CasSampling: Exploring Efficient Cascade Graph Learning for Popularity Prediction

Guixiang Cheng[1,2], Xin Yan[1,2(✉)], Shengxiang Gao[1,2], Guangyi Xu[3], and Xianghua Miao[1,2]

[1] Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming, China
kg_yanxin@sina.com

[2] Yunnan Key Laboratory of Artificial Intelligence, Kunming University of Science and Technology, Kunming, China

[3] Yunnan Nantian Electronics Information Co., Ltd., Kunming, China

Abstract. Predicting the growth size of an information cascade is one of the primary challenges in understanding the diffusion of information. Recent efforts focus on utilizing graph neural networks to capture graph structure. However, there is considerable variance in the information cascade size (from few to million). From the perspective of efficiency and performance, the method of modeling each node is inappropriate for graph neural networks. In this paper, we propose a novel deep learning framework for popularity prediction called CasSampling. Firstly, we exploit a heuristic algorithm to sample the critical part of cascade graph. For the loss of structure information due to sampling, we keep outdegree of sampled node in the global graph as part of the node feature into the graph attention networks. For the loss of temporal information due to sampling, we utilize the time series to learn the global propagation time flow. Then, we design an attention aggregator for node-level representation to better integrate local-level propagation into the global-level time flow. Experiments conducted on two benchmark datasets demonstrate that our method significantly outperforms the state-of-the-art methods for popularity prediction. Additionally, the computation cost is much less than the baselines. Code and (public) datasets are available at https://github.com/Gration-Cheng/CasSampling.

Keywords: Popularity prediction · Cascade graph sampling · Graph neural network

1 Introduction

With the improvement of communication technology, the rapid development of online social media has promoted the propagation and interaction of massive information. Through social media, people spread news, politics, and life hot spots in a cascading way. Therefore, the prediction of the information propagation cascade is significant, and the effective prediction of the number of retweets

D. Koutra et al. (Eds.): ECML PKDD 2023, LNAI 14171, pp. 70–86, 2023.
https://doi.org/10.1007/978-3-031-43418-1_5

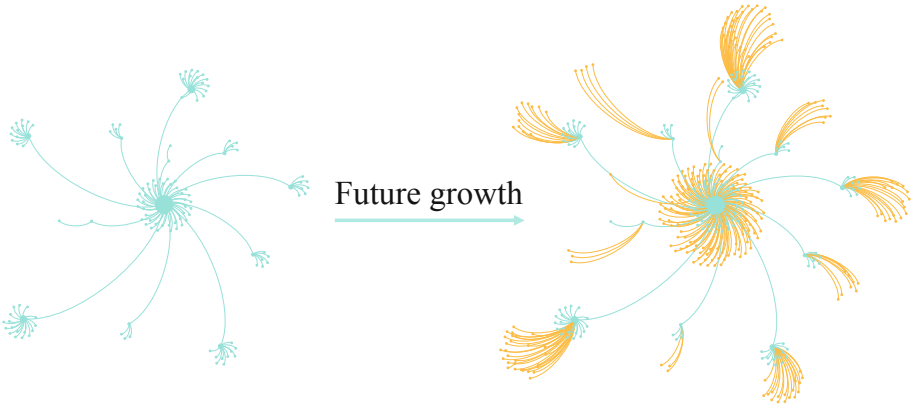


Fig. 1. A real-world example of a propagation graph in the Weibo dataset. The left figure shows the propagation graph during the observation time 1 h, while the right figure shows the propagation graph after 23 h. The blue nodes denote the observed nodes, while the orange nodes represent the nodes that will propagate in the future.

in a period is beneficial for understanding the cascade, which has attracted considerable attention in academia and industry. Cascading propagation plays a crucial role in many downstream tasks, such as accelerating or suppressing the spread of information [1,2], rumor detection, and even epidemic prediction. However, social media are large open platforms, and the uncertainty about cascading effects makes popularity prediction an extremely challenging problem.

In recent years, Most of the research uses deep learning-based approaches to learn the representation of cascades. Most of them [3–6], view an information propagation as a sequence of events and use subgraphs or subsequences to represent the cascade. However, modeling subgraph or subsequence is difficult to learn the cascade effect of information propagation.

Recent research has focused on using graph neural networks to capture the cascade effect [4,6–8]. Graph Neural Networks (GNN) are able to effectively model graph-structured data by integrating node attributes and topology. However, when facing a large number of nodes, GNNs can be computationally expensive and inefficient. As shown in Fig. 1, many of the nodes within the observation time do not bring new forwarding propagation. These nodes have little effect on propagation. It is obvious that nodes with more propagation during the observation time will have a larger cascading effect.

Note that with these problems, existing methods confront several challenges: (1) Some methods model each node to learn node-level representation, but it is not efficient because of the cascade size (from few to million). (2) The method of modeling subgraphs or subsequences is difficult to learn the cascade effect of information propagation. (3) Time is crucial information. Existing methods lack the extensive use of time information both at the local-level and the global-level.

To address these challenges, we proposed a novel neural network model named CasSampling. The model focuses on sampling the cascade graph and compensates for the loss of time and structure information due to sampling, making the model more efficient and performing better. Our main contributions can be summarized as follows:

- **Efficiency graph representation**. We implemented a heuristic algorithm to sample the key part of cascade graph, which address the problem of the large variable size of graphs that make it difficult to model with GNNs. To compensate for the loss of graph structure due to sampling, we retain the outdegree of sampled nodes as part of the sampled nodes feature. It efficiently models cascade graphs with the large variable sizes and is effective for explicitly capturing cascading effects.
- **Multi-scale time information**. We design an attention aggregator that combines the node's propagation embedding with the time stamp of nodes. To compensate for the loss of time information caused by graph sampling, we use time series to learn the global propagation time flow. We have successfully integrated two types of temporal information for the first time, which enables us capture the potential information between the retweet time of the active node and the global propagation time flow, and it can more fully model the information diffusion process.
- **Evaluation on benchmark datasets.** We conduct extensive evaluations on two publicly available benchmark datasets, demonstrating that CasSampling significantly outperforms the state-of-the-art (SOTA) baselines and reduces the computational cost.

2 Related Work

We review the related work grouped into three main categories: featured-based, generative process and deep learning-based approaches.

Featured-Based Approaches. It usually extracts features from specific platforms, such as Arxiv; Weibo [3]; Twitter [9]; These features include user-related features [10,11], content-related features [12], cascades structural [13,14] and temporal features [15]. However, feature-based approaches extract features from different platforms, making the learned features difficult to generalize into different scenarios, and the prediction performance heavily relies on the quality of the hand-crafted feature.

Generative Process Approaches. It mainly regards the process of message diffusion as an arrival time sequence [16–19]. These methods focus on modeling the intensity function for the arrival process. The popularity prediction can be obtained by event simulation of the intensity function. These methods demonstrate enhanced comprehensibility, but they rely on certain assumptions, and we do not know whether these assumptions are valid in real situations, limiting model performance.

Deep Learning-Based Approaches. Generally, existing methods for popularity prediction mainly focus on four types of information, i.e., temporal information, node representation, structure representation, and content.

For node and structure representation. DeepHawkes [3] integrates an interpretable Hawkes process for information cascade prediction. With the booming development of graph neural networks (GNN), CasCN [4] uses it to model the structural information of cascade subgraphs (via cascade Laplacian). CasFlow [20] uses a variational autoencoder to learn the uncertainty of the cascade. TempCas [5] exploits a heuristic algorithm to sample critical paths and utilizes some handcraft features as compensation for the sampled graph. All the work above depicts the subgraph of participating users. However, modeling subgraphs or subsequences is difficult to learn the cascade effect of information propagation. CouledGNN [7] uses the global propagation graph to capture the interaction between node activation states and diffusion; CasGCN [21] considers that the cascade effect is bidirectional, uses in-coming and out-going adjacency matrices as the representation of graph structure, and combines the time information with the graph structure by attention mechanism. These GNN-based methods model each node for graph embedding. However, from the perspective of efficiency and performance, the method of modeling each node is inappropriate for GNN.

For temporal information modeling, DeepHawkes [3] introduced a nonparametric time decay function into the path modeling of recurrent neural networks. DFTC [22] represents temporal information by time series, which used a Convolution-1d neural network for capturing short-term outbreaks and LSTM for long-term fluctuations. TempCas [5] improves the DFTC [22] by combining the short-term and long-term rather than capturing them separately. However, these methods do not fully utilize global-level and local-level time information.

Note these problems. We propose a novel model, called CasSampling, which implements a heuristic algorithm to sample the cascade graph and compensate for the loss of time and structure information caused by sampling. Compared with the same model, we sample cascade at the graph-level for the first time and fully utilize multi-scale time information. It achieved better performance and less computation cost.

3 Preliminaries

We now present the essential background and formally define the popularity prediction problem.

Definition 1 (Cascade Graph). *Suppose that we have n posts,* $P = \{p_c, c \in [1, n]\}$. *For each post* p_c, *there is a cascade graph denoted by* $\mathcal{G}_c = (\mathcal{V}_c, \mathcal{E}_c, \mathcal{T}_c)$, *where* $\mathcal{V}_c$ *is a set of nodes that have been involved in the cascade during the observation time* T, *a directed edge* $(v_i, v_j) \in \mathcal{E}_c$ *represents that node* v_j *retweet from node* v_i, *and a tuple of node time label* $(v_i : t_i) \in \mathcal{T}_c$ *denotes the time elapsed between the original post and node* v*'s retweet.*

Definition 2 (Growth size). *It is defined as the amount of cascade growth number over observation time window T after it has spread for Δt. According to* **Definition 1**, *we obtain $\mathcal{G}_c = (\mathcal{V}_c, \mathcal{E}_c, \mathcal{T}_c), \mathcal{T}_c < T$. Our task is to predict the growth size ΔS_i of a cascade after a given time interval Δt. The growth size can be defined as $\Delta S_i = |\mathcal{V}_i^{T+\Delta t}| - |\mathcal{V}_i^{T}|$.*

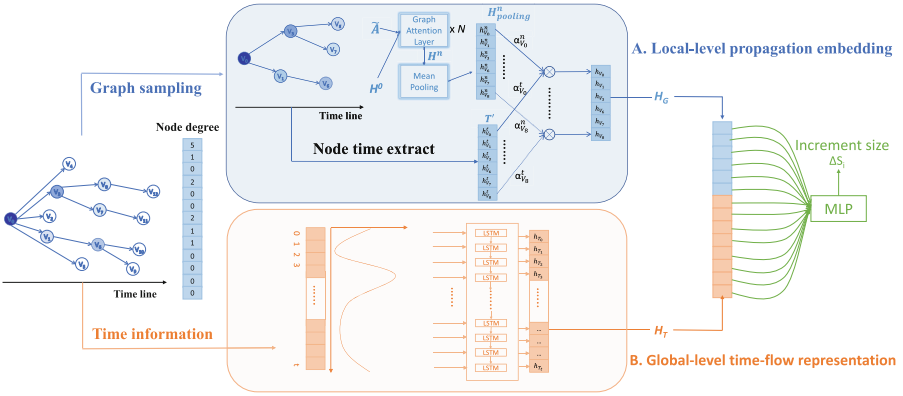


Fig. 2. The framework of CasSampling for popularity prediction.

4 Method

Before introducing the details of the CasSampling model, we present the overall framework of CasSampling in Fig. 2. It contains four major parts:(1)A heuristic algorithm to sample the critical parts of a graph. (2) Local-level propagation embedding model graph structure with GAT and uses the attention aggregator to combine it with node time information. (3) Global-level time flow representation adopts LSTM on time series to capture the propagation trend. (4) Prediction layer concatenates local-level propagation and global-level time flow into the self-attention layer to fuse each other and feed into Multilayer-Perception(MLP) to predict the increment size.

4.1 Graph Sampling

Efficient node representation is challenging due to the variable size of cascades(from few to million). Specifically, the millions of nodes for the GNN are computation expensive. To achieve an efficient graph representation, we used rule-based graph sampling to reduce the number of nodes while preserving the original graph information as much as possible.

Algorithm 1 Graph sampling

Input: A cascade graph $\mathcal{G}_c$, degree vector D of $\mathcal{G}_c$; the maximum number of nodes K.
Output: Sampled graph$\mathcal{G}_c^{sampled}$ and out-degree vector $D_{sampled}$ of $\mathcal{G}_c^{sampled}$

$\mathcal{V}_c, \mathcal{E}_c, \mathcal{T}_c \leftarrow \mathcal{G}_c$
if $|\mathcal{V}_c| < K$ **then**
  return $\mathcal{G}_c, D$
end if
$D_{sorted}, \mathcal{V}_{sorted}$=sort$(D, \mathcal{V}_c, \mathcal{T}_c)$# Sort by Rule 1 and Rule 2
$D_{sampled}$,$\mathcal{V}_{sampled}$=selectTop(D_{sorted},$\mathcal{V}_{sorted}$,K)
$\mathcal{E}_{sampled} \leftarrow \varnothing, \mathcal{T}_{sampled} \leftarrow \varnothing$
for each $\{v_i, v_j\}$ in $\mathcal{E}_c$ **do**
  if $v_j \notin \mathcal{V}_{sampled}$ **then**
    continue
  else if $v_i \in \mathcal{V}_{sampled}$ **then**
    $\mathcal{E}_{sampled} \leftarrow \mathcal{E}_{sampled} \cup \{v_i, v_j\}$
  else
    v_k=findAncestor(v_i,$\mathcal{E}_c$,$\mathcal{V}_{sampled}$) # Find the nearest ancestor of node v_i in the $\mathcal{V}_{sampled}$.
    $\mathcal{E}_{sampled} \leftarrow \mathcal{E}_{sampled} \cup \{v_k, v_j\}$
  end if
end for
for each v in $\mathcal{V}_{sampled}$ **do**
  $\mathcal{T}_{sampled} \leftarrow \mathcal{T}_{sampled} \cup (v : \mathcal{T}(v))$
end for
$\mathcal{G}_c^{sampled} = \{\mathcal{V}_{sampled}, \mathcal{E}_{sampled}, \mathcal{T}_{sampled}\}$
return $\mathcal{G}_c^{sampled}$,$D_{sampled}$

Given a cascade graph $\mathcal{G}_c$ and the adjacency matrix $\mathbf{A_c}$.The outdegree vector $D = \{d_i, i \in [1, n]\}$ can be computed with:

$$d_i = log_2(\sum_{j=1}^{N} a_{ij}), \tag{1}$$

where N is the number of nodes and a_{ij} is one element of $\mathbf{A_c}$. The d_i denotes the outdegree of node i after logarithmic scaling.

Since the node with a larger out-degree is more critical, and according to the Hawkes process [3], the point closer to the occurrence of time has the more significant influence, there are two rules for sorting:

Rule 1: Sort the $\mathcal{V}_c$ by the outdegree D of nodes.(from large to small).
Rule 2: For nodes with the same outdegree, make a second sort according to their time(from late to early).

To reduce computation costs and improve performance, we sampled nodes based on their sorted out-degree vector and selected the top K nodes. However, this process may result in some nodes missing parent nodes. To address this issue, we identified the nearest ancestor node that was not filtered out and used

it as the parent node. To preserve the original graph information, we used the out-degree of the original graph node as the feature for the sampled node.

The process of graph sampling is shown in Algorithm 1.

4.2 Local-Level Propagation Embedding

Graph Attention Layer. Recently, Graph neural networks have been advanced in graph learning. To capture the local information and achieve node-level representation, we utilize graph neural networks to learn hidden information among cascade nodes. Each tweet has a different cascade graph which is an inductive task, so we choose Graph Attention networks [23] (GAT) to learn the graph structure.

The input of GAT consists of two parts, adjacency of the graph and the node's feature matrix. After graph sampling, we obtain the $\mathcal{G}_c^{sampled}$ and $D^{sampled}$ of a cascade. To retain the original graph information, we reserve the outdegrees of nodes as part of nodes feature to learn the node influence and the size of the original cascade. The node input feature H^0 can be defined as:

$$H^0 = \mathbf{A} + \mathrm{diag}(D_{sampled}) = \begin{bmatrix} - & h_1^0 & - \\ - & h_2^0 & - \\ & \vdots & \\ - & h_N^0 & - \end{bmatrix}, \tag{2}$$

where $\mathbf{A}$ is the adjacency matrix of $\mathcal{G}_c^{sampled}$. $\mathrm{diag}(D_{sampled})$ indicates diagonalizing the $D_{sampled}$ vector. $h_i^0 \in \mathbb{R}^F$ is the input feature of node i, $F = N$, and N is the number of nodes.

For the adjacency matrix, we add self-connection to prevent loss of self-information during aggregation. The Adjacency $\tilde{\mathbf{A}}$ is formulated as:

$$\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I_n}, \tag{3}$$

where $\mathbf{I_n} \in \mathbb{R}^{N \times N}$ is an identity matrix.

The main idea of GAT is to aggregate node features by calculating the attention weight between connected nodes. After n layers of GAT, the node receives messages from other nodes within n-hops. The aggregate function is as follows:

$$h_i^n = \sigma\Big(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W} h_j^{n-1}\Big), \tag{4}$$

where $\mathcal{N}_i$ is the neighborhood of node v_i in the graph, which can be obtained from $\tilde{\mathbf{A}}$. The h_j^{n-1} represents embedding of node j after n-1 layers of GAT. The α_{ij} is the attention score between node i and node j. It can be calculated by:

$$\alpha_{ij} = \frac{\exp(\mathrm{LeakyReLU}(\mathbf{a}^T[\mathbf{W}h_i \| \mathbf{W}h_j]))}{\sum_{k \in \mathcal{N}_i} \exp(\mathrm{LeakyReLU}(\mathbf{a}^T[\mathbf{W}h_i \| \mathbf{W}h_k]))}, \tag{5}$$

where $\mathbf{a} \in \mathbb{R}^{2F'}$ denote the learnable parameters. LeakyReLU is an activation function. $\|$ is the concatenation operation. A weight matrix $\mathbf{W} \in \mathbb{R}^{F' \times F}$ is shared to every node.

We do mean pooling with the output H^n of the n layers of GAT, and pooling the feature of each node.

$$H^n_{pooling} = \mathbf{MeanPooling}(H^n). \tag{6}$$

The final graph embedding can be expressed as: $H^n_{pooling} = [h^n_1, h^n_2, \ldots, h^n_N]^T$, $H^n \in \mathbb{R}^N$, where $h^n_i \in \mathbb{R}$ is the embedding of node i.

Node Time Information. To integrate graph structure and global temporal flow, we preserve the original time information of nodes instead of incorporating it into GAT input features. We utilize the attention aggregator to merge node structure and time information as the graph represents. Let $T' = [h^t_1, h^t_2, \ldots, h^t_N]^T$, $T' \in \mathbb{R}^N$ denote the time stamp of the node, which is obtained from $\mathcal{T}_{sampled}$. $h'_i = [h^n_i \| h^t_i]$ represents node i concatenating GAT output and time information.

$$e^j_i = \mathbf{v}^T tanh(\mathbf{W_a}[h^j_i \| h'_i]), \tag{7}$$

where h^j_i denotes one of the node feature of h'_i. $\mathbf{W_a}$ and $\mathbf{v}$ denote the learnable parameter.

$$\alpha^j_i = \frac{\exp(e^j_i)}{\sum_{k \in \{t,n\}} \exp(e^k_i)}, \tag{8}$$

where α^j_i represents the attention score of h_{ij}.

Aggregate the features of node i, and the aggregate function is as follows:

$$h_i = \sum_{j \in \{t,n\}} \alpha^j_i h^j_i. \tag{9}$$

$H_G = [h_1, h_2, \ldots, h_N], H_G \in \mathbb{R}^N$ contains nodes structure embedding and nodes time information.

4.3 Global-Level Time Flow Representation

Given a fixed interval of time t_s, We have T/t_s time slots. From $\mathcal{G}_c$, we can get the global graph node's time information, then calculate the time slot of each node. Let $R_T = \{r_1, r_2, \ldots, r_{T/t_s}\}, R_T \in \mathbb{R}^{T/t_s}$ denote the temporal flow sequence. We utilize LSTM to capture the time flow information.

$$H_T = \mathbf{LSTM}(R_T), \tag{10}$$

where $H_T \in \mathbb{R}^{T/t_s}$ is the global propagation time flow representation.

4.4 Prediction Layer

Each node in H_G represents graph structure information. H_T contains global temporal information. We concatenate the two parts into Self-Attention [24] layer to fuse each other and feed into MLP to predict the increment size.

$$H = H_G \oplus H_T, \tag{11}$$

$$H' = \textbf{Self Attention}(H) + H, \tag{12}$$

$$\Delta S_i = \textbf{MLP}(H^{'}), \tag{13}$$

Our ultimate task is to predict the increment size for a fixed time interval, which can be done by minimizing the following loss function:

$$\mathcal{L}(\Delta S_i, \Delta \hat{S}_i) = \frac{1}{P}\sum_{i=1}^{P}(\log_2 \Delta S_i - \log_2 \Delta \hat{S}_i)^2, \tag{14}$$

where P is the number of posts, ΔS_i is the predicted amount of growth, $\Delta \hat{S}_i$ is the ground truth.

4.5 Complexity Analysis

The complexity based on the sparse matrix operation of GAT is $\mathcal{O}(|\mathcal{V}_c|F'F + |\mathcal{E}_c|F)$, where F is the number of input features and F' is the number of output features. The complexity of the node-level time attention mechanism is $\mathcal{O}(|\mathcal{V}_c|)$. Compare with the subsequence-based method, our method can be parallelized. The complexity of global time flow representation is $\mathcal{O}(T/t_s)$. The complexity of Self-Attention layer is $\mathcal{O}((|\mathcal{V}_c| + T/t_s)^2)$. Sum up, the complexity approximates to $\mathcal{O}(|\mathcal{V}_c|F'F + |\mathcal{E}_c|F + (|\mathcal{V}_c| + T/t_s)^2)$.

5 Experiments

To evaluate the performance of our model, we compare CasSampling with several SOTA methods on two benchmark datasets under various evaluation metrics.

5.1 Datasets

Sina Weibo. The dataset [3] comes from Sina Weibo, a major microblogging site that is similar to Twitter. It contains posts that were published between 0:00 to 24:00 on June 1, 2016.

Twitter. The dataset is collected by [9] and contains public English written tweets published between Mar 24 and Apr 25, 2012.

The observation time T for Weibo is set to 0.5 h, 1 h and 2 h, and 1 days,2 days and 3 days for Twitter. We select 24 h as the prediction time for the Weibo dataset and 32 days for the Twitter dataset. Following earlier methods [3,20],

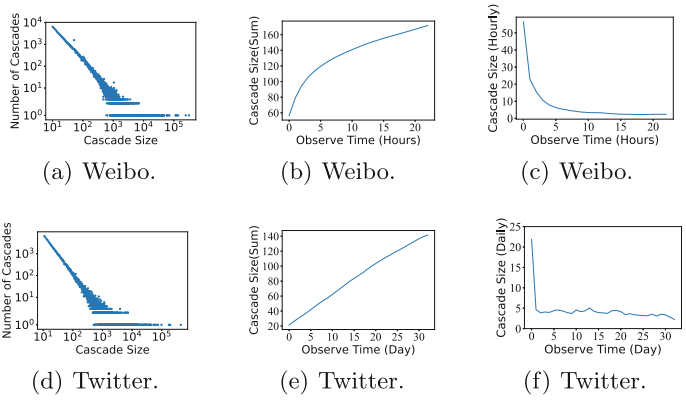
we filter out cascades whose $|\mathcal{V}_c| < 10$, and due to the effect of diurnal rhythm in Weibo, we focused on tweets published between 8 a.m. and 6 p.m. to give each tweet at least 6 h to gain retweets. For Twitter, we only tracked tweets published before Apr 10, ensuring at least 15 days for each tweet to grow adopters. For each of two datasets, we randomly split it into training set (70%), validation set (15%), and test set (15%). The statistics and visualization of these two datasets are shown in Table 1 and Fig. 3.

5.2 Baseline

To validate CasSampling's performance in popularity prediction, we chose the following SOTA baselines for comparison:

Table 1. Descriptive statistics of two datasets.

Dataset	Ori. Cascade	Avg. path length	Avg. popularity	0.5 h/1 day			1 h/2 days			2h/3days		
				Train	Val	Test	Train	Val	Test	Train	Val	Test
Weibo	119313	1.217	171.098	21461	4598	4598	27353	5860	5860	32943	7059	7059
Twitter	88440	1.201	142.672	9640	2065	2065	12740	2729	2729	15777	3380	3380



(a) Weibo. (b) Weibo. (c) Weibo.

(d) Twitter. (e) Twitter. (f) Twitter.

Fig. 3. Cascade size distribution of each dataset. In the 1st column, each figure shows the distribution of cascade sizes. The 2nd column denotes the mean sum cascade size changing over observation time. The 3rd column describes the mean hourly cascade size change over the observation time.

Feature-Linear and Feature-Deep. We have extracted all the predictable features from recent research [10,12,13,15]. Then, we feed it into a linear regression model and a fully-connected layer to predict the increment size.

DeepHawkes [3]. DeepHawkes considers three important aspects of the Hawkes process: user influence, time decay effect, and self-exciting mechanism.

CasCN [4]**.** CasCN is the first GNN-based framework exploiting both structural and employs a sequence of sub-cascade graphs with cascade Laplacian.

CasFlow [20]**.** CasFlow combines the local structure of cascade graph with global social collaboration network.

TempCas [5]**.** TempCas sample the critical path of cascade and utilizes handcrafted features to compensate for structural loss. It uses LSTM and attention CNN to model long-short term time information.

5.3 Evaluation Metrics

Following existing works [3,4,25], we use Mean Square Logarithmic Error (MSLE) and Symmetric Mean Absolute Percentage Error (SMAPE) for prediction performance evaluation, which are defined as:

$$\text{MSLE} = \frac{1}{P}\sum_{i=1}^{P}(\log_2 \Delta S_i - \log_2 \Delta \hat{S}_i)^2, \tag{15}$$

$$\text{SMAPE} = \frac{1}{P}\sum_{i=1}^{P}\frac{|\log_2 \Delta S_i - \log_2 \Delta \hat{S}_i|}{(\log_2 \Delta S_i + \log_2 \Delta \hat{S}_i)/2}, \tag{16}$$

where P is the number of posts, ΔS_i is the predicted amount of growth, $\Delta \hat{S}_i$ is the ground truth.

Table 2. Results on Weibo and Twitter dataset.

Model	Weibo						Twitter					
	0.5 h		1 h		2 h		1 Day		2 Days		3 Days	
	MSLE	SMAPE	MSLE	SMAPE	MSLE	SMAPE	MSLE	SMAPE	MSLE	SMAPE	MSLE	SMAPE
Feature-Linear	3.025	0.305	2.653	0.323	2.451	0.332	9.123	0.698	6.729	0.632	5.833	0.602
Feature-Deep	2.891	0.281	2.612	0.319	2.332	0.311	7.801	0.669	6.330	0.599	5.439	0.574
DeepHawkes	2.674	0.277	2.538	0.303	2.312	0.302	6.874	0.635	5.085	0.545	4.281	0.463
CasCN	2.660	0.275	2.613	0.323	2.452	0.310	7.121	0.638	5.438	0.560	4.482	0.463
CasFlow	2.418	0.247	2.298	0.257	2.003	0.281	6.989	0.625	5.143	0.552	4.102	0.449
TempCas	2.332	0.243	2.219	0.248	2.001	0.278	6.232	0.611	4.332	0.525	3.680	0.455
CasSampling-Struct	2.553	0.246	2.483	0.283	2.339	0.307	7.329	0.667	5.773	0.591	4.681	0.483
CasSampling-St.ND	2.793	0.287	2.681	0.312	2.539	0.352	7.811	0.687	5.983	0.610	4.997	0.498
CasSampling-TimeFlow	2.388	0.244	2.241	0.248	2.021	0.281	6.322	0.602	4.349	0.521	3.757	0.457
CasSampling-NNT	2.311	0.244	2.178	0.247	1.988	**0.273**	6.198	0.599	4.298	0.516	3.658	0.451
CasSampling	**2.194**	**0.227**	**2.113**	**0.243**	**1.883**	0.276	**5.908**	**0.594**	**4.125**	**0.512**	**3.433**	**0.447**

Table 3. Computation cost on Weibo dataset with 1h observation time.

Models	Time cost			Parameter
	Preprocessing	Trainning	Inference	
DeepHawkes	∼1 min	∼40 min	323 samples/s	∼103M
CasCN	∼3 h	∼2 h	158 samples/s	∼210M
CasFlow	∼28 min	∼15 min	1432 samples/s	∼11M
TempCas	∼6 min	∼13 min	1591 samples/s	∼12M
CasSampling	∼2 min	∼5 min	6328 samples/s	∼720K

5.4 Experiment Settings

Parameter Settings. For baselines, we follow the settings of their works. In our experiment, the maximum number of nodes in cascade graph K is set to 128. For local propagation embedding, CasSampling contains 2 layers of GAT, the hidden dimension feature is set to 512 and the output dimension of the node feature is set to 8. For the global propagation time flow, the number of time slots is set to 64. For the prediction layer, the number of neurons in each layer of the MLP is {64,32}. The optimizer is Adam with learning rate = 0.0001. We set the batch size as 64 and the training epochs as 50.

Experimental Environment. We ran the experiment on a PC with an AMD 5600X 3.70 Ghz, an NVIDIA GTX 3090 24 GB, and 64 GB memory. CasSampling was trained by using PyTorch 1.11.0.

6 Results and Analysis

In this section, we report experimental results and conduct further analysis.

6.1 Experiment Results

The experimental results are shown in Table 2. Our approach outperforms the baseline methods for all metrics. CasCN and DeepHawkes only focus on node-level modeling, which is inadequate for large graphs. CasFlow combines cascade graphs with a global social network for structure learning. However, the method above does not take into account the importance of time information, which may be the reason to limit their performance. TempCas implements a heuristic algorithm to sample critical paths, but compensates for structural losses using hand-crafted features without alignment with structural representation. Although it uses LSTM and attention CNN to model long-short term global propagation time information, the node-level time information is not integrated into the structure representation, which may result in poor integration between the structure embedding and the time-flow representation.

Our proposed CasSampling model beats all counterparts on all datasets. Compared with the classic models DeepHawkes and CasCN, our method has

Table 4. Effect of varying maximum number of node K on the performance of the model.

Model	Weibo Dataset								
	MSLE								
	0.5 h			1 h			2 h		
	K = 64	K = 128	K = 256	K = 64	K = 128	K = 256	K = 64	K = 128	K = 256
CasSampling-Struct	2.612	2.553	2.501	2.551	2.483	2.432	2.389	2.339	2.302
CasSampling	2.282	2.231	**2.228**	2.158	**2.113**	2.128	1.924	**1.883**	1.891

(a) Global graph (b) Sampled graph

Fig. 4. An example of graph sampling. The left is the original global graph, which has over 3000 nodes. The right is the sampled graph, with colored edges and points being selected (The max number of node K is 128). The depth of color and the size of nodes indicate the activity of nodes, which is derived from the degree of sampling nodes in the global graph.

improved by 10%–25% on each evaluation metrics. Compared with the recent SOTA models (CasFlow, TempCas), our method also has improved by 5%–15% on MSLE. This shows that our method is significantly better than the baseline in information popularity prediction.

We compute the time cost and parameter for baselines and CasSampling, as shown in Table 3. It demonstrates that CasSampling is more efficient compared with all the SOTA baselines.

Table 4 illustrates the impact of different maximum node number of K on the model's performance. As K increases, we observe a consistent improvement in the performance of CasSampling-Struct, since a larger number of nodes leads to more temporal information being captured. However, the overall performance of the model is best at $K = 128$, indicating that an appropriate value of K can facilitate better capturing of cascading effects.

6.2 Ablation Study

To study the relative importance of each module in the CasSampling, we conduct ablation studies over the different parts of the model as follows:

- **CasSampling-Struct**. It only uses the local-level propagation embedding part to predict increment size.
- **CasSampling-St.ND**. It only uses the local-level propagation embedding part and removes the outdegree feature of node.
- **CasSampling-TimeFlow**. It only uses global-level time flow representation to predict increment size.

– **CasSampling-NNT**. It removed the attention aggregator module of node time information.

The results are shown in Table 2. **CasSampling-Struct** demonstrates that the sampled graph can still represent global propagation. Figure 4 is an example of graph sampling. The performance of **CasSampling-St.ND** decreases considerably, which shows that the importance of maintaining the outdegree of the node plays a great role in compensating for the graph structure. The performance of **CasSampling-NNT** is not as good as **CasSampling**, which proves that adding time information to nodes can make local-level propagation embedding better integrated with global-level time flow representation. **CasSampling-TimeFlow** shows an interesting result that only time information is better than most models, so we did a further analysis to explore the underlying reason.

6.3 Further Analysis

We select samples based on the average path length of the graph to further analysis the time-flow based method and graph-embedding based method.

We link the performance with the graph structure. Figure 5 indicates that CasSampling-TimeFlow's performance slightly decreases with longer average path lengths in the cascade graph, while CasSampling-Struct performs better under these conditions. This suggests that CasSampling-Struct can learn the intrinsic information of complex cascades, while CasSampling-TimeFlow cannot.

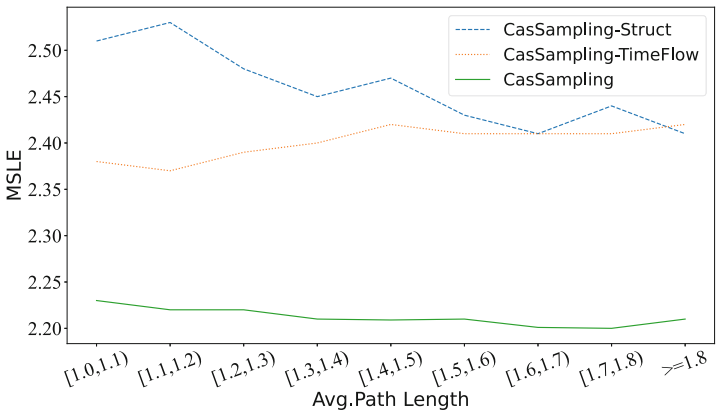


Fig. 5. The Relationship between performance and average path length of cascade graph.

7 Conclusion

We present CasSampling, a new deep learning framework for efficient popularity prediction. It captures cascade graph structure and leverages multi-scale time information. CasSampling consists of four main components: (1) a heuristic algorithm for sampling critical parts of a graph, (2) a local-level propagation embedding model that uses GAT and an attention aggregator to combine graph structure with node time information, (3) A global-level time flow representation using LSTM to capture propagation trends, and (4) a prediction layer that fusing local-level propagation and global-level time flow and feeds it into an MLP to predict the increment size. We conducted extensive experiments on Weibo and Twitter datasets, and achieved SOTA performance on information cascade size prediction with much less computation cost than the baselines.

Our future work mainly focuses on the following aspects: (1) Exploit a better strategy to sample graph. (2) Explore the relationship between structural information and temporal information, and better integrate each other.

Acknowledgments. The work was supported by National Natural Science Foundation of China (Grant Nos. 61966020, 61972186, U21B2027), Yunnan high-tech industry development project (Grant No. 201606), Yunnan provincial major science and technology special plan projects (Grant No. 202103AA080015, 202002AD080001-5), Yunnan Basic Research Project (Grant No. 202001AS070014), and Talents and Platform Program of Science and Technology of Yunnan (Grant No. 202105AC160018).

Ethical Statement. The purpose of this paper is to explore efficient and effective methods for learning cascade graphs for popularity prediction while adhering to academic integrity and research ethics requirements. We used publicly available data from social media datasets that have been authorized by Twitter and Weibo officials. To ensure the confidentiality of personal information, all data is anonymized and stored securely. We obtained approval and permission from the ethics committee of our institution to conduct this research.

The models and algorithms used in this study are based on publicly available data and previous research results, and we have thoroughly tested and verified them. We commit to conducting a transparent and fair evaluation of the algorithms and models used in this research, and we will present them fully in the paper.

Throughout this study, we will adhere to academic standards and ethical requirements, striving to avoid any behavior that may violate these requirements. We hope that this research will contribute to the development of cascade graph learning and popularity prediction, promoting further research in this area.

References

1. Mishra, S., Rizoiu, M.A., Xie, L.: Modeling popularity in asynchronous social media streams with recurrent neural networks. In: Twelfth International AAAI Conference on Web and Social Media (2018)

2. Li, G., Chen, S., Feng, J., Tan, K.l., Li, W.S.: Efficient location-aware influence maximization. In: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, pp. 87–98 (2014)
3. Cao, Q., Shen, H., Cen, K., Ouyang, W., Cheng, X.: Deephawkes: bridging the gap between prediction and understanding of information cascades. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 1149–1158 (2017)
4. Chen, X., Zhou, F., Zhang, K., Trajcevski, G., Zhong, T., Zhang, F.: Information diffusion prediction via recurrent cascades convolution. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE), pp. 770–781. IEEE (2019)
5. Tang, X., Liao, D., Huang, W., Xu, J., Zhu, L., Shen, M.: Fully exploiting cascade graphs for real-time forwarding prediction. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 582–590 (2021)
6. Yuan, C., Li, J., Zhou, W., Lu, Y., Zhang, X., Hu, S.: DyHGCN: a dynamic heterogeneous graph convolutional network to learn users' dynamic preferences for information diffusion prediction. In: Hutter, F., Kersting, K., Lijffijt, J., Valera, I. (eds.) ECML PKDD 2020. LNCS (LNAI), vol. 12459, pp. 347–363. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-67664-3_21
7. Cao, Q., Shen, H., Gao, J., Wei, B., Cheng, X.: Popularity prediction on social platforms with coupled graph neural networks. In: Proceedings of the 13th International Conference on Web Search and Data Mining, pp. 70–78 (2020)
8. Wu, Z., Zhou, J., Liu, L., Li, C., Gu, F.: Deep popularity prediction in multi-source cascade with HERI-GCN. In: 2022 IEEE 38th International Conference on Data Engineering (ICDE) (2022)
9. Weng, L., Menczer, F., Ahn, Y.Y.: Virality prediction and community structure in social networks. Sci. Rep. **3**(1), 1–6 (2013)
10. Cui, P., Jin, S., Yu, L., Wang, F., Zhu, W., Yang, S.: Cascading outbreak prediction in networks: a data-driven approach. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 901–909 (2013)
11. Ma, Z., Sun, A., Cong, G.: On predicting the popularity of newly emerging hashtags in twitter. J. Am. Soc. Inform. Sci. Technol. **64**(7), 1399–1410 (2013)
12. Petrovic, S., Osborne, M., Lavrenko, V.: Rt to win! predicting message propagation in twitter. In: Proceedings of the International AAAI Conference on Web and Social Media, vol. 5, pp. 586–589 (2011)
13. Shulman, B., Sharma, A., Cosley, D.: Predictability of popularity: gaps between prediction and understanding. In: Tenth International Conference on Web and Social Media (2016)
14. Bao, P., Shen, H.W., Huang, J., Cheng, X.Q.: Popularity prediction in microblogging network: a case study on Sina Weibo. In: Proceedings of the 22nd International Conference on World Wide Web, pp. 177–178 (2013)
15. Cheng, J., Adamic, L., Dow, P.A., Kleinberg, J.M., Leskovec, J.: Can cascades be predicted? In: Proceedings of the 23rd International Conference on World Wide Web, pp. 925–936 (2014)
16. Rizoiu, M.A., Xie, L., Sanner, S., Cebrian, M., Yu, H., Van Hentenryck, P.: Expecting to be hip: hawkes intensity processes for social media popularity. In: Proceedings of the 26th International Conference on World Wide Web, pp. 735–744 (2017)
17. Mishra, S., Rizoiu, M.A., Xie, L.: Feature driven and point process approaches for popularity prediction. In: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, pp. 1069–1078 (2016)

18. Shen, H., Wang, D., Song, C., Barabási, A.L.: Modeling and predicting popularity dynamics via reinforced poisson processes. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 28 (2014)
19. Yang, S.H., Zha, H.: Mixture of mutually exciting processes for viral diffusion. In: International Conference on Machine Learning, pp. 1–9. PMLR (2013)
20. Xu, X., Zhou, F., Zhang, K., Liu, S., Trajcevski, G.: Casflow: exploring hierarchical structures and propagation uncertainty for cascade prediction. IEEE Trans. Knowl. Data Eng. (2021)
21. Xu, Z., Qian, M., Huang, X., Meng, J.: CasGCN: predicting future cascade growth based on information diffusion graph. arXiv preprint arXiv:2009.05152 (2020)
22. Liao, D., Xu, J., Li, G., Huang, W., Liu, W., Li, J.: Popularity prediction on online articles with deep fusion of temporal process and content features. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 200–207 (2019)
23. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: International Conference on Learning Representations (2018, accepted as poster). https://openreview.net/forum?id=rJXMpikCZ
24. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
25. Chen, X., Zhang, F., Zhou, F., Bonsangue, M.: Multi-scale graph capsule with influence attention for information cascades prediction. Int. J. Intell. Syst. **37**(3), 2584–2611 (2022)