN layers L (d) Timeline error weight α_1 (e) CE loss weight α_2

Figure 11: The sensitivity of T, S, L, α_1 and α_2 on the MSC dataset.



(a) Popularity-MSC (b) Density-MSC (c) Trans.-MSC (d) Kurtosis-MSC (e) Skew-MSC

(f) Popularity-SSC (g) Density-SSC (h) Trans.-SSC (i) Kurtosis-SSC (j) Skew-SSC

(k) Popularity-Twitter (l) Density-Twitter (m) Trans.-Twitter (n) Kurtosis-Twitter (o) Skew-Twitter

Figure 12: Visualization of learned information cascade embedding and manually extracted properties, for each point, the darker the color, the higher the property value. "Trans." is the abbreviation of transitivity.

A.6 Interpretability Analysis

In this section, we visualize the learned information cascade embedding by projecting the embeddings as 2D points via t-SNE[4] and coloring these points by hand-crafted properties. The visualization results are drawn in Figure 12 under 1-hour observation.

The properties to be visualized include the final popularity (i.e., the ground-truth of our task), the density of the cascade graph, the transitivity of sampled followership network, the kurtosis, and the skew of the time series. The former three are the spatial properties coming from the cascade graph and the followership network. The latter two temporal properties come from the (re)posting time series, where the skewness reflects the stage of information diffusion. In a complete diffusion process, the popularity is in the tendency of low - high - low. The kurtosis coefficient demonstrates the explosion of information diffusion. These two temporal coefficients jointly reflect the state of the information diffusion process. We measure the HIF's learning ability of spatial-temporal features by the correlation between the learned embeddings (the position of points) and these manually extracted properties (the color of points).

We can observe a noticeable correlation between the color gradient and the position of the points (e.g., in Figure 12(l) indicating that the embeddings learned by our framework can better reflect the spatial-temporal properties of the original data, i.e., HIF learns spatial-temporal features well.

The multi-source cascade aggregates sub-cascades with different kurtosises and skewnesses and released at different times, so the distribution of the MSC temporal feature scatter is more uniform. The observation duration in our settings (i.e., 0.5 h and 1 h) is too short to observe a prominent diffusion process (as shown in Figure 8a), resulting in a uniform distribution of temporal property scatters on SSC and Twitter datasets. However, we still observe that each point has a similar skewness or kurtosis as its nearest neighbor.