Information diffusion analysis in online social networks based on deep representation learning

Chen, X.

Part I

Information Cascades Modeling

Chapter 4

Learning Structural-temporal Features for Cascades Modeling

4.1 Chapter Overview

As described in Chapter 2, the plethora of approaches proposed to tackle the cascade prediction problem fall into three main categories:

1. *handcrafted feature-based approaches* – mostly focusing on identifying and incorporating complicated hand-crafted features, e.g., structural [27, 37, 54], content [22, 38, 51, 52], temporal [28, 55], etc. Their performance strongly depends on extracted features requiring extensive domain knowledge, which is hard to be generalized to new domains;

2. *point process-based approaches* – typically relying on Hawkes point process [9, 30, 49], 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

3. *deep learning-based methods*, especially Recurrent Neural Networks (RNN) based approaches [8, 9, 35, 36] – which automatically learn temporal characteristics but fall short in the intrinsic structural information of cascades, essential for cascade prediction [1].

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 [114]. This impedes many global structure based approaches since obtaining a complete graph or further embedding into it is hard, if not impossible. (2) efficient representation of cascades – difficult due to their varying size (from very few to millions [1]), 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 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 working with 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 *temporal*, *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.

This chapter is based on the following publication[10]:

- **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 '19 (2019) 770–781

4.2 Problem Statement

Recall (see Definition 4 in chapter 3) that the observed cascade graph of a post m_i before time t_o is denoted as $C_i(t_o)$. In this chapter, we use $g_i^{t_j} = (U_i^{t_j}, E_i^{t_j}, t_j)$ as a short-

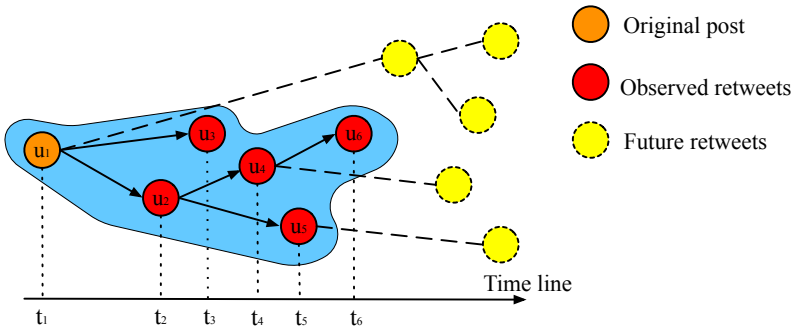


Figure 4.1: The observed cascade graph of a post m_i. Node u_0 initiates the original message m_i.

hand for the snapshot of $C_i(t_o)$ reflecting the diffusion status of a post m_i at time t_j ($t_j \leq t_o$). For example: a node $u_x \in U_i^{t_j}$ represents a user who (for the first time) re-tweets the post m_i from some other users in Twitter (or a paper in a citation network). An edge $\left(u_x, u_y\right) \in E_i^{t_j}$ represents a re-tweet (or a citation) of u_x from u_y ; and t_j is the time-instant when the last re-tweeting (or the citation) behavior occurs in the current snapshot. Figure 4.1 illustrates how the cascade graph can be represented as $g_i^{t_1} = \left(\left(u_1\right), \varnothing, t_1\right), \cdots, g_i^{t_6} = \left(\left(u_1, u_2, \cdots, u_5, u_6\right), \left[\left(u_1, u_2\right), \left(u_1, u_3\right), \cdots, \left(u_4, u_6\right)\right], t_6\right)$.

As depicted above, from observed cascade graph $C_i(t_o)$, we can get different snapshots $g_i^{t_j}$, so that $C_i(t_o)$ can be further represented as a sequence of sub-cascade graphs $G_i^{t_o} = \left\{g_i^{t_j} | t_j \leq t_o\right\}$. In this chapter, our task is to predict the increment size ΔS_i regarding the post m_i for a fixed time interval Δt ($\Delta t = t_p - t_o$), i.e., $\Delta S_i = \left|U_i^{t_o+\Delta t}\right| - \left|U_i^{t_o}\right|$.

Definition 8 *The cascade size predictor is a function $f\left(\cdot\right)$ that is to be learned, mapping $G_i^{t_o} = \left\{g_i^{t_1}, \cdots, g_i^{t_j}, \cdots; t_j \leq t_o\right\}$ to ΔS_i for the time interval Δt.*

Table 4.1: Main notations used throughout this chapter.

Symbol	Description
$g_i^{t_j}, \mathbf{a}_i^{t_j}$	A sub-cascade graph of $C_i(t_o)$ at diffusion time t_j and it's adjacency matrix.
$G_i^{t_o}, \mathbf{A}_i^{t_o}$	A sequence of sub-cascade graphs of $C_i(t_o)$ and the corresponding adjacency matrices.
Δt	The fixed time interval.
ΔS_i	The increment size of m_i after Δt.
$\mathbf{P}_c$	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.

4.3 CasCN: Information diffusion prediction via recurrent cascades convolution

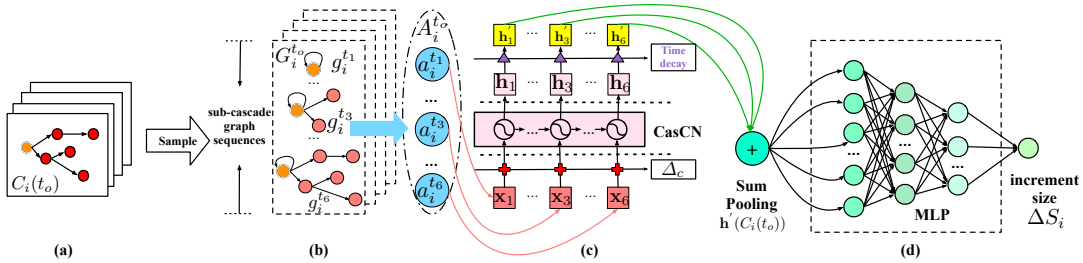


Figure 4.2: Overview of *CasCN*: (a) The input is a cascade graph $C_i(t_o)$ for a given time window t_o of a certain post m_i. (b) We obtain a sequence of sub-cascade graphs from $C_i(t_o)$, and use an adjacency matrix $\mathbf{a}_i^{t_j}$ to represent an instance $g_i^{t_j}$ of the sub-cascade graph. We refer to $\mathbf{A}_i^{t_o} = \{\mathbf{a}_i^{t_1}, \mathbf{a}_i^{t_2}, \cdots\}$ as signals. (c) We feed the signals and the Laplacian matrix Δ_c of cascade $C_i(t_o)$ into *CasCN*. The output $\mathbf{h}_t$ of *CasCN* is transformed to a new representation $\mathbf{h}_t^{'}$ by multiplying it by a time decay factor. All $\mathbf{h}_t^{'}$'s will be assembled via a sum pooling to form the final $C_i(t_o)$ representation: $\mathbf{h}^{'}(C_i(t_o))$. (d) Finally, we use a MLP to predict the increment size of cascade (ΔS_i) for a fixed time interval Δt.

Our deep learning framework *CasCN* takes the observed cascade graph $C_i(t_o)$ as an input and predicts the increment size ΔS_i regarding certain information (e.g., a post) m_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.

CasCN is an end-to-end type of framework consisting of three basic components, depicted in Figure 4.2: (1) Cascade graph sampling: it 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: it 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 a 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 the previous steps.

4.3.1 Cascade Graph as Sub-cascade Graph Sequences

Given a post m_i, the first step in *CasCN* is to initialize the representation of its corresponding cascading graph $C_i(t_o)$. 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

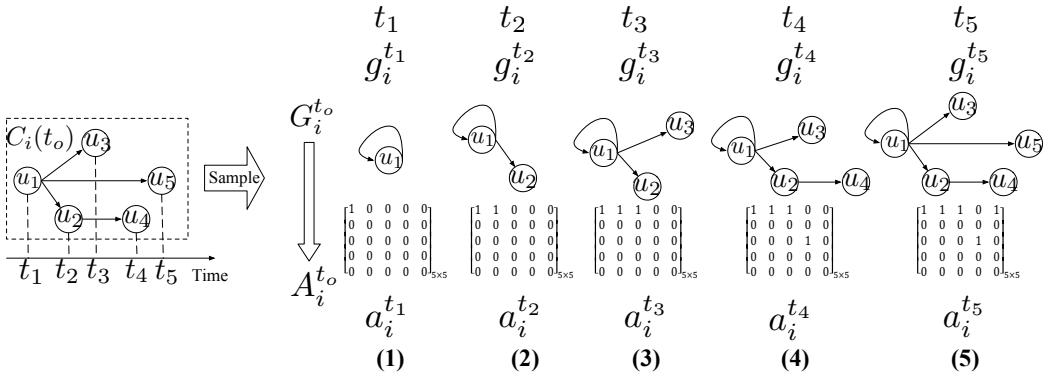


Figure 4.3: Illustration of sampling and representation of sub-cascade graph sequence.

process could be generalized as performing a random walk over a cascade graph similar to DeepWalk which, however, fails to consider the dynamics of cascades – one of the most important factor in information diffusion. DeepHawkes [9] 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 $G_i^{t_o}$ which is used to represent cascades within the observation time t_o. $G_i^{t_o}$ is denoted as:

$$G_i^{t_o} = \left\{ g_i^{t_1}, \cdots, g_i^{t_j}, \cdots \right\}, t_j \leq t_o. \tag{4.1}$$

The sampling process is illustrated in Figure 4.3, 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 $G_i^{t_o}$ only contains one single node because it is the generator of the post m_i, so we add a self-connection for this initiator. Thus, $G_i^{t_o}$ is represented with a sequence of adjacency matrices $\mathbf{A}_i^{t_o} = \left\{ \mathbf{a}_i^{t_1}, \cdots, \mathbf{a}_i^{t_j}, \cdots \right\}$.

4.3.2 Laplacian Transformation of Cascades

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 trees (DATs, which belong to dynamic directed graphs). As mentioned above, the graph Laplacian for an undirected graph is a symmetric difference operator $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D}$ is the degree matrix and $\mathbf{W}$ is the weight matrix of the graph, which cannot be adapted in DAGs.

Recently, Li et al. [115] 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:

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

where $\mathbf{D}^{-1}\mathbf{W}$ is the random walk matrix used to replace Laplacian $\widetilde{\mathbf{L}}$ in Eq. (3.13) This operation is actually a diffusion process convolution proposed by Atwood and Towsley [116] 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 DATs, all of which require incorporating special structure and direction information rather than a *single* and *fixed* sensor network in [115]. To overcome this challenge, we introduce Laplacian of cascade Δ_c, called *CasLaplacian*, for modeling the convolution operation over a single cascade signal $\mathbf{X}$ as:

$$y = g_\theta * \mathcal{G}\mathbf{X} = \sum_{k=0}^{K} \theta_k T_k \left(\widetilde{\Delta}_c \right) \mathbf{X} \tag{4.2}$$

where $\widetilde{\Delta}_c = \frac{2}{\lambda_{max}} \Delta_c - \mathbf{I}$ 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:

$$\mathcal{L} = \mathbf{I} - \frac{\Phi^{\frac{1}{2}}\mathbf{P}\Phi^{-\frac{1}{2}} + \Phi^{-\frac{1}{2}}\mathbf{P}^T\Phi^{\frac{1}{2}}}{2}, \tag{4.3}$$

(cf. [117]) where $\mathbf{P}$ 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 $\mathbf{P}$.

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 $\mathbf{P}_c$, 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* [118] which computes Laplacian of DAGs as:

$$\Gamma = \Phi^{\frac{1}{2}} (\mathbf{I} - \mathbf{P}) \Phi^{-\frac{1}{2}} \tag{4.4}$$

where the transition probability matrix $\mathbf{P}$ is defined as $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$ with the hypothesis that the graph is strongly connected (SCGs) [118]. In contrast, our cascade graphs are not SCGs. Thus, we define a transition probability matrix $\mathbf{P}_c$ of given cascade graph as:

$$\mathbf{P}_c = (1 - \alpha) \frac{\mathbf{E}}{n} + \alpha \left(\mathbf{D}^{-1}\mathbf{W} \right), \tag{4.5}$$

where $\mathbf{E} \in \mathbb{R}^{n \times n}$ is an all–one matrix and $\alpha \in (0, 1)$ is an initial probability, used to restrict the state transition matrix $\mathbf{D}^{-1}\mathbf{W}$ to be a strongly connected matrix without 0 anywhere. Then the transition matrix $\mathbf{P}_c$ 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 $\phi^T \mathbf{P}_c = \phi^T$ 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:

$$\Delta_c = \Phi^{\frac{1}{2}} \left(\mathbf{I} - \mathbf{P}_c\right) \Phi^{-\frac{1}{2}}. \tag{4.6}$$

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 $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$, 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 $\phi^T \mathbf{P} = \phi^T$. Note that, as for undirected graph, the normalized Laplacian $\mathbf{L}$ can be transformed as:

$$\mathbf{L} = \mathbf{D}^{-\frac{1}{2}} \left(\mathbf{D} - \mathbf{A}\right) \mathbf{D}^{-\frac{1}{2}} = \mathbf{D}^{\frac{1}{2}} \left(\mathbf{I} - \mathbf{P}\right) \mathbf{D}^{-\frac{1}{2}}. \tag{4.7}$$

Also, the stationary probabilities ϕ of an undirected graph can be calculated as

$$\phi_i = \frac{d_i}{\sum_k d_k} = \frac{d_i}{d}, i = 1, 2, \cdots, n, \tag{4.8}$$

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 the degree matrix $\mathbf{D}$:

$$\mathbf{L} = \mathbf{D}^{\frac{1}{2}} \left(\mathbf{I} - \mathbf{P}\right) \mathbf{D}^{-\frac{1}{2}} \approx \Phi^{\frac{1}{2}} \left(\mathbf{I} - \mathbf{P}\right) \Phi^{-\frac{1}{2}}. \tag{4.9}$$

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

4.3.3 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 $\mathbf{A}_i^{t_o}$ 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) [40], which is a stable and powerful variant of RNNs. We replace the multiplications by dense matrices W with graph

Algorithm 1: Laplacian of cascade.

Input: A cascade graph C, initial probability α.
Output: *CasLaplacian*–Laplacian of cascade Δ_c.

1: Compute degree matrix $\mathbf{D}$ and weighted adjacency matrix $\mathbf{W}$ of a cascade graph C.
2: Compute transition probability matrix $\mathbf{P}_c$ of cascade graph according to Eq. (4.5) .
3: Solve the eigenvalue problem $\phi^T \mathbf{P}_c = \phi^T$ subject to a normalized equation $\phi^T e = 1$ to get $\{\phi_i\}$.
4: $\Phi = \mathrm{diag}(\phi)$.
5: Compute *CasLaplacian* Δ_c according to Eq. (4.6).

convolutions to incorporate the structural information:

$$\begin{aligned}
\mathbf{i}_t &= \sigma\left(\mathbf{W}_i * \mathcal{G}\mathbf{X}_t + \mathbf{U}_i * \mathcal{G}\mathbf{h}_{t-1} + \mathbf{V}_i \odot \mathbf{c}_{t-1} + \mathbf{b}_i\right) \\
\mathbf{f}_t &= \sigma\left(\mathbf{W}_f * \mathcal{G}\mathbf{X}_t + \mathbf{U}_f * \mathcal{G}\mathbf{h}_{t-1} + \mathbf{V}_f \odot \mathbf{c}_{t-1} + \mathbf{b}_f\right) \\
\mathbf{o}_t &= \sigma\left(\mathbf{W}_o * \mathcal{G}\mathbf{X}_t + \mathbf{U}_o * \mathcal{G}\mathbf{h}_{t-1} + \mathbf{V}_o \odot \mathbf{c}_t + \mathbf{b}_o\right)
\end{aligned} \tag{4.10}$$

where $*\mathcal{G}$ denotes the graph convolution defined in Eq.(4.2), signal $\mathbf{X}_t$ is the cascade graph sequences $\mathbf{A}_i^{t_o} \in \mathbb{R}^{d_T \times n \times n}$, d_T denotes the number of diffusion time steps of post m_i. We leverage $\mathbf{W}_i * \mathcal{G}\mathbf{X}_t$ to mean a graph convolution of signal $\mathbf{X}_t$ with $d_h \times n$ filters which are functions of the cascade Laplacian Δ_c parametrized by K Chebyshev coefficients. $\sigma(\cdot)$ is the logistic sigmoid function and $\mathbf{i}_t, \mathbf{f}_t, \mathbf{o}_t, \mathbf{b}_*$ are respectively the input gate, forget gate, output gate and bias vector. The matrices $\mathbf{W} \in \mathbb{R}^{K \times n \times d_h}$, $\mathbf{U} \in \mathbb{R}^{K \times d_h \times d_h}$ and $\mathbf{V} \in \mathbb{R}^{n \times d_h}$ 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 $\mathbf{c}_t$ is updated by replacing the existing memory unit with a new cell $\mathbf{c}_t$ as:

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh\left(\mathbf{W}_c * \mathcal{G}\mathbf{X}_t + \mathbf{U}_c * \mathcal{G}\mathbf{h}_{t-1} + \mathbf{b}_c\right) \tag{4.11}$$

The hidden state is then updated by

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh\left(\mathbf{c}_t\right) \tag{4.12}$$

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

4.3.4 Cascades Size Prediction

Previous works [30, 49] 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 $\phi^p(\tau) = (\tau + c)^{-(1+\theta)}$; (2)

exponential functions $\phi^e(\tau) = e^{-\theta\tau}$; (3) Rayleigh functions $\phi^r(\tau) = e^{-\frac{1}{2}\theta\tau^2}$, where $\tau = t_j - t_i$ and $t_i \leq t_j$. In practice, the choice of such function varies for different scenarios, e.g., exponential functions are suitable for financial data while Rayleigh functions perform better for epidemiology and power law functions are more applicable in geophysics and social networks [119, 120].

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 chapter, 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 $[t_1, t_o]$, and then split the time window into l disjoint time intervals $\{[t_1, t_2), [t_2, t_3), \cdots, [t_{l-1}, t_l = t_o]\}$ to approximate the continuous time window by discrete time intervals. 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_j:

$$m = \lfloor \frac{(t_j - t_1)}{\lceil t_o/l \rceil} \rfloor \tag{4.13}$$

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

For a cascade graph $C_i(t_o)$ regarding post m_i within the observation time window $[t_1, t_o]$, we get the hidden states $\{\mathbf{h}_1, \cdots, \mathbf{h}_t, \cdots, \mathbf{h}_{d_T}\}$ and we multiply a time decay effect λ_m for each hidden state to obtain $\left\{\mathbf{h}_1^{'}, \cdots, \mathbf{h}_t^{'}, \cdots, \mathbf{h}_{d_T}^{'}\right\}$ by:

$$\mathbf{h}_t^{'} = \lambda_m \mathbf{h}_t \tag{4.14}$$

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

$$\mathbf{h}^{'}\left(C_i(t_o)\right) = \sum_{t=1}^{d_T} \mathbf{h}_t^{'} \tag{4.15}$$

The last component of *CasCN* is a multi-layer perceptron (MLP) with one final output unit. Given the representation $\mathbf{h}^{'}\left(C_i(t_o)\right)$, we calculate the increment size ΔS_i as:

$$\Delta S_i = f\left(C_i(t_o)\right) = \mathrm{MLP}\left(\mathbf{h}^{'}\left(C_i(t_o)\right)\right) \tag{4.16}$$

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

$$\ell\left(\Delta S_i, \Delta\widetilde{S}_i\right) = \frac{1}{P}\sum_{i=1}^{P}\left(\log \Delta S_i - \log \Delta\widetilde{S}_i\right)^2 \tag{4.17}$$

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

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

Algorithm 2: Learning with *CasCN*.

Input: sequences of adjacency matrices of cascade graphs $\mathbf{A} = \{\mathbf{A}_1^{t_o}, \mathbf{A}_2^{t_o}, \cdots\}$ within an observation time t_o; Laplacian sequence for cascade graphs $\Delta = \{\Delta_c^1, \Delta_c^2, \cdots\}$, batch size b.

Output: Increment sizes $\Delta S = \{\Delta S_1, \Delta S_2, \cdots\}$ of cascades.

repeat
  b = 1, 2, ...
  for adjacency matrix sequence $\mathbf{A}_i^{t_o}$ and corespoding Laplacian Δ_c^i in batch b **do**
    Compute the Structural and Temporal information $\mathbf{h}_t$ of cascade $C_i(t_o)$ according to Eq. (4.10) - Eq. (4.12).
    Multiply each hidden state $\mathbf{h}_t$ with time decay effect λ_m to get $\mathbf{h}_t^{'}$, according to Eq. (4.14).
    $\mathbf{h}^{'}\left(C_i(t_o)\right) \leftarrow \text{Aggregate}\Big(\Big\{\mathbf{h}_t^{'}, t \in [1, d_T]\Big\}\Big)$
    Feed $\mathbf{h}^{'}\left(C_i(t_o)\right)$ into MLP to compute increment size ΔS_i of cascade, according to Eq. (4.16)
    Use Adaptive moment estimation (Adam) to optimize the objective function in Eq. (4.17) and update parameters in Eq. (4.10), (4.11), (4.13)
  end for
until convergence;

4.4 Evaluation

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. For reproducability of our results, supplemental materials, implementation details and instructions are available online at https://github.com/ChenNed/CasCN.

4.4.1 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 4.2.

Table 4.2: Statistics of datasets

	Data sets	Sina Weibo			HEP-PH		
posts-papers	All	119,313			34,546		
edges	All	8,466,858			421,578		
t_o		1hour	2hours	3hours	3years	5years	7years
cascades	train	25,145	29,515	31,780	3,458	3,467	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

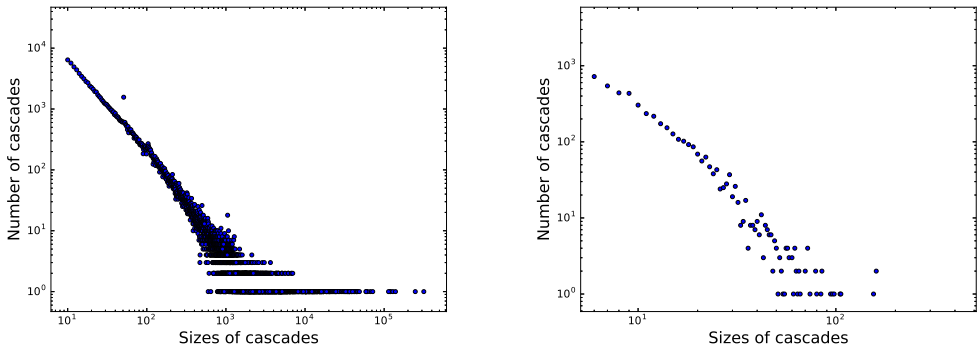
• **Sina Weibo**[1]: The first dataset is Sina Weibo, a popular Chinese microblog platform [2], provided in [9] – which collects all original posts generated on June 1^{st}, 2016, and tracks all re-tweets of each post within the next 24 hours. It includes $119,313$ posts in total. Figure 4.5a shows that the popularity of cascades saturates after 24 hours since publishing. Figure 4.4a shows the distribution of cascade size (the number of re-tweets of each post). We follow similar experimental setup as in DeepHawkes [9] – i.e., the length t_o of the observation time window being $t_o = 1$ 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**[3]: 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 [121]. For the observation window, we choose $t_o = 3$, 4 and 5 years corresponding to the year that the popularity reaches about 50%, 60% and 70% of the final size, as shown in Figure 4.5b. Then, we pick up 70% of cascades for training and the rest for validation and testing via even split.

4.4.2 Baselines

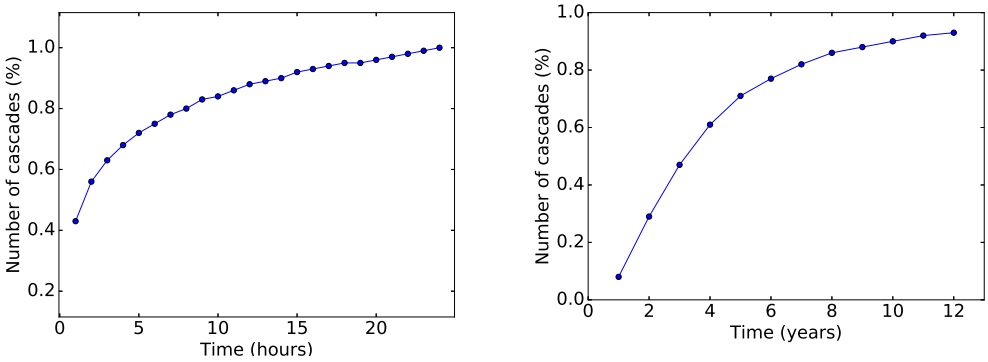
We have already seen that existing relevant methods for information cascade prediction are mainly falling into three categories: (1) Handcrafted feature-based approaches, (2) Point process-based approaches, and (3) Deep learning-based approaches. Therefore, we select several methods in each group as baselines. As

[1]https://github.com/CaoQi92/DeepHawkes
[2]http://www.weibo.com
[3]http://snap.stanford.edu/data/cit-HepPh.html



(a) Weibo dataset **(b)** HEP-PH

Figure 4.4: 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

Figure 4.5: Percentage distribution between time and the number of cascades

deep learning methods, we select three representative methods: DeepCas [8], DeepHawkes [9] and Topo-LSTM [35]. Note that DeepHawkes is also regarded as a successful implementation of Hawkes process – i.e., point process-based approaches. Furthermore, we include a network representation method to enrich our experiment – Node2Vec. We also introduce a baseline named LIS [13] from diffusion model-based approaches, which used to model cascades dynamics. The baselines and their implementation details are as follows:

Feature-based: Recent studies [1, 39, 49, 56] 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 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. The details of the L_2-coefficient setting can be found in Section 4.4.5.
- **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 [13]: 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 [122]: Node2Vec is selected as a representative of node embedding methods, and can be replaced with any other embedding methods, e.g., DeepWalk [123] and LINE [124]. 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 [9]: 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 point process-based approaches and deep learning-based approaches.

Topo-LSTM [35]: 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.

4.4.3 Variants of CasCN

In addition to the 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 $\mathbf{h}_t$ by a linear interpolation between the last state $\mathbf{h}_{t-1}$ and the candidate state $\widetilde{\mathbf{h}}_t$.

CasCN-Path: In CasCN-Path, we use random walks to represent a cascade graph (shown in Figure 4.6) 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 $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$.

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

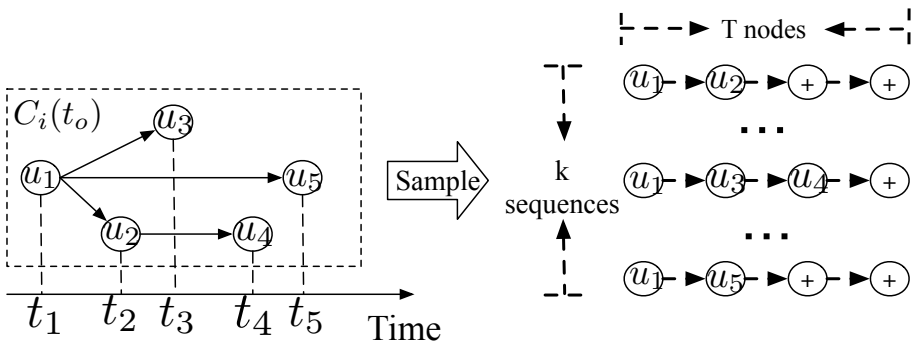


Figure 4.6: Sampling the cascade graph as random walks.

4.4.4 Evaluation Metric

Following the existing works, we choose standard evaluation metrics – *MSLE* (mean square log-transformed error) in our experiments [8, 9, 35]. Note that the smaller *MSLE*, the better the prediction performance. Specifically, *MSLE* is the metric for evaluating the linking accuracy, defined as:

$$MSLE = \frac{1}{P}\sum_{i=1}^{P}\left(\log \Delta S_i - \log \Delta \widetilde{S}_i\right) \tag{4.18}$$

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

4.4.5 Hyper-parameters

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 [13], the maximum number of epochs 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 [122], 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 the cascade Laplacian.

4.4.6 Overall performance

Table 4.3 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.,

Table 4.3: Overall performance comparison of information cascades prediction among different approaches. M: model; t_o: observation time.

Datasets	Weibo Dataset			HEP-PH		
Metric	MSLE					
M \ t_o	1 hour	2 hours	3 hours	3 years	5 years	7 years
Features-deep	3.680	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	1.470	1.233
CasCN	**2.242**	**2.036**	**1.910**	**1.353**	**1.164**	**0.851**

Table 4.4: Performance comparison between CasCN and its variants. M: model; t_o: observation time.

Datasets	Weibo Dataset			HEP-PH		
Metric	MSLE					
M \ t_o	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

feature-based and point process-based approaches, and it is superior to the state-of-the-art deep learning methods, with a statistically significant drop of MSLE.

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 depends on hand-crafted features which are difficult to select for different scenarios in practice.

For embedding methods, Node2Vec [122] 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 diffusion

model-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 time t_o, we clearly see a general pattern that the larger the t_o, the easier to make a good prediction. It is mainly because of the fact that longer t_o reveals more information for prediction.

4.4.7 Ablation study

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 4.4.3.

The experimental results are shown in Table 4.4, from which we can see that our 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.

4.4.8 Parameter analysis in CasCN

We now turn to investigating whether the parameters of *CasCN* have impact on the performance of cascade size prediction. The results are summarized in Table 4.5. We consider two vital parameters of graph convolutional kernel, i.e., the Chebyshev coefficient K and the largest eigenvalue λ_{max} of the Laplacian matrix Δ_c. For the

Table 4.5: Analysis of parameter impact on performance.

Dataset	Weibo Dataset		
Metric	MSLE		
Parameter \ t_o	1 hour	2 hours	3 hours
K=1	2.284	2.061	1.932
K=2	**2.242**	**2.036**	**1.910**
K=3	2.312	2.078	1.939
$\lambda_{max} \approx 2$	2.418	2.217	2.046
λ_{max} = real	**2.242**	**2.036**	**1.910**

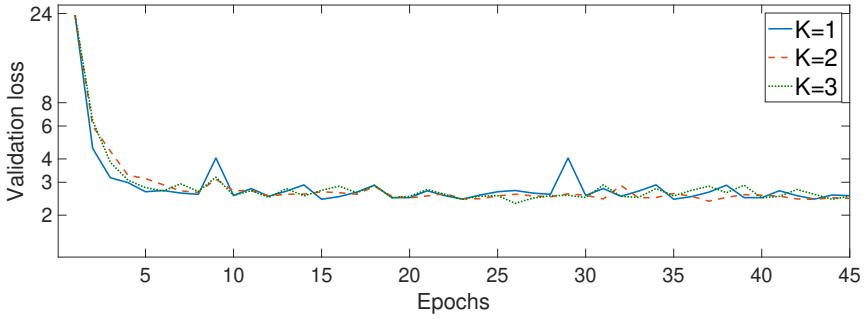
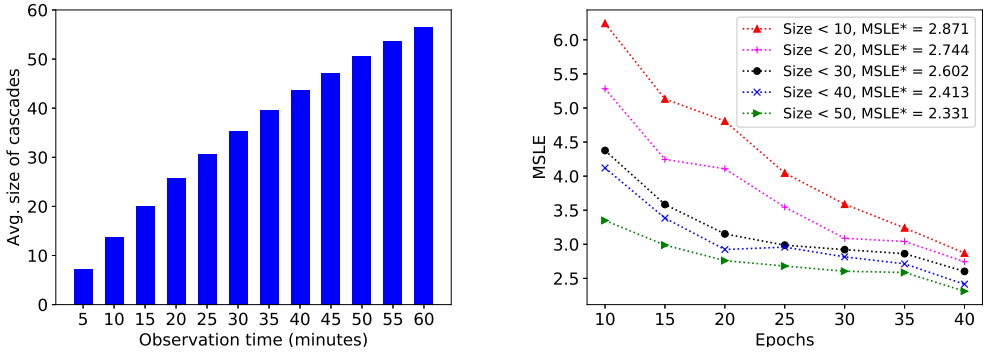


Figure 4.7: Loss of CasCN on the validation set with varying K



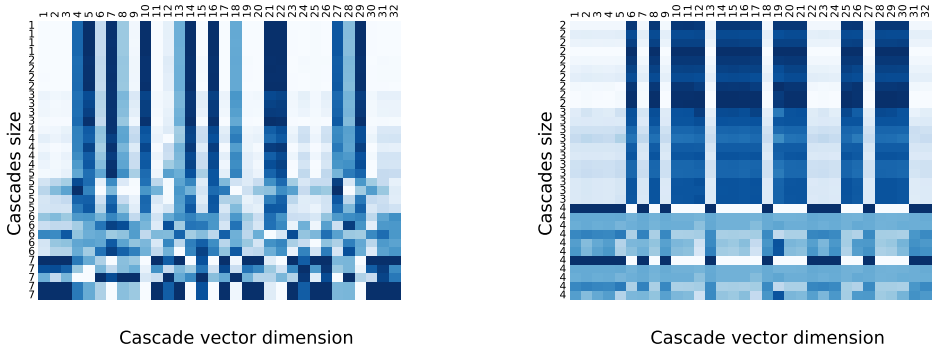
(a) Avg. cascade size distribution. **(b)** Results.

Figure 4.8: Impact of smaller-size observations.

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 4.5 shows that *CasCN* with $K = 2$ can achieve better performance than K=1 and 3. And in Figure 4.7, 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 at a higher computational cost.

We also investigated the performance of *CasCN* when the observed cascades are small – e.g., the size of the cascades is within 10, 20. Figure 4.8a gives the statistics of Weibo dataset illustrating the average cascade size increasing with time. Figure 4.8b shows that the MSLE results for various size decrease with training epochs. Apparently, the larger the size of the observed cascade, the lower the MSLE value *CasCN* achieves.

4.4.9 Discussions on feature learning

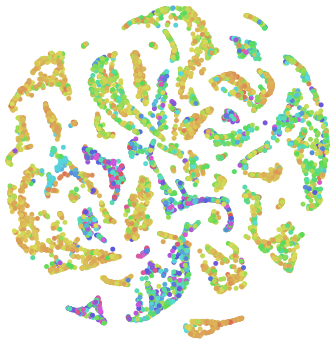


(a) Heatmap of Weibo **(b)** Heatmap of HEP-PH

Figure 4.9: Feature visualization. Figure (a) and (b) are learned representations by *CasCN*.

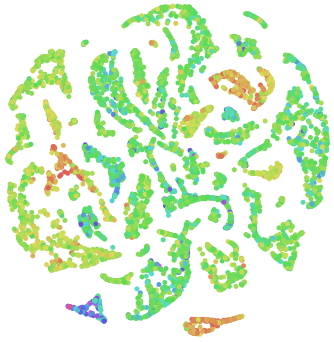
Finally, we discuss and demonstrate the capability of *CasCN* on feature learning in a visual way. We use the latent representation of each cascade graph $C_i(t_o) : \mathbf{h}^{'}(C_i(t_o))$ to plot a heatmap (as shown in Figure 4.9). The value in each dimension corresponds to some implicit or explicit features related to predicting the cascade size. Figure 4.9 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 [125] to project the vector representation summarized in $\mathbf{h}^{'}(C_i(t_o))$ for the cascade graph $C_i(t_o)$ 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

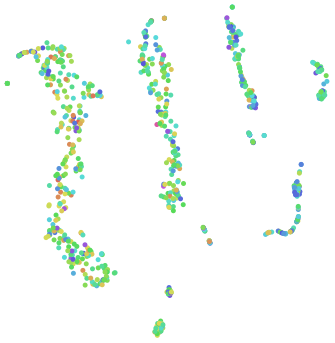


(a) f: Leaf nodes plot for Weibo

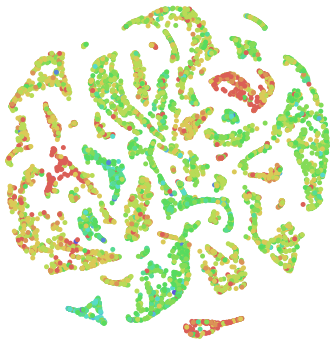
(b) f: Leaf nodes plot for HEP-PH



(c) f: Mean time plot for Weibo



(d) f: Mean time plot for HEP-PH



(e) Increment size plot for Weibo

(f) Increment size plot for HEP-PH

Figure 4.10: Feature visualization. In Figure (a) - (f), 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").

for cascade size prediction. Take the Weibo dataset as an example: Figure4.10a and 4.10c have similar color distributions with the true increment size 4.10e – we believe that leaf nodes and mean time are two important features for cascade size prediction.

4.5 Summary

In this chapter we proposed a novel deep learning based framework – CasCN. It is an end-to-end modeling framework for cascade growth prediction that does not rely heavily on feature engineering and can be easily generalized; 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, 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.