Knowledge-Based Systems 255 (2022) 109740

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Transformer-enhanced Hawkes process with decoupling training for information cascade prediction

Liu Yu [a,1], Xovee Xu [a,1], Goce Trajcevski [b], Fan Zhou [a,*]

[a] University of Electronic Science and Technology of China (UESTC), Chengdu, Sichuan 610054, China
[b] Iowa State University, Ames, IA 50011, USA

ARTICLE INFO

Article history:
Received 15 March 2022
Received in revised form 16 August 2022
Accepted 17 August 2022
Available online 23 August 2022

Keywords:
Information diffusion
Information cascade prediction
Hawkes point process
Attention mechanism
Popularity prediction

ABSTRACT

The ability to model the information diffusion process and predict its size is crucial to understanding information propagation mechanism and is useful for many applications such as popularity prediction and fake news detection. Recent research works have attempted to address the problem of information cascade prediction using two basic paradigms: (1) sequential methods, e.g., recurrent neural networks (RNNs), and (2) graph learning techniques to retain the topological information and enable consideration of structural relationships among diffusion participants. However, existing models consider the topological and temporal features separately, falling short of simulating their entanglement in the diffusion process. As a consequence, since the evolving directed acyclic graph (DAG) of information diffusion is intrinsically coupled with both topological and temporal dependencies, there is a loss of cross-domain information. In this paper, we propose a transformer enhanced Hawkes process (**Hawkesformer**), which links the hierarchical attention mechanism to Hawkes self-exciting point process for information cascade prediction. Specifically, we extend traditional Hawkes process with a topological horizon and efficiently acquire knowledge from continuous-time domain. A two-level attention architecture is used to parameterize the intensity function of Hawkesformer. At the first-level, we disentangle the *primary* and *non-primary* paths to simulate the coupled topological and temporal information for capturing the *global* dependencies between the nodes in a graph. At the second-level, a *local* pooling attentive module is proposed to embed the cascade evolution rate for modeling short-term outbreaks. Extensive experiments on two real-world datasets demonstrate the significant performance improvements of Hawkesformer over existing state-of-the-art models.

1. Introduction

Sharing content through social media platforms such as Twitter, Facebook, and Weibo has become the main channel for expressing individuals' opinions and reactions regarding various "everyday" topics. This trend, in turn, has brought an unprecedented convenience for generating, delivering and propagating information items, along with vast volumes of data describing how the information was spread. The initial information (e.g., tweet), along with the subsequent resharing (e.g., retweet) form an information cascade representing what is commonly referred to as information diffusion process [1,2]. A similar phenomenon has been identified in other (non-social media) settings: paper citations [3], blogging space [4,5], and email forwarding [6]. A large number of user sharing behaviors contributes to the rapid and massive dissemination of information. Predicting the size of an information cascade after a certain time-period brings enormous economic and social values in many downstream applications such as advertising, influence maximization, misinformation and hate speech reduction [7–12]. The retweet records of an information cascade are not simplified sequential events but are equipped with an underlying diffusion topology, often captured via directed acyclic graph (DAG). The core challenge of information cascade prediction is how to model the underlying diffusion process governing the popularity dynamics of information cascades.

Existing information cascade prediction models fall into three main categories: (1) *Feature-Engineering* approaches [13–15] extract and compute a large number of hand-crafted factors such as basic user features, content features, temporal/structural features. The main bottlenecks of feature-based methods are generalizability and privacy concerns. Features identified by experts and domain knowledge can only apply to certain platforms or particular type of information. On the other hand, features of users are often unavailable due to privacy policies, limiting the scalability

* Corresponding author.
E-mail addresses: liu.yu@std.uestc.edu.cn (L. Yu), xovee@ieee.org (X. Xu), gocet25@iastate.edu (G. Trajcevski), fan.zhou@uestc.edu.cn (F. Zhou).

[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.knosys.2022.109740
0950-7051/

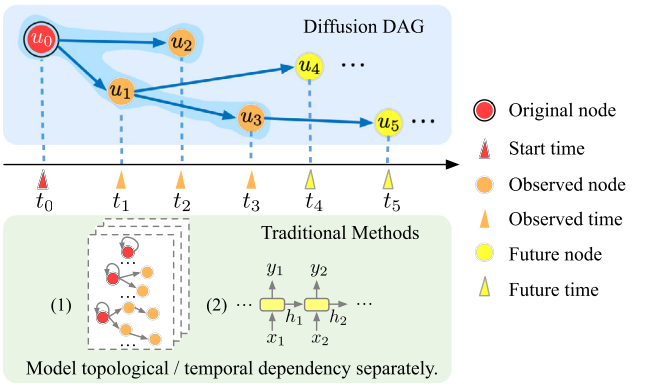


Fig. 1. Top: Information cascade diffusion process as a directed acyclic graph (DAG)—with the coupled temporal and topological dependencies. Bottom: An example of a cross-dependence problem in separate modeling.

of the models. Many researchers [13,14,16] summarized that *temporal* and *topological* features are the most effective, e.g., the arrival rate, time intervals, retweet relationships. (2) *Probabilistic Generative* approaches aim to model the event occurrence by classic examples of temporal point processes (TPPs), including Poisson process [17] and Hawkes self-exciting process [18]. These processes are used to simulate/reproduce the cascade diffusion mechanism—however, they often underestimate or simplify the complex underlying mechanisms governing the success of information cascades [15,19]. Despite their efficiency and enhanced interpretability, conventional generative models are less powerful in making precise predictions. (3) *Deep Learning*-based approaches learn the complex diffusion mechanisms with deep neural networks, which have millions of parameters that can be updated by gradient-descent algorithms. Recurrent neural network (RNN) based sequential models are utilized to capture the temporal dependencies among cascade events [20,21]. Structural characteristics, in contrast, are often modeled by graph neural networks (GNNs) and graph learning techniques [22–24].

Although the aforementioned methods have achieved significant improvements in information cascade prediction, they exhibit four notable drawbacks: (i) Feature-based models explore feature-sets extracted from content, structure, time-series, and basic profile from the users. Nevertheless, they rely on extensive hand-crafted feature engineering that cannot be generalized from one domain to another and are not easy to implement [25]. (ii) Probabilistic generative models simplify the arrival rate of events, e.g., exponential or power-law functions, and make strong assumptions on the diffusion mechanisms in the cascaded events, limiting their learning power on large-scale cascade data. (iii) Recent deep learning-based methods focus more on static temporal cascade diffusion within specific snapshots in the discrete-time domain. Little attention has been paid to the whole evolution process and fully exploiting all cascade dynamics. For instance, in [26,27] the authors learned topological–temporal dependencies in isolation, losing cross-domain information, see Fig. 1. (iv) Conventional methods for cascade prediction face the challenge of data imbalance impact. The long-tailed distribution of cascade sizes hinders the training process and degrades the prediction performance.

To tackle the aforementioned limitations of the existing models, in this work, we present **Hawkesformer** (Transformer-Enhanced Hawkes process). Essentially, it links two-level attention modules in a hierarchy to parameterize the intensity function of self-exciting Hawkes point process. The intensity function models the coupled temporal and topological dependencies of cascades in the continuous-time domain. At the first level, we design a global dependency module to dynamically capture the long-term diffusion process of past events in cascade, where we make a *primary*/*non-primary* path assumption to adaptively integrate the diffusion process on the underlying DAG. This also enables each node attending the cascade at any position to update its current hidden states. At the second level, we design a local pattern module that considers the short-term evolution rate of an information cascade and encodes the local representation in a time-slice window, which extends self-attention by noticing the local fluctuations. In addition, the decoupling training of representation extractor and regressor allows us to avoid the predictions skewed by instance-rich data when jointly training the whole network.

Our main contributions can be summarized as follows:

- We present Hawkesformer, a deep information diffusion learning model that learns the coupled temporal and topological features of information diffusion in the continuous-time domain. It extends the traditional Hawkes process with a topological horizon and can efficiently acquire the coupled knowledge for information popularity prediction.
- We propose a two-level attention architecture that parameterizes the intensity function of traditional Hawkes process, which first uses a *global* dependency module to capture the long-term diffusion between nodes in dynamic diffusion topology, and then leverages a *local* dependency module to capture the short-term evolution rate in a fixed time-slice.
- We customize a learnable *position-wise* user embedding method for unattributed information DAGs, bringing high parameter efficiency and faster computation and leveraging decoupled training scheme from [28] to cope with the long-tailed distribution in cascade sizes.
- Extensive experimental evaluations on three real-world datasets show that Hawkesformer outperforms nine state-of-the-art baselines from three paradigms. Ablation studies and visualizations are also provided to demonstrate its effectiveness and interpretability.

In addition to the greatly enhanced experimental evaluations and discussion of the relevant literature, the present work also extends the results in [28] (which focused on the impact of long-tailed distribution on the training) with two novel aspects: (1) a new architecture to capture both long-term and short-term dependencies; and (2) an augmented and learnable embedding approach to increase efficiency and performance.

The rest of this paper is organized as follows. Section 2 reviews the related literature and positions the contributions of Hawkesformer. Section 3 introduces the preliminaries of the information cascade popularity prediction problem. In Section 4, we present the details of our solution for information cascade prediction. We report the main results, ablation study and visualization results in Section 5. We conclude our work and point out future directions in Section 6.

2. Related work

In this section, we first review the related literature on information cascades and discuss their relations to our proposed Hawkesformer model. Since our main focus is on information cascade popularity prediction with unattributed DAG, the proposed model mostly relates to the Hawkes process built on attention neural network—which is why we present a more detailed review of the related literature from those two domains in separate subsections.

2.1. Information cascade popularity prediction

Modeling information cascades and predicting information diffusion in social (as well as other kinds of) networks have been studied extensively in recent years [1,29]. Macroscopic, mesoscopic, and microscopic approaches [1,2,30,31] have been proposed for a range of practical downstream prediction tasks such as popularity prediction [20], rumor detection [32], outbreak prediction [33], user activation prediction [34], and social influence locality [35]. In this work, we mainly study the information cascade popularity prediction problem, i.e., given an information cascade, we predict its future size (macroscopic) by observing its early stage condition.

Initial studies of information cascade prediction have explored various kinds of features with respect to the related information in cascade diffusion. They rely on machine learning algorithms to connect cascade popularity with many hand-crafted features, including but not limited to content, diffusion topology, user profiles, and event time-series. For example, four types of determinants of cascade popularity are extracted in [15]: (a) basic user features; (b) temporal features; (c) volume features; and (d) past user success—which are modeled by a Random Forest regressor. The findings reported that user-related features and temporal activity features perform competitively in popularity prediction. However, many proposed specific context features and private user profiles are either data-source specific or rely on information that is not publicly available, which requires manual effort and extensive human domain expertise, limiting such models' scalability. Examples of context-specific features include topics or hashtag content [36], propagation network measurements [37], and sophisticated features relating to audience behavior [38]. Many researchers [13,14,16,39] also concluded that *temporal* features and *structural/topological* features – e.g., the arrival rate, time interval, retweet relationship – are the most effective predictors.

Another body of works relies on probabilistic generative models, describing the diffusion process as an event sequence to capture the temporal dynamics of information diffusion. Notably, the family of temporal point processes (TPPs) has been extensively studied in the literature, among which Hawkes point process [18,40] and Poisson point process are dominant examples. They incorporate different endogenous and exogenous factors to model the conditional intensity functions. A double stochastic process was employed in SEISMIC [19] to account for infectiousness and the arrival time of events in microblogging network, and a reinforced Poisson process with the 'rich-get-richer' mechanism was utilized in [3] to track the citation dynamics of a cascade. Some of works [15,41] discussed the combination of Hawkes process and three important factors (user influence, decaying social attention and content quality) for social network popularity prediction. Time-Dependent Hawkes process [42] takes into account the circadian nature of the users and the aging of the information, and predicts a popularity curve evolving in time and Hawkes intensity process [43] supplies the missing link between external promotions and inherent factors to explain the complex popularity history, etc. [44] propose a dual mixture self-exciting process to jointly model the separable content virality and influence decay of cascades, which directly characterizes the spread dynamics of online items and supplies interpretable quantities for predicting the final content popularity. Although such methods have gained success and are highly interpretable in certain contexts, their capacity for modeling large-scale data and the simplifying assumptions for the underlying diffusion processes constrain the practicality and expressiveness of the models. Moreover, they lack a description of the cascade diffusion structure (i.e., DAGs), which is important for fully exploiting the whole cascade dynamics [21].

Recently, inspired by the success of deep learning in various real-world applications, researchers have explored deep learning-based models to improve the performance of information cascade prediction. Deep neural networks – which do not require special assumptions regarding cascade diffusion mechanisms and can be trained in an end-to-end manner – are capable to learning and representing massive event data with less reliance on model selection. Recurrent neural networks (RNNs) and graph neural networks (GNNs) are most common tools to characterize temporal and structural information embedded in the diffusion process. Extensive works combined the temporal sequence and information cascade graph within snapshots for prediction. For example, DeepCas [20] and TempCas [45] proposed heuristic strategies to sample cascade path as input sequences to RNNs. DeepHawkes [46] is the first attempt to combine Hawkes theory with deep neural network. It borrows the interpretable factors from Hawkes process to model the self-exciting mechanism by feeding cascade paths into RNNs. However, sequential models cannot fully exploit the structural dynamics of information diffusion with underlying DAGs.

Graph-based models such as CasCN [27], CasSeqGCN [47] and VaCas [26], used node or graph embedding techniques, e.g., spectral graph wavelets [48] and graph convolutional networks [49], for obtaining the structural representations of cascade graph. They also employed RNN-based modules to capture the temporal dependencies of cascade events. However, temporal and spatial features are captured separately, which cannot intuitively and effectively simulate the whole diffusion process, losing cross-domain information. First, training long sequence of events with RNNs is notoriously difficult because of gradient explosion and vanishing, which hinders the model for capturing the long-term dependencies between cascade events and mapping the cascade events into continuous-time domain [26,45,50]. Second, GNN-based models make up for the lack of retweeting relationship (structural information) of RNN-based sequential models, which enforces many redundant dependencies, e.g., CasCN [27] apply convolutional operation on a sequence of sub-graphs, which is computationally intensive and indirect for modeling the cascade diffusion process. Consequently, those particular downsides plus the increased computational burdens make these models inefficient. Several other deep learning techniques such as reinforcement learning [51], attention mechanism [20,52,53], and auto-encoder [26,54] are also used for information cascade prediction. Instead of solely depending on observed information cascade, the work [55] relies on parameters that fit previous similar cascades and infers new parameters accordingly. As a result, it can predict the final cascade size and shape with few observations. They found that as more observation data become available, the performance of predictions tends to improve with time. Other works try to solve specific problems inherent in cascade learning, such as generalization capability [56] and long-tailed data distribution [28], etc.

For a comprehensive review of the recent advances in information cascade prediction, the reader is referred to [1,2,29,30]. We note that what separates the present work from the existing results is that we introduce a novel approach for coupling the temporal and structural information and tackle both long-term and short-term predictions which, as demonstrated by the experimental observations (cf. Section 5), yield significant improvements.

2.2. Hawkes point process

Temporal point processes (TPPs) provide an elegant mathematical tool for modeling event occurrences in the continuous time domain. The self-exciting Hawkes point process [18] so far

has been a dominant point process in both theoretical studies and real-world applications [57]. Recently it spaned a wide range of applications such as finance [58] and bioinformatics [59], criminology [60], and social science [15,19,61], to name a few.

Hawkes-based models view individual broadcasts of information as events in a stochastic point process, and the triggering intensity can be calculated based on the activation time interval.

Traditional Hawkes Processes. They can be divided into two categories: parametric and nonparametric [62]. The parametric models [15,63] suffer from model mis-specification if the manually pre-specified intensity function does not fit the actual event behavior. Its intensity function is a classic additive form defined as:

$$\lambda(t) = \mu + \sum_{j:t_j<t} \phi(t - t_j), \tag{1}$$

where $\mu \geq 0$ is the base intensity and $\phi(\cdot)$ denotes an excitation term, which is devised as a tailored function for tailored model. Choices for the excitation function include parameterized exponential or power-law functions. Here, Eq. (1) makes a strong assumption that historical excitation is positive and additive over past events, which might be simplified in the (complex) real world. As for the nonparametric models [64], although they do not have an explicit specification on intensity function and have a higher data capacity, their learning algorithm can be mathematically more complex [62], preventing them from adoption by practitioners. In sum, traditional TPP-based models excel at clear interpretation on the problem for learning and fitting the data. This, however, is a simplification that may overrate the cascade size owing to its self-exciting mechanism or underrate the cascade size due to its time decay mechanism.

Variants of Hawkes Process. With the fast development of deep learning theory and techniques, a notable trend is resorting to the neural network based approach for modeling more flexible TPPs. We name it neural point process, which distinguishes itself from the traditional point process. Specifically, recurrent neural network and its variants, e.g. long-short term memory (LSTM) were utilized in [61,65] to build network blocks, which can be learned in an end-to-end manner with no need for devising tailored intensity functions and learning algorithms that required in traditional TPPs. For example, the intensity function of Neural Hawkes Process (NHP) [65] is defined by:

$$\begin{aligned} &\lambda(t) = f(\mathbf{w}^\top \mathbf{h}(t)), \\ &f(x) = \beta \log\Big(1 + \exp(\frac{x}{\beta})\Big), \end{aligned} \tag{2}$$

where $\mathbf{h}(t)$ is the hidden states encoding the event history obtained by a continuous-time LSTM. A softplus function $f(\cdot)$ with parameter β guarantees a positive intensity. One notable limitation of those RNNs-based model is unable to capture complex long-term dependencies [66]. Hence, inspired by the development of natural language processing field, several attention-based models (e.g., THP [67], SAHP [68]) have emerged in TPPs to address this issue. UTHP [69] introduces RNN in the encoding process and CNN in the position-wise feed-forward neural network to overcome the problems in THP, including parallel processing, recursive learning, and abstracting local salient properties. As THPs do not effectively utilize the temporal information in the asynchronous events, TAA-THP [70] modifies the traditional dot-product attention structure and incorporates the temporal encoding into the attention structure. Their two key advantages are that those neural point processes free the need for explicit parametric intensify form selection and can efficiently perform using off-the-shelf solvers (e.g., stochastic gradient descent) and tools in an end-to-end manner.

However, they are only capable of modeling simple event streams on the next activated user prediction task and thus cannot be applied to dynamic DAG structures that we are considering in this work.

2.3. Attention mechanism

Attention mechanism was originally used in neural machine translation (NMT) [71,72], which enables NMT models to focus on a subset of the input sequence. Self-attention is a special case of the attention mechanism recently proposed in Transformer [66], which has been broadly applied in a variety of NLP tasks such as machine translation [73] and language modeling [74], yielding considerable improvements. The usage of Transformer is also common in computer vision in recent years [75].

Essentially, it encodes sequences of input tokens by relating these tokens to each other, and the input consists of queries, keys (dimension is d_k) and values (dimension is d_v). In practice, in order to implement them by using highly optimized matrix multiplication, the queries, keys, and values are packed together into matrices Q, K, and V. The matrix form of the attention function is:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3}$$

Self-attention computes the dot products of the query with all keys, divides each by $\sqrt{d_k}$, and applies a softmax function to obtain the weights on values. It measures the dependency between each token pair from the same input sequence. Since self-attention contains no order of the sequence, a positional encoder is used to encode the absolute position of the tokens in the sequence. Consequently, self-attention encodes both token dependency and positional information. For RNNs, single-layer and two-layer architectures are the most common types. However, they may fail to model more intricate dependencies among data. On the contrary, the non-recurrent structure of self-attention facilitates efficient training of multi-layer models. Transformer-based architectures can be as deep as dozens of layers [73,74], which enables capturing higher order dependencies and makes them more powerful than RNNs. Moreover, the self-attention mechanism has faster convergence than RNNs.

Despite its success ranging from natural language processing and computer vision [76], to graph neural network [23], it has rarely been used in other areas. We remark that the Transformer architecture cannot be directly applied to model temporal point processes within diffusion DAGs. Specifically, it does not consider the dynamic diffusion topology. Moreover, time intervals between any two events of a cascade can be arbitrary (irregularly), while for languages, words are observed on regularly spaced time intervals. Therefore, we need to generalize the architecture to a continuous-time domain while considering the dynamic diffusion DAGs, which is what separates the current work from the existing results.

3. Preliminaries

We now introduce the basic terminology and settings, provide a formal definition of the problem, and discuss some context and assumptions. We note that, for convenience, a list of symbols used throughout the paper, along with their concise definitions, is provided in Table 1. In the rest of this study, we consider tweet cascade as an example-setting; however, we note that our approach is applicable to other types of information cascades, e.g., academic publications, news articles, online forums, video, and streaming media, etc.

Table 1
Frequently used symbols and their definitions.

Notation	Interpretation
I	Information item, e.g., tweet, news, or research paper.
u_j, t_j	Current user and retweet timestamp.
t_o, t_p	Observation time ($t_o = t_L$), prediction time.
L	Number of events in the observed period. $L = max(j)$
$\lambda(t)$	Conditional intensity – or event rate at time t – omit history $\mathcal{H}$ for conciseness.
$C_k(t)$	The kth information cascade observed at time t.
$\mathcal{H}_k$	The history of kth cascade within observation time.
$\boldsymbol{s}_t$	The sequence of event timestamps up to but not including time t.
D	Dimensionality of embedding.
N	Number of cascades.
$\mathbf{U}$	A learnable shared matrix for position-wise embedding.
$\mathbf{V}$	The set of all position-wise user indices.
$\mathbf{Z}$	Temporal encodings matrix.
$\mathbf{X}$	User retweet embeddings; $\mathbf{X} = \mathbf{UV} + \mathbf{Z}$
$\mathbf{H}, \mathbf{h}(t_j)$	The hidden state of the first level, i.e., global dependency of current node u_j; $\mathbf{H}$ consists of $\mathbf{h}(t_j)$
$\boldsymbol{\Upsilon}, \boldsymbol{\upsilon}(t_j)$	The hidden state of the second level, i.e., local dependency of current node u_j; $\boldsymbol{\Upsilon}$ consists of $\boldsymbol{\upsilon}(t_j)$
$\mathcal{V}$	All predecessor nodes of current node u_j.
$\mathcal{V}^P, \mathcal{V}^N$	Nodes on the primary/non-primary path, respectively. $\mathcal{V} = \mathcal{V}^P + \mathcal{V}^N$
$\mathbf{P}_{\mathbf{score}}, \mathbf{N}_{\mathbf{score}}$	Primary/non-primary attention scores.
$\mathbf{A}$	The dependency attention probability matrix, including attention from primary/non-primary path.
$\mathbf{M}^P, \mathbf{M}^N$	Matrices used to retain the attention scores of nodes on primary/non-primary paths, respectively.
$y_k(t_p), \hat{y}_k(t_p)$	Ground truth and predictive popularity at prediction time t_p.

Let I be an information item diffused in a network. Consider a Twitter user u_0 who posts a tweet I at time-stamp t_0. Later, other users see this tweet (through the user feed, searching, or recommendation) and interact with I, e.g., "commenting", "liking", and "retweeting". In this paper, we consider "retweeting" as the major source of information diffusion in the Twitter social network.

Given a tweet I where a user u_0 starts it at t_0, we denote all its retweets during a period of time as an *information cascade* C_I, consisting of retweet history $\mathcal{H}_I = \{(t_j, v_j, u_j)\}_{j=1}^{L}$, where $t_j \in (0, t_o]$ and L is the total number of retweets during the observation window $(t_0, t_0 + t_o]$. For each pair (t_j, v_j, u_j), user u_j retweets user v_j's retweet (or the original tweet I) at time-stamp t_j. Here v_j belongs to one of the users in $\{u_0, u_1, \ldots, u_{j-1}\}$. For example in Fig. 2, $\mathcal{H}_I$ consists of five retweets: $\{(t_1, u_0, u_1), (t_2, u_0, u_2), (t_3, u_1, u_3), (t_4, u_1, u_4), (t_5, u_3, u_5)\}$. Now we give the formal definition of information cascade popularity prediction:

Definition 1 (*Information Cascade Popularity Prediction*)**.** Given observed retweet history $\mathcal{H}_I$ of information cascade I at observation time t_o, we predict its size (popularity) $\hat{y}_I(t_p)$, i.e., the number of users who perform the retweeting action to the original tweet I after observation time t_o and before prediction time $t_p \gg t_o$.

Overall, recall that for N observed cascades (e.g., N tweets) $\{C_k\}_{1 \le k \le N}$, the popularity prediction can be formalized as a complex and nonlinear regression task solved by minimizing the following loss function:

$$\mathcal{L}(\Theta) = \frac{1}{N} \sum_{k=1}^{N} (\hat{y}_k(t_p) - y_k(t_p))^2, \qquad (4)$$
$$\hat{y}_k(t_p) = \text{Model}_{\Theta}(\mathcal{H}_k),$$

where $y_k(t_p)$ is the ground truth popularity for cascade C_k at prediction time t_p, and Θ are model parameters.

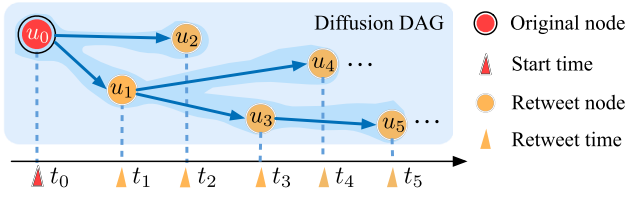


Fig. 2. An example of information cascade.

In the scenario of cascade prediction, each cascade generally corresponds to a directed acyclic graph (DAG) representing the information diffusion—and causing cross-dependence in cascades. In other words, a particular retweet behavior could be triggered by its non-immediate predecessor in the memory chain. As shown in Fig. 2, the retweet behavior of u_2 depends on u_0 rather than its immediate predecessor u_1. Every time a new retweeting behavior occurs, the underlying diffusion structure updates. Thus the diffusion DAGs are associated with the coupled topological–temporal dependencies, which cannot be captured by chain-structured sequential models (like RNNs).

Global and local dependencies. We note that predecessor nodes may either promote or inhibit the occurrence of successor nodes. Thus we consider all the predecessor nodes as the global dependency of the current node, where these predecessor nodes may be located far away from the current node in the memory chain. Since many previous works [2,9,45] have shown that the local dependency (e.g., short-term outbreaks) is important for predicting the future size of information cascade, it is necessary to encode the local dependency.

Primary and non-primary assumption. Information cascade graphs have many distinct (sub-)structures [77]. Some of them are based on simple broadcast propagation, while others have hub structures and deep depth. We hypothesize that hub nodes (nodes

on primary path) are more important for the diffusion process than the, so called, leaf nodes (nodes on non-primary path) which are relatively less impactful. We note that many existing works represent cascade structures with similar strategies, e.g., in DeepCas [20] and SI-HDGNN [78] the transition probabilities are assigned by node degrees to sample the critical paths that cover the most influential nodes; in TempCas [45] the root node is fixed as the starting node that obtains the full cascade paths and can cover more successor nodes.

4. Model

In this section we present the details of our proposed approach. We start with a global overview of Hawkesformer. Next, we discuss in detail the four critical components to capture the *coupled structural and temporal information* with *evolving directed acyclic graph structure* over time in Hawkesformer, which are: (i) learnable user embedding method and a temporal encoding block; (ii) path-aware attention module to learn the *global* dependencies among past cascade events; (iii) pooling attention module to learn the *local* patterns; and (iv) fully connected predictive layer consists of expected popularity (the integral of TTP over time) and topological hidden state. We finalize the section with a complexity analysis.

4.1. Transformer enhanced Hawkes process

In order to capture the global dependencies and local patterns of the entire information cascade, we design a Transformer Enhanced Hawkes Process (**Hawkesformer**), the architecture of which is illustrated in Fig. 3. In essence, it links a two-level attention framework to parameterize the intensity function of Hawkes process. Hawkesformer is expressive and flexible for both global and local dependencies and it captures the coupled spatial–temporal dynamics from continuous-time domain. We characterize the global dependence as $\mathbf{h}(t)$ and local patterns as $\boldsymbol{v}(t)$. Given an information cascade C_k and its retweet history $\mathcal{H}_k = \{(t_j, v_j, u_j)\}_{j=1}^{L}$. Let $\boldsymbol{s}_{t_j} = \{t_0, t_1, \ldots, t_n \mid t_n < t_j\}$ be the sequence of time stamps up to but not including time t_j – we model the continuous dynamics of temporal point process with the following conditional intensity function:

$$\lambda(t|\Theta, \boldsymbol{s}_{t_j}) = f\Big(\underbrace{\alpha \frac{t - t_j}{t_j}}_{current} + \underbrace{\boldsymbol{w_1}^\top \mathbf{h}(t_j)}_{global} + \underbrace{\boldsymbol{w_2}^\top \boldsymbol{v}(t_j)}_{local}\Big) \tag{5}$$

where Θ denotes the model parameters; retweet time t is defined on interval $[t_j, t_{j+1})$; and $f(x) = \beta \log(1 + \exp(x/\beta))$ is the softplus function, with "softness" parameter β to constrain the intensity function to be positive. The first term *current* indicates the evolutionary process in the continuous-time domain; $\boldsymbol{w_1}, \boldsymbol{w_2}$ are learned weights for global dependency and local patterns at time t, respectively, and $\lambda(t)$ denotes the tweet arrival rate at a given time t. We set the base intensity $\mu = 0$ in Eq. (1) since all retweets are considered spawning from the original tweet. Note that different from traditional additive form in Eq. (1), the arrival rate $\lambda(t|\Theta, \boldsymbol{s}_{t_j})$ in Hawkesformer provides learnable influences of all the predecessor to each current node.

4.2. User embeddings

An important component of Hawkesformer is the user embeddings, which is different from the existing models which learn static node embeddings and only preserve topology proximity [48,79], or utilize a learnable global user matrix which is computationally intensive [46]. Specifically, we propose an efficient *position-wise* embedding method. Inspired by word embedding techniques [73] in NLP, we train a shared embedding matrix $\mathbf{U} \in \mathbb{R}^{D \times L}$ where the jth column of $\mathbf{U}$ denotes the D-dimensional embedding for jth retweet user in each cascade, i.e., the same positional nodes in each cascade share an embedding. This greatly reduces the computational overhead and makes our model more efficient. The matrix $\mathbf{U}$ is randomly initialized and it dynamically updates subsequent contextualized embeddings with respect to the underlying cascade. The global dependency module (cf. Section 4.3) dynamically adjusts the initial user embedding based on the contextual node position in the graph structure, no matter where the nodes are located in the cascade structure (e.g., hub or leaf nodes). For a retweet user u_j in the jth position, if we let $\mathbf{v}_j$ denote the index vector of u_j, then its position-wise embedding is indexed by $\mathbf{U}\mathbf{v}_j$. Here $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_L] \in \mathbb{R}^{L \times L}$ is the collection of user one-hot indices.

Temporal encodings. The key ingredient of our proposed Hawkesformer is a two-level attention framework, which is specifically designed for information cascade graph prediction tasks. Since the attention mechanism discards the sequential dependency of RNNs and all tokens of the input sequence are fed into the network with no special order or position. We characterize the temporal information by utilizing a temporal encoding pattern, which is analogous to the "vanilla" absolute positional encoding [66] for each node in cascade. We denote this temporal encoding pattern as $\mathbf{z}$, which is defined as:

$$[\mathbf{z}(t_j)]_i = \begin{cases} \cos\left(t_j / 10000^{\frac{i-1}{D}}\right), & \text{if } i \text{ is odd} \\ \sin\left(t_j / 10000^{\frac{i}{D}}\right), & \text{if } i \text{ is even} \end{cases} \tag{6}$$

Here we utilize a trigonometric function to define the temporal encoding $\mathbf{z}(t_j) \in \mathbb{R}^D$ for each timestamp t_j, where D is the encoding dimension. The subscript i of $[\mathbf{z}(t_j)]_i$ is the index of the encoding dimension, where $i \in [1, D]$. Thus, the embedding of the retweet sequence $\mathcal{H}$ is specified as $\mathbf{X} = \mathbf{UV} + \mathbf{Z}$, where $\mathbf{UV}$ are position-wise embeddings of all users in the cascade, $\mathbf{Z} = [\mathbf{z}(t_0), \mathbf{z}(t_1), \ldots, \mathbf{z}(t_L)] \in \mathbb{R}^{D \times L}$ is the concatenation of temporal encodings. Each column of $\mathbf{X} = [\mathbf{x}(t_0), \mathbf{x}(t_1), \ldots, \mathbf{x}(t_L)] \in \mathbb{R}^{D \times L}$ corresponds to the initial embedding of a specific user in the cascade. Note that our initial user embedding $\mathbf{X}$ is composed of a learned position-wise embedding $\mathbf{UV}$ and a fixed temporal encoding $\mathbf{Z}$.

4.3. Modeling global dependence by self-attention mechanism

The global dependence of an information cascade can be regarded as an indication of who (i.e., which user) could possibly infect or inhibit whom (i.e., another user) in the long-term diffusion process. After obtaining the initial user and temporal embeddings, we propose a path-aware attention module to parameterize the Hawkesformer intensity function, which allows us to capture the long-term diffusion dependencies among input cascade events and extract the coupled topological–temporal dependency-aware user features.

As signified by the Hawkes process, the final popularity of an information cascade is related to the past participation of users, which means that predecessor users influence not only its immediate retweeters but also non-immediate retweeters by way of transitivity. We rely on self-attention as a desirable tool to capture these long-term coupled topological–temporal dependencies. To represent the influence of predecessor nodes on cascade future popularity from a topological view, we remark that for the current node u_j, all the predecessor nodes $\mathcal{V} = \{u_1, u_2, \ldots, u_{j-1}\}$ are divided into two types (as discussed in Section 3): nodes on the *primary* path $\mathcal{V}^p = \{a_1, a_2, \ldots, a_p\} \subset \mathcal{V}$, which includes all the nodes on the entire retweet path from the current node u_j

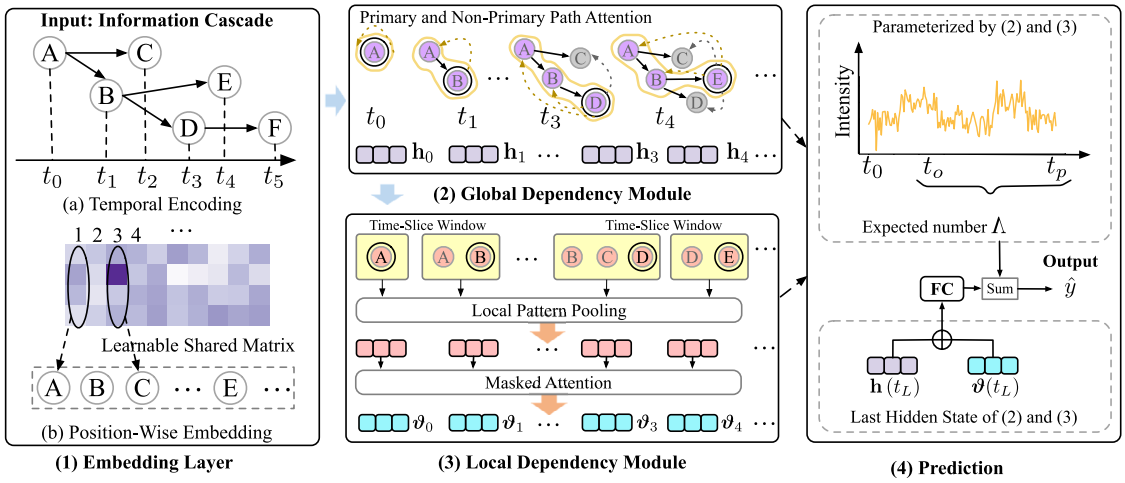


Fig. 3. Overview of our proposed Hawkesformer.

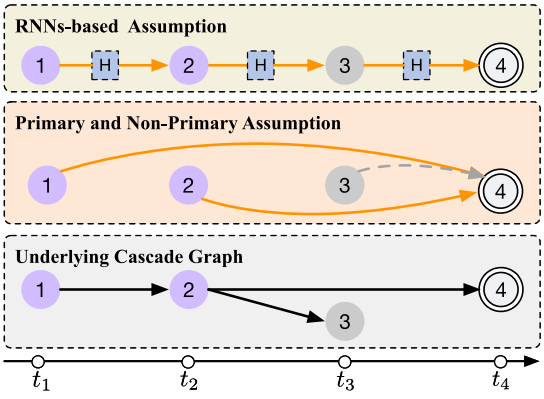


Fig. 4. Primary and non-primary paths in the global dependencies for a current node u_4. Purple (respectively, gray) dots denote nodes on the primary (respectively, non-primary) path.

back to the root node u_0; and other nodes in *non-primary* path $\mathcal{V}^N = \mathcal{V} - \mathcal{V}^P = \{b_1, b_2, \ldots, b_n\}$. We consider the current node u_j traced from root node as *primary* path $\mathcal{V}^P$ which contributes the most to the arrival time at u_j. On the contrary, the nodes on the non-primary path $\mathcal{V}^N$ have relatively small contributions to u_j, but also have an impact on final popularity. An illustrative example is shown in Fig. 4 where, for the current node u_4, the purple nodes indicate the primary path and the gray ones are the nodes on the non-primary path.

Based on the above interpretation, a long-term dependency module is employed to deal with each input user in the information cascade. Given the current node u_j, we obtain the long-term dependency attention score for user $a_l \in \mathcal{V}^P$ in the *primary* path as follows:

$$\alpha_{lj} = \frac{\exp(\langle \mathbf{a}_l, \widetilde{\mathbf{x}}_j \rangle)}{\sum_{l=1}^{p} \exp(\langle \mathbf{a}_l, \widetilde{\mathbf{x}}_j \rangle) + \sum_{m=1}^{n} \exp(\langle \mathbf{b}_m, \widetilde{\mathbf{x}}_j \rangle)}, \tag{7}$$

where $\mathbf{a}$, $\mathbf{b}$ and $\widetilde{\mathbf{x}}$ are the vectors of the users' corresponding embeddings from $\mathbf{X}$ after being transformed by primary query matrix $\mathbf{W}^P \in \mathbb{R}^{D\times D}$, non-primary query matrix $\mathbf{W}^N \in \mathbb{R}^{D\times D}$, and key matrix $\mathbf{W}^K \in \mathbb{R}^{D\times D}$, respectively. Function $\langle \cdot, \cdot \rangle$ is the inner product. The transformation matrices are utilized to differentiate the user's roles in unidirectional dependencies. Similarly, the long-term dependency attention score α_{mj} from a user $b_m \in \mathcal{V}^N$ in *non-primary* path is defined as:

$$\alpha_{mj} = \frac{\exp(\langle \mathbf{b}_m, \widetilde{\mathbf{x}}_j \rangle)}{\sum_{l=1}^{p} \exp(\langle \mathbf{a}_l, \widetilde{\mathbf{x}}_j \rangle) + \sum_{m=1}^{n} \exp(\langle \mathbf{b}_m, \widetilde{\mathbf{x}}_j \rangle)}. \tag{8}$$

Now we have the complete diffusion dependency $\mathbf{e}_j$ for user u_j, computed via the following attention weighted sum:

$$\mathbf{e}_j = \sum_{l=1}^{p} \alpha_{lj} \hat{\mathbf{x}}_l + \sum_{m=1}^{n} \alpha_{mj} \hat{\mathbf{x}}_m. \tag{9}$$

where $\hat{\mathbf{x}}$ is the vectors from $\mathbf{X}$ after being transformed by value matrix $\mathbf{W}^V$. So the diffusion dependency for all nodes is: $\mathbf{E} = \{\mathbf{e}_0, \ldots, \mathbf{e}_j, \ldots, \mathbf{e}_L\}$. The aforementioned long-term dependency attention process for a specific node u_4 is illustrated in Fig. 4. We can observe that unlike RNNs-based models which sequentially obtain compressed states of historical users, our attention-based module enables each user attend the information cascade at any position to update its current hidden state.

Since dependency attentions are computed independently for each user, the computation of primary attention scores $\mathbf{P}_{\text{score}}$ and non-primary attention scores $\mathbf{N}_{\text{score}}$ can be paralleled as matrix multiplication:

$$\begin{aligned} \mathbf{P}_{\text{score}} &= (\mathbf{W}^P \mathbf{X})^\top (\mathbf{W}^K \mathbf{X}) \odot \mathbf{M}^P, \\ \mathbf{N}_{\text{score}} &= (\mathbf{W}^N \mathbf{X})^\top (\mathbf{W}^K \mathbf{X}) \odot \mathbf{M}^N, \end{aligned} \tag{10}$$

where $\mathbf{M}^P$, $\mathbf{M}^N \in \mathbb{R}^{L\times L}$ are matrices used to retain the attention scores of nodes on primary and non-primary paths, respectively. The element $M^P_{i,j} = 1$ if $u_i \in \mathcal{V}^P$ else 0. Similarly, $M^N_{i,j} = 1$ if $u_i \in \mathcal{V}^N$ else 0.

To avoid "data leakage" issue, we use a masked matrix $\mathbf{M} \in \mathbb{R}^{L\times L}$ in the attention module where the element $M_{i,j} = 0$ if $i < j$ otherwise $M_{i,j} = -\infty$. Like most attention mechanisms, we leverage softmax function to derive the probability distribution. Finally, the dependency attention probability matrix is derived as:

$$\mathbf{A} = \text{softmax}\left(\frac{\mathbf{P}_{\text{score}} + \mathbf{N}_{\text{score}}}{\sqrt{D}} + \mathbf{M}\right), \tag{11}$$

where D is the dimension of the input embedding. The masked matrix $\mathbf{M}$ forces the softmax function to assign valid long-term dependency attentions only over u_j's previous retweeters and assign 0 to subsequent users (i.e., users activated later than t_j). Each column vector $\boldsymbol{\alpha}_j$ in $\mathbf{A}$ represents u_j's attentions over its previous states including *primary* and *non-primary* path. Then the

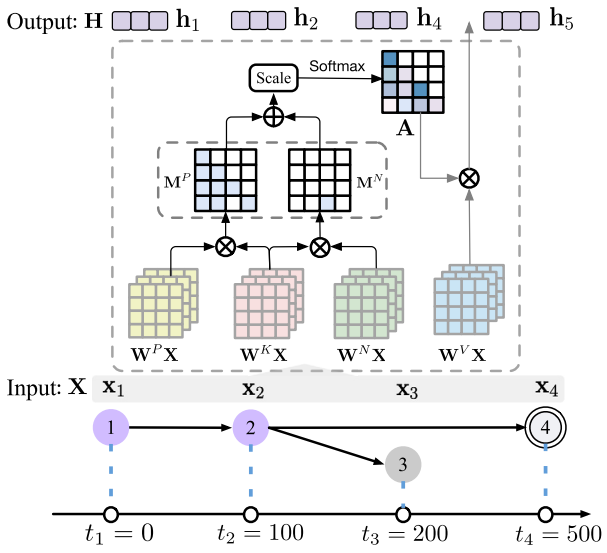


Fig. 5. The implement details of primary and non-primary assumption.

final matrix form of contextualized embeddings for an information cascade can be obtained by $\mathbf{E} = (\mathbf{W}^V\mathbf{X})\mathbf{A}$. The whole matrix computing process of global dependency is illustrated in Fig. 5.

To learn more semantic information, we map multiple heads to different subspaces, which can be run in parallel using matrix multiplication. We compute multiple heads $\mathbf{E}_1, \mathbf{E}_2, \ldots, \mathbf{E}_H$ and then concatenate them to the final $\mathbf{E}$:

$$\mathbf{E} = \mathbf{W}^O \,\text{Concat}\,(\mathbf{E}_1, \mathbf{E}_2, \ldots, \mathbf{E}_H)\,, \tag{12}$$

where $\mathbf{W}^O \in \mathbb{R}^{D\times HD}$ is an aggregation matrix. To learn higher-level retweet representations, the output $\mathbf{E}$ is then fed through fully connected networks. Formally,

$$\begin{aligned}\mathbf{H} &= \mathbf{W}_2^{\text{FC}}\,\text{ReLU}(\mathbf{W}_1^{\text{FC}}\mathbf{E} + \mathbf{b}_1) + \mathbf{b}_2,\\ \mathbf{h}(t_j) &= \mathbf{H}(:, j),\end{aligned} \tag{13}$$

where $\mathbf{W}_1^{\text{FC}} \in \mathbb{R}^{D_F\times D}, \mathbf{W}_2^{\text{FC}} \in \mathbb{R}^{D\times D_F}, \mathbf{b}_1 \in \mathbb{R}^{D_F}, \mathbf{b}_2 \in \mathbb{R}^{D}$ are parameters of the neural network, $\mathbf{h}(t_j)$ is the representation of user u_j related to long-term dependency, and each column of $\mathbf{H} \in \mathbb{R}^{D\times L}$ corresponds to a particular retweet. In practice we stack multiple long-dependency modules together, and inputs are passed through each of these modules sequentially. In this way our model is able to capture high level dependencies.

4.4. Modeling local patterns with pooling attention

The acceleration of retweets number in information cascade is not stable and short-term outbreaks in the diffusion process has a great impact on the final popularity. To better embed the evolution rate of information cascade, we utilize a pooling attention module to parameterize the intensity function of Hawkesformer for capturing the local patterns.

Pooling operation on local patterns within time-slice. Based on the learned dependency-aware user representation described in Section 4.3, for each current user u_j in one cascade, we perform a pooling operation on its *local* pattern, which composed of the current u_j and predecessor users within a time-slice window t_s. Specifically, we firstly formalize $\boldsymbol{\vartheta} = \{\boldsymbol{\vartheta}(t_0), \ldots, \boldsymbol{\vartheta}(t_j), \ldots, \boldsymbol{\vartheta}(t_L)\}$ as a collection of all local patterns of an information cascade. Each local pattern $\boldsymbol{\vartheta}(t_j)$ is composed of hidden states within a time-slice in Eq. (13), i.e., $\boldsymbol{\vartheta}(t_j) = [\mathbf{h}(t_j - t_s); \ldots; \mathbf{h}(t_j)]$. Note that t_s is a fixed value, and each current node u_j has its corresponding local pattern $\boldsymbol{\vartheta}(t_j)$. And then for each current node u_j, we generate a new representation $\widetilde{\mathbf{h}}_{t_j}$ by conducting a pooling operation on $\boldsymbol{\vartheta}(t_j)$ to aggregate the features in a time-slice t_s. Hence, the total pooling representations of all local patterns $\boldsymbol{\vartheta}$ can be formalized as:

$$\widetilde{\mathbf{H}} := \text{Pool}([\boldsymbol{\vartheta}(t_0); \ldots; \boldsymbol{\vartheta}(t_L)]), \tag{14}$$

where $\widetilde{\mathbf{H}} = \{\widetilde{\mathbf{h}}_{t_0}, \ldots, \widetilde{\mathbf{h}}_{t_L}\}$ and each column of $\widetilde{\mathbf{H}}$ corresponds to the pooling representation of its local pattern.

Merging evolution of local patterns with masked attention. To merge the impacts of local patterns, which capture the evolution rate related to rise/fall behavior, we stack an attention layer. Masked attention mechanism is applied to dynamically aggregate the local pattern embeddings before the current time t_j. The input of the attention layer is the pooling representations of local patterns $\widetilde{\mathbf{H}}$. For jth local pattern, a hidden vector is generated to summarize the evolving process of all previous patterns:

$$\boldsymbol{\upsilon}(t_j) = \frac{\sum_{i=1}^{j-1} f(\tilde{\mathbf{h}}_{t_i}, \tilde{\mathbf{h}}_{t_j}) g(\tilde{\mathbf{h}}_{t_j})}{\sum_{i=1}^{j-1} f(\tilde{\mathbf{h}}_{t_i}, \tilde{\mathbf{h}}_{t_j})}, \tag{15}$$

where $\boldsymbol{\Upsilon} = \{\boldsymbol{\upsilon}(t_1), \ldots, \boldsymbol{\upsilon}(t_j), \ldots, \boldsymbol{\upsilon}(t_L)\}$ is a collection of local dependencies of evolving rate, $g(\cdot)$ is a linear transformation function. The similarity function $f(\cdot,\cdot)$ is specified as:

$$f\left(\tilde{\mathbf{h}}_{t_i}, \tilde{\mathbf{h}}_{t_j}\right) = \exp\left(\tilde{\mathbf{h}}_{t_i}\tilde{\mathbf{h}}_{t_j}^T\right). \tag{16}$$

4.5. Prediction and optimization

Having obtained the global and local dependencies from two-level attention-based framework in Sections 4.3 and 4.4, we constructed a stronger model Hawkesformer with richer coupled topological–temporal information. The two-level transformer-based framework can be used to model the cascade diffusion DGAs. TPP provides a principled treatment by directly absorbing the raw timestamp such that the time information is accurately kept. We combine those topological hidden states and temporal point process for popularity prediction, which extend Hawkes process with a topological horizon in continuous-time domain. Next, we proceed with the detailed optimization and prediction processes.

4.5.1. Prediction

Given an information cascade C_k and its retweet history $\mathcal{H}_k = \{(t_j, v_j, u_j)\}_{j=1}^{L}$ over an observation window $(t_0, t_0 + t_o]$, let $\boldsymbol{s}_t = \{t_0, t_1, \ldots, t_n \mid t_n < t\}$ denote the sequence of event times before current time t. Recall that its conditional intensity function $\lambda(t|\Theta, \boldsymbol{s}_{t_j})$ was specified in Eq. (5). According to the Hawkes theory, the survival function at time t is defined as:

$$S(t \mid \boldsymbol{s}_t) = \exp(-\int_{t_n}^{t} \lambda(\tau)d\tau), \tag{17}$$

where the conditional probability indicates that no event occurs in the window $[t_n, t]$. Then, we can further derive the likelihood of observing an event at an arbitrary time t' $(t' > t_o)$ using:

$$\begin{aligned}P(t' \mid \boldsymbol{s}_t) &= \lambda(t' \mid \boldsymbol{s}_t)S(t' \mid \boldsymbol{s}_t)\\ &= \lambda(t' \mid \boldsymbol{s}_t)\exp\Big(-\int_{t_o}^{t'} \lambda(\tau \mid \boldsymbol{s}_\tau)d\tau\Big).\end{aligned} \tag{18}$$

With the above conditional density function and topological dependency described in Sections 4.3 and 4.4, we combine topological hidden states from the two-level attention architecture

and temporal point process for the final popularity prediction as:

$$\log_2 \hat{y} = \underbrace{\int_{t_o}^{t_p} t \cdot P(t \mid \boldsymbol{s}_t) dt}_{\Lambda} + FC\Big(\mathbf{h}(t_L) \oplus \boldsymbol{\upsilon}(t_L)\Big) \tag{19}$$

where t_p is the prediction time and t_o is the observation time. The global path-aware dependence $\mathbf{h}\left(t_j\right)$ contains the coupled topological–temporal information. The local pattern results $\boldsymbol{\upsilon}(t_j)$ focus on the sudden rise/fall in the current time-slice. The first term Λ is the integration on Hawkes process based on intensity function (cf. Eq. (5)), which indicates the expected number of popularity. The second term is the fusion concatenation of the last hidden states from the global dependency $\mathbf{h}\,(t_L)$ and local patterns $\boldsymbol{\upsilon}(t_L)$ through a fully-connected layer. We combine both of them to get the predicted incremental popularity $\hat{y}$. An overview of Hawkesformer is shown in Algorithm 1.

Algorithm 1: Optimization of Hawkesformer

input : Observed information cascade $C_k(t_o)$ with its retweet history $\mathcal{H}_k = \{(t_j, v_j, u_j)\}_{j=1}^{L}$ and its corresponding time stamps sequence $\boldsymbol{s}_{t_j} = \{t_0, t_1, \ldots, t_n \mid t_n < t_j\}$.

output: Predicted cascade size $S(t_p|\mathcal{H}_k) = \hat{y}_k(t_p)$

Random initialize the shared matrix $\mathbf{U}$ with normal distribution;
Obtain the collection of all user one-hot indices $\mathbf{V}$;
Compute the initial *position-wise* embeddings $\mathbf{UV}$;
Compute temporal encoding $\mathbf{Z}$ (Eq. (6));
Obtain initial user embedding $\mathbf{X} = \mathbf{UV} + \mathbf{Z}$;
while *not converged* **do**
  Train the *global* dependency module to obtain the primary/non-primary attention scores $\mathbf{P}_{\mathbf{score}}/\mathbf{N}_{\mathbf{score}}$ (Eq. (10));
  Obtain attention probability matrix $\mathbf{A}$, contextualized embeddings $\mathbf{E}$, long-term dependency $\mathbf{H}$;
  Train the *local* pattern module to obtain evolution rate encodings $\boldsymbol{\Upsilon}$;
  Obtain intensity function $\lambda(t)$ at time t (Eq. (5));
  Calculate the likelihood $P(t' \mid \boldsymbol{s}_t)$ at time t';
  Predict cascade size $\hat{y}$ via Eq. (19);
end

4.5.2. Optimization

We can express the joint likelihood of observing retweets $\boldsymbol{s}_{t_o} = \{t_0, t_1, \ldots, t_n \mid t_n < t_o\}$ up to an observation time t_o as:

$$\begin{aligned}
\ell(C_k) &= P(\{t_0, t_1, \ldots, t_n \mid t_n < t_o\}) \\
&= \prod_{t_j \in \boldsymbol{s}_{t_o}} f(t_j \mid \boldsymbol{s}_{t_j}) \\
&= \prod_{t_j \in \boldsymbol{s}_{t_o}} \lambda(t_j \mid \boldsymbol{s}_{t_j}) \cdot \exp\Big(-\int_{t_0}^{t_o} \lambda(\tau \mid \boldsymbol{s}_\tau) d\tau\Big) \\
&= \underbrace{\sum_{j=0}^{L} \log \lambda(t_j \mid \boldsymbol{s}_{t_j})}_{\text{retweet}} - \underbrace{\int_{t_0}^{t_L} \lambda(t \mid \boldsymbol{s}_t) dt}_{\text{non-retweet}},
\end{aligned} \tag{20}$$

where $\mathbf{s}_{t_i}$ denotes the retweets up to time t_i of information cascade C_k. Though using the intensity function of Hawkesformer we can predict the size of cascades by simulations. However, the model still has no supervision signals to guide the way towards better performance. In this paper, we utilize the ground truth popularity as the supervision signal. Specifically, suppose we have N information cascades $\{C_1, C_2, \ldots, C_N\}$, our model is optimized by minimizing the Mean Squared Logarithmic Error (MSLE) loss over popularity and maximizing the log-likelihood across all information cascades:

$$\mathcal{L} = \frac{1}{N} \sum_{k=1}^{N} \Big(\big(\log_2 y_k - \log_2 \hat{y}_k\big)^2 - \log \ell(C_k) \Big) \tag{21}$$

Here $\ell(C_k)$ is the joint log-likelihood that needs to be maximized given by Eq. (20). However, it is challenging to compute the non-retweet log-likelihood $\Phi = \int_{t_0}^{t_L} \lambda(t \mid \boldsymbol{s}_t) dt$ in Eq. (20), since there is no closed-form for this integral Φ because of the softplus function. Thus, we resort to applying a numerical integration [80] to approximate the value of the integral, which is faster due to the elimination of sampling process:

$$\widehat{\Phi}_{\text{NU}} = \sum_{j=2}^{L} \frac{t_j - t_{j-1}}{2} \big(\lambda(t_j \mid \mathbf{s}_j) + \lambda(t_{j-1} \mid \mathbf{s}_{j-1})\big) \tag{22}$$

Essentially, the above formula corresponds to the trapezoidal rule for approximating the definite integral corresponding to Φ. Even though approximations build upon numerical integration algorithms are biased, in practice the errors are affordable. This is because the conditional intensity (Eq. (5)) uses softplus as its activation function, which is highly smooth and ensures small bias introduced by linear interpolations (Eq. (22)) between consecutive events.

4.6. Complexity analysis

We close this section with an analysis of the computational complexity of Hawkesformer. For the dynamic DAGs we studied in this paper, how to construct user embeddings efficiently is critical in practical applications. Despite sensitive user information (privacy and biased characteristics), the content of user retweets, structures of the diffusion, as well as temporal features, can be considered as qualified user embeddings.

- *Complexity for computing position-wise embeddings of nodes:* Compared to conventional graph cascade models – especially those random walk-based [20] and GNN-based models [27,52] – the position-wise embedding approach allows Hawkesformer to handle large cascade efficiently. The time and space complexities are both $O(L)$ – i.e., linear in the number of nodes during the observation time window.
- *Complexity for computing global dependency of nodes:* The primary and non-primary assumption in global dependency module are based on self-attention mechanism. The matrix multiplication of self-attention is $O(L^2 \cdot D)$, which is the same as the vanilla self-attention [66] (D corresponds to the dimension of embeddings).
- *Complexity for computing local dependency of nodes:* The second level of Hawkesformer – the merging evolution of local patterns with masked attention in Eq. (15) – is also based on self-attention, and the time complexity is $O(L^2 \cdot D)$, too.

We note that the time and space complexities of fully-connected layers are related to the input dimensions of the hidden states.

5. Experiments

We now discuss in detail the experimental settings and quantitative observations obtained, regarding the benefits of Hawkesformer. We note that in addition to the comparative results with respect to the state-of-the-art, we also conducted an ablation study regarding the impact of the different constituents of Hawkesformer, as well as case studies illustrating the interpretation aspect.

Table 2
Statistics of the datasets.

Dataset	Twitter	Weibo	APS
#Cascade	88,440	119,313	207,685
#Node	490,474	6,738,040	616,316
Avg. popularity	142	240	51
Avg. sequence length	2.196	2.237	3.999
Number of cascades in two different observation settings			
Train (1d/ 0.5 h/ 3y)	9,643	21,294	19,462
Val (1d/ 0.5 h/ 3y)	2,066	4,563	4,170
Test (1d/ 0.5 h/ 3y)	2,066	4,563	4,170
Train (2d/ 1 h/ 5y)	12,734	27,011	33,840
Val (2d/ 1 h/ 5y)	2,729	5,788	7,251
Test (2d/ 1 h/ 5y)	2,728	5,788	7,251

5.1. Datasets

In our evaluation we used three large-scale publicly available information cascade datasets—Twitter, Sina Weibo, and APS (American Physical Society). There are two types of information: tweets in social networks and publications in an academic network, which we use in two settings: (1) predicting the number of retweets of tweets; and (2) forecasting the citation count of academic papers. Both types of cascades are instances of real-world information diffusion processes, providing a relevant performance comparison between Hawkesformer and baselines. In addition, using two different scenarios allows us to verify the generalizability of the proposed model (i.e., avoid the risk of limiting the model to a specific type of application). The statistics of the datasets are summarized in Table 2, and we describe them in detail below:

- **Twitter** hashtag cascade dataset used in this work is the one collected by [77]. It contains public English written tweets published between Mar 24 and Apr 25, 2012. A Hashtag and its adopters form an information cascade. We set the observation time t_o to 1 day or 2 days, and the prediction time t_p to 32 days.
- **Sina Weibo** dataset was collected from the largest microblog platform in China, as reported in [46]. It contains 119,313 tweets posted on Jun 1, 2016, and tracks all retweets of each post within the next 24 h. Each tweet and its retweets form an information retweet cascade. We set the observation time t_o to 0.5 h or 1 h, and the prediction time t_p to 24 h.
- **ASP** dataset is from the American Physical Society (https://journals.aps.org/datasets), and it includes all papers published in 17 APS journals between 1893 and 2017. Each paper and the ones that cite it form an information citation cascade. We set the observation time t_o to 3 years or 5 years, and the prediction time t_p to 20 years.

Distribution aspects. The above datasets face the challenge of data imbalance which, in turn, yields unsatisfactory prediction performance in conventional methods. Our recent study [28] has addressed this issue by introducing a general decoupling framework. We visualize the distribution of three datasets used in our work in Fig. 6 and, as can be seen, they all follow the long-tailed distribution. When learning with long-tailed data, a common challenge is that instance-rich (or majority) cascades dominate the training procedure. The learned regression model tends to perform better on these instance-rich data, while performance is significantly worse for instance-scarce (or minority) cascades. Considering VaCas [26] model as an example, we conduct experiments on three long-tailed datasets. We plot the model prediction and ground truth in Fig. 7, noting that the horizontal and vertical axes are log-scale. The illustration indicates that feeding the heavy-tailed data directly into prediction model makes the instance-rich (or head) data dominate the training procedure, influencing model's predictions to be conservative and distributed in the relatively middle range and lowering the prediction performance. We take the initiative to use decoupling training method [28] to cope with the long-tailed data distribution, and we note that the other details of the experimental configuration are discussed in Section 5.4.

Following previous works [27,46], we filter out cascades whose $|C(t_o)| < 10$ and $|C(t_o)| > 1,000$ and we only select first 1000 participants. We track Twitter hashtags before Apr 10, ensuring at least 15 days during the observation window for each hashtag to grow adopters. Due to the effect of diurnal rhythm in the Weibo dataset, we focus on Weibo tweets posted between 8:00 AM and 6:00 PM, ensuring at least 6 h for retweets growth. As for the APS dataset, we consider papers published between 1893 and 1997—so that each paper has at least 20 years (1997–2017) to gain its citations.

5.2. Baselines

To demonstrate the effectiveness of Hawkesformer for predicting the information cascade popularity, we consider nine baselines from three categories. Recall that Hawkesformer is inspired by Hawkes Process and attention mechanism, which takes topological–temporal coupling into account for cascade modeling. Hence, we selected several strong methods covering Hawkes process based models, graph/recurrent neural networks, as well as attention based models. Moreover, we added feature-based models.

- *Feature engineering-based models*: are widely used for information cascade prediction. Features extracted from cascades are fed into various machine learning models for prediction, e.g., [81] use observed popularity $S_k(t_o)$ to predict $\hat{S}_k(t_p)$ of news articles and online videos, we denote this method as **Feature-P**. In addition, we extract features mentioned in [14,15,39] and feed them into a linear regression model and a two-layer MLP model, and denote these two methods as **Feature-Linear** and **Feature-Deep**. These features include: (1) *structural features*: the number of leaf nodes, the average degree (both in-degree and out-degree), average and max length of retweet path of cascades. (2) *temporal features*: time between original and first participant, cumulative popularity series, mean time between the first half and the second half of participants. (3) *Other features*: We use node ids as *node identity* feature.
- *Statistical model-based:* **SEISMIC** [19] is a statistical and generative cascade popularity prediction approach with an implementation of traditional self-exciting Hawkes point process, which uses the pre-specified power-law function to fit the time decay effect in the intensity function. **DeepHawkes** [46] combines both deep neural network and three interpretable factors (the influence of users, self-exciting mechanism, and time decay) of Hawkes process for prediction. It utilizes RNNs on retweet path to model the self-exciting mechanism.
- *Deep learning-based:* **CasCN** [27] is a hybrid model which stacks GCNs to obtain node embeddings and uses RNNs to model the evolving process of the diffusion. **FOREST** [51] combines reinforcement learning and RNNs to handle the multi-scale cascade prediction problem. **VaCas** [26] employs hierarchical variational autoencoders to embed both

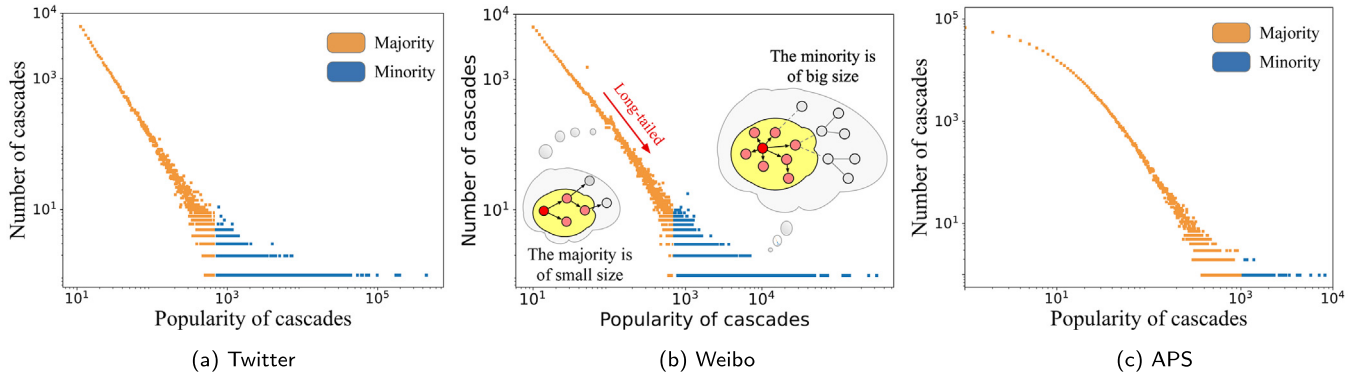


Fig. 6. Visualization of popularity on three datasets. We can see all of the three datasets follows the long-tailed distribution. The majority (instance-rich data) is of small size, while the minority (instance-scarce data) is of extreme size/outliers. The distinction between Minority and Majority is through the x-axis of 1000.

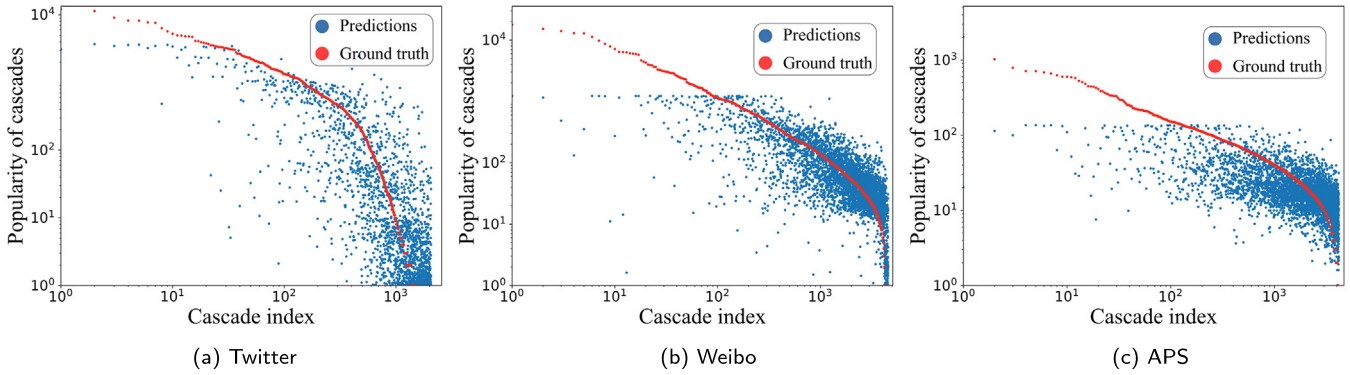


Fig. 7. The ground truth and the predictions on three datasets made by VaCas.

user-level behavior and cascade-level diffusion and utilizes Bi-GRU to generate context-dependent cascade representations. **TempCas** [45] implements a heuristic method with RNNs to embed cascade graph and uses LSTM over a specifically designed attention-based CNN to learn the historical short-term variation trend.

We note that Hawkesformer focuses on information cascade popularity prediction by observing participants during an observation window. Some of the existing cascade prediction models focusing on micro-level prediction [20,68,82,83] (e.g., predicting node activation) or requiring special cascade features [9,84] (e.g., content features and private information), are omitted for comparison.

5.3. Metrics

We use two widely adopted evaluation metrics following previous works [26–28] – mean square logarithmic error (MSLE) and mean absolute percentage error (MAPE) — respectivelly defined as:

$$\text{MSLE} = \frac{1}{N}\sum_{k=1}^{N}\left(\log_2 \Delta\hat{y}_k - \log_2 \Delta y_k\right)^2, \tag{23}$$

$$\text{MAPE} = \frac{1}{N}\sum_{i=k}^{N}\frac{\left|\log_2 \Delta\hat{y}_k - \log_2 \Delta y_k\right|}{\log_2 \Delta y_k}, \tag{24}$$

where N is the total number of cascades, where $\Delta\hat{y}_k = \hat{y}_k(t_p) - \hat{y}_k(t_o)$ is the incremental popularity, $\Delta y_k = y_k(t_p) - y_k(t_o)$ is the increment of ground truth.

5.4. Experimental settings

The node embedding dimension D is 64, batch size is 64, max sequence length is 1000. For global dependency module, we stacked 4 blocks and 4 attention heads. For local pattern module, we stacked 1 block and 4 attention heads. We employ the AdamW optimizer [85] with a linear warm-up, a linear decay learning rate scheduler, and a peak learning rate of $3e^{-5}$. All model trainings are early stopped with a patience of 10 epochs on validation loss. To be fair, the dimension of the embeddings in all deep learning based models is set to 64. We randomly split each dataset into a training set (70%), a validation set (15%), and a test set (15%).

Configurations for decoupling training: Since infrequent cascades (e.g., tweets with many retweets) are few during training, models trained with unbalanced data are prone to under-fit uncommon cascades. However, in practice, we expect predictions not to be affected by extreme values/outliers and the model generalizes well to all sizes of cascades. We consider decoupling the learned representation from the regression perspective to mitigate the long-tailed cascade prediction problem [28]. As mentioned in [28], the best performance is achieved by combining *instance-balanced* sampling strategy and the η-*norm*+SUB decoupling scheme. Hence, we separate the whole training into two stages: representation extractor and regressor. Specifically,

1. The input data are divided into three classes of popularity in decreasing order based on our observation of cascade sizes, i.e., many-shot (20%) represents the top 20% with large popularity in the dataset, medium-shot (60%) represents the 60% with middle popularity in the dataset, and

Table 3
Descriptive statistics of tree long-tailed datasets in decoupling training.

Observation setting	Dataset	Twitter			Weibo			APS		
		Few	*Medium*	*Many*	*Few*	*Medium*	*Many*	*Few*	*Medium*	*Many*
(1d/ 0.5 h/ 3y)	#Cascade	1,929	5,785	1,929	4,259	12,776	4,259	3,893	11,676	3,893
	Range	12–18	19–358	>358	11–33	34–228	>228	11–21	22–65	>65
	Avg. popularity	15	94	1,217	23	90	908	17	37	138
(2d/ 1 h/ 5y)	#Cascade	2,547	7,640	2,547	5,403	16,205	5,403	6,768	20,304	6,768
	Range	12–19	20–305	>305	11–26	27–179	>179	11–18	19–52	>52
	Avg. popularity	15	94	1217	19	70	673	15	30	107

Table 4
Performance comparison between Hawkesformer and the baselines on three datasets, measured by 10 runs of mean MSLEs (lower is better). The background colors denote long-tailed prediction after applying decoupling scheme. ↑ means the percentage of performance improvement (lower metrics), ↓ means the percentage of performance reduction (higher metrics), a short dash (-) means no significant improvement. The bold values indicate better results than other configurations. The bottom right number of MSLE is the standard deviation. A paired t-test is performed, and $*$ denotes the statistical significance ($p < 0.05$) compared to the best baseline method.

Model	Twitter		Weibo		APS	
	1 Day	2 Day	0.5 h	1 h	3 Years	5 Years
Feature-P	$14.792_{\pm 0.11}$	$13.515_{\pm 0.12}$	$4.455_{\pm 0.09}$	$4.001_{\pm 0.10}$	$2.382_{\pm 0.03}$	$2.348_{\pm 0.05}$
Feature-Linear	$9.326_{\pm 0.10}$	$6.758_{\pm 0.09}$	$2.959_{\pm 0.07}$	$2.640_{\pm 0.11}$	$1.852_{\pm 0.02}$	$1.728_{\pm 0.02}$
SEISMIC [19]	$9.401_{\pm 0.06}$	$8.336_{\pm 0.08}$	$4.678_{\pm 0.03}$	$3.934_{\pm 0.15}$	$1.704_{\pm 0.03}$	$1.736_{\pm 0.02}$
Feature-Deep	$7.438_{\pm 0.18}$	$6.357_{\pm 0.14}$	$2.715_{\pm 0.13}$	$2.546_{\pm 0.06}$	$1.844_{\pm 0.03}$	$1.666_{\pm 0.11}$
	↑ 2.1%	↑ 5.9%	↑ 3.8%	↑ 2.4%	↑ 5.7%	↑ 3.0%
DeepHawkes [46]	$6.916_{\pm 0.02}$	$5.311_{\pm 0.05}$	$2.556_{\pm 0.03}$	$2.488_{\pm 0.01}$	$1.612_{\pm 0.02}$	$1.576_{\pm 0.03}$
	↑ 1.5%	↑ 1.9%	↑ 1.2%	↑ 2.7%	↑ 3.2%	↑ 7.0%
CasCN [27]	$6.837_{\pm 0.08}$	$5.086_{\pm 0.05}$	$2.513_{\pm 0.13}$	$2.419_{\pm 0.08}$	$1.562_{\pm 0.07}$	$1.421_{\pm 0.04}$
	↓ 1.1%	↑ 0.3%	↑ 1.2%	-	↑ 2.7%	↑ 2.7%
FOREST [52]	$6.802_{\pm 0.11}$	$5.105_{\pm 0.06}$	$2.477_{\pm 0.04}$	$2.402_{\pm 0.02}$	$1.557_{\pm 0.03}$	$1.376_{\pm 0.09}$
	↑ 0.4%	↑ 1.9%	↑ 2.0%	-	↓ 0.4%	-
VaCas [26]	$6.483_{\pm 0.07}$	$4.944_{\pm 0.14}$	$2.132_{\pm 0.06}$	$2.221_{\pm 0.03}$	$1.354_{\pm 0.03}$	$1.346_{\pm 0.02}$
	↑ 2.1%	↑ 3.3%	↑ 2.4%	↑ 1.5%	↑ 6.6%	↑ 3.1%
TempCas [45]	$6.470_{\pm 0.03}$	$4.881_{\pm 0.02}$	$2.115_{\pm 0.12}$	$2.187_{\pm 0.04}$	$1.351_{\pm 0.01}$	$1.329_{\pm 0.03}$
	↑ 2.8%	↑ 9.6%	↑ 4.0%	↑ 1.1%	↑ 4.7%	↑ 1.8%
Hawkesformer	$\mathbf{6.355}^{*}_{\pm 0.03}$	$\mathbf{4.317}^{*}_{\pm 0.06}$	$\mathbf{1.837}^{*}_{\pm 0.07}$	$\mathbf{2.166}^{*}_{\pm 0.01}$	$\mathbf{1.193}^{*}_{\pm 0.05}$	$\mathbf{1.288}^{*}_{\pm 0.08}$
	↑ 3.1%	↑ 0.3%	↑ 1.5%	-	↑ 1.7%	-

few-shot (20%) represents the last 20% with less popularity. We report the descriptive statistics in Table 3.

2. We first train the backbone to extract information cascade representations (using instance-balanced sampling strategy), which learns the structural or temporal characteristics of information cascades until convergence.
3. Secondly, we fix the parameters of backbone and then fine-tune the regressor (using η-norm decoupling scheme), which combines original prediction values and weighted biases rectified by a specifically designed sub-network SUB.

All the deep-learning baselines (Feature-Deep, DeepHawkes, CasCN, FOREST, Vacas, TempCas), including ours, except for using unmodified original data in a plain training way, are also applied to decoupling framework. The experimental results after applying the decoupling training method are shown in Tables 4 and 5, annotated with background colors.

5.5. Main result

The overall performance comparison between our proposed Hawkesformer and baselines is shown in Tables 4 and 5, from which we can see that Hawkesformer consistently outperforms all the baselines, in terms of MSLE and MAPE, where the results of *Hawkesformer* surpass the best baseline (TempCas) by 13.1%∼16.7%. The superior performance of Hawkesformer lies in its consideration of both global dependencies and local patterns for short-term outbreaks in information cascades, which are jointly modeled by the arrival intensity of Hawkes process and the hierarchical attention framework. We have the following four **O**bservations:

(O1): The performance of Feature-Deep is superior than Feature-Linear, suggesting that other than feature selection, model selection is also important for feature-based cascade prediction. Notably, feature-based methods sometimes have better performance than diffusion-based methods and deep learning methods. However, their performance heavily depends on hand-crafted features that are difficult to select for different scenarios in practice.

(O2): SEISMIC performs poorly since its excitation is independent and additive over the past retweets. It makes a strong and simplified hypothesis on the diffusion mechanism, which may over-estimate the cascade size. Our framework lifts this restriction and uses attention mechanism to model the dependencies between patterns in information cascade such as the order, timing, and the number of the past retweets. As for DeepHawkes, it models the diffusion process by feeding the retweet path into RNNs, which is computationally intensive. It also determines the

Table 5
Performance comparison between Hawkesformer and the baselines on three datasets, measured by 10 runs of mean MAPEs (lower is better). The meaning of background color, ↑, ↓, and a short dash (-) are the same as Table 4. The bold values indicate better results than other configurations. The bottom right number of MAPE is the standard deviation. A paired t-test is performed, and $*$ denotes the statistical significance ($p < 0.05$) compared to the best baseline method.

Model	Twitter		Weibo		APS	
	1 Day	2 Day	0.5 h	1 h	3 Years	5 Years
Feature-P	$0.961_{\pm 0.01}$	$0.983_{\pm 0.01}$	$0.391_{\pm 0.02}$	$0.398_{\pm 0.00}$	$0.316_{\pm 0.01}$	$0.352_{\pm 0.03}$
Feature-Linear	$0.520_{\pm 0.00}$	$0.459_{\pm 0.02}$	$0.258_{\pm 0.01}$	$0.271_{\pm 0.01}$	$0.272_{\pm 0.02}$	$0.291_{\pm 0.00}$
SEISMIC [19]	$0.403_{\pm 0.03}$	$0.434_{\pm 0.01}$	$0.412_{\pm 0.02}$	$0.319_{\pm 0.04}$	$0.332_{\pm 0.00}$	$0.325_{\pm 0.03}$
Feature-Deep	$0.485_{\pm 0.02}$	$0.500_{\pm 0.00}$	$0.228_{\pm 0.05}$	$0.272_{\pm 0.02}$	$0.270_{\pm 0.07}$	$0.282_{\pm 0.04}$
	↑ 2.7%	↑ 3.3%	↑ 3.6%	-	↑ 4.6%	-
DeepHawkes [46]	$0.551_{\pm 0.01}$	$0.502_{\pm 0.00}$	$0.320_{\pm 0.00}$	$0.300_{\pm 0.08}$	$0.266_{\pm 0.04}$	$0.295_{\pm 0.00}$
	↑ 4.0%	↑ 2.2%	↑ 4.4%	-	↑ 1.2%	↑ 3.5%
CasCN [27]	$0.495_{\pm 0.01}$	$0.471_{\pm 0.00}$	$0.306_{\pm 0.06}$	$0.371_{\pm 0.00}$	$0.275_{\pm 0.02}$	$0.281_{\pm 0.02}$
	-	↓ 2.0%	↓ 4.9%	-	↑ 2.7%	↑ 2.7%
FOREST [52]	$0.511_{\pm 0.00}$	$0.493_{\pm 0.04}$	$0.328_{\pm 0.04}$	$0.377_{\pm 0.02}$	$0.278_{\pm 0.00}$	$0.280_{\pm 0.00}$
	↓ 0.7%	-	↑ 1.6%	-	↑ 3.7%	↑ 4.0%
VaCas [26]	$0.437_{\pm 0.00}$	$0.353_{\pm 0.00}$	$0.246_{\pm 0.03}$	$0.250_{\pm 0.01}$	$0.224_{\pm 0.04}$	$0.278_{\pm 0.00}$
	↑ 7.3%	↑ 8.9%	↑ 7.8%	↑ 4.1%	↑ 4.5%	↑ 1.9%
TempCas [45]	$0.406_{\pm 0.04}$	$0.335_{\pm 0.03}$	$0.263_{\pm 0.00}$	$0.237_{\pm 0.03}$	$0.203_{\pm 0.07}$	$0.252_{\pm 0.00}$
	↑ 8.7%	-	↑ 12.8%	↑ 13.6%	↑ 5.1%	↑ 8.0%
Hawkesformer	$\mathbf{0.357}^{*}_{\pm 0.04}$	$\mathbf{0.321}^{*}_{\pm 0.03}$	$\mathbf{0.226}^{*}_{\pm 0.02}$	$\mathbf{0.203}^{*}_{\pm 0.01}$	$\mathbf{0.194}^{*}_{\pm 0.00}$	$\mathbf{0.216}^{*}_{\pm 0.03}$
	↑ 2.8%	↑ 1.4%	-	↑ 1.4%	↑ 0.7%	-

influence of predecessor nodes only on the current node from the retweet path, ignoring the implicit but important nodes out of the path. According to Hawkes theory, any triggered users prior to the current time will influence the arrival rate at the current time. Instead, our proposed Hawkesformer computes the long-term dependency attentions on all infected predecessors, which is more effective in explicitly capturing node influences.

(O3): GNN/RNN-based models generally outperform the Hawkes process-based models due to their capabilities in learning the topological and temporal representations of cascades. However, such hybrid models learn structural and temporal features separately, which cannot intuitively and effectively simulate the whole diffusion process, losing cross-domain information. FOREST [51], VaCas [26] and CasCN [27] use GNNs or graph learning techniques to characterize the structure information within the cascade snapshots. For example, CasCN applies convolutional operation on a sequence of sub-graphs, which is redundant and indirect for modeling the cascade diffusion process; Moreover, these methods use RNNs to model the active nodes, which could lead to error accumulation or gradient explosion/vanishing problems when the cascade length is long. Hawkesformer with self-attention mechanism does not aggregate node information sequentially but generates the cascade embedding at once. Thus, it is expressive and flexible for both long-term and local dependencies (which used to be modeled by RNNs or 1d CNNs), capturing the coupled temporal and topological dependency of cascade from the continuous-time domain.

(O4): TempCas proposes heuristic strategies to sample critical path and extracts graph scale features as supplementary to prevent the loss of other informative features, which cannot fully exploit the diffusion process of DAGs. It concatenates two LSTM based modules and feeds them to an attention layer. Attention in its RNN modules also aims at computing long-term dependencies to alleviate the forgetting problem. This finding is consistent with our argument that modeling long-term dependency is important for diffusion prediction.

5.6. Results on decoupling scheme

In addition to the results under standard training, we show the results after applying the decoupling framework in Tables 4 and 5 (the line with background colors after each model's performance). The experimental setting is shown in Section 5.4. We first re-sample the cascade data using an instance-balanced sampling strategy, and then re-train the regressor to rectify decision boundaries through fine-tuning, enabling the regressor to distinguish different cascade classes. From the results, we can see that under a total of 84 conditions, most models (64/84) enjoyed performance improvements within the decoupling scheme, some models (15/84) showed no significant improvements, and only a small number of models (5/84) encountered performance degradation. These results demonstrate the effectiveness of the decoupling scheme (*instance-balanced* sampling strategy and the *η-norm*+SUB network [28]). The performance degradation mainly occurs for CasCN and FOREST. As implied by [86], data imbalance might not be a major issue when learning high-quality representations. Thus we speculate that the backbone networks of FOREST and CasCN, or their ways of modeling cascade structures, are incapable of learning discriminative representations, and therefore responsible for the performance degradation. It is worth noting that the decoupling scheme does not improve Hawkesformer on four conditions. Possible reasons include (1) performance saturation on Weibo and APS datasets; (2) the local-pattern module in Hawkesformer is able to capture the short-term outbreaks and distinguish abnormal cascades (with extreme values/outliers) and thus learn better cascade representations.

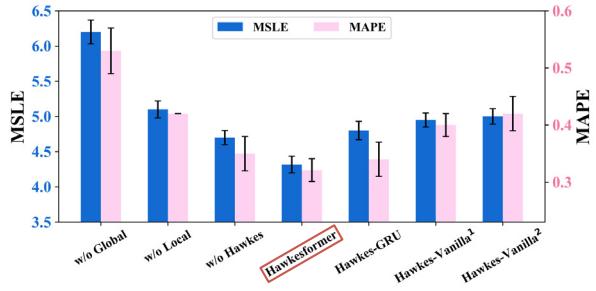
5.7. Ablation study

To further investigate the impact of each component in Hawkesformer, we devised several variants described below.

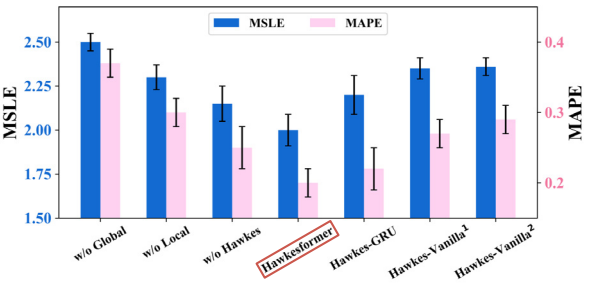
5.7.1. Variants of Hawkesformer

Following are the descriptions of the six variants of Hawkesformer that we built:

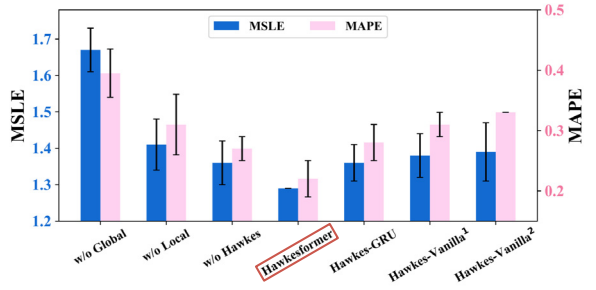
- (v1) `w/o Global` denotes removing *Global Dependency* from Hawkesformer;
- (v2) `w/o Local` denotes removing *Local Dependency* module from Hawkesformer;



(a) Twitter dataset (observation time is 2 days)

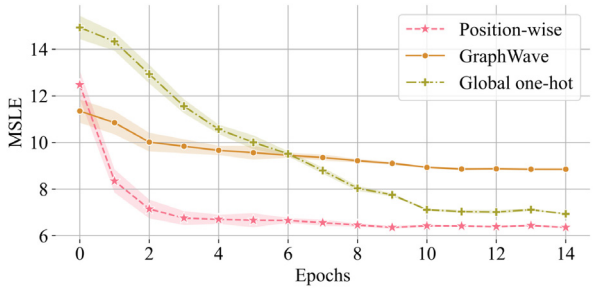


(b) Weibo dataset (observation time is 1 hour)

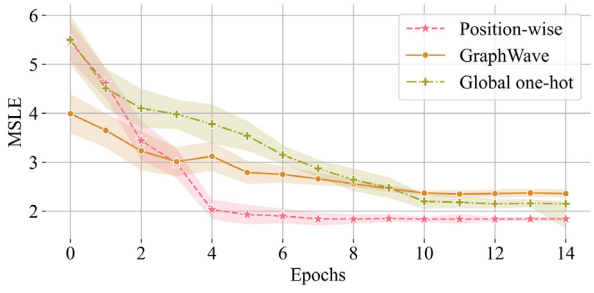


(c) APS dataset (observation time is 5 years)

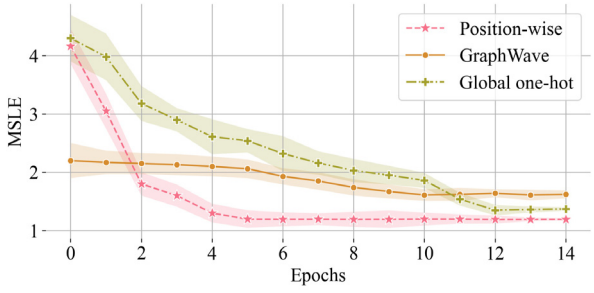
Fig. 8. Ablation experiments on six variants of Hawkesformer. Mean MSLEs and MAPEs are reported by 5 runs with standard deviations.



(a) Twitter dataset (observation time is 1 days)



(b) Weibo dataset (observation time is 0.5 hour)



(c) Weibo dataset (observation time is 3 years)

Fig. 9. The comparison result of proposed position-wise embedding with global one-hot and GraphWave on Hawkesformer. The *X*-axis is epochs, while the *Y*-axis represents the trend of the MSLE metric. Mean MSLEs are reported by 5 runs with standard deviations.

- (v3) `w/o Hawkes` denotes removing Hawkes process and retaining the two-level attention framework from Hawkesformer, which means modeling in discrete-time domain;
- (v4) Hawkes-GRU denotes replacing our devised attention network by a sequential model (two layers of Bi-GRU).
- (v5) Hawkes-Vanilla[1] denotes replacing our *global* dependency (i.e. primary/non-primary assumption) and *local* module with vanilla transformer encoder block.
- (v6) Hawkes-Vanilla[2] is the same as THP [67]. The only difference between Hawkes-Vanilla[2] and Hawkes-Vanilla[1] is that Hawkes-Vanilla[2] (THP) uses global one-hot, rather than our proposed *position-wise user embedding* used in Hawkes-Vanilla[1].

The performance of Hawkesformer variants are shown in Fig. 8. We can see that all variants are inferior to Hawkesformer. Specifically, we have the following observations:
(i) `w/o Global` performs worst among variants, demonstrating that *Global Dependency* module is a crucial and powerful cascade feature extractor, which dynamically learn contextualized cascade representations based on the position-wise user embeddings. The *Local Dependency* module is also important compared to other variants. This verifies the necessity of detecting short-term outbreaks.
(ii) The performance of `w/o Hawkes` slightly decreases. Without absorbing the raw timestamps by Hawkes process, variant `w/o Hawkes` converts event sequence by aggregation based on predefined time interval, which has the unwanted discretization error [87], i.e., the learning is not sensitive to the choice of the interval length for aggregation. Hence, the performance decrease can reflect the importance of TPP in retaining the time information, where the temporal history can be incorporated to facilitate the prediction task.
(iii) The performance of Hawkes-GRU is inferior to Hawkesformer due to the sequential training of RNNs which is notoriously tricky when the sequence is too long. As a consequence, RNNs may not be able to model sophisticated, long, and non-sequential dependencies among cascade input, nor do they process all the cascade events in parallel for efficient computation.
(iv) Vanilla transformer architecture fails to differentiate primary or non-primary paths in cascade structure. It treats the current node equally with infected predecessors and thus is unable to

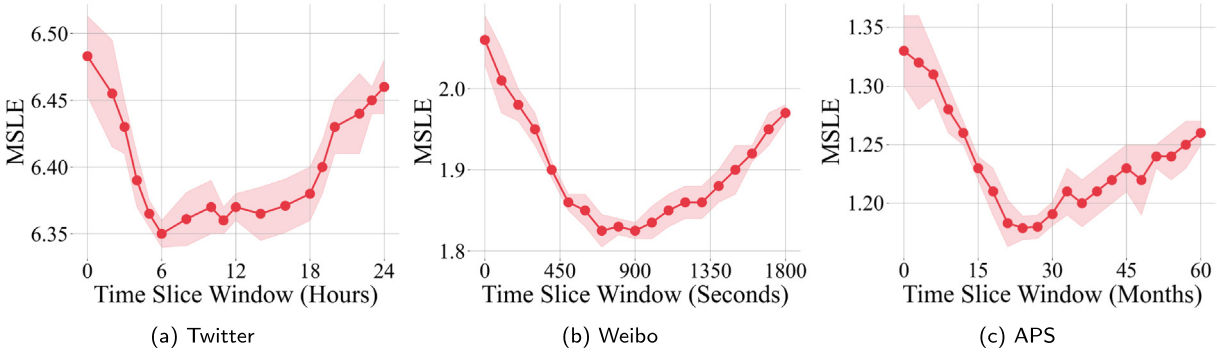


Fig. 10. Impact of the time-slice window t_s on prediction performance, where 24 h, 1800 s, and 60 months are observation times of Twitter, Weibo, and APS datasets, respectively. Mean MSLEs are reported by 5 runs with standard deviations.

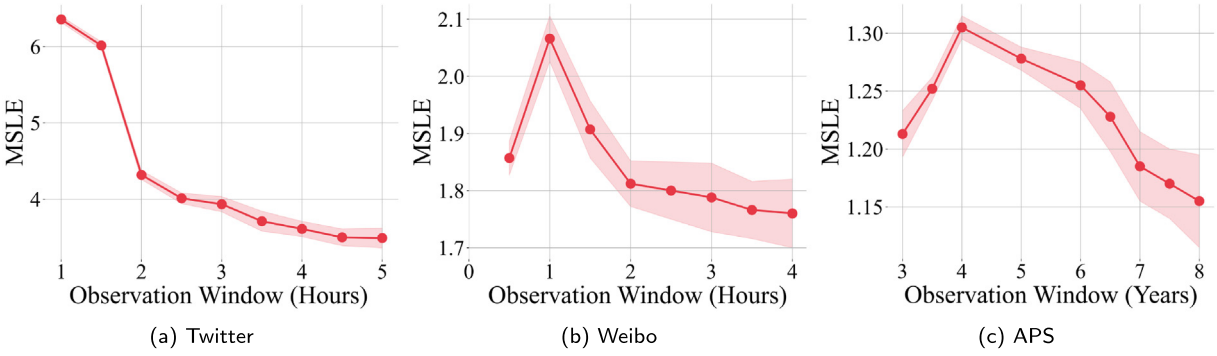


Fig. 11. Impact of sliding observation window t_o on prediction performance of Twitter, Weibo, and APS datasets, respectively. Mean MSLEs are reported by 5 runs with standard deviations.

learn the main diffusion process. It also proves the effectiveness of our primary and non-primary assumption. Specially, the performance of Hawkes-Vanilla^2 is slightly inferior to Hawkes-Vanilla^1, and Hawkes-Vanilla^2 converges slower in training. This mainly owes to the usage of position-wise user embedding in Hawkes-Vanilla^1, which reduces the cost of updating user contextualized embedding and therefore converges much faster than global one-hot embedding (cf. Section 5.7.2).

Note that Hawkes-Vanilla^2 treats data as a simple event sequence, which turns all nodes in graph into "one path" (cf. Section 5.8). On the contrary, the first-level (Global Dependency) module of Hawkesformer considers the impact of different paths—therefore, the topological information exists in it. For one current node, we consider the difference between primary and non-primary paths; for all nodes in the entire generated DAG at the predicted time, their primary paths and non-primary paths are different. Hawkesformer also considers the evolution rate of information cascade in a fixed time slice window to capture the rise/fall behaviors in the second-level (Local Dependency) module. In other words, Hawkesformer can improve the performance of information cascade prediction from the above two aspects.

5.7.2. Position-wise embedding

To reveal the effect of our proposed position-wise embedding, we replace it by a global one-hot vector (as previously used in DeepHawkes [46]), or replace it by GraphWave (as previously used in VaCas [26]). The comparison result is shown in Fig. 9. We can observe that none of two variants are superior than our designed position-wise user embedding, which is more suitable for the dynamic learning based on transformer architecture. With the increase of the number of epochs, the loss (MSLE) of position-wise embedding converges much faster than the other two methods. Moreover, the global one-hot approach requires 0.4 M, 6 M, and 0.6 M individual user embeddings for Twitter, Weibo, and APS, respectively, bringing considerable memory/computation overheads and converging slowly. In contrast, Hawkesformer's position-wise embedding dramatically reduces the cost of updating user embeddings and speeds up the training process. Graph wavelet used in [26] is a static node representation method for learning structural similarity among nodes, which is insufficient to learn dynamic diffusion processes and therefore performs the worst.

5.7.3. Parameter sensitivity of time-slice

For the local pattern module used in Hawkesformer, we further explore the impact of time-slice t_s on prediction performance. As illustrated in Fig. 10, we obtained consistent results on all three datasets. When the time-slice window is small, there are fewer nodes in the same time-slice window. Thus, the diffusion trends captured by local pattern module are rather similar to the trends captured by long-term dependency module, which brings redundant information. When the time-slice window is getting larger, the differences between each window are also small, making the short-breaks hard to be identified. We conclude that an appropriate choice for time-slice window t_s is around 6 h for Twitter, 900 s for the Weibo, and 20 months for the APS.

5.7.4. Pooling function

We further tried three pooling functions in local dependency module and the comparison results is shown in Table 6. We can see that pooling function significantly affects the prediction performance. Specifically: average pooling performs the best in comparison to max/sum poolings on three datasets; max pooling performs poorly, which could be the result of the inadequacy of considering only the most influential node to reflect the evolution rate of an information cascade.

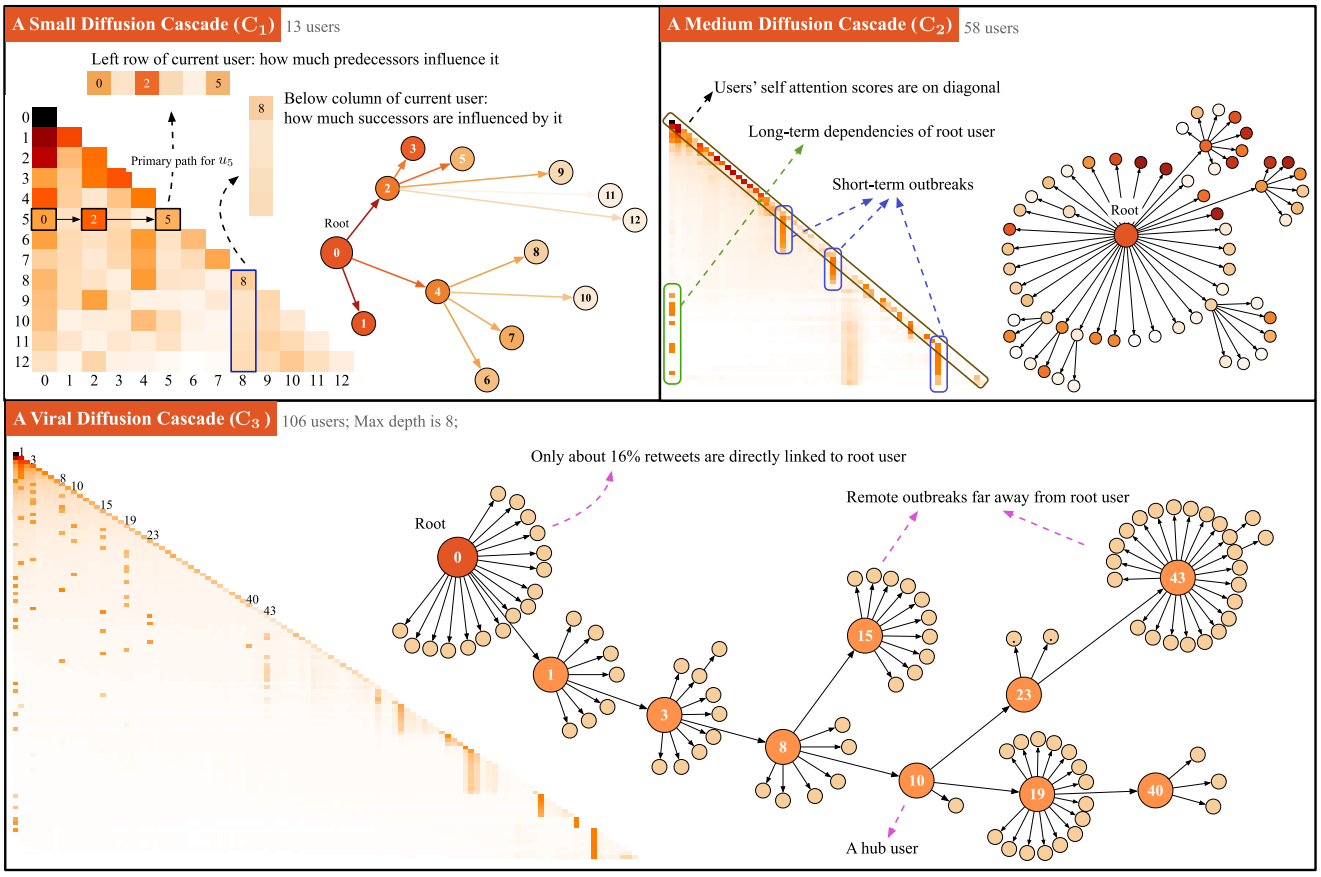


Fig. 12. Attention pattern visualization of three representative information cascades (small, medium, and viral) on the Weibo dataset. The attention scores are averaged from all heads from the last layer of Hawkesformer. For each current node (on the diagonal), its left row scores represent how much predecessors influence it; and the below column scores represent how much its successors are influenced by it.

Table 6
Investigation the MAPE metric of different pooling functions on Hawkesformer. The observation setting is (1 day/0.5 h/5 years) for three datasets.

Pooling	Twitter	Weibo	APS
Max	0.411	0.253	0.226
Sum	0.383	0.237	0.211
Average	0.357	0.226	0.216

5.7.5. Effect of observation time

The most critical parameter in Hawkesformer (also the case for baselines) is the observation window t_o. Here we try out a sliding window and report the changes in MSLE results over time on Hawkesformer. As illustrated in Fig. 11, we can see that as the observation time increases, the amount of the cascade information known about diffusion process and the final size also increases. As we expected, as time elapses, the more cascade information (topological and temporal feature) is available, the more accurate prediction Hawkesformer can make, which is a natural outcome of increasing the training data.

5.8. Case studies on diffusion dependency

The ablation study results have shown that the long-term dependency module plays a vital role in our approach. Here, we plot heatmaps and cascade structures of three cascades in Fig. 12 to visualize the attention probability matrix **A** (extracted from all attention heads from the last layer of Hawkesformer, cf. Eq. (11)) of three representative information cascades on Weibo dataset. Each entry in matrix **A** is ranged in [0, 1] – the darker the entry color, the higher the attention pattern – where each row of matrix **A** represents the attention probability attended from its predecessor nodes in the current history $\mathcal{H}$. The sum of each row is 1.

For information cascade C_1 which is a small diffusion (13 users), our model pays more attention to the primary path as well as the early users participating in the diffusion. For each row of matrix **A**, we can see that the users on the primary path generally have the most significant scores, which supports our primary/non-primary assumption during the information diffusion (similar phenomena are also shown in cascade C_2 and C_3). Hawkesformer can distinguish different paths. For node 9 in cascade C_1, Hawkesformer detects that $0 \rightarrow 2 \rightarrow 9$ is the primary path. Also, node 0 and node 2 have a slightly greater impact on node 9, while other nodes on the non-primary path have a smaller impact on node 9's retweet behavior. For node 8, Hawkesformer detects that $0 \rightarrow 4 \rightarrow 8$ is the primary path, node 0 and node 4 have a greater impact on node 8, while other nodes on the non-primary have a smaller impact on the retweeting behavior of node 8. For information cascade C_2 which is a medium diffusion (58 users), we observed that there are long-term dependencies between root user and successors. This indicates that modeling the long-term dependencies between nodes in dynamic diffusion topology is needed since sequential models such as RNNs are hard to train on long-sequence and face gradient vanishing/exploding issues when the sequence is too long. We also observed several short-term outbreaks, which again verifies our assumption of modeling local patterns for information cascade learning. At last, we showed a viral spreading

cascade C_3 which has 106 users and a max depth of 8. Most of the retweets (∼84%) in this cascade are from indirect users to the root. We found that even "tail" users on the end of the diffusion tree can trigger outbreaks (e.g., users 19 and 43). Some users served as hubs connecting other influential users to bring them into the process of information diffusion.

6. Conclusion

We presented Hawkesformer, a novel methodology for effectively modeling the information cascade diffusion process by linking the merits of Transformer to Hawkes process. At the first level, our designed *global* path-aware attention module can effectively capture the long-term diffusion dependencies between nodes in the dynamic diffusion topology. At the second level, our proposed *local* pooling attention module can accurately capture rise/fall trend of information cascades. Mutual cooperation of two modules enables Hawkesformer to better predict the cascade behaviors and propagation popularity. Besides, we customized a position-wise embedding approach for learning user influences in diffusion DAGs. Unlike existing models that the topological and temporal features separately, we highlighted the loss of cross-domain information, and proposed a new learning scheme exploiting coupled topological and temporal modeling to effectively simulate their entanglement in the diffusion process. Through the attention visualization results, we found that there are long-term dependencies between root user and successors, short-term outbreaks, and users on the primary path generally have the most significant scores, which supports the global–local user dependencies assumption and the primary/non-primary assumption.

An immediate next step of our work is to extend Hawkesformer to incorporate other informative features, e.g., fusing multimodal content features such as texts, topics and figures of microblogs, etc. Another direction of our ongoing work is to incorporate memory-efficient continuous models such as neural ordinary differential equations (NODE) [88] into Hawkesformer, to learn a smoother hidden states. We will also investigate the benefits of applying Hawkesformer to other business-related contexts, e.g., effective advertising and interpretation of viral information spreading such as rumors and epidemics.

CRediT authorship contribution statement

Liu Yu: Writing, Coding – original draft, Visualization, Experiment, Conceptualization, Methodology, Formal analysis, Revision. **Xovee Xu:** Conceptualization, Visualization, Investigation, Writing – review & editing. **Goce Trajcevski:** Writing – review & editing, Investigation, Funding acquisition, Revision. **Fan Zhou:** Supervision, Funding acquisition, Investigation, Resources, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 62176043 and No. 62072077), Natural Science Foundation of Sichuan Province, China (Grant No. 2022NSFSC0505 and No. 2022NSFSC0956), Sichuan Science and Technology Program, China (Grant No. 2022YFSY0006), and the NSF SWIFT, USA grant 2030249.

References

[1] F. Zhou, X. Xu, G. Trajcevski, K. Zhang, A survey of information cascade analysis: Models, predictions, and recent advances, ACM Comput. Surv. 54 (2) (2021).
[2] H. Li, C. Xia, T. Wang, S. Wen, C. Chen, Y. Xiang, Capturing dynamics of information diffusion in SNS: A survey of methodology and techniques, ACM Comput. Surv. 55 (1) (2021) 1–51.
[3] H. Shen, D. Wang, C. Song, A.-L. Barabási, Modeling and predicting popularity dynamics via reinforced poisson processes, in: AAAI, Vol.28, (1) 2014.
[4] D. Gruhl, R. Guha, D. Liben-Nowell, A. Tomkins, Information diffusion through blogspace, in: WWW, 2004, pp. 491–501.
[5] J. Leskovec, M. McGlohon, C. Faloutsos, N. Glance, M. Hurst, Patterns of cascading behavior in large blog graphs, in: Proceedings of the 2007 SIAM International Conference on Data Mining, SIAM, 2007, pp. 551–556.
[6] B. Golub, M.O. Jackson, Using selection bias to explain the observed structure of internet diffusions, Proc. Natl. Acad. Sci. USA 107 (24) (2010) 10833–10836.
[7] M. Gomez-Rodriguez, J. Leskovec, A. Krause, Inferring networks of diffusion and influence, TKDD 5 (4) (2012) 1–37.
[8] S. Masud, S. Dutta, S. Makkar, C. Jain, V. Goyal, A. Das, T. Chakraborty, Hate is the new infodemic: A topic-aware modeling of hate speech diffusion on Twitter, in: ICDE, IEEE, 2021, pp. 504–515.
[9] D. Liao, J. Xu, G. Li, W. Huang, W. Liu, J. Li, Popularity prediction on online articles with deep fusion of temporal process and content features, in: AAAI, Vol. 33, (01) 2019, pp. 200–207.
[10] S. Vosoughi, D. Roy, S. Aral, The spread of true and false news online, Science 359 (6380) (2018) 1146–1151.
[11] K.-Y. Lin, R.K.-W. Lee, W. Gao, W.-C. Peng, Early prediction of hate speech propagation, in: International Conference on Data Mining Workshops, ICDMW, IEEE, 2021, pp. 967–974.
[12] S. Thirumuruganathan, M. Simpson, L.V. Lakshmanan, To intervene or not to intervene: Cost based intervention for combating fake news, in: International Conference on Management of Data, 2021, pp. 2300–2309.
[13] K.Y. Kamath, J. Caverlee, Spatio-temporal meme prediction: learning what hashtags will be popular where, in: CIKM, 2013, pp. 1341–1350.
[14] J. Cheng, L. Adamic, P.A. Dow, J.M. Kleinberg, J. Leskovec, Can cascades be predicted? in: WWW, 2014, pp. 925–936.
[15] S. Mishra, M.-A. Rizoiu, L. Xie, Feature driven and point process approaches for popularity prediction, in: CIKM, 2016, pp. 1069–1078.
[16] F. Davletov, A.S. Aydin, A. Cakmak, High impact academic paper prediction using temporal and topological features, in: CIKM, 2014, pp. 491–498.
[17] D.J. Daley, D. Vere-Jones, An Introduction to the Theory of Point Processes: Volume I: Elementary Theory and Methods, Springer, 2003.
[18] A.G. Hawkes, Spectra of some self-exciting and mutually exciting point processes, Biometrika 58 (1) (1971) 83–90.
[19] Q. Zhao, M.A. Erdogdu, H.Y. He, A. Rajaraman, J. Leskovec, Seismic: A self-exciting point process model for predicting tweet popularity, in: KDD, 2015, pp. 1513–1522.
[20] C. Li, J. Ma, X. Guo, Q. Mei, Deepcas: An end-to-end predictor of information cascades, in: WWW, 2017, pp. 577–586.
[21] J. Wang, V.W. Zheng, Z. Liu, K.C.-C. Chang, Topological recurrent neural network for diffusion prediction, in: ICDM, IEEE, 2017, pp. 475–484.
[22] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, 2013, ArXiv:1312.6203.
[23] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, 2017, ArXiv:1710.10903.
[24] X. Feng, Q. Zhao, Y. Li, AECasN: An information cascade predictor by learning the structural representation of the whole cascade network with autoencoder, Expert Syst. Appl. 191 (2022) 116260.
[25] X. Xu, F. Zhou, K. Zhang, S. Liu, G. Trajcevski, CasFlow: Exploring hierarchical structures and propagation uncertainty for cascade prediction, IEEE Trans. Knowl. Data Eng. (TKDE) (2021).
[26] F. Zhou, X. Xu, K. Zhang, G. Trajcevski, T. Zhong, Variational information diffusion for probabilistic cascades prediction, in: INFOCOM, IEEE, 2020, pp. 1618–1627.
[27] X. Chen, F. Zhou, K. Zhang, G. Trajcevski, T. Zhong, F. Zhang, Information diffusion prediction via recurrent cascades convolution, in: ICDE, IEEE, 2019, pp. 770–781.
[28] F. Zhou, L. Yu, X. Xu, G. Trajcevski, Decoupling representation and regressor for long-tailed information cascade prediction, in: SIGIR, 2021, pp. 1875–1879.
[29] L. Gao, B. Zhou, Y. Jia, H. Tu, Y. Wang, C. Chen, H. Wang, H. Zhuang, Deep learning for social network information cascade analysis: a survey, in: IEEE International Conference on Data Science in Cyberspace, DSC, IEEE, 2020, pp. 89–97.
[30] X. Gao, Z. Cao, S. Li, B. Yao, G. Chen, S. Tang, Taxonomy and evaluation for microblog popularity prediction, ACM Trans. Knowl. Discov. Data (TKDD) 13 (2) (2019) 1–40.

[31] J. Wang, W. Jiang, K. Li, G. Wang, K. Li, Incremental group-level popularity prediction in online social networks, ACM Trans. Internet Technol. (TOIT) 22 (1) (2021) 1–26.
[32] X. Chen, F. Zhou, F. Zhang, M. Bonsangue, Modeling microscopic and macroscopic information diffusion for rumor detection, Int. J. Intell. Syst. 36 (10) (2021) 5449–5471.
[33] C. Gou, H. Shen, P. Du, D. Wu, Y. Liu, X. Cheng, Learning sequential features for cascade outbreak prediction, Knowl. Inf. Syst. 57 (3) (2018) 721–739.
[34] C. Yang, M. Sun, H. Liu, S. Han, Z. Liu, H. Luan, Neural diffusion model for microscopic cascade study, IEEE Trans. Knowl. Data Eng. (TKDE) (2019).
[35] J. Qiu, J. Tang, H. Ma, Y. Dong, K. Wang, J. Tang, Deepinf: Social influence prediction with deep learning, in: KDD, 2018, pp. 2110–2119.
[36] S. Petrovic, M. Osborne, V. Lavrenko, Rt to win! predicting message propagation in twitter, in: ICWSM, Vol. 5, (1) 2011.
[37] H. Li, X. Ma, F. Wang, J. Liu, K. Xu, On popularity prediction of videos shared in online social networks, in: CIKM, 2013, pp. 169–178.
[38] B. Chang, H. Zhu, Y. Ge, E. Chen, H. Xiong, C. Tan, Predicting the popularity of online serials with autoregressive models, in: CIKM, 2014, pp. 1339–1348.
[39] B. Shulman, A. Sharma, D. Cosley, Predictability of popularity: Gaps between prediction and understanding, in: ICWSM, Vol. 10, (1) 2016.
[40] V. Isham, M. Westcott, A self-correcting point process, Stochastic Process. Appl. 8 (3) (1979) 335–347.
[41] M.-A. Rizoiu, Y. Lee, S. Mishra, L. Xie, A tutorial on hawkes processes for events in social media, 2017, ArXiv:1708.06401.
[42] R. Kobayashi, R. Lambiotte, Tideh: Time-dependent hawkes process for predicting retweet dynamics, in: ICWSM, 2016.
[43] M.-A. Rizoiu, L. Xie, S. Sanner, M. Cebrian, H. Yu, P. Van Hentenryck, Expecting to be HIP: Hawkes intensity processes for social media popularity, in: WWW, 2017, pp. 735–744.
[44] Q. Kong, M.-A. Rizoiu, L. Xie, Describing and predicting online items with reshare cascades via dual mixture self-exciting processes, in: CIKM, 2020, pp. 645–654.
[45] X. Tang, D. Liao, W. Huang, J. Xu, L. Zhu, M. Shen, Fully exploiting cascade graphs for real-time forwarding prediction, in: AAAI, Vol. 35, (1) 2021, pp. 582–590.
[46] Q. Cao, H. Shen, K. Cen, W. Ouyang, X. Cheng, Deephawkes: Bridging the gap between prediction and understanding of information cascades, in: CIKM, 2017, pp. 1149–1158.
[47] Y. Wang, X. Wang, R. Michalski, Y. Ran, T. Jia, CasSeqGCN: Combining network structure and temporal sequence to predict information cascades, 2021, ArXiv:2110.06836.
[48] C. Donnat, M. Zitnik, D. Hallac, J. Leskovec, Learning structural node embeddings via diffusion wavelets, in: KDD, 2018, pp. 1320–1329.
[49] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, 2016, ArXiv:1609.02907.
[50] Q. Zhang, R. Luo, Y. Yang, Y. Liu, Benchmarking deep sequential models on volatility predictions for financial time series, 2018, ArXiv:1811.03711.
[51] C. Yang, J. Tang, M. Sun, G. Cui, Z. Liu, Multi-scale information diffusion prediction with reinforced recurrent networks, in: IJCAI, 2019, pp. 4033–4039.
[52] Q. Cao, H. Shen, J. Gao, B. Wei, X. Cheng, Popularity prediction on social platforms with coupled graph neural networks, in: WSDM, 2020, pp. 70–78.
[53] L. Yu, X. Xu, G. Trajcevski, F. Zhou, Linking transformer to hawkes process for information cascade prediction (student abstract), in: AAAI, 2022.
[54] Y. Zhao, N. Yang, T. Lin, S.Y. Philip, Deep collaborative embedding for information cascade prediction, Knowl.-Based Syst. 193 (2020) 105502.
[55] R. Krohn, T. Weninger, Modelling online comment threads from their start, in: 2019 IEEE International Conference on Big Data, Big Data, IEEE, 2019, pp. 820–829.
[56] X. Xu, F. Zhou, K. Zhang, S. Liu, CCGL: Contrastive cascade graph learning, IEEE Trans. Knowl. Data Eng. (TKDE) (2021).
[57] A. Veen, F.P. Schoenberg, Estimation of space–time branching process models in seismology using an em–type algorithm, J. Amer. Statist. Assoc. 103 (482) (2008) 614–624.
[58] E. Errais, K. Giesecke, L.R. Goldberg, Affine point processes and portfolio credit risk, SIAM J. Financial Math. 1 (1) (2010) 642–665.
[59] P. Reynaud-Bouret, S. Schbath, Adaptive estimation for hawkes processes; application to genome analysis, Ann. Statist. 38 (5) (2010) 2781–2822.
[60] G.O. Mohler, M.B. Short, P.J. Brantingham, F.P. Schoenberg, G.E. Tita, Self-exciting point process modeling of crime, J. Amer. Statist. Assoc. 106 (493) (2011) 100–108.
[61] N. Du, H. Dai, R. Trivedi, U. Upadhyay, M. Gomez-Rodriguez, L. Song, Recurrent marked temporal point processes: Embedding event history to vector, in: KDD, 2016, pp. 1555–1564.
[62] J. Yan, Recent advance in temporal point process: from machine learning perspective, SJTU Tech. Rep. (2019).
[63] D. Luo, H. Xu, Y. Zhen, X. Ning, H. Zha, X. Yang, W. Zhang, Multi-task multi-dimensional hawkes processes for modeling event sequences, in: AAAI, 2015.
[64] W. Lian, R. Henao, V. Rao, J. Lucas, L. Carin, A multitask point process predictive model, in: ICML, 2015, pp. 2030–2038.
[65] H. Mei, J. Eisner, The neural hawkes process: A neurally self-modulating multivariate point process, 2016, ArXiv:1612.09328.
[66] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: NIPS, 2017, pp. 5998–6008.
[67] S. Zuo, H. Jiang, Z. Li, T. Zhao, H. Zha, Transformer hawkes process, in: ICML, 2020, pp. 11692–11702.
[68] Q. Zhang, A. Lipani, O. Kirnap, E. Yilmaz, Self-attentive hawkes process, in: ICML, 2020, pp. 11183–11193.
[69] L.-n. Zhang, J.-w. Liu, Z.-y. Song, X. Zuo, W.-m. Li, Z.-y. Liu, Universal transformer hawkes process, in: 2021 International Joint Conference on Neural Networks, IJCNN, IEEE, 2021, pp. 1–7.
[70] L.-n. Zhang, J.-w. Liu, Z.-y. Song, X. Zuo, Temporal attention augmented transformer Hawkes process, Neural Comput. Appl. 34 (5) (2022) 3795–3809.
[71] D. Walther, U. Rutishauser, C. Koch, P. Perona, On the usefulness of attention for object recognition, in: Workshop on Attention and Performance in Computational Vision At ECCV, Vol. 1, Citeseer, 2004.
[72] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, 2014, ArXiv:1409.0473.
[73] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, 2018, ArXiv:1810.04805.
[74] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al., Language models are unsupervised multitask learners, OpenAI Blog 1 (8) (2019) 9.
[75] S.H. Khan, M. Naseer, M. Hayat, S.W. Zamir, F.S. Khan, M. Shah, Transformers in vision: A survey, 2021, ArXiv:2101.01169.
[76] J. Hu, L. Shen, G. Sun, Squeeze-and-excitation networks, in: CVPR, 2018, pp. 7132–7141.
[77] L. Weng, F. Menczer, Y.-Y. Ahn, Virality prediction and community structure in social networks, Sci. Rep. 3 (1) (2013) 1–6.
[78] X. Xu, T. Zhong, C. Li, G. Trajcevski, F. Zhou, Heterogeneous dynamical academic network for learning scientific impact propagation, Knowl.-Based Syst. 238 (2022) 107839, http://dx.doi.org/10.1016/j.knosys.2021.107839.
[79] A. Grover, J. Leskovec, Node2vec: Scalable feature learning for networks, in: KDD, 2016, pp. 855–864.
[80] J. Stoer, R. Bulirsch, Introduction to Numerical Analysis, Vol. 12, Springer Science & Business Media, 2013.
[81] G. Szabo, B.A. Huberman, Predicting the popularity of online content, Commun. ACM 53 (8) (2010) 80–88.
[82] A. Sankar, X. Zhang, A. Krishnan, J. Han, Inf-vae: A variational autoencoder framework to integrate homophily and influence in diffusion prediction, in: WSDM, 2020, pp. 510–518.
[83] Y. Gu, Attentive neural point processes for event forecasting, in: AAAI, Vol. 35, (9) 2021, pp. 7592–7600.
[84] G. Chen, Q. Kong, N. Xu, W. Mao, NPP: A neural popularity prediction model for social media content, Neurocomputing 333 (2019) 221–230.
[85] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, in: ICLR, 2019.
[86] B. Kang, S. Xie, M. Rohrbach, Z. Yan, A. Gordo, J. Feng, Y. Kalantidis, Decoupling representation and classifier for long-tailed recognition, 2019, ArXiv:1910.09217.
[87] A.S. Fotheringham, D.W. Wong, The modifiable areal unit problem in multivariate statistical analysis, Environ. Plan. A 23 (7) (1991) 1025–1044.
[88] R.T. Chen, Y. Rubanova, J. Bettencourt, D.K. Duvenaud, Neural ordinary differential equations, NeurIPS 31 (2018).