2019 IEEE 35th International Conference on Data Engineering (ICDE)

Information Diffusion Prediction via Recurrent Cascades Convolution

Xueqin Chen*, Fan Zhou*†, Kunpeng Zhang‡, Goce Trajcevski§, Ting Zhong*, Fengli Zhang*
*School of Information and Software Engineering, University of Electronic Science and Technology of China
‡Department of Decision, Operations & Information Technologies, University of Maryland, college park MD
§Department of Electrical and Computer Engineering, Iowa State University, Ames IA
†Corresponding author: fan.zhou@uestc.edu.cn

***Abstract*—Effectively predicting the size of an information cascade is critical for many applications spanning from identifying viral marketing and fake news to precise recommendation and online advertising. Traditional approaches either heavily depend on underlying diffusion models and are not optimized for popularity prediction, or use complicated hand-crafted features that cannot be easily generalized to different types of cascades. Recent generative approaches allow for understanding the spreading mechanisms, but with unsatisfactory prediction accuracy.**

To capture both the underlying structures governing the spread of information and inherent dependencies between re-tweeting behaviors of users, we propose a semi-supervised method, called Recurrent *Cas*cades *C*onvolutional *N*etworks (*CasCN*), which explicitly models and predicts cascades through learning the latent representation of both structural and temporal information, without involving any other features. In contrast to the existing single, undirected and stationary Graph Convolutional Networks (GCNs), CasCN is a novel multi-directional/dynamic GCN. Our experiments conducted on real-world datasets show that CasCN significantly improves the prediction accuracy and reduces the computational cost compared to state-of-the-art approaches.

***Index Terms*—information cascade, structural-temporal information, cascade size prediction, deep learning**

I. Introduction

Online social platforms allow their users to generate and share various contents and communicate on topics of mutual interest. Such activities facilitate fast diffusion of information and, consequently, spur the phenomenon of information cascades. The phenomenon is ubiquitous – i.e., it has been identified in various settings: paper citations [1], blogging space [2], [3], email forwarding [4], [5]; as well as in social sites (e.g., Sina Weibo [6] and Twitter [7], [8]). A body of research in various domains has focused on modeling cascades, with significant implications for a number of applications, such as marketing viral discrimination [9], influence maximization [10], [11], media advertising [12] and fake news detection [13]–[15]. Cascade prediction problem turns out to be of utmost importance since it enables controlling (or accelerating) information spreading in various scenarios.

The plethora of approaches proposed to tackle cascade prediction problem fall into four main categories: (1) *diffusion model-based approaches* [16]–[19], which characterize the diffusion process of information – but heavily depend on assumed underlying diffusion models and are not optimized for cascade prediction; (2) *feature-based approaches* – mostly focusing on identifying and incorporating complicated hand-crafted features, e.g., structural [20]–[22], content [23]–[26], temporal [27], [28], etc. Their performance strongly depends on extracted features requiring extensive domain knowledge, which is hard to be generalized to new domains; (3) *generative approaches* – typically relying on Hawkes point process [6], [29], [30], which models the intensity function of the arrival process for each message independently, enabling knowledge regarding the popularity dynamics of information – but with less desirable predictive power; and (4) *deep learning based methods*, especially Recurrent Neural Networks (RNN) based approaches [6], [8], [31], [32] – which automatically learn temporal characteristics but fall short in the intrinsic structural information of cascades, essential for cascade prediction [33].

Challenges and Our Approach: Effective and efficient prediction of the size of cascades has several challenges: (1) lack of knowledge of complete network structure through which the cascades propagate [34]. This impedes many global structure based approaches since obtaining or further embedding a complete graph is hard, if not impossible. (2) efficient representation of cascades – difficult due to their varying size (from very few to millions [33]), making the random walk based cascade sampling methods biased and ill-suited. (3) modeling diffusion dynamics of information cascades not only requires locally structural characteristics (e.g., community size and activity degree of users) but also needs some temporal characteristics – e.g., information within the first few hours plays crucial role in determining the cascades' size.

To address above challenges, we propose a novel framework called *CasCN* (Recurrent **Cas**cades **C**onvolutional **N**etworks) which, while relying on existing paradigms, incorporates both structural and temporal characteristics for predicting the future size of a given cascade. Specifically, *CasCN* samples sub-cascade graphs rather than a set of random-walk sequences from a cascade, and learns the local structures of each sub-cascades by graph convolutional operations. The convoluted spatial structures are then fed into a recurrent neural network for training and capturing the evolving process of a cascade structure. Our main contributions and advantages of *CasCN* are:

- *Use of less information*: We rely solely on the structural and

2375-026X/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDE.2019.00074

temporal information of cascades, avoiding massive and complex feature engineering, and our model is more generalizable to new domains. In addition, *CasCN* leverages deep learning to learn latent semantics of cascades in an end-to-end manner.
• *Representation of a cascade graph*: We sample a cascade graph as a sequence of sub-cascade graphs and use an adjacency matrix to represent each sub-cascade graph. This fully preserves the structural dynamics of cascades as well as the topological structure at each diffusion time, while eliminating the intensive computational cost when operating large graphs.
• *Additional impacting factors*: *CasCN* takes into account two additional crucial factors for estimating cascade size – the directionality of cascade graphs and the time of re-tweeting (e.g., decay effects).
• *Multi-cascade convolutional networks*: We propose a holistic approach, with variants capturing *spatial*, *structural*, and *directional* patterns in multiple sub-graphs, aware of temporal evolution of dynamic graphs – making our methodology readily applicable to other spatio-temporal data prediction tasks.
• *Evaluations on real-world datasets*: We conduct extensive evaluations on several publicly available benchmark datasets, demonstrating that *CasCN* significantly outperforms the state-of-the-art baselines.

Organization: In the rest of this paper, Section II reviews the related work, followed by Section III which formalizes the problem and introduces the preliminary background. In Section IV, we describe the main aspects of *CasCN* methodology in details. Experimental evaluations quantifying the benefits of our approach are presented in Section V and Section VI concludes the paper and outlines directions for future work.

II. Related Work

There exists a large body of related research on information cascade prediction and graph representation learning.

A. Information Cascades Modeling & Prediction

The works on information cascades modeling mainly focus on two levels: (1) At micro-level, local patterns of social influence are studied – e.g., inferring the action status of a user [31], [32]. The approaches predict the likelihood of a user propagating a particular piece of information, or forecast when the next propagation might occur given a certain information cascade [32]. (2) At macro-level, typical studies include cascade growth prediction [6], [8], [22], [23], [31] and outbreak prediction (above a certain threshold) [7], [22], [33], [35].

The methods on information cascades prediction fall into the following four categories:

Diffusion model-based approaches strongly assume that the underlying diffusion model is known as a prior. Typical examples include independent cascade model [16]–[19] and linear threshold model [17]. Specifically, latent influence and susceptibility (LIS) model to directly learn user-specific influence and susceptibility, and naturally capture context-dependent factors is proposed in. [18]. Information propagation via survival theory is described in [16], and [17] implements both independent cascade model and linear threshold model for information propagation, assuming a uniform or a degree-modulated propagation probability for a piece of information. While these methods gain success at characterizing the diffusion process of information, they heavily depend on the underlying diffusion model and are not quite appropriate for cascade prediction.

Generative process approaches focus on modeling the intensity function for each message arrival independently. Typically, they observe every event and learn parameters by maximizing the probability of events occurring during the observation time window. There exist two typical generative processes: (1) Poisson process – [1], [36], mainly modeling the stochastic popularity dynamics by employing Reinforced Poisson Process and incorporate it into the Bayesian framework for external factor inference and parameter estimation. (2) Hawkes process [6], [29], [30] – constructing predictors that combine Hawkes self-exciting point process for modeling each cascade and leverage feature-driven method for estimating the content virality, memory decay, and user influence (cf. [29]). These methods demonstrate an enhanced comprehensibility, but are unable to fully leverage the implicit information in the cascade dynamics for satisfactory prediction.

Feature-based approaches extract various hand-crafted features from raw data, typically including information content features [23]–[26], user characteristics [35], [37], [38], cascade's structural [20]–[22] and temporal features [27], [28] – and then feed them into discriminative machine learning algorithms to perform prediction. Combining content information with other types of features, e.g., temporal and structural features, can significantly reduce the prediction error [23] . Incorporating features related to early-adopters (cf. [37]) demonstrated that user features are informative predictors; and recent results comparing the prediction power of models using different sets of features (cf. [39]), found that temporal features have largest impact on prediction. However, [33] concluded that both temporal and structural features are almost equally effective in predicting cascade size. All these features heavily depend on domain knowledge. The non-existence of a standard and systematic way to design features, makes such methods hard to generalize. Moreover, the conclusions of existing works are sometimes contradictory, largely due to the heterogeneity among different types of social networks.

Deep learning based approaches are inspired by the recent successes of deep learning in many fields, and cascade prediction using deep neural networks has achieved significant performance improvements [6], [8], [31], [32]. The first deep learning based predictor of information cascades (DeepCas), presented [8], transforms the cascade graph as node sequences through random walk and automatically learns the representation of individual graphs. A deep learning based process that inherits the high interpretability of Hawkes process, with the high predictive power of deep learning methods was proposed in [6]. Coverage and attention mechanisms for capturing the cross-dependencies in cascades and alignments to better reflect the structural information was introduced in [32], whereas [31] employs topological RNNs to explore

the dynamic directed acyclic graph diffusion structure and tailor it for the task of node activation prediction. Overall, these approaches treat the cascade modeling as a sequence modeling problem using RNN – avoiding strong prior knowledge imposed by the diffusion models and feature design, while flexibly capturing sequential dependence in cascades. However, they lack good learning abilities in modeling structural information and dynamics in cascade, largely due to the biased cascade sampling methods and inefficient local structure embedding. Furthermore, the methods incur intensive computation overhead from the node sampling and subsequent embedding, especially for larger cascades.

B. Graph Convolutional Network

Graph convolutional networks (GCN [40]) learn a convolutional operation in Fourier domain by computing the eigen-decomposition of the Laplacian graph, bridging the spectral graph theory and deep neural networks on graph learning. An extension on fast localized convolutional filters on graphs is presented in [41] where the filters are approximated via a Chebyshev expansion of the Laplacian. GCNs model proposed in [42] simplifies previous works by restricting the filters in a first-order approximation of spectral graph convolution. Recently, a self-attention mechanism was introduced into GCN to reduce the dependency on graph structure [43]. Also, [44] propose FastGCN, which interprets graph convolutions as integral transform of embedding functions, under probability measures to solve the time and memory challenges for training large and dense graphs. Merging CNN and RNN for graph-structured data to identify dynamic patterns for modeling and predicting time-varying graph-based data was proposed in [45]. Authors extend GCN to model multivariate time series distributed on a network [46]. This is similar to recent work on sequential generalization of GCN [45], but the focus is on continuous time prediction and long-term forecasting via encoder-decoder architecture and scheduled sampling techniques.

While we rely on existing paradigms (combining graph convolutional mechanisms and RNN) to model and predict the cascades, our main differences are: (1) we use GCN to model the cascades prediction; (2) we propose a semi-supervised learning approach for capturing spatial dependence and temporal dynamics of information diffusion in an end-to-end manner; (3) we train RNN with a sequence of sub-graphs (via cascade Laplacian) rather than individual node embeddings, which better represents both global and local structures and improves the cascade prediction performance (cf. Section V).

III. Preliminaries

We now present the necessary background and formally define the problem, and introduce the preliminaries regarding structural and temporal modeling in information cascades.

A. Problem Definition

We cast the cascade size prediction as a regression problem aiming at predicting the size of information cascades, used on a social network to describe the process of information diffusion (e.g., specific tweets, rumors, etc.), where individuals can merely observe their immediate neighbors.

TABLE I
NOTATIONS USED THROUGHOUT THE PAPER

Symbol	Description
p_i	A message post, e.g., a tweet or a paper.
$C_i(t)$	A cascade graph regarding post p_i.
T	Observation time window.
$g_i^{t_j}, a_i^{t_j}$	A sub-cascade graph of $C_i(t)$ at diffusion time t_j and it's adjacency matrix.
G_i^T, A_i^T	A sequence of sub-cascade graphs of $C_i(t)$ and the corresponding adjacency matrices.
Δt	The fixed time interval.
ΔS_i	The increment size of p_i after Δt.
P_c	Transition matrix of a cascade.
ϕ, Φ	Stationary transition distribution and diagonalized ϕ
Δ_c	Laplacian of a cascade.
λ_{max}	The largest eigenvalue of Laplacian.
K	Maximum steps from the central node, i.e., K^{th}-order neighborhood or Chebyshev coefficients.

Definition 1. ***Cascade Graph.*** *Suppose we have n posts, $P = \{p_i, i \in [1, n]\}$. An evolving cascade graph for a given message post p_i, is a sequence $C_i(t) = [(U_i(t_1), E_i(t_1)), \ldots, (U_i(t_m), E_i(t_m))]$ where $U_i(t_j)$ is a set of nodes and $E_i(t_j) \subset U_i(t_j) \times U_i(t_j)$ is a set of edges, corresponding to the nodes and edges of $C_i(t)$ that are associated with p_i at time-instant t_j. Thus, the cascade graphs are evolving sequences of directed acyclic graphs (DAG).*

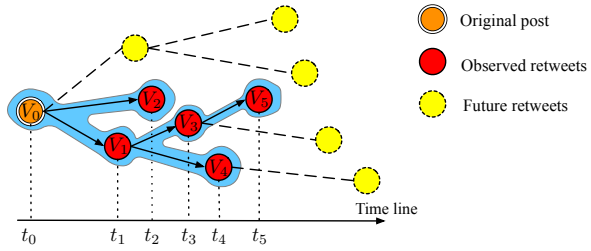


Fig. 1. The cascade graph of a post p_i. Node V_0 initiates the original message p_i.

We use $g_i^{t_j} = \left\{U_i^{t_j}, E_i^{t_j}, t_j\right\}$ as a shorthand for the snapshot of cascade graph $C_i(t)$ that reflecting the diffusion status of post p_i at t_j. For example: a node $x \in U_i^{t_j}$ represents a user who tweets or re-tweets the post p_i from some sources (e.g., other users) in Twitter or a paper in the citation networks; an edge $(x, y) \in E_i^{t_j}$ represents a relationship between x and y (e.g., re-tweet or citation); and t_j represents the time-instant when the re-tweeting or citation behavior occurs. Fig. 1 illustrates how the cascade graph can be represented as $g_i^{t_0} = \{(V_0), \{\varnothing\}, t_0\}, g_i^{t_1} = \{(V_0, V_1), \{(V_0, V_1)\}, t_1\}, \cdots, g_i^{t_5} = \{(V_0, V_1, V_2, V_3, V_4, V_5), \{(V_0, V_1), (V_0, V_2), (V_1, V_3), (V_1, V_4), (V_3, V_5)\}, t_5\}$.

Given a cascade graph $C_i(t)$ of a post p_i within an observation time window T, we can get different snapshots

$g_i^{t_j}$ of $C_i(t)$, which form a sequence of sub-cascade graphs G_i^T. In this paper, our task is to predict the increment size ΔS_i regarding the post p_i for a fixed time interval Δt, i.e., $\Delta S_i = \left|U_i^{T+\Delta t}\right| - \left|U_i^T\right|$.

Definition 2. *The cascade size predictor is a function $f(\cdot)$ that is to be learned, mapping G_i^T to ΔS_i:*

$$G_i^T = \left\{g_i^{t_0}, \cdots, g_i^{t_{m-1}}; t_j \in [0,T)\right\} \xrightarrow{f(\cdot)} \Delta S_i \qquad (1)$$

B. Recurrent Neural Network & Graph Convolution Networks

The common method to model temporal dependencies is to leverage recurrent neural networks (RNNs), typically including popular Long Short-Term Memory (LSTM) [47] and Gated Recurrent Units (GRU) [48]. In particular, LSTM has been proven stable and powerful for modeling long-range dependencies in various general-purpose sequence modeling tasks [49]–[51] and cascade prediction [8], [31].

We model the structural information of cascades using Defferrards graph convNet [41] – a popular Graph Convolutional Network (GCN) method that defines a spectral formulation in Fourier domain for the convolution operator on graphs $*\mathcal{G}$. More specifically, given a graph signal $x \in \mathbb{R}^{n \times d_x}$ (a graph signal can be a node or a graph), and the corresponding weighted adjacency matrix $W \in \mathbb{R}^{n \times n}$ and diagonal degree matrix $D \in \mathbb{R}^{n \times n}$ with $D_{ii} = \sum_j W_{ij}$, the normalized graph Laplacian is constructed as $L = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}}WD^{-\frac{1}{2}}$. The spectral convolutions on graphs are defined as:

$$y = g_\theta * \mathcal{G}x = g_\theta(L)\,x = g_\theta\left(U\Lambda U^T\right)x = U g_\theta(\Lambda)\,U^T x,$$

where $g_\theta = diag(\theta)$ is a filter parameterized by $\theta \in \mathbb{R}^n$ in the Fourier domain, $U = [u_0, u_1 \cdots, u_{n-1}] \in \mathbb{R}^{n \times n}$ is the matrix of eigenvectors of L, $\Lambda = [\lambda_0, \lambda_1, \cdots, \lambda_{n-1}] \in \mathbb{R}^{n \times n}$ is the diagonal matrix of eigenvalues of L. The Laplacian can be diagonalized as $L = U\Lambda U^T \in \mathbb{R}^{n \times n}$ with $U^T x$ is the graph Fourier transform of graph signal x. However, computing the eigen-decomposition of L in the first place might be prohibitively expensive for large graphs and the complexity of multiplication with U is $\mathcal{O}\left(n^2\right)$. Defferrard et al. [41] approximate $g_\theta(\Lambda)$ with a truncated expansion in terms of Chebyshev polynomials $T_k(x)$ up to K^{th} order:

$$g_\theta{}'(\Lambda) \approx \sum_{k=0}^{K} \theta_k' T_k\left(\widetilde{\Lambda}\right), \qquad (2)$$

with $\widetilde{\Lambda} = \frac{2}{\lambda_{max}}\Lambda - I_N$ (λ_{max} denotes the largest eigenvalue of L), identity matrix $I_N \in \mathbb{R}^{n \times n}$ and a vector of Chebyshev coefficients θ_k'. The Chebyshev polynomials of order K are recursively defined as $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$, with $T_0(x) = 1$ and $T_1(x) = x$. The graph filtering operation can now be written as:

$$y \approx g_\theta{}'(L)\,x \approx \sum_{k=0}^{K} \theta_k' T_k\left(\widetilde{L}\right)x, \qquad (3)$$

where $\widetilde{L} = \frac{2}{\lambda_{max}}L - I_N$. Note that as Eq.(3) is now an order K polynomial of the Laplacian, the complexity is reduced to $\mathcal{O}(K|\varepsilon|)$. We refer the readers to [41] for details and an in-depth discussion.

IV. CasCN: Model, Approach and Properties

Our deep learning framework *CasCN* takes a cascade graph $C_i(t)$ as an input and predicts the increment size ΔS_i regarding certain information (e.g., a post) p_i. *CasCN* leverages LSTM and GCN to fully extract temporal and structural information from the cascade graph. After an overview of *CasCN*, we focus on the details in the respective sub-sections.

An end-to-end type of framework, *CasCN* consists of three basic components, depicted in Fig. 2: (1) Cascade graph sampling: dynamically samples a sequence of sub-cascade graphs from the original cascade graph, and then represents sub-cascade graphs as a sequence of adjacency matrices; (2) Structural and temporal modeling: feeds the adjacency matrix sequences and the structural information of cascade graphs (i.e., the Laplacian matrices of cascade graphs) within an observation window into a neural network. It combines recurrent neural networks and graph convolutional networks with time decay function to learn the representation of cascades; (3) Prediction network: a Multi-Layer Perceptron (MLP) is used to predict the increment cascade size based on the representation learned from previous steps.

A. Cascade Graph as Sub-cascade Graph Sequences

Given a post p_i, the first step in *CasCN* is to initialize the representation of its corresponding cascading graph, $C_i(t)$. Existing methods typically treat the graph in two ways: either sampling the graph as a bag of nodes, which ignores both local and global structural information, or denoting the graph as a set of paths. For example, DeepCas [8] samples a set of paths from each cascade. The sampling process could be generalized as performing a random walk over a cascade graph similar to DeepWalk which, however, fails to consider dynamics of cascades – one of the most important factors in information diffusion. DeepHawkes [6] transforms the cascade graph into a set of diffusion paths according to the diffusion time, each of which depicts the process of information propagation between users within the observation time; however, this method ignores the structural information of cascade graphs.

Our approach samples the cascade graph $C_i(t)$ to obtain a sequence of sub-cascade graphs G_i^T which is used to represent cascades within the observation time T. G_i^T is denoted as:

$$G_i^T = \left\{g_i^{t_1}, g_i^{t_2}, \cdots, g_i^{t_{m-1}}\right\}, t_j \in [0,T)\ \ and\ j \in [1,m).$$

The sampling process is illustrated in Fig. 3, where each sub-cascade graph is represented by an adjacency matrix: the rows correspond to the alphabetical order of nodes' labels (top to bottom) and the columns correspond to edges, as illustrated above each instance of the adjacency matrix. The first sub-cascade graph in G_i^T only contains one single node because it is the generator of the post p_i, so we add a self-connection for this initiator. Thus, G_i^T is represented with a sequence of adjacency matrices $A_i^T = \left\{a_i^{t_1}, a_i^{t_2}, \cdots, a_i^{t_{m-1}}\right\}$.

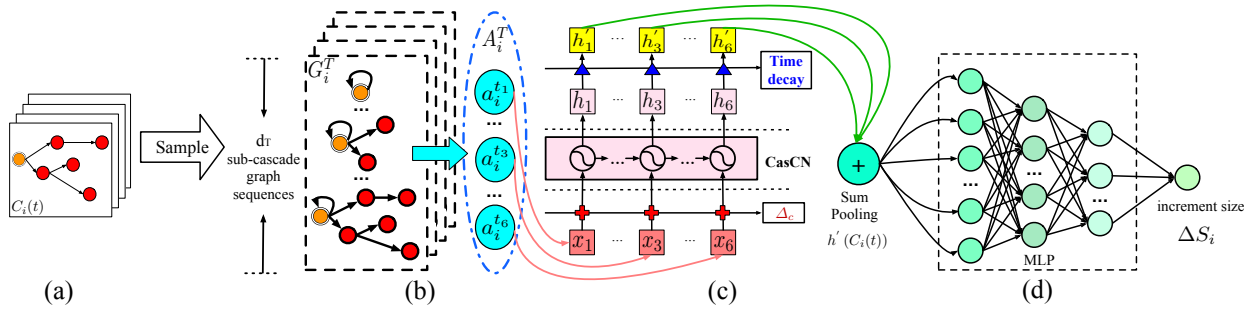


Fig. 2. Overview of *CasCN*: (a) The input is a cascade graph $C_i(t)$ for a given time window T and a certain post p_i. (b) We obtain a sequence of sub-cascade graphs from $C_i(t)$, and use an adjacency matrix to represent instances of sub-cascade graph $g_i^{t_j}$. Thus, we have $A_i^T = \{a_i^{t_1}, a_i^{t_2}, \cdots\}$, referred as signals. (c) We feed the signals and the Laplacian matrix Δ_c of cascade $C_i(t)$ into *CasCN*. The output h_t of *CasCN* is transformed to a new representation $h_t^{'}$ by multiplying a time decay factor. All $h_t^{'}$'s will be assembled via a sum pooling to new $C_i(t)$ representation: $h^{'}(C_i(t))$. (d) Finally, we use a MLP to predict the increment size of cascade (ΔS_i) for a fixed time interval Δt.

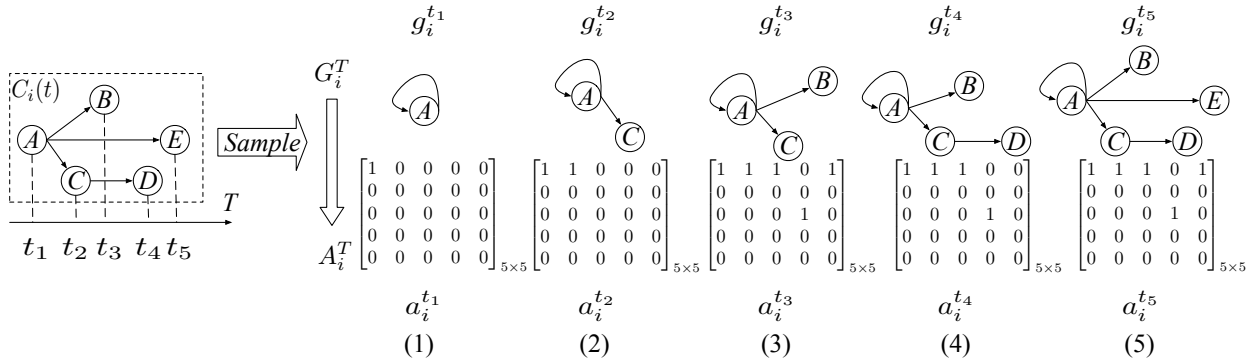


Fig. 3. Illustration of sampling and representation of sub-cascade graph sequence.

B. Laplacian Transformation of Cascades

CasCN is inspired by GCN [41] in the sense of using spectral graph theory to define a convolutional operator. However, classical GCN methods cannot be applied for cascades modeling since they focus on *fixed* and *undirected* graphs which, in turn, cannot consider the temporal information of cascade evolution. In contrast, cascade graphs in our problem are dynamic directed graphs (DAGs). As mentioned, the graph Laplacian for an undirected graph is a symmetric difference operator $L = D - W$, where D is the degree matrix and W is the weight matrix of the graph, which cannot be adapted in DAGs.

Recently, Li et al. [46] propose DCRNN to model the traffic flow as a diffusion process on a *fixed* DAG (a *directed* sensor graph), and define a diffusion convolution as:

$$y = g_\theta * \mathcal{G}x = \sum_{k=0}^{K} \left(\theta_{k,1} \left(D_O^{-1} W \right)^K + \theta_{k,2} \left(D_I^{-1} W^T \right)^K \right) x$$

where $-D^{-1}W$ is the random walk matrix used to replace Laplacian $\widetilde{L}$ in Eq.(3). This operation is actually a diffusion process convolution proposed by Atwood and Towsley [52] where the diffusion process is modeled as Markov process and may converge to a stationary distribution $\mathcal{P} \in \mathbb{R}^{n \times n}$ after many steps, and the i^{th} row $\mathcal{P}_{i,:} \in \mathbb{R}^n$ represents the likelihood of diffusion from node v_i.

In our settings, various cascades are different DAGs, all of which require incorporating special structure and direction information rather than a *single* and *fixed* sensor network in [46]. To overcome this challenge, we introduce Laplacian of cascade Δ_c, called *CasLaplacian*, for modeling the convolution operation over a single cascade signal X as:

$$y = g_\theta * \mathcal{G}X = \sum_{k=0}^{K} \theta_k T_k \left(\widetilde{\Delta}_c \right) X \tag{4}$$

where $\widetilde{\Delta}_c = \frac{2}{\lambda_{max}} \Delta_c - I_N$ is a scaled Laplacian.

Now we introduce the way of computing Laplacian of cascade Δ_c, which can capture special structural and directional characteristics of different cascades. For a *directed* graph, we define the normalized *directed* Laplacian as:

$$\mathcal{L} = I - \frac{\Phi^{\frac{1}{2}} P \Phi^{-\frac{1}{2}} + \Phi^{-\frac{1}{2}} P^T \Phi^{\frac{1}{2}}}{2}, \tag{5}$$

(cf. [53]) where P is a transition probability matrix, Φ is a diagonal matrix with entries $\Phi(v, v) = \phi(v)$, and $\phi = [\phi_i]_{1 \le i \le n}$ is the *column* vector of the stationary probabilities distribution of P.

However, such a symmetrical $\mathcal{L}$ can not capture the unique characteristic of the random walk on the different cascades. For example, given a cascade with transition probability matrix P_c, there exist cascades which have the same stationary distribution matrix $\mathcal{P}_c$, such that all these cascades have the same Laplacian matrix. To solve this problem, we relied on *Diplacian* [54] which computes Laplacian of DAGs as:

$$\Gamma = \Phi^{\frac{1}{2}} (I - P) \Phi^{-\frac{1}{2}} \tag{6}$$

where the transition probability matrix P was defined as $P = D^{-1}W$ with the hypothesis that the graph is strongly connected graphs (SCGs) [54]. In contrast, our cascade graphs are not SCGs. Thus, we define a transition probability matrix P_c of given cascade graph as:

$$P_c = (1 - \alpha) \frac{E}{n} + \alpha \left(D^{-1}W\right), \tag{7}$$

where $E \in \mathbb{R}^{n \times n}$ is an all–one matrix and $\alpha \in (0, 1)$ is an initial probability, used to restrict the state transition matrix $D^{-1}W$ to be a strongly connected matrix without 0 anywhere. Then the transition matrix P_c is irreducible, and has a unique stationary probability distribution $\{\phi_i | \phi_i > 0, 1 \leq i \leq n\}$. The stationary distribution vector $\{\phi_i\}$ can be obtained by solving an eigenvalue problem $\phi^T P_c = \phi^T$ subject to a normalized equation $\phi^T e = 1$, where $e \in \mathbb{R}^n$ is an all-one vector.

Finally, we can compute *CasLaplacian* as:

$$\Delta_c = \Phi^{\frac{1}{2}} (I - P_c) \Phi^{-\frac{1}{2}}. \tag{8}$$

Relationship with GCN: We now explain the relationship between our directed CasLaplacian and the normalized one in GCN, as well as the rationale behind CasLaplacian.

A random walk on *undirected* graph G is a Markov chain defined on G with the transition probability matrix $P = D^{-1}A$, and there exists a unique stationary distribution $\{\phi_1, \phi_2, \ldots, \phi_n\}$. Let $\phi = [\phi_i]_{1 \leq i \leq n}$ be the *column* vector of the stationary probabilities, where $\phi^T P = \phi^T$. Note that, as for undirected graph, the normalized Laplacian L can be transformed as:

$$L = D^{-\frac{1}{2}} (D - A) D^{-\frac{1}{2}} = D^{\frac{1}{2}} (I - P) D^{-\frac{1}{2}}. \tag{9}$$

Also, the ϕ of undirected graph can be caculated as

$$\phi_i = \frac{d_i}{\sum_k d_k} = \frac{d_i}{d}, i = 1, 2, \cdots, n, \tag{10}$$

and $\Phi^{\frac{1}{2}} = \sqrt{\text{diag}(\phi)}$, where $\phi = [\phi_1, \phi_2, \cdots, \phi_n]^T$, which can be used to approximate degree matrix D:

$$L = D^{\frac{1}{2}} (I - P) D^{-\frac{1}{2}} \approx \Phi^{\frac{1}{2}} (I - P) \Phi^{-\frac{1}{2}}. \tag{11}$$

Algorithm 1 formalizes the process of constructing the Laplacian of cascades.

Algorithm 1 Laplacian of cascade.

Input: A cascade graph C, initial probability α.
Output: *CasLaplacian*–Laplacian of cascade Δ_c.
1: Compute degree matrix D and weighted adjacency matrix W of a cascade graph C.
2: Compute transition probability matrix P_c of cascade graph according to Eq.(7) .
3: Solve the eigenvalue problem $\phi^T P_c = \phi^T$ subject to a normalized equation $\phi^T e = 1$ to get $\{\phi_i\}$.
4: $\Phi = \text{diag}(\phi)$.
5: Compute *CasLaplacian* Δ_c according to Eq.(8).

C. Structural and Temporal Modeling

We represent and model the cascade graph in a structural-temporal way. After obtaining the adjacency representation of sub-cascade graph sequence A_i^T and the Laplacian matrix Δ_c for each cascade graph, *CasCN* turns to learn the structural and temporal patterns via the combination of classical LSTM and GCN.

We leverage the RNNs to model the temporal dependence of diffusion – in particular, using the Long Short-Term Memory (LSTM) [47], which is a stable and powerful variant of RNNs. We replace the multiplications by dense matrices W with graph convolutions to incorporate the structural information:

$$\begin{aligned} i_t &= \sigma (W_i * \mathcal{G} X_t + U_i * \mathcal{G} h_{t-1} + V_i \odot c_{t-1} + b_i) \\ f_t &= \sigma (W_f * \mathcal{G} X_t + U_f * \mathcal{G} h_{t-1} + V_f \odot c_{t-1} + b_f) \\ o_t &= \sigma (W_o * \mathcal{G} X_t + U_o * \mathcal{G} h_{t-1} + V_o \odot c_t + b_o) \end{aligned} \tag{12}$$

where $*\mathcal{G}$ denotes the graph convolution defined in Eq.(4), signal X_t is the cascade graph sequences $A_i^T \in \mathbb{R}^{d_T \times n \times n}$, d_T denotes the number of diffusion time steps of post p_i. We leverage $W_i * \mathcal{G} X_t$ to mean a graph convolution of signal X_t with $d_h \times n$ filters which are functions of the graph Laplacian L parametrized by K Chebyshev coefficients. $\sigma(\cdot)$ is the logistic sigmoid function and i_t, f_t, o_t, b_* are respectively the input gate, forget gate, output gate and bias vector. The matrices $W \in \mathbb{R}^{K \times n \times d_h}$, $U \in \mathbb{R}^{K \times d_h \times d_h}$ and $V \in \mathbb{R}^{n \times d_h}$ are the different gate parameters, and n denotes the number of nodes in a cascade graph, and d_h is the size of cell states.

In particular, the memory cell c_t is updated by replacing the existing memory unit with a new cell c_t as:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh (W_c * \mathcal{G} X_t + U_c * \mathcal{G} h_{t-1} + b_c) \tag{13}$$

The hidden state is then updated by

$$h_t = o_t \odot \tanh (c_t) \tag{14}$$

where $\tanh(\cdot)$ refer to the hyperbolic tangent function, and $\odot$ is the entry-wise product.

D. Cascades Size Prediction

Previous works [29], [30] have shown the existence of time decay effect – i.e., that the influence of a node on other nodes will decrease over time. Various time decay functions have been defined: (1) power-low functions $\phi^t(T) = (T + c)^{-(1+\theta)}$; (2) exponential functions $\phi^e(T) = e^{-\theta T}$; (3) Reyleigh functions $\phi^T(T) = e^{-\frac{1}{2}\theta T^2}$. In practice, the choice of such function varies for different scenarios, e.g.,

exponential functions are suitable for financial data while Reyleigh functions perform better for epidemiology and power law functions are more applicable in geophysics and social networks [55], [56].

However, all the above time-decay functions have the limitation of parametric assumption which is greatly influenced by assumed prior distribution (and intuition). In this paper, we employ a non-parametric way to define the time decay function. More specifically, we assume that the time window of the observed cascade is $[0, T]$, and then split the time window into l disjoint time intervals $\{[t_0 = 0, t_1), [t_1, t_2), [t_2, t_3), \cdots, [t_{l-1}, t_l = T]\}$ to make the continuously time window into discrete. It not only allocates each diffusion time a corresponding interval, but also allows us to learn the discrete variable of time decay effect $\lambda = \{\lambda_m, m \in (1, 2, \cdots, l)\}$. Therefore, we define a function to compute the corresponding time interval m of time decay effect for a re-tweet at time t:

$$m = \lfloor \frac{(t - t_0)}{\lceil T/l \rceil} \rfloor \tag{15}$$

Where t_0 is the time of original post, l is the number of time interval, $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ are floor and ceiling operation.

For a cascade graph $C_i(t)$ regarding post p_i within the observation time window $[0, T]$, we get the hidden states $\{h_1, h_2, \cdots, h_T\}$ and we multiply a time decay effect λ_m for each hidden state to obtain $\left\{h_1^{'}, h_2^{'}, \cdots, h_T^{'}\right\}$ by:

$$h_t^{'} = \lambda_m h_t \tag{16}$$

summed up to get the representation vector for the cascade graph $C_i(t)$:

$$h^{'}\left(C_i(t)\right) = \sum_{t=1}^{d_T} h_t^{'} \tag{17}$$

The last component of *CasCN* is a multi-layer perceptron (MLP) with one final output unit. Given the representation $h^{'}\left(C_i(t)\right)$, we calculate the increment size ΔS_i as:

$$\Delta S_i = f\left(C_i(t)\right) = \text{MLP}\left(h^{'}\left(C_i(t)\right)\right) \tag{18}$$

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

$$\ell\left(\Delta S_i, \Delta \widetilde{S}_i\right) = \frac{1}{P} \sum_{i=1}^{P} \left(\log \Delta S_i - \log \Delta \widetilde{S}_i\right)^2 \tag{19}$$

where P is the number of posts, ΔS_i is the predicted incremental size for post p_i, and $\Delta \widetilde{S}_i$ is the ground truth.

The process of training *CasCN* is shown in Algorithm 2.

V. Experiments

In this section, we compare the performance of our proposed model *CasCN* with several state-of-the-art approaches that we use as baselines, and a few variants of *CasCN* itself, for cascade size prediction using two real-world datasets. To allow readers to reproduce our results, we make supplemental materials, implementation details and instructions available online at https://github.com/ChenNed/CasCN.

Algorithm 2 Learning with *CasCN*.

Input: sequences of adjacency matrices of cascade graphs $A = \{A_1^T, A_2^T, \cdots\}$ within an observation window T; Laplacian sequence for cascade graphs $L = \{L_1, L_2, \cdots\}$, batch size b.
Output: Increment sizes $\Delta S = \{S_1, S_2, \cdots\}$ of cascades.

```
repeat
    b = 1, 2, ...
    for adjacency matrix sequence A_i^T and corespoding Laplacian L_i in batch b do
        Compute the Structural and Temporal information h_t of cascade C_i(t) according to Eq.(12) - Eq.(14).
        Multiply each hidden state h_t with time decay effect λ_m to get h_t', according to Eq.(16).
        h'(C_i(t)) ← Aggregate({h_t', t ∈ [1, d_T]})
        Feed h'(C_i(t)) into MLP to compute increment size ΔS_i of cascade, according to Eq.(18)
        Use Adaptive moment estimation (Adam) to optimize the objective function in Eq.(19) and update parameters in Eq.(12), (13), (15)
    end for
until convergence;
```

TABLE II
Statistics of datasets

	Data sets	Sina Weibo			HEP-PH		
posts-papers	All	119,313			34,546		
edges	All	8,466,858			421,578		
T		1hour	2hours	3hours	3years	5years	7years
cascades	train	25,145	29,515	31,780	3,458	3467	3,478
	val	5,386	6,324	6,810	837	839	848
	test	5,386	6,324	6,810	837	839	848
Avg. nodes	train	28.58	29.30	29.48	5.27	5.27	5.27
	val	28.71	29.47	29.69	4.32	4.93	4.27
	test	29.11	29.77	30.21	4.91	4.27	4.28
Avg. edges	train	27.78	28.54	28.74	4.27	4.27	4.27
	val	27.91	28.70	28.94	3.31	3.93	3.95
	test	28.32	29.01	29.48	3.91	3.27	3.28

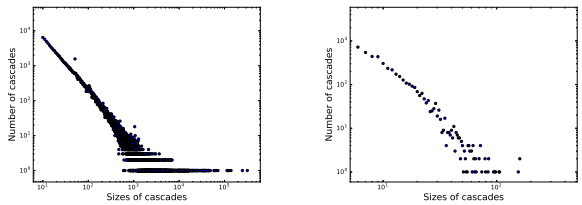
A. Datasets

We evaluate the effectiveness and generalizability of *CasCN* on two scenarios of information cascade prediction, and compare with previous works such as DeepCas and DeepHawkes – using publicly available datasets. The first one is to forecast the size of re-tweet cascades on Sina Weibo and the second one is to predict the citation count of papers in Citation dataset HEP-PH. The statistics of the datasets as shown in Table II.

• **Sina Weibo** (https://github.com/CaoQi92/DeepHawkes): The first dataset is Sina Weibo, a popular Chinese microblog platform, provided in [6] – which collects all original posts generated on June 1^{st}, 2016, and tracks all re-tweets of each post within the next 24 hours. It includes $119,313$ posts in total. Fig. 5(a) shows that the popularity of cascades saturates after 24 hours since publishing. Fig. 4(a) shows the distribution of cascade size (the number of re-tweets of each post). We follow similar experimental setup as in DeepHawkes [6] – i.e., the length T of the observation time window being $T = 1$ hour, 2 hours and 3 hours, and the cascades with the publication time before 8 am and after 6 pm being filtered out. Finally, we sort the cascades in terms of their publication time after preprocessing and choose the first 70% of cascades for

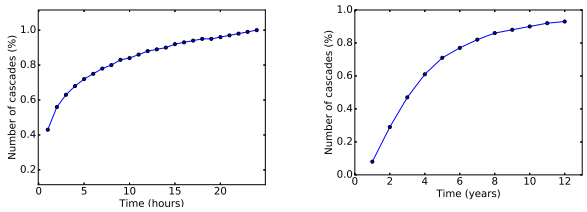
training and the rest for validation and testing via even split.

- **HEP-PH** (http://snap.stanford.edu/data/cit-HepPh.html): HEP-PH dataset is from the e-print arXiv and covers papers in the period from January 1993 to April 2003 (124 months). If a paper i cites paper j, the graph contains a directed edge from i to j. The data was originally released as a part of 2003 KDD Cup [57]. For the observation window, we choose $T = 3$, 4 and 5 years corresponding to the year that the popularity reaches about 50%, 60% and 70% of the final size, as shown in Fig. 5(b). Then, we pick up 70% of cascades for training and the rest for validation and testing via even split.



(a) Weibo dataset (b) HEP-PH

Fig. 4. Distribution of cascades size, the X axis is the size of cascades, and the Y axis is the number of cascades corresponding to the different sizes.



(a) Weibo dataset (b) HEP-PH

Fig. 5. Percentage distribution between time and the number of cascades

B. Baselines

In section II, we mentioned that existing relevant methods for information cascade prediction are mainly falling into four categories: (1) Diffusion Model-based approaches, (2) Generative process approaches, (3) Feature-based approaches, and (4) Deep learning-based approaches. Therefore, we select several methods in each group as baselines. For deep learning methods, we select three representative methods: DeepCas [8], DeepHawkes [6] and Topo-LSTM [31]. Note that DeepHawkes is also regarded as a successful implementation of Hawkes process – i.e., generative approaches. Furthermore, we include a network representation method to enrich our experiment – Node2Vec. The baselines and their implementation details are as follows:

Feature-based: Recent studies [29], [33], [35], [39] show that structural features, temporal features, and other features (e.g., content features and user features) are informative for information cascade prediction. In our implementations, we include all features mentioned above that could be generalized across datasets. These features include:

Structural features: We count the number of leaf nodes, the average degree (both in-degree and out-degree), average and max length of retweet path of cascades as measures of structural features.

Temporal features: We extract the time elapsed since the initial post for each retweet, the cumulative growth and incremental growth every 10 minutes for Sina Weibo and every 31 days for HEP-PH, for the reason that the time in Sina Weibo can be accurate to minutes, and the unit in HEP-PH is a day.

Other features: We use node ids as *node identity* feature.

After we extracting all the cascade features, we use two models, i.e., **Feature-linear** and **Feature-deep**, to perform information cascade prediction. The label (incremental size of cascade) has been logarithmically transformed before feeding into models, so that the baseline of feature-based methods optimizes the same loss function as CasCN.

- **Feature-linear**: We feed the features into a linear regression model with L_2 regularization, and the details of the L_2-coefficient setting can be found in Section V-E.
- **Feature-deep**: For fairness of comparison of the performance of the feature-based approaches with CasCN, we propose a strong baseline denoted as *Feature-deep*, which also uses a MLP model to predict the incremental size of cascade with hand-craft feature vectors.

LIS [18]: LIS is a diffusion model-based approach. This method models the cascade dynamics by learning two low-dimensional latent vectors for messages from observed cascades to capture their influence and susceptibility respectively.

Node2Vec [58]: Node2Vec is selected as a representative of node embedding methods, and can be replaced with any other embedding methods, e.g., DeepWalk [59] and LINE [60]. We conduct random walks from cascade graphs and generate embedding vectors for each node. Next, the embeddings of all nodes in a cascade graph are fed into MLP to make predictions.

DeepCas [8]: The first deep learning architecture for information cascades prediction, which represents a cascade graph as a set of random walk paths and piped through bi-directional GRU neural network with an attention mechanism to predict the size of the cascade. It mainly utilizes the information of structure and node identities for prediction.

DeepHawkes [6]: DeepHawkes model integrates the high prediction power of end-to-end deep learning into interpretable factors of Hawkes process for popularity prediction. The marriage between deep learning technique and a well-established interpretable process for modeling cascade dynamics bridges the gap between prediction and understanding of information cascades. This method belongs to both generative approaches and deep learning-based approaches.

Topo-LSTM [31]: A novel topological recurrent neural network which is a directed acyclic graph-structured (DAG-structured) RNN takes dynamic DAGs as inputs and generates a topology-aware embedding for each node in the DAGs as outputs. The original application of Topo-LSTM is to predict node activations. We replace the logistic classifier in Topo-

LSTM with a diffusion size regressor to predict the size of cascades.

C. Variants of CasCN

In addition to comparison with existing baselines, we also derive a few variants of CasCN:

CasCN-GL: CasCN-GL replaces the structural-temporal modeling component of CasCN with the combination of GCN and LSTM for modeling structural and temporal patterns, respectively.

CasCN-GRU: This method replaces the LSTM of CasCN with GRU. Similar to LSTM, CasCN with GRU models structural-temporal information using extra gating units, but without separated memory cells. Formally, we update the state of h_t by a linear interpolation between the last state h_{t-1} and the candidate state $\widetilde{h_t}$.

CasCN-Path: In CasCN-Path, we use random walks to represent a cascade graph (shown in Fig. 6) rather than sub-cascade graphs used in CasCN. Therefore, we first embed users into a 50-dimensional space to represent the latent (re-tweeting) relationships among users in a cascade graph. Next, we use random walks to sample sufficient number of sequences for all cascade graphs. Finally, we feed them to CasCN and predict the size of information cascades.

CasCN-Undierected: In CasCN-Undierected, we regard the cascade graphs as undirected graphs and calculate the normalized Laplacian according to $L = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$.

CasCN-Time: In CasCN-Time, we do not consider the time decay effect of re-tweeting.

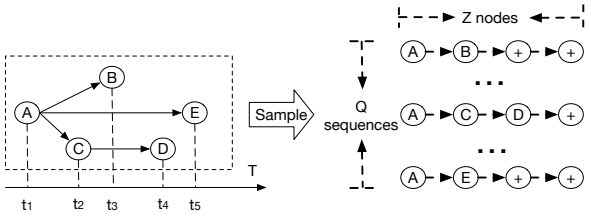


Fig. 6. Sampling the cascade graph as random walks.

D. Evaluation Metric

Following the existing works, we choose standard evaluation metrics – *MSLE* (mean square log-transformed error) in our experiments [6], [8], [31]. Note that the smaller *MSLE*, the better the prediction performance. Specifically, *MSLE* is the metric for evaluating the linking accuracy, defined as:

$$MSLE = \frac{1}{P}\sum_{i=1}^{P}\left(\log \Delta S_i - \log \Delta \widetilde{S}_i\right) \tag{20}$$

where P is the total number of posts, ΔS_i is the predicted incremental size for post p_i, and $\Delta \widetilde{S}_i$ is the ground truth.

E. Hyper-parameter

Models mentioned above involve several hyper-parameters. For example, L_2 coefficient in **Feature-linear** are chosen from $\{1, 0.5, 0.1, 0.05, ..., 10^{-8}\}$. For **Feature-deep**, parameters are similar to deep learning-based approaches.

For **LIS**, we follow the work in [18], the maximum epoch M is 1×10^5. We use random values to initialize regularization parameters γ_I and γ_S.

For **Node2Vec**, we follow the work in [58], i.e., parameters p and q are selected from $\{0.25, 0.50, 1, 2, 4\}$, the length of walk is chosen from $\{10, 25, 50, 75, 100\}$, and the number of walks per node varies from $\{5, 10, 15, 20\}$.

For **DeepCas**, **DeepHawkes** and **Topo-LSTM**, we follow the setting of DeepCas [8], where the embedding dimensionality of users is 50, the hidden layer of each GRU has 32 units and the hidden dimensions of the two-layer MLP are 32 and 16, respectively. The learning rate for user embeddings is 5×10^{-4} and the learning rate for other variables is 5×10^{-3}. The batch size for each iteration is 32 and the training process will stop when the loss of validation set does not decline for 10 consecutive iterations. The time interval of DeepHawkes is set to 10 minutes for Sina Weibo and 2 months for HEP-PH. For ***CasCN***, the basic parameters (e.g., learning rate and batch size, etc.) are the same as above deep learning-based approaches, except that we choose the support $K = 2$ of GCN and calculate the max eigenvalue λ_{max} of cascade Laplacian.

TABLE III
OVERALL PERFORMANCE COMPARISON OF INFORMATION CASCADES PREDICTION AMONG DIFFERENT APPROACHES. M: MODEL; T: OBSERVATION TIME WINDOW.

Datasets	Weibo Dataset			HEP-PH		
Metric	MSLE					
M \ T	1 hour	2 hours	3 hours	3 years	5 years	7 years
Features-deep	3.68	3.361	3.296	1.893	1.623	1.619
Features-linear	3.501	3.435	3.324	1.715	1.522	1.471
LIS	3.731	3.621	3.457	2.144	1.798	1.787
Node2Vec	3.795	3.523	3.513	2.479	2.157	2.096
DeepCas	2.958	2.689	2.647	1.765	1.538	1.462
Topo-LSTM	2.772	2.643	2.423	1.684	1.653	1.573
Deep-Hawkes	2.441	2.287	2.252	1.581	1.47	1.233
CasCN	**2.242**	**2.036**	**1.91**	**1.353**	**1.164**	**0.851**

TABLE IV
PERFORMANCE COMPARISON BETWEEN CASCN AND ITS VARIANTS. M: MODEL; T: OBSERVATION TIME WINDOW.

Datasets	Weibo Dataset			HEP-PH		
Metric	MSLE					
M \ T	1 hour	2 hours	3 hours	3 years	5 years	7 years
CasCN	**2.242**	**2.036**	**1.916**	1.35	**1.164**	**0.851**
CasCN-GRU	2.288	2.052	1.965	**1.347**	1.166	0.874
CasCN-Path	2.557	2.483	2.404	1.664	1.437	1.332
CasCN-GL	2.312	2.028	1.942	1.364	1.357	1.302
CasCN-Undierected	2.309	2.132	1.978	1.562	1.425	1.118
CasCN-Time	2.652	2.547	2.363	1.732	1.512	1.451

F. Performance comparison

We first report the performance of various methods on cascade size prediction. The results are illustrated in Table III. Then we compare the performance of *CasCN* with a few variants, and the results are shown in Table IV. Lastly, we do some analyses with different parameter settings of *CasCN*,

and the results can be found in Table V. The following sections will describe these empirical observations in details.

CasCN vs. Baselines: Table III summarizes the performance comparison among *CasCN* and baselines on both Sina Weibo dataset and HEP-PH dataset. In general, the proposed *CasCN* model performs relatively well on information cascade prediction for both datasets (post re-tweeting and paper citing). It outperforms traditional approaches, e.g., feature-based and generative approaches, as well as superior to the state-of-the-art deep learning methods, with a statistically significant drop of MSLE. Now we step into the details of comparison:

The performance gap between these Feature-deep and Feature-linear is quite small meaning that if we have a set of representative features of information cascades, deep learning does not always perform better than traditional predicting methods. However, as discussed earlier, the performance of such methods heavily depend on hand-crafted features which are difficult to select for different scenarios in practice.

For embedding methods, Node2Vec [58] does not perform well. Through the comparison with DeepCas [8], it proves that only taking the node embeddings as the graph representation is not enough and is not comparable with representing the graph as a set of random paths.

DeepCas, as the first proposed end-to-end deep learning method for cascade size prediction, exhibits better performance than feature-based approaches and traditional generative process-based approaches. But it still way worse than other deep learning based approaches, because of failing to consider temporal information and the topological structure of cascade graphs. The latest method Topo-LSTM also lacks time feature, so that it performs a little bit worse than DeepHawkes and our model. DeepHawkes, while successful in modeling cascades in a deep generative way, it does not perform the best due to its weak ability to learn structural information.

Finally, our proposed *CasCN* model, which purely relies on and fully explores structural and temporal information, significantly outperforms all baselines. For example, it achieves excellent prediction results with MSLE = 1.916 when observing for 3 hours in Sina Weibo and MSLE = 0.851 when observing for 7 years in HEP-PH, respectively. It reduces the prediction error by 15.2% and 30.9% comparing to the second best DeepHawkes.

When comparing methods with different observation window T, we clearly see a general pattern that the larger the T, the easier to make a good prediction. It is mainly because of the fact that longer T reveals more information for prediction.

CasCN with Variants: To investigate and demonstrate the effectiveness of each component of our model (e.g., to understand the effect of the sampling part of *CasCN*), we present five variants of *CasCN*, where all are built upon the original *CasCN* model with some components changed. Their details can be found in Section V-C.

The experimental results are shown in Table IV, from which we can see that original *CasCN* leads to a certain reduction of prediction error when compared with other variants. From the comparison to CasCN-Undierected and CasCN-Time, we find that directionality and time decay effect are proved to be indispensable for cascade size prediction. Similarly, *CasCN*-Path brings a remarkable decrease of the prediction performance, which tells the necessity and effectiveness of sampling in *CasCN*. This indicates that the way to sample cascade graph as sub-cascade graph sequence can better reflect the dynamics of the cascade structure and the topological structure of each diffusion time.

In summary, sub-graphs sampling and structural-temporal modeling are critical components in *CasCN*, both of which essentially improve the performance of information cascade prediction as presented in the results.

TABLE V
ANALYSIS OF PARAMETER IMPACT ON PERFORMANCE.

Dataset	Weibo Dataset		
Metric	MSLE		
Parameter \ T	1 hour	2 hours	3 hours
K=1	2.284	2.061	1.932
K=2	**2.242**	**2.036**	**1.91**
K=3	2.312	2.078	1.9386
$\lambda_{max} \approx 2$	2.418	2.217	2.046
$\lambda_{max} =$ real	**2.242**	**2.036**	**1.91**

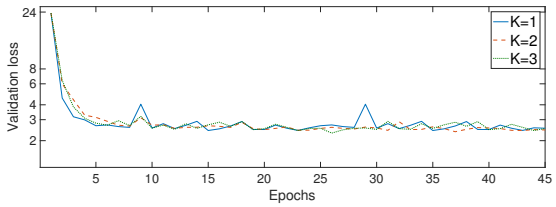
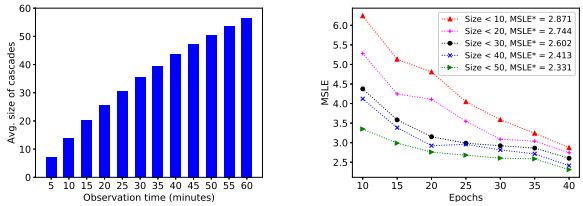


Fig. 7. Loss of CasCN on the validation set with varying K

Parameter analysis in CasCN: We now turn to investigating whether the parameters of *CasCN* have impact on the performance of cascade size prediction. Results are summarized in Table V. We consider two vital parameters of graph convolutional kernel, i.e., Chebyshev coefficients K and the largest eigenvalue λ_{max} of Laplacian matrix Δ_c. For Chebyshev coefficients K we selected from $\{1, 2, 3\}$. To obtain λ_{max}, we have two ways: the first is to approximate it as $\lambda_{max} \approx 2$, and the second is to compute the exact value of λ_{max} for each cascade graph. Table V shows that *CasCN* with $K = 2$ can achieve better performance than K=1 and 3. And in Fig. 7, the validation loss in each epoch steadily declines. There is no evidence showing that a larger or smaller K is better than a median one. Further, bigger K will increase the computational cost. For the value of λ_{max}, precise values can lead to better prediction results.

We also investigated the performance of *CasCN* when the observed cascades are small – e.g., the size of the cascades is within 10, 20. Fig. 8(a) gives the statistics of Weibo dataset illustrating the average cascade size increasing with time. Fig. 8(b) shows that the MSLE results for various size de-



(a) Avg. cascade size distribution. (b) Results.

Fig. 8. Impact of smaller-size observations.

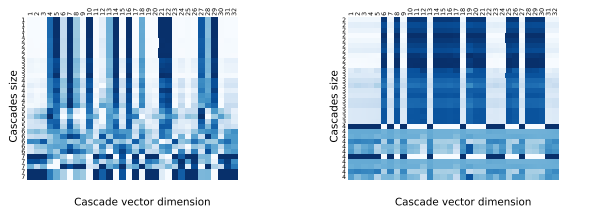
crease with training epochs. Apparently, the larger of observed cascade size, the lower MSLE value *CasCN* achieves.

Discussions on feature learning: Finally, we discuss and demonstrate the capability of *CasCN* on feature learning in a visual way. We use the latent representation of each cascade graph $C_i(t) : h^{'}(C_i(t))$ to plot a heatmap (as shown in Fig. 9). The value in each dimension corresponds to some implicit or explicit features related to predicting the cascade size. Fig. 9 tells us that the latent representation varies with cascade size. And surprisingly, there exists a clear pattern separation between outbreak (larger cascades) and non-outbreak (smaller) cascades, which indicates that *CasCN* is able to learn a good latent representation of cascades with different sizes and thus can be applied for outbreaking prediction.

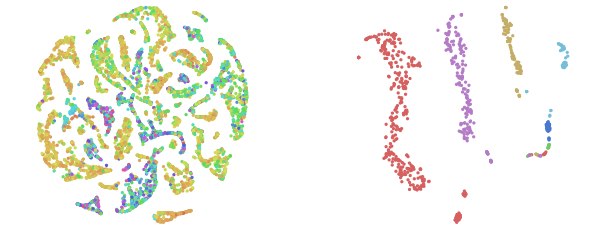
Next, we try to understand/interpret the importance of some hand-crafted features in cascade size prediction. First, we use t-SNE [61] to project the vector representation summarized in $h^{'}(C_i(t))$ for the cascade graph $C_i(t)$ to one data point in a 2-D space. Note that cascade graphs with similar vector representations are placed closely. Second, we color each data point (transformed from a cascade graph) using different strategies, such as based on the value of a certain feature f (e.g., number of leaf nodes, mean time, etc.), or the true increment size (the ground-truth label). The distribution of colors suggests a connection between the learned representations and network properties. That is, if a colored plot based on a certain feature f is well correlated with that of the true increment size, this feature is positively useful for cascade size prediction. Take the Weibo dataset as an example: Fig.9(c) and 9(e) have similar color distributions with the true increment size 9(g) – we believe that leaf nodes and mean time are two important features for cascade size prediction.

VI. Conclusions and Future Work

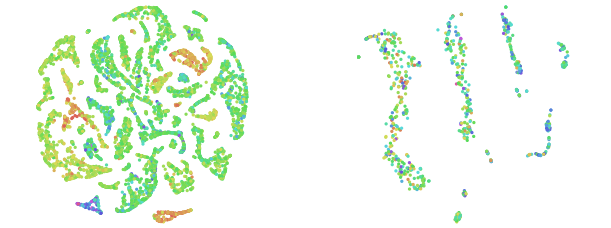
We proposed a novel deep learning based framework – *CasCN* – an end-to-end modeling framework for cascade growth prediction that does not rely heavily on feature engineering and can be easily generalized; and enabling the information cascade prediction by exploiting both structural and temporal information. The *CasCN* model can learn a better latent representation for cascade graph with less information, using structural and temporal information of cascades within a deep learning framework. Incorporating the directionality of cascades and time decay effect further improves the prediction performance. Our experiments conducted on two scenarios,



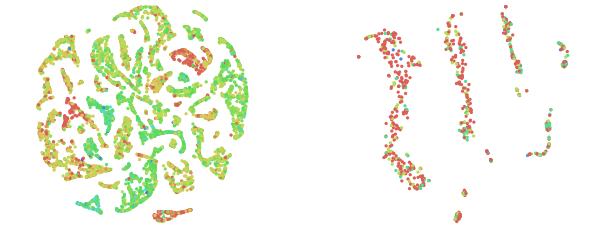
(a) Heatmap of Weibo (b) Heatmap of HEP-PH



(c) f: Leaf nodes plot for Weibo (d) f: Leaf nodes plot for HEP-PH



(e) f: Mean time plot for Weibo (f) f: Mean time plot for HEP-PH



(g) Increment size plot for Weibo (h) Increment size plot for HEP-PH

Fig. 9. Feature visualization. Figure (a) and (b) are learned representations by *CasCN*. In (c) - (h), we layout the cascade graphs as data points in the test set to a 2-D space using t-SNE. Every layout is colored using hand-crafted network properties or the ground-truth (captioned "f: feature").

i.e., forecasting the size of re-tweeting cascades in Sina Weibo and predicting the citation of papers in HEP-PH, demonstrate that *CasCN* outperforms various state-of-the-art methods.

Our future work focuses on three tasks: (1) introducing more sophisticated training techniques (e.g., attention mechanism or batch training), to transform *CasCN* to inductive model and model larger graphs; (2) efficient incorporation of updates into LSTM; and (3) investigating the potential coupling CasCN with model-based approaches to improve the accuracy.

ACKNOWLEDGEMENTS

Work supported by National Natural Science Foundation of China (Grants No.61602097 and No.61472064), NSF Grants III 1213038 and CNS 1646107, ONR grant N00014-14-10215, and the Fundamental Research Funds for the Central Universities (No.ZYGX2015J072).

REFERENCES

[1] H.-W. Shen, D. Wang, C. Song, and A.-L. Barabási, "Modeling and predicting popularity dynamics via reinforced poisson processes." in *AAAI*, 2014.
[2] D. Gruhl, R. Guha, D. Liben-Nowell, and A. Tomkins, "Information diffusion through blogspace," in *WWW*, 2004.
[3] J. Leskovec, M. McGlohon, C. Faloutsos, N. Glance, and M. Hurst, "Cascading behavior in large blog graphs," in *ICDM*, 2007.
[4] D. Liben-Nowell and J. Kleinberg, "Tracing information flow on a global scale using internet chain-letter data," *PNAS*, vol. 105, no. 12, 2008.
[5] B. Golub and M. O. Jackson, "Using selection bias to explain the observed structure of internet diffusions," *PNAS*, vol. 107, no. 24, 2010.
[6] 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.
[7] M. Jenders, G. Kasneci, and F. Naumann, "Analyzing and predicting viral tweets," in *WWW*, 2013.
[8] C. Li, J. Ma, X. Guo, and Q. Mei, "Deepcas: An end-to-end predictor of information cascades," in *WWW*, 2017.
[9] J. Leskovec, L. A. Adamic, and B. A. Huberman, "The dynamics of viral marketing," *TWEB*, vol. 1, no. 1, 2007.
[10] J. Tang, X. Tang, X. Xiao, and J. Yuan, "Online processing algorithms for influence maximization," in *SIGMOD*, 2018.
[11] K. Huang, S. Wang, G. Bevilacqua, X. Xiao, and L. V. Lakshmanan, "Revisiting the stop-and-stare algorithms for influence maximization," in *VLDB*, 2017.
[12] H. Li, X. Ma, F. Wang, J. Liu, and K. Xu, "On popularity prediction of videos shared in online social networks," in *CIKM*, 2013.
[13] S. Vosoughi, D. Roy, and S. Aral, "The spread of true and false news online," *Science*, vol. 359, no. 6380, 2018.
[14] C. Song, W. Hsu, and M. Lee, "Temporal influence blocking: Minimizing the effect of misinformation in social networks," in *ICDE*, 2017.
[15] Y. Liu and Y. B. Wu, "Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks," in *AAAI*, 2018.
[16] M. Gomez-Rodriguez, J. Leskovec, and B. Schölkopf, "Modeling information propagation with survival theory," in *ICML*, 2013.
[17] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *SIGKDD*, 2003.
[18] Y. Wang, H. Shen, S. Liu, and X. Cheng, "Learning user-specific latent influence and susceptibility from information cascades," in *AAAI*, 2015.
[19] N. Ohsaka, T. Sonobe, S. Fujita, and K. Kawarabayashi, "Coarsening massive influence networks for scalable diffusion analysis," in *SIGMOD*, 2017.
[20] D. M. Romero, C. Tan, and J. Ugander, "On the interplay between social and topical structure," in *AAAI*, 2013.
[21] 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.
[22] L. Weng, F. Menczer, and Y. Y. Ahn, "Predicting successful memes using network and community structure," in *AAAI*, 2014.
[23] O. Tsur and A. Rappoport, "What's in a hashtag?: content based prediction of the spread of ideas in microblogging communities," in *WSDM*, 2012.
[24] S. Petrovic, M. Osborne, and V. Lavrenko, "Rt to win! predicting message propagation in twitter," in *AAAI*, 2011.
[25] Z. Ma, A. Sun, and G. Cong, "On predicting the popularity of newly emerging hashtags in twitter," *JASIST*, vol. 64, no. 7, 2013.
[26] L. Hong, O. Dan, and B. D. Davison, "Predicting popular messages in twitter," in *WWW*, 2011.
[27] G. Szabo and B. A. Huberman, "Predicting the popularity of online content," *Communications of the Acm*, vol. 53, no. 8, 2008.
[28] H. Pinto and J. M. Almeida, "Using early view patterns to predict the popularity of youtube videos," in *WSDM*, 2013.
[29] S. Mishra, M. A. Rizoiu, and L. Xie, "Feature driven and point process approaches for popularity prediction," in *CIKM*, 2016.
[30] S. Gao, J. Ma, and Z. Chen, "Modeling and predicting retweeting dynamics on microblogging platforms," in *WSDM*, 2015.
[31] J. Wang, V. W. Zheng, Z. Liu, and C. C. Chang, "Topological recurrent neural network for diffusion prediction," in *ICDM*, 2017.
[32] Y. Wang, H. Shen, S. Liu, J. Gao, X. Cheng, Y. Wang, H. Shen, S. Liu, J. Gao, and X. Cheng, "Cascade dynamics modeling with attention-based recurrent neural network," in *IJCAI*, 2017.
[33] J. Cheng, L. Adamic, P. A. Dow, J. M. Kleinberg, and J. Leskovec, "Can cascades be predicted?" in *WWW*, 2014.
[34] K. Wegrzycki, P. Sankowski, A. Pacuk, and P. Wygocki, "Why Do Cascade Sizes Follow a Power-Law?" in *WWW*, 2017.
[35] P. Cui, S. Jin, L. Yu, F. Wang, W. Zhu, and S. Yang, "Cascading outbreak prediction in networks:a data-driven approach," in *SIGKDD*, 2013.
[36] J. Gao, H. Shen, S. Liu, and X. Cheng, "Modeling and predicting retweeting dynamics via a mixture process," in *WWW*, 2016.
[37] E. Bakshy, J. M. Hofman, W. A. Mason, and D. J. Watts, "Everyone's an influencer: quantifying influence on twitter," in *WSDM*, 2011.
[38] K. Lerman and A. Galstyan, "Analysis of social voting patterns on digg," in *WOSN*, 2008.
[39] B. Shulman, A. Sharma, and D. Cosley, "Predictability of popularity: Gaps between prediction and understanding." in *ICWSM*, 2016.
[40] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *ICML*, 2014.
[41] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *NIPS*, 2016.
[42] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.
[43] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.
[44] J. Chen, T. Ma, and C. Xiao, "Fastgcn: Fast learning with graph convolutional networks via importance sampling," 2018.
[45] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," *arXiv preprint arXiv:1612.07659*, 2016.
[46] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *ICLR*, 2018.
[47] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, 1997.
[48] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *NIPS*, 2014.
[49] A. Graves, "Generating sequences with recurrent neural networks," *Computer Science*, 2013.
[50] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using lstms," in *ICML*, 2015.
[51] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *NIPS*, 2014.
[52] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *NIPS*, 2016.
[53] C. Fan, "Laplacians and the cheeger inequality for directed graphs," *Annals of Combinatorics*, vol. 9, no. 1, 2005.
[54] Y. Li and Z.-L. Zhang, "Digraph laplacian and the degree of asymmetry," *Internet Mathematics*, vol. 8, no. 4, 2012.
[55] D. Sornette, "Apparent criticality and calibration issues in the hawkes self-excited point process model: application to high-frequency financial data," *Social Science Electronic Publishing*, no. 8, 2013.
[56] J. Wallinga and P. Teunis, "Different epidemic curves for severe acute respiratory syndrome reveal similar impacts of control measures." *American Journal of Epidemiology*, vol. 160, no. 6, 2004.
[57] J. Gehrke, P. Ginsparg, and J. Kleinberg, "Overview of the 2003 kdd cup," *Acm SIGKDD Explorations Newsletter*, vol. 5, no. 2, 2003.
[58] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *SIGKDD*, 2016.
[59] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *SIGKDD*, 2014.
[60] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *WWW*, 2015.
[61] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, 2008.