Scilight

Transactions on Artificial Intelligence
https://www.sciltp.com/journals/tai

Article

LTVGN: Mastering Predictions of Information Transmissibility in Time-Varying Information Networks

Xinrui Shi and Yupeng Li *

Department of Interactive Media, Hong Kong Baptist University, Hong Kong SAR 000000, China
* Correspondence: ypengl@hkbu.edu.hk

How To Cite: Shi, X.; Li, Y. LTVGN: Mastering Predictions of Information Transmissibility in Time-Varying Information Networks. *Transactions on Artificial Intelligence* **2026**, 2(1), 1–14. https://doi.org/10.53941/tai.2026.100001

Received: 13 September 2025
Revised: 9 October 2025
Accepted: 14 November 2025
Published: 4 January 2026

Abstract: In the era of information overload, various types of information interconnect to form complex networks. To better manage diffusion paths within networks, we propose predicting information transmissibility—the probability of information being transmitted under the influence of other information in the network. Accurate transmissibility prediction has practical applications in recommendation systems and misinformation control, enabling relevant information to reach appropriate audiences while curbing the spread of less useful content. Given the characteristics of information networks, text-attributed graphs provide a natural representation that captures both network structure and content semantics. However, existing text-attributed graph representation methods fail to capture diffusion dynamics and incur high computational costs. Therefore, we propose a novel efficient textual-graph model, Language Temporal Variation Graph Network(LTVGN), to predict transmissibility by capturing time-varying features, structural information and textual information. Our proposed model is evaluated on the citation dataset HEP-TH. The results demonstrate that our model outperforms state-of-the-art models, achieving a low estimation error.

Keywords: information network; transmissibility; text-attributed graph

1. Introduction

Complex information networks, formed by interconnected entities such as social media posts, are the foundation of modern information propagation. Understanding the underlying structure of these networks is essential for tasks ranging from targeted recommendations to misinformation control. In the realm of social influence studies, the classic problem of influence spread estimation focuses on quantifying the aggregate reach of a seed set [1]. Yet, this macroscopic perspective fails to capture entity-specific transmission dynamics, limiting its utility in scenarios that demand granular precision. For example, effectively managing a marketing campaign or curbing a rumor requires knowing not just how many nodes are influenced, but the specific probability that a particular post will be transmitted. Consequently, we propose predicting information transmissibility—a method to calculate the likelihood of transmission for individual entities under the influence of seed nodes [2]. This shift to entity-level prediction is pivotal for optimizing information flow, providing the necessary granularity to strategically amplify desirable content and effectively dampen the reach of undesirable material.

Existing research on influence estimation from an entity perspective primarily focuses on predicting susceptibility—the probability of a user being influenced within a social network [3,4]. Pioneering works such as DeepIS [3] established susceptibility prediction in static networks, while subsequent frameworks like DySuse [4] extended these capabilities to dynamic settings. However, these studies overlook a critical factor in diffusion dynamics: information content. Empirical evidence demonstrates that textual characteristics (e.g., sentiment [5], topic [6], and linguistic style [7]) are primary drivers of propagation in real-world scenarios, such as the virality of health messages or the spread of product opinions. Hence, there is a pressing need to develop models that incorporate both network structure and textual content to capture the complex mechanisms driving information diffusion. With the integration of textual content, information networks can be structured as text-attributed graphs [8], where nodes

represent information items with textual content and edges denote connections (e.g., retweets) between items, with weights signifying influence strength. Existing approaches model these text-attributed graphs using a cascaded LLM-GNN architecture [8], where language models first embed each node's textual information as node representations, after which graph neural networks learn structural patterns based on these representations. While this integration leverages the strengths of both components, it combines numerous parameters, resulting in a significant computational burden and creating an inevitable trade-off between computational efficiency and model performance. Furthermore, current LLM-GNN models inadequately capture diffusion dynamics and temporal information that are crucial for understanding information spread. Therefore, we propose Language Temporal Variation Graph Network (LTVGN), a lightweight temporal-aware framework that effectively captures diffusion dynamics without sacrificing model performance.

Specifically, LTVGN is designed to efficiently learn temporal and structural characteristics through a multi-stage process. To handle textual content, the framework first leverages ChatGPT-4o to extract key information, which is subsequently encoded by BERT. This strategy significantly reduces computational costs by avoiding redundant encoding of raw text. For temporal dynamics, one-hot seed vectors are multiplied with time-weighted adjacency matrices to calculate node activation probabilities, which capture both temporal patterns and graph structure. At the core of our architecture lies the Temporal-Spatial Transformer module, which incorporates positional encoding and attention mechanisms to process the spatiotemporal relationships between nodes. Crucially, to ensure the model remains lightweight, we implement Low-Rank Adaptation (LoRA) [9]. By decomposing weight updates into low-rank matrices, LoRA dramatically reduces the number of trainable parameters while maintaining model expressivity, resulting in faster convergence and lower memory requirements. Our contribution is summarized as follows. To the best of our knowledge, this is the first work to propose a comprehensive framework for estimating information transmissibility under the diffusion model paradigm. Unlike previous approaches that focus on either network structure or content analysis, our model integrates both dimensions to capture the complex interplay between textual content and network topology in driving information spread. To accurately estimate information transmissibility during the diffusion dynamic, we propose LTVGN, a computationally efficient framework that processes both temporal and structural information in text-attributed graphs.

2. Related Work

In this section, we discuss related work on influence spread estimation and text-attributed graph.

2.1. Influence Spread Estimation

Influence spread estimation is a classic problem that estimates the extent of influence from a specified seed set within a network [1]. This fundamental research question has inspired investigations across various network scales. At a fine-grained level, studies such as DeepIS [3] focus on entity-level influence dynamics to predict the probability of individual users being influenced. While [4] extends this to dynamic networks by accounting for topological and feature evolution, current approaches face two critical limitations. First, they predominantly ignore the role of information content. Research in network analysis demonstrates that textual information, including emotional tone [10, 11], thematic presentation [12], and verbal expression [7], serves as a primary driver of propagation, determining which advisories or product evaluations gain widespread attention. Second, regarding structural evolution, previous works often model general dynamic networks where nodes and edges appear or disappear. However, many real-world social systems consist of stable participants with fluctuating relationship strengths. To address these gaps, we propose a model that integrates textual content to estimate influence at an entity-specific level. Furthermore, we specifically formulate the environment as a temporal network, characterized by a static node set with time-varying edge weights, rather than a dynamic network [4]. This formulation more accurately captures social environments where the community remains stable, but the intensity of interactions evolves over time.

2.2. Text-Attributed Graph

Our study models information networks as text-attributed graphs, where nodes represent information items (e.g., social media posts) containing rich textual content, and weighted edges signify the influence strength between them. To process such graphs, recent studies have introduced Large Language Models (LLMs) as encoders within Graph Neural Networks (GNNs) [8]. These approaches typically adopt a cascaded architecture, where LLMs are first employed to extract textual features, followed by GNNs to encode the graph structure. This architecture aims to leverage the superior text understanding capabilities of LLMs to enhance node and edge representations within the GNN framework. The standard pipeline operates as follows:

$$\mathbf{x}_{v_i} = \text{LLM}(d_{v_i}) \tag{1}$$

$$\mathbf{h}_{v_i} = \text{GNN}(\mathbf{X}, \mathcal{G}) \tag{2}$$

where $\mathbf{x}_{v_i}$ denotes the textual feature vector of node v_i, obtained by encoding the associated textual content d_{v_i} via a Large Language Model (LLM). Subsequently, $\mathbf{h}_{v_i}$ represents the final node embedding generated by the Graph Neural Network (GNN), which takes the feature matrix $\mathbf{X} = \{\mathbf{x}_{v_i}\}$ and the graph structure $\mathcal{G}$ as input. This architecture effectively integrates textual semantics with graph structure, producing representations suitable for downstream tasks such as node classification and link prediction.

Large Language Models (LLMs) serve as powerful encoders that transform node-level textual information into feature vectors, thereby equipping Graph Neural Networks (GNNs) with rich semantic representations. Current integration approaches typically fall into two categories: one-step and two-step training [8]. One-step methods (e.g., TextGNN, AdsGNN) implement end-to-end joint training of LLMs and GNNs. While conceptually simple, this approach faces significant hurdles: the substantial memory demands restrict neighbor sampling, thereby limiting structural awareness, and joint optimization often risks convergence to suboptimal solutions. Conversely, two-step training (e.g., SimTeG [13]) decouples the process by adapting LLMs to the graph structure before fixing their parameters for GNN training. Although this staged optimization improves textual feature quality, it incurs higher computational costs. Beyond these architectural trade-offs, a critical limitation persists across both paradigms: existing models typically treat graphs as static snapshots, failing to capture the dynamic nature of information propagation and the evolving influence relationships between nodes. To address this, we propose LTVGN, a novel framework that not only optimizes the fusion of textual and structural information but also effectively models the temporal interactions between entities.

3. System Model

In this section, we introduce diffusion models and formally define our prediction problem. While research on information diffusion has yielded various mathematical frameworks, we focus on the Independent Cascade (IC) model [14] due to its principled formalization of stochastic spreading processes. The IC model characterizes diffusion as a probabilistic process wherein activated nodes independently influence their neighbors based on transmission probabilities assigned to the connecting edges. We adopt this well-established model as our foundation, as it effectively captures the stochastic nature of information transmission while remaining computationally tractable. Building upon this, we then provide a formal definition of the prediction problem, outlining the mathematical objectives for estimating transmission dynamics within text-attributed temporal networks.

Before formulating the problem, we introduce the concept of influence probability. Within an information network, this metric quantifies the likelihood that one information entity will influence another, serving as a proxy for interaction strength. Crucially, this probability is dynamic rather than static. For instance, consider two social media posts, A and B. Initially, they may exhibit a weak relationship based solely on semantic similarity. However, if observations reveal that users engaging with post A subsequently share post B, the influence probability increases significantly. By accounting for these temporal dynamics, our framework accurately captures the evolution of network relationships.

The symbols used in this paper are detailed in Table 1.

Table 1. Symbols in the article.

Symbol	Description
G	Information network
V	Node (information entity) set of G
E	Edge set of G
C	Text set of G
p_{ij}^t	Influence probability from i to j at time t
P_t	Influence strength matrix at time t
S	Seed node (information entity) set
v_i, v_j	Nodes (information entities)
D	An instance of diffusion process
$I_{S,D}^t(v_i\|C)$	The state of v_i given S, D at t, conditioned on C
$\delta_S^t(v_i\|C)$	Information transmissibility of v_i given S at t, conditioned on C

3.1. Independent Cascade Model

In this study, we consider a graph $G = (V, E, C)$, where V, E, and C represent the node set, edge set, and text set, respectively. Each node $v_i \in V$ is associated with textual information $c_{v_i} \in C$, which captures the semantic content of the information entity. To model temporal dynamics, each directed edge $e_{ij} = (v_i, v_j) \in E$ is assigned a time-varying weight p_{ij}^t. This weight represents the influence probability—the likelihood that node v_i will successfully activate node v_j at timestep t. The diffusion process follows the Independent Cascade (IC) model logic. It begins with a predefined set of seed nodes S activated at timestep $t = 0$. At each subsequent timestep $t > 0$, every node v_i that was newly activated at timestep $t - 1$ has a single opportunity to activate each of its inactive out-neighbors v_j, with success determined by the probability p_{ij}^t. Once activated, nodes remain active indefinitely. The process terminates when no new nodes are activated.

3.2. The Transmissibility Prediction Problem

Diffusion processes are inherently stochastic and dependent on both the seed set S and the textual content C. We model the propagation dynamics using a time-dependent influence strength matrix P_t, where each entry p_{ij}^t reflects the probability of node v_i activating v_j at time t, conditioned on the semantic relevance between c_{v_i} and c_{v_j}. Let D denote an instance of the diffusion process initiated by S. For a given instance D, let $I_{S,D}^t(v_i \mid C) \in \{0, 1\}$ represent the activation state of node v_i at time t. Our primary objective is to estimate the transmissibility $\delta_S^t(v_i \mid C)$ of node v_i under seed set S at time t, defined as the expectation:

$$\delta_S^t(v_i \mid C) = E_{D \sim p(D|C)}[I_{S,D}^t(v_i \mid C)] \tag{3}$$

where $p(D \mid C)$ denotes the probability distribution over the space of possible diffusion instances, conditioned on the textual content C.

4. LTVGN Model

Our model consists of four modules (Figure 1): (1) a feature construction module that derives upper-bound activation probabilities for each entity across timesteps while encoding network structure; (2) a language module that extracts entity-specific semantic features; (3) a Temporal-Spatial Transformer designed to jointly capture temporal dynamics and structural dependencies; and (4) a diffusion propagation module that infers entity transmissibility by aggregating neighborhood information. Collectively, these components enable comprehensive modeling (Algorithm 1) of information propagation by integrating structural, semantic, temporal, and spatial aspects.

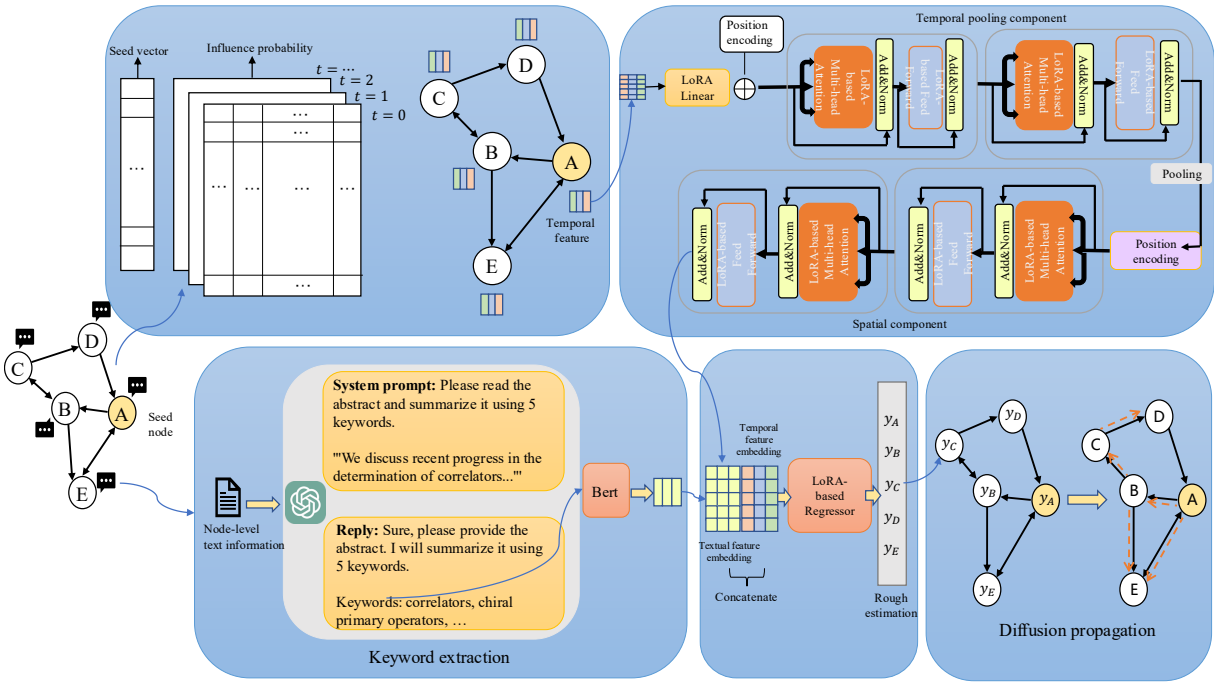

Figure 1. LTVGN structure.

Algorithm 1 LTVGN - Language Temporal Variation Graph Network

Require: Graph $G = (V, E, C)$, Seed set S, Influence strength matrix P_t, Time step t, In-Neighbors $N(v_i)$ of node v_i

Ensure: Node transmissibility predictions $\delta_S^t(v_i|C)$

Feature Construction Module

Initialize seed vector $\mathbf{x}$ where $\mathbf{x}[i] = 1$ if $v_i \in S$, else 0

Construct feature matrix: $\mathbf{X} = [\mathbf{x}, P_1^T\mathbf{x}, P_1^T P_2^T\mathbf{x}, \ldots]$

Language Module

for each node $v_i \in V$ **do**

  Extract keywords: $k_{v_i} = \text{LLM}(c_{v_i}, \text{system\_prompt})$

  Encode textual features: $h_i = \text{Linear}(\text{BERT}(K_{v_i}))$

end for

Temporal-Spatial Transformer Module

1. Temporal Encoding

$X_{\text{emb}} \leftarrow \text{LoRA}_{\text{proj}}(X) + \text{PosEncoding}$

$H_{\text{temporal}} \leftarrow \text{TemporalTransformer}(X_{\text{emb}})$

$Z_{\text{node}} \leftarrow \text{MeanPooling}(H_{\text{temporal}})$ ▷ Aggregate over time dimension

2. Spatial Dependency Modeling

$Z_{\text{spatial}} \leftarrow Z_{\text{node}} + E_{\text{spatial}}$ ▷ Add node identity

$H_{\text{spatial}} \leftarrow \text{SpatialTransformer}(Z_{\text{spatial}})$

3. Regressor

$\hat{y} \leftarrow \text{LoRA}_{\text{reg}}(H_{\text{spatial}})$

Feature Fusion

Combine features: $Z_{\text{combined}} = \text{Concat}(h_i, \hat{y}_i)$

Generate coarse predictions: $\hat{\mathbf{y}} = \text{LoRA}(Z_{\text{combined}})$

Diffusion Propagation Module

for each non-seed node $v_i \in V \setminus S$ **do**

  $\widetilde{\delta}_{v_i}^t = 1 - \prod_{j \in N(v_i)} (1 - P_{ji}^t \widetilde{\delta}_{v_j}^t)$

end for

Training

Compute MAE loss and update parameters via backpropagation

return Transmissibility predictions $\delta_S^t(v_i|C)$

4.1. Feature Construction

This module constructs the necessary features. Seed set is one of the most important factors, as information transmission is dependent on seed nodes. Formally, we represent the seed nodes using a multi-hot vector x, where $x_{[i]} = 1$ if node $i \in S$, and $x_{[i]} = 0$ otherwise. Another key factor is the influence strength matrix P_t, which preserves the graph's topology and the influence probability between any two nodes at timestep t. According to DeepIS [3], these two factors are sufficient for accurate prediction. Therefore, we use them to form the feature matrix.

$$X = [x, P_1^T x, P_1^T P_2^T x, P_1^T P_2^T P_3^T x, \ldots, P_1^T P_2^T P_3^T \ldots P_t^T x] \tag{4}$$

$P_1^T P_2^T P_3^T \ldots P_t^T x$ represents the upper bound of the activation probability for all nodes within t timesteps under the Independent Cascade (IC) model [15]. These values serve as upper bounds because standard matrix multiplication sums probabilities across all propagation paths independently. Consequently, this method does not account for the mutual exclusivity of activation events (i.e., a node cannot be activated twice), leading to an overestimation when multiple paths converge on the same node.

4.2. Language Module

To capture the semantic information of each information entity, we leverage large language models (LLMs) to extract keywords from textual content via prompt engineering, rather than directly embedding the entire text. These extracted keywords are then encoded into dense vector representations using BERT. This keyword-focused approach significantly reduces computational overhead and memory consumption while preserving semantic fidelity.

The inputs of the LLM $\text{LLM}(\cdot)$ are the system prompt sp and textual content c_{v_i}. System prompt sp is a predefined instruction that guides the LLM to generate task-specific outputs [16]. In citation network, we define the system prompt as: Please read the abstract and summarize it using 5 keywords. Then, the key information k_{v_i} of node v_i is extracted as:

$$k_{v_i} = \text{LLM}(c_{v_i}, sp) \tag{5}$$

Given the keywords $K_{v_i} = k_{v_i}^1, k_{v_i}^2, \ldots, k_{v_i}^5$ extracted for node v_i, we employ a pre-trained BERT model followed by a projection layer to obtain the final semantic embedding:

$$h_i = \text{Linear}(\text{BERT}(K_{v_i})) \tag{6}$$

where $\text{BERT}(\cdot)$ encodes the keywords into a 768-dimensional vector, and $\text{Linear}(\cdot)$ maps this output to the desired embedding dimension.

4.3. Temporal-Spatial Transformer

Our Temporal-Spatial Transformer processes spatiotemporal data through a hierarchical encoding scheme designed to capture both temporal evolution and spatial dependencies. Given input $X \in \mathbb{R}^{B\times N\times T}$, where B is the batch size, $N = 27{,}400$ is the number of nodes, and T is the number of historical timesteps, the model operates in three stages: (1) temporal encoding, which extracts temporal patterns for each node independently; (2) spatial encoding, which captures spatial interaction patterns among nodes; and (3)a regression head, which makes coarse node-level predictions.

4.3.1. LoRA Linear Layer

To enable parameter-efficient fine-tuning, we employ Low-Rank Adaptation (LoRA) [9]. Given a pre-trained, frozen weight matrix $W \in \mathbb{R}^{d_{\text{out}}\times d_{\text{in}}}$, LoRA introduces a trainable low-rank update $\Delta W = BA$. Here, the matrices are defined as $B \in \mathbb{R}^{d_{\text{out}}\times r}$ and $A \in \mathbb{R}^{r\times d_{\text{in}}}$, where the rank r satisfies $r \ll \min(d_{\text{out}}, d_{\text{in}})$. For an input x, the modified forward pass is defined as:

$$\text{LoRA}(x; W, B, A) = Wx + \frac{\alpha}{r}BAx \tag{7}$$

where α is a scaling hyperparameter. During training, W remains fixed, and only the low-rank matrices A and B are optimized.

To ensure that the model behavior remains unchanged at the beginning of training (i.e., $\Delta W = 0$), we initialize A using Kaiming uniform initialization [17] and set B to zero. This approach significantly reduces the number of trainable parameters from $d_{\text{out}} \times d_{\text{in}}$ to $r(d_{\text{out}} + d_{\text{in}})$. In our implementation, we apply LoRA with a rank of $r = 4$ and a scaling factor of $\alpha = 1$ to all linear transformations within the model, including projection layers, attention mechanisms, and feed-forward networks. This uniform application ensures architectural consistency and facilitates efficient adaptation in future fine-tuning scenarios.

4.3.2. Temporal Encoding

The temporal encoding module processes T time steps for each node in the information network to extract temporal dynamics. We employ a Transformer-based encoder that captures both short-term fluctuations (step-to-step variations) and longer-term trends (multi-step patterns) through self-attention across all time steps.

Value Projection and Scaling

Each scalar time step value is first projected into a higher-dimensional space to enable rich feature learning. We employ a LoRA-adapted linear projection:

$$X_{\text{proj}} = \sqrt{d_{\text{model}}} \cdot \text{LoRA}(X; W, A, B) \tag{8}$$

The scaling factor $\sqrt{d_{\text{model}}}$ is used to control the magnitude of the projected features, so that their scale is comparable to that of the subsequent positional encodings. This helps maintain a balanced contribution of input features and positional information in the input representation, consistent with standard Transformer architectures [18].

Sinusoidal Positional Encoding

Since a vanilla Transformer encoder does not encode sequence order by itself, we inject temporal ordering information via sinusoidal positional encodings [18]:

$$PE(t, 2i) = \sin\left(\frac{t}{10000^{2i/d_{\text{model}}}}\right), \quad PE(t, 2i+1) = \cos\left(\frac{t}{10000^{2i/d_{\text{model}}}}\right) \tag{9}$$

where $t \in \{0, 1, \ldots, T-1\}$ is the time-step index and $i \in \{0, 1, \ldots, d_{\text{model}}/2 - 1\}$ is the dimension index. The temporal encoded input is then obtained as

$$H^{(0)}_{\text{temporal}} = \text{Dropout}(X_{\text{proj}} + PE) \tag{10}$$

Temporal Transformer Layers

Two Transformer encoder layers process the temporal sequence to capture dependencies across time steps:

$$H^{(\ell)}_{\text{temporal}} = \text{TransformerLayer}_{\ell}(H^{(\ell-1)}_{\text{temporal}}), \quad \ell \in \{1, 2\} \tag{11}$$

For each Transformer encoder layer, a multi-head self-attention mechanism is used to capture dependencies across different positions in the input sequence, followed by a feed-forward network to enhance feature representation. Residual connections and layer normalization are applied after each sub-layer to ensure training stability and facilitate gradient flow.

Temporal Pooling

Following the Transformer layers, we aggregate the temporal information using mean pooling:

$$z_{\text{node}} = \frac{1}{T}\sum_{t=1}^{T} H^{(2)}_{\text{temporal}}[:, t, :] \in \mathbb{R}^{B \times N \times d_{\text{model}}} \tag{12}$$

This operation distills the sequence into a single vector per node, summarizing its behavior over the T-step history. We select mean pooling over max pooling or last-step selection to ensure that information from all time steps contributes equally to the final representation. This is particularly crucial for capturing temporal patterns that are distributed across the entire temporal window rather than localized to specific moments.

4.3.3. Spatial Encoding

Following temporal encoding that generates the initial node representations, we model spatial interactions among the $N = 27{,}400$ nodes using spatial Transformer layers. This stage is critical for capturing dependencies across spatially distributed locations.

Spatial Positional Embedding

Since our spatial nodes lack explicit geometric coordinates (e.g., latitude/longitude), we use learnable positional embeddings to provide spatial context:

$$H_{\text{spatial}} = z_{\text{node}} + E_{\text{spatial}} \tag{13}$$

where $E_{\text{spatial}} \in \mathbb{R}^{1 \times N \times d_{\text{model}}}$ is a trainable parameter. These embeddings are learned during training to encode the implicit spatial structure in the data, allowing the model to discover which nodes are functionally related even without explicit distance metrics.

Spatial Transformer Layers

We model spatial interactions using two Transformer encoder layers, configured with 2 attention heads and a feed-forward dimension of 32:

$$H^{(\ell)}_{\text{spatial}} = \text{TransformerLayer}_{\ell}(H^{(\ell-1)}_{\text{spatial}}), \quad \ell \in \{1, 2\} \tag{14}$$

In each layer, the multi-head attention mechanism enables the model to capture diverse spatial interaction patterns simultaneously. By leveraging self-attention, this component identifies content-based similarities between nodes based on learned feature patterns rather than predefined connections, thereby uncovering implicit relationships. Concurrently, it extracts distributed global patterns across the node feature space while learning to assign differential importance to nodes based on their predictive value. Most notably, this mechanism synthesizes new insights from node combinations, revealing higher-order relationships that remain invisible when examining nodes in isolation.

4.3.4. Regression Head

After spatial encoding, we apply a two-layer MLP independently to each node to generate predictions:

$$h_i = \text{Dropout}\Big(\text{ReLU}\big(\text{LoRA}(H^{(2)}_{\text{spatial},i}; W_{\text{hidden}}, B_{\text{hidden}}, A_{\text{hidden}})\big)\Big) \tag{15}$$

$$\hat{y}_i = \text{LoRA}(h_i; W_{\text{out}}, B_{\text{out}}, A_{\text{out}}) \tag{16}$$

where $W_{\text{hidden}} \in \mathbb{R}^{d_{\text{model}} \times (d_{\text{model}}/2)}$ compresses the spatial feature representation, and $W_{\text{out}} \in \mathbb{R}^{(d_{\text{model}}/2) \times 1}$ maps the hidden state to a scalar prediction. Both layers incorporate LoRA adapters to enable parameter-efficient learning. The subscript $i \in \{1, \ldots, N\}$ indexes the nodes. This design allows node-specific predictions while sharing learned spatial patterns across all nodes. The output $\hat{y} \in \mathbb{R}^{N \times 1}$ contains predictions for all 27,400 nodes simultaneously.

4.4. Fusion Module

The fused representation is then processed by a two-layer network, where both layers are equipped with LoRA adapters to enable parameter-efficient transformation:

$$Z_{\text{combined}} = \text{Concat}(h_i, \hat{y}_i) \tag{17}$$

The fusion mechanism processes this combined representation through a two-layer network with LoRA adaptation for making predictions:

$$Z_{\text{middle}} = \text{ReLU}(\text{LoRA}(Z_{\text{combined}}; W_1, B_1, A_1)) \tag{18}$$

$$\hat{y} = \text{LoRA}(Z_{\text{middle}}; W_2, B_2, A_2) \tag{19}$$

where the LoRA function is defined in Equation (7).

4.5. Diffusion propogation

This module aggregates neighbor information to make fine-grained predictions. Following [3], we employ the approximate stationary distribution $\widetilde{\delta}$ under the IC model. For each node v_i, we consider the sub-network formed by its in-neighbors $N(v_i)$, the set of nodes capable of activating v_i. Aggregating influence from these sub-networks allows for a precise estimation of activation timing. The transmissibility is calculated as:

$$\widetilde{\delta}^t_{v_i} = 1 - \prod_{j \in N(v_i)} (1 - P^t_{ji} \widetilde{\delta}^t_{v_j}) \tag{20}$$

where $\widetilde{\delta}^t_{v_i}$ represents the transmissibility of node v_i at timestep t, and P^t_{ji} denotes the influence probability from neighbor v_j to v_i.

In practice, the exact computation of Equation (20) requires enumerating all possible propagation paths from seed nodes, which has exponential complexity. We therefore adopt an efficient approximation using the static influence matrix initialized at $t = 0$, leveraging the empirical observation that influence probabilities remain relatively stable within short propagation windows. This preserves accuracy while ensuring scalability. By definition, seed nodes are assigned $\widetilde{\delta}^t = 1$ and bypass this aggregation process.

4.6. Training

We adopt the mean absolute error (MAE) as the loss function for the node-level regression task.

$$L_{\text{MAE}} = \frac{1}{M} \sum_{m=1}^{M} \frac{1}{|V|} \sum_{i=1}^{|V|} | \hat{y}_m(i) - y_m(i) | \tag{21}$$

5. Evaluation

5.1. Datasets

We use HEP–TH (high energy physics theory) citation network [19]. We follow previous studies [3,4] to use the largest component network consisting of 27,400 articles and 352,542 citation links.

5.2. Data Preparation

In this section, we detail the data preparation process for computing ground truth labels and the splitting strategy for model training and evaluation.

Due to the unavailability of explicit time-varying influence probabilities in real-world scenarios, we follow established practices from prior literature. We initialize the influence probability at timestep $t = 0$ based on the cosine similarity between article embeddings. To capture the temporal dynamics of influence, we model the probability at subsequent timesteps using an exponential decay function:

$$P(t) = P_0 e^{-\alpha t} \tag{22}$$

where $P(t)$ represents the influence probability at timestep t, and $P_0 = P(0)$ is the initial influence probability derived from semantic similarity. α is the decay constant. We set α to ensure the influence probability decays gradually over a reasonable time horizon, reflecting the diminishing impact of information over time in citation networks.

Ground truth. To simulate the information diffusion process, we employ the Independent Cascade (IC) model with varying initial conditions. We define seed sets with sizes ranging from 100 to 1000 nodes. For each simulation instance, nodes are randomly selected to form the initial seed set that triggers the diffusion.

The propagation follows a stochastic mechanism wherein each newly activated node attempts to activate its inactive neighbors. This activation is determined by comparing the time-varying influence probability of the neighbor at the current timestep against a randomly generated threshold drawn from a uniform distribution $[0, 1]$. A neighbor is successfully activated (labeled as 1) if the influence probability exceeds this threshold; otherwise, it remains inactive (labeled as 0).

Given the inherent randomness of the IC model, we utilize Monte Carlo simulations to ensure statistical robustness. We repeat the diffusion process 1000 times for each configuration and average the results to compute the final transmissibility score for every node. Consequently, the ground-truth dataset consists of a seed vector, a multi-hot vector indicating the seed nodes (1 for seed nodes, 0 otherwise), and the corresponding transmissibility value for each node.

Data splitting. For all diffusion instances, we randomly sample 80% as training data, 10% as validation data, and 10% as test data.

5.3. Baselines

In this section, we introduce the baseline models employed to establish performance benchmarks for our study. As our model performs node-level regression, we compare it against several established graph neural networks and a specialized influence estimation model.

GCN [20]: The Graph Convolutional Network aggregates information from a node and its neighbors using a normalized averaging mechanism, allowing for the smoothing of features across the graph structure. GraphSAGE [21]: This inductive framework generates embeddings by sampling and aggregating features from a node's local neighborhood. Unlike GCN, GraphSAGE concatenates the node's own features with the aggregated neighbor information, explicitly distinguishing self-information from neighbor context. GAT [22]: Graph Attention Networks introduce an attention mechanism that allows nodes to assign different importance weights to different neighbors. By learning these attention coefficients, the model dynamically determines the contribution of each neighbor during aggregation, resulting in more adaptive and context-aware representations. SGC [23]: Simplifying Graph Convolutional Networks reduces the complexity of traditional GCNs. It collapses the non-linearities between GCN layers, effectively transforming the multi-layer non-linear model into a single linear transformation while retaining comparable performance. DeepIS [3]: DeepIS is designed for influence estimation. It constructs a feature matrix combining seed node vectors with fixed influence probabilities. This matrix is processed by a fully connected network for a coarse estimation, followed by an aggregation of in-neighbor features for refinement. In our experimental setting, we utilize the influence probabilities at timestep 0 to construct the input feature matrix, as the original DeepIS architecture is designed for static influence probabilities.

To ensure a fair comparison, we equipped all baseline models with the same feature construction and language modules used in our proposed LTVGN. This standardization ensures that all methods have access to identical

information and that any observed performance differences are attributable solely to variations in model architecture rather than discrepancies in feature extraction.

5.4. Implementation Details

To ensure fairness, all models operate on the feature matrix defined in Equation (1). Each baseline model uses a 2-layer structure with 64 hidden units. Besides, the GAT model employs 4 attention heads.

Our TemporalSpatialTransformer module uses an embedding dimension of 16, substantially reduced from the standard 256 dimensions typical in transformer architectures. This reduction balances computational efficiency with model expressiveness for our large graph (27,400 nodes). The attention mechanism employs 2 heads across 2 encoder layers, ensuring sufficient capacity to capture spatiotemporal dependencies while maintaining computational tractability.

For parameter efficiency, we integrate Low-Rank Adaptation (LoRA) with rank $r = 4$ and scaling factor $\alpha = 1.0$. LoRA is applied to all query, key, value, and output projection matrices within the transformer, achieving significant parameter reduction without sacrificing expressiveness.

In language module, textual features are processed using the pre-trained bert-base-uncased model, which yields 768-dimensional representations. To mitigate the high cost of encoding raw text for 27,400 nodes, we implement a two-stage approach: (1) we utilize GPT-4 to extract exactly five keywords per node via prompt engineering; (2) these keywords are encoded by BERT and subsequently projected to 16 dimensions via a linear layer to match the transformer's embedding size.

5.5. Evaluation Metrics

Mean Absolute Error (MAE): The average of the absolute differences between the predicted $\hat{y}_i$ and actual y_i transmissibility of each node, averaged over multiple instances M.

$$MAE = \frac{1}{M}\sum_{m=1}^{M}\frac{1}{|V|}\sum_{i=1}^{|V|}|\,\hat{y}_m(i) - y_m(i)\,| \tag{23}$$

5.6. Result

Figure 2 shows the mean absolute error across all models. LTVGN consistently outperforms all baselines across different seed sizes.

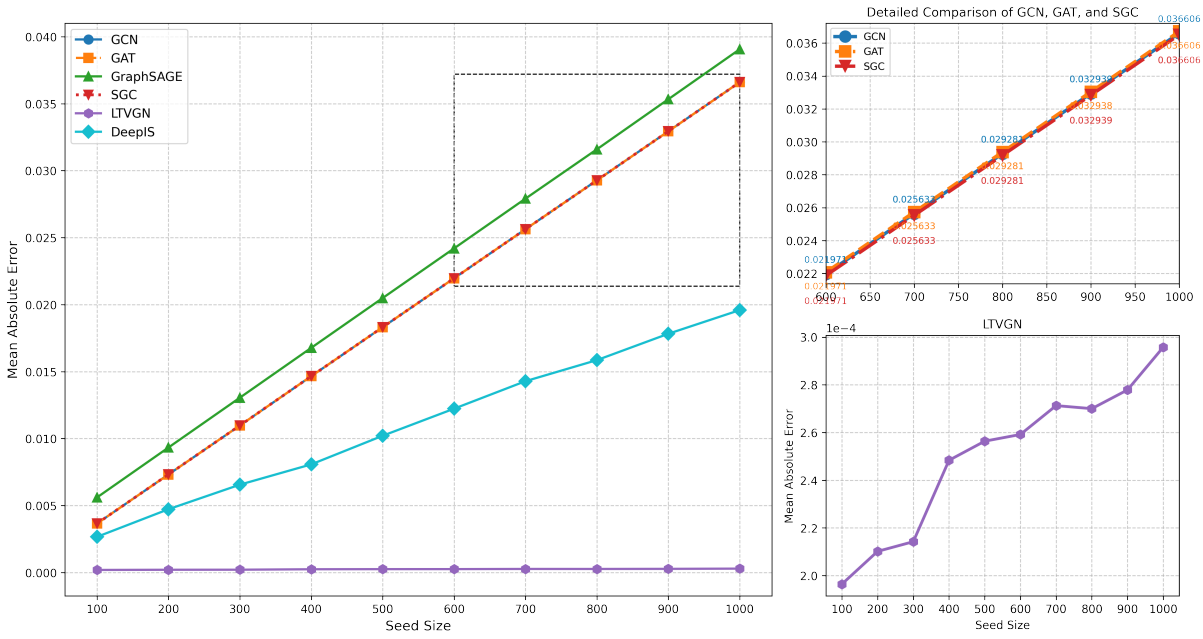

Figure 2. Mean Absolute Error of different models.

Traditional GNN-based methods (GCN, GraphSAGE, GAT, SGC) exhibit suboptimal performance because they treat the sequential upper-bound probabilities at each timestep as independent features, thereby failing to model essential temporal dependencies. Furthermore, their generic aggregation mechanisms (e.g., mean pooling, concatenation, attention) are ill-suited for representing specific diffusion dynamics. While DeepIS outperforms these

traditional GNNs by explicitly modeling diffusion characteristics—utilizing a two-layer fully connected network and a propagation scheme that approximates stationary probabilities under the Independent Cascade model—it remains limited by its inability to capture temporal dynamics or leverage textual semantics. Notably, performance across all baseline methods degrades as the seed size increases, as larger seed sets generate complex diffusion patterns that these models struggle to represent.

In contrast, LTVGN achieves a substantial performance gain, reducing the mean absolute error by an order of magnitude compared to the baselines. This improvement is driven by three synergistic components: (1) the TemporalSpatialTransformer, which effectively learns temporal dependencies from sequential upper-bound calculations while capturing spatial node relationships; (2) the language module, which incorporates node semantics that correlate with diffusion propensity; and (3) a specialized diffusion propagation mechanism that refines predictions by aggregating influence from neighbors. By unifying these elements, LTVGN successfully models the inherently spatiotemporal and semantic nature of information diffusion, enabling it to capture complex propagation patterns that static, structure-only methods cannot represent.

5.7. Ablation Study

To evaluate the contribution of key components in LTVGN, we conduct an ablation study comparing the full model against two variants: one without diffusion propagation and one without the TemporalSpatialTransformer. As shown in Figure 3a, the full LTVGN model demonstrates superior performance, maintaining the lowest Mean Absolute Error across all seed sizes.

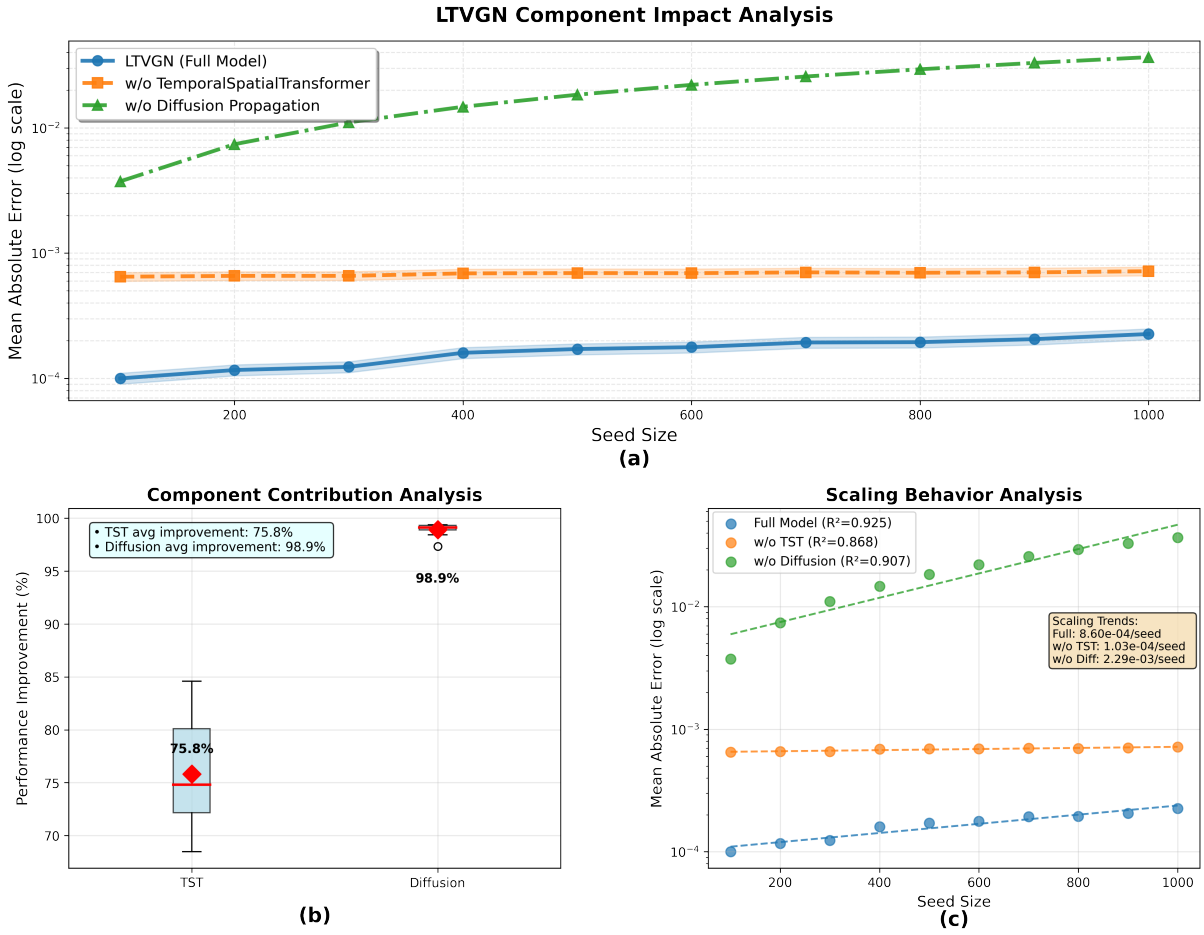

Figure 3. Mean Absolute Error of Ablation Study.

This visual gap is quantified in Figure 3b, which highlights the relative importance of each module. The inclusion of Diffusion Propagation is shown to be foundational, yielding an average performance improvement of 98.9%. Meanwhile, the TST proves to be a significant enhancement, contributing an average improvement of 75.8%.

Figure 3c illustrates the scaling behavior analysis, utilizing linear regression to delineate the trends in prediction error (log scale) relative to seed size across different model variants. The Full Model demonstrates superior robustness, maintaining the lowest error baseline while exhibiting a high goodness-of-fit ($R^2 = 0.925$) and the

most gradual error growth rate (8.60×10^{-4}). In contrast, the variant without diffusion propagation (w/o Diffusion) displays the steepest ascending slope (2.29×10^{-3}), indicating that the diffusion component is critical for mitigating error accumulation as data scale increases. Meanwhile, the removal of the Tempora-Spatial Transformer (w/o TST) results in a consistently higher error plateau compared to the full model, despite a relatively stable slope (1.03×10^{-4}).

5.8. Parameter Analysis

The parameter d defines the dimension of the feature vector constructed in the model's first module. Specifically, it corresponds to the timestep t considered in Equation (4) when tracking node activation by seed nodes. We evaluated the model's performance with $d \in \{3, 4, 5\}$. Figure 4 illustrates the impact of the feature dimension on prediction error across varying seed set sizes.

Contrary to the intuition that longer sequences capture richer temporal information, our results indicate that $d = 3$ consistently yields the lowest mean absolute error across all seed sizes. The error for $d = 3$ ranges from 8.38×10^{-5} (at 100 seeds) to 1.78×10^{-4} (at 1000 seeds). Interestingly, the performance does not degrade linearly with dimension size; $d = 5$ ranks second (error ranging from 9.99×10^{-5} to 2.26×10^{-4}), while $d = 4$ exhibits significantly higher error rates (approximately 9.5×10^{-4} to 1.02×10^{-3}).

These findings suggest that in our scenario, the most relevant information are concentrated within the first three timesteps. Extending the feature dimension beyond this point appears to introduce noise rather than useful temporal dependencies, potentially confusing the model. Furthermore, a smaller d offers advantages in computational efficiency. Consequently, we selected $d = 3$ as the default value, as it provides the optimal balance between predictive accuracy and computational cost.

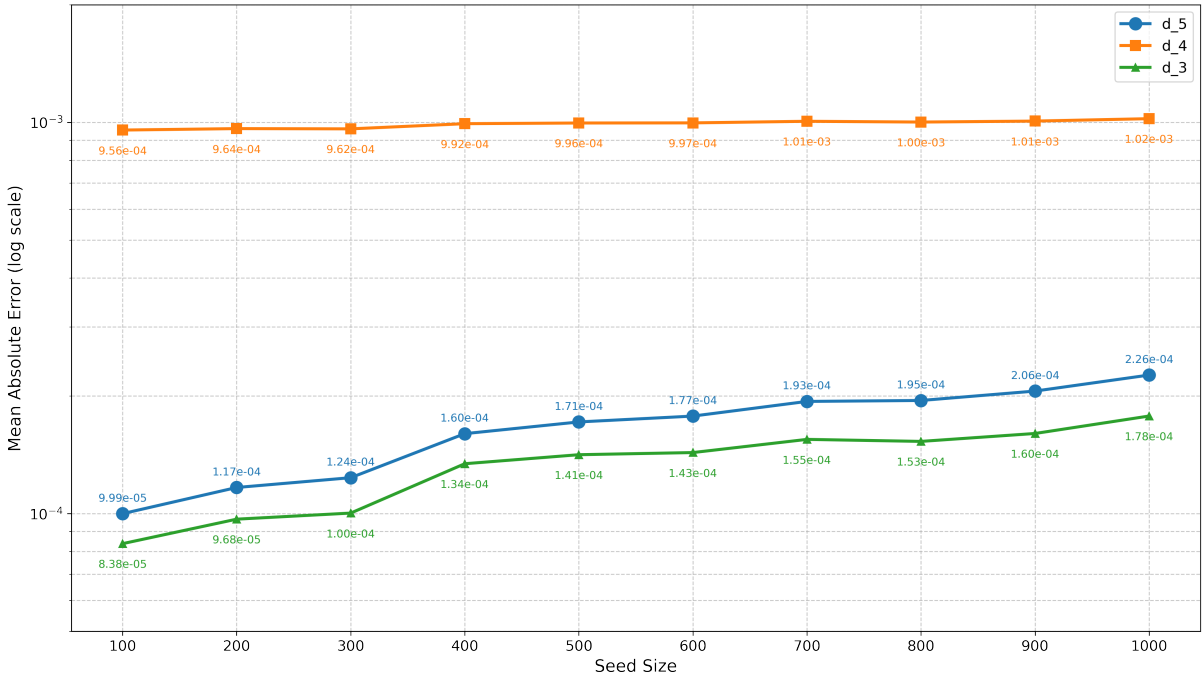

Figure 4. Mean Absolute Error of Parameter Analysis

6. Conclusions

In this paper, we address the problem of estimating information transmissibility in networks with time-varying influence probabilities and textual content. We propose LTVGN, a graph neural network that integrates four key components: (1) a feature construction module that computes upper bound activation probabilities at each timestep while encoding network structure; (2) a language module that encodes node semantics via large language models; (3) a Temporal-Spatial Transformer that jointly models temporal dependencies and spatial relationships; (4) a diffusion propagation module that refines transmissibility estimates using neighborhood aggregation. Experimental results demonstrate that LTVGN consistently outperforms state-of-the-art baselines, reducing mean absolute error by an order of magnitude. This framework enables accurate transmissibility prediction for applications such as recommendation systems and targeted information dissemination.

Author Contributions

X.S.: conceptualization, data curation, original draft, reviewing and editing; Y.L.: conceptualization, supervision, reviewing and editing. All authors have read and agreed to the published version of the manuscript.

Funding

This research was supported by National Natural Science Foundation of China (No. 62202402), Guangdong and Hong Kong Universities "1 + 1 + 1" Joint Research Collaboration Scheme, Project No. 2025A0505000001, the Initiation Grant for Faculty Niche Research Areas 2023/24 (No. RC-FNRA-IG/23-24/COMM/01), the Early Career Scheme (ECS) from the Research Grants Council of HKSAR (HKBU 22202423), the General Research Fund (GRF) from the Research Grants Council of HKSAR (HKBU 12203425), and Germany/Hong Kong Joint Research Scheme sponsored by the Research Grants Council of Hong Kong and the German Academic Exchange Service of Germany (No. G-HKBU203/22).

Institutional Review Board Statement

Not applicable.

Informed Consent Statement

Not applicable.

Data Availability Statement

No data is being made available.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Chen, W.; Yuan, Y.; Zhang, L. Scalable Influence Maximization in Social Networks under the Linear Threshold Model. In Proceedings of the 2010 IEEE International Conference on Data Mining, Sydney, NSW, Australia, 13–17 December 2010; pp. 88–97.
2. Shi, X.; Li, Y. TVGN: Mastering Predictions of Information Transmissibility in Time-Varying Networks. In Proceedings of the International Conference on Computational Data and Social Networks, Bangkok, Thailand, 16–18 December 2024; pp. 148–160.
3. Xia, W.; Li, Y.; Wu, J.; et al. Deepis: Susceptibility Estimation on Social Networks. In Proceedings of the 14th ACM International Conference on Web Search and Data Mining, New York, NY, USA, 8–12 March 2021; pp. 761–769.
4. Shi, Y.; Zhou, J.; Zhang, C. DySuse: Susceptibility estimation in dynamic social networks. *Expert Syst. Appl.* **2023**, *234*, 121042.
5. Ferrara, E.; Yang, Z. Quantifying the effect of sentiment on information diffusion in social media. *PeerJ Comput. Sci.* **2015**, *1*, e26.
6. Berry, C. The Diffusion of Information: The Impact of Sentiment and Topic on Retweets. In Proceedings of the 2020 IEEE International Conference on Big Data (Big Data), Atlanta, GA, USA, 10–13 December 2020; pp. 5636–5638.
7. Džanko, L.; Suitner, C.; Erseghe, T.; et al. Linguistic features influencing information diffusion in social networks: A systematic review. *Comput. Hum. Behav. Rep.* **2025**, *e18*, 100626.
8. Jin, B.; Liu, G.; Han, C.; et al. Large Language Models on Graphs: A Comprehensive Survey. In *IEEE Transactions on Knowledge and Data Engineering*; IEEE: New York, NY, USA, 2024.
9. Hu, E.J.; Shen, Y.; Wallis, P.; et al. Lora: Low-rank adaptation of large language models. *ICLR* **2022**, *1*, 3.
10. Stieglitz, S.; Dang-Xuan, L. Emotions and information diffusion in social media—sentiment of microblogs and sharing behavior. *J. Manag. Inf. Syst.* **2013**, *29*, 217–248.
11. Wang, X.; Lee, E.W. Negative emotions shape the diffusion of cancer tweets: toward an integrated social network–text analytics approach. *Internet Res.* **2021**, *31*, 401–418.
12. Zhao, Y.; Wang, C.; Han, H.; et al. Unfolding the Mixed and Intertwined: A Multilevel View of Topic Evolution on Twitter. In Proceedings of the International Conference on Advanced Data Mining and Applications. Dalian, China, 21–23 November 2019; pp. 359–369.
13. Duan, K.; Liu, Q.; Chua, T.S.; et al. Simteg: A frustratingly simple approach improves textual graph learning. *arXiv* **2023**, arXiv:2308.02565.

14. Lu, F.; Zhang, W.; Shao, L.; et al. Scalable influence maximization under independent cascade model. *J. Netw. Comput. Appl.* **2017**, *86*, 15–23.
15. Zhou, C.; Zhang, P.; Guo, J.; et al. Ublf: An upper bound based approach to discover influential nodes in social networks. In Proceedings of the 2013 IEEE 13th International Conference on Data Mining, Dallas, TX, USA, 7–10 December 2013; pp. 907–916.
16. Li, Z.; Peng, B.; He, P.; et al. Guiding large language models via directional stimulus prompting. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 62630–62656.
17. He, K.; Zhang, X.; Ren, S.; et al. Delving Deep into Rectifiers: Surpassing Human-Level Performance on Imagenet Classification. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1026–1034.
18. Vaswani, A.; Shazeer, N.; Parmar, N.; et al. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.
19. Leskovec, J.; Kleinberg, J.; Faloutsos, C. Graphs Over Time: Densification Laws, Shrinking Diameters and Possible Explanations. In Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, Chicago, IL, USA, 21–24 August 2005; pp. 177–187.
20. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv* **2016**, arXiv:1609.02907.
21. Hamilton, W.; Ying, Z.; Leskovec, J. Inductive representation learning on large graphs. *arXiv* **2017**, arXiv:1706.02216.
22. Velickovic, P.; Cucurull, G.; Casanova, A.; et al. Graph attention networks. *arXiv* **2017**, arXiv:1710.10903.
23. Wu, F.; Souza, A.; Zhang, T.; et al. Simplifying Graph Convolutional Networks. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 6861–6871.