Enhancing Dynamic Text-Attributed Graph Modeling via Instruction Tuning

Weili Shi
University of Virginia
rhs2rr@virginia.edu

Haoming Li
University of Texas at Dallas
dal170519@utdallas.edu

Sheng Li
University of Virginia
vga8uf@virginia.edu

Abstract

Unlike conventional static graphs, dynamic graphs evolve over time as their links change. Recent benchmarks, such as the Dynamic Text-attributed Graph Benchmark, further enable the study of dynamic graph learning with rich textual information on nodes and edges. The success of graph instruction tuning on static text-attributed graphs motivates us to extend this approach to dynamic graphs. However, previous works suggest that directly using LLMs as predictors on dynamic graphs often underperforms traditional dynamic graph models. To address this issue, we propose a novel two-stage training framework consisting of **D**ynamic **T**ext-attributed **G**raph **I**nstruction **T**uning (**DTGIT**) followed by standard supervised training. Unlike prior approaches that employ instruction tuning to enhance LLMs' understanding of graph structures and directly use LLMs as predictors for downstream tasks, our proposed instruction tuning framework serves as a warm-up stage before standard supervised training. Specifically, during the instruction tuning stage, the embeddings produced by the dynamic graph model are aligned with temporal semantics in the natural language space, while the LLM itself is not directly used as a predictor for downstream tasks. In the second stage, the dynamic graph model is initialized with the knowledge acquired during instruction tuning and further fine-tuned in a supervised manner for downstream tasks. Extensive experiments demonstrate that existing dynamic graph models (e.g., DyGFormer) can achieve improved performance on the downstream tasks, particularly on link prediction, when trained with our framework.

1 Introduction

In contrast to traditional static graphs, where the connections between nodes remain static, dynamic graphs feature evolving relationships between nodes over time. These dynamic graphs

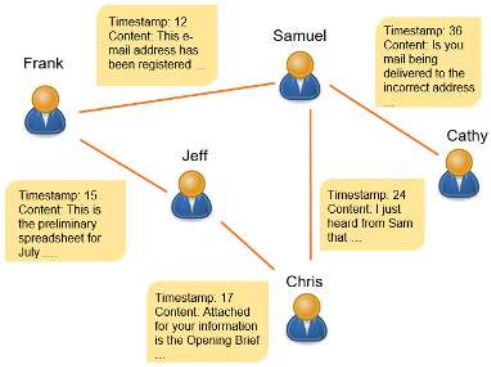


Figure 1: The illustration of a dynamic text-attributed graph. The graph represents email communications between employees over time. There is rich textual information (e.g., names of employees) in the graph.

are common in real-world applications, such as social networks, where interactions between users change continuously. To capture the temporal dependencies in such graphs, many temporal graph learning methods (Yu et al., 2023; Tian et al., 2023; Li et al., 2025b; Shi et al., 2026a) have been developed. These approaches aggregate both structural and temporal information from neighboring nodes, allowing them to effectively model evolving behaviors of nodes within the dynamic graph and make accurate predictions about future interactions.

Similar to text-attributed graphs, where rich textual information is associated with nodes and edges in static graphs, recent work by Zhang et al. (Zhang et al., 2024a) introduced the Dynamic Text-Attributed Graph Benchmark (DTGB) to facilitate the exploitation of textual information for temporal dependency modeling. As illustrated in Figure 1, each dataset in this benchmark is augmented with semantic textual descriptions that characterize both nodes and their interactions over time. Motivated by the strong generalization and interpretability capabilities demonstrated by graph instruction tuning methods, such as GraphGPT (Tang et al., 2024) and LLaGA (Chen et al., 2024a), across diverse downstream tasks on text-attributed graphs, it

Findings of the Association for Computational Linguistics: IJCNLP-AACL 2026, pages 1314–1329
November 6-10, 2026 ©2026 Association for Computational Linguistics

is appealing to extend these approaches to dynamic text-attributed graph settings. However, applying graph instruction tuning approaches to dynamic graphs introduces several inherent challenges. First, these methods are primarily designed to encode the structural information of static graphs and therefore do not readily accommodate dynamic graphs. Moreover, recent studies (Lei et al., 2025; Zhang et al., 2024d, 2025b) indicate that naively adopting LLMs as predictors or agents often leads to inferior performance compared to conventional dynamic graph models. Thus, leveraging the benefits of embedding alignment through instruction tuning while mitigating the limitations of directly applying LLMs to dynamic graph tasks remains a non-trivial challenge.

To address these challenges, we propose a novel training framework that enhances existing dynamic graph models by integrating textual information to better capture evolving patterns. Our framework consists of two stages: **D**ynamic **T**ext-attributed **G**raph **I**nstruction **T**uning (**DTGIT**), which leverages temporal dependencies in dynamic graphs as natural language instructions, followed by standard supervised training. In the first stage, evolving event data in dynamic graphs are encoded in a form compatible with LLM inputs. Compared to static graphs, the structural information of dynamic graphs is often more amenable to LLMs, as temporal interactions can be naturally expressed in natural language. For instance, as shown in Figure 1, consider the event tuple (u, v, t), where u and v represent entities such as *Frank* and *Jeff*, respectively. This interaction can be easily translated into a natural language statement like "*Frank sent an email to Jeff at timestamp 15*", which can be directly processed by LLMs. Specifically, an existing dynamic graph model and a projector are employed to generate token-level embeddings for node entities. These node token embeddings, along with contextual information from the event, are then incorporated into a specifically designed instruction to guide the LLM to determine whether a given node pair constitutes a positive or negative example.

Notably, during instruction tuning, only the dynamic graph model and the projector are trainable, and the LLM remains frozen. Moreover, our proposed instruction tuning differs from the prior graph alignment tuning methods such as GraphGPT (Tang et al., 2024) and LLaGA (Chen et al., 2024a). First, prior methods primarily focus on instruction tuning on traditional static graphs, which cannot be directly adapted to dynamic graph settings. In contrast, our proposed approach explicitly targets alignment tuning for dynamic graphs. Unlike prior works that employ instruction tuning to enhance the structural understanding of LLMs and directly use LLMs as predictors for downstream tasks, our approach leverages instruction tuning to align model embeddings with temporal patterns expressed in natural language. The acquired semantic knowledge is then transferred to standard supervised training. This is achieved by aligning node embeddings with the behavioral semantics of entities in the natural language space (i.e., having the LLM answer "Yes" or "No" for positive or negative examples, respectively). With the designed negative examples, the dynamic graph model is also encouraged to identify the recurring patterns in dynamic graphs.

While the first stage focuses on language-level alignment, the resulting dynamic graph model is not directly suitable for downstream tasks due to the mismatch between natural language outputs and probabilistic predictions. Therefore, we introduce a second stage that applies standard supervised training. In this stage, temporal dependencies are modeled in the probability space using binary cross-entropy loss, assigning higher scores to positive examples and lower scores to negative ones. The trained model is subsequently evaluated on downstream tasks.

Compared to prior works (Zhang et al., 2025b; Wang et al., 2025; Lei et al., 2025) on dynamic text-attributed graph modeling, our approach is fundamentally different in methodology. Existing methods typically focus on either extending existing dynamic graph models with new modules or employing LLMs directly for prediction. For example, GraphAgent-Dynamic (Lei et al., 2025) adopts a multi-agent LLM system as the predictor, while another line of work, such as CROSS (Zhang et al., 2025b), introduces new model components to fuse temporal semantics with structural information for representation learning. LKD4DyTAG (Roy et al., 2025) proposes a novel distillation framework to acquire temporal information embedded in edges from LLMs. In contrast, our work proposes a novel training paradigm that goes beyond conventional supervised learning by leveraging the alignment induced through instruction tuning to enhance the modeling of evolving patterns in dynamic graph models. Extensive experiments demonstrate that,

when trained with our proposed framework, existing dynamic graph models generally achieve improved performance on downstream tasks such as link prediction and node retrieval. These results indicate the effectiveness and generality of our training paradigm for dynamic text-attributed graphs. The contributions of our work are summarized as follows:

- We propose a novel two-stage training framework for dynamic text-attributed graphs, which comprises Dynamic Text-attributed Graph Instruction Tuning (DTGIT) and a subsequent standard supervised training stage.
- Unlike prior graph instruction-tuning approaches designed for static graphs or methods that directly use LLMs as predictors, we employ instruction tuning as a warm-up stage prior to supervised training, enabling dynamic graph models to acquire semantic knowledge through behavior alignment.
- Extensive experiments on multiple dynamic text-attributed graph benchmarks demonstrate that our framework generally improves the performance of existing dynamic graph models on downstream tasks, including link prediction and node retrieval.

2 Related Work

LLMs for graph learning. Inspired by the remarkable success of large language models (LLMs) in natural language processing, there has been growing interest in extending LLMs to graph-based applications. This interest is particularly strong in scenarios where structured graph data is paired with rich textual information, such as e-commerce networks and academic networks, which are ubiquitous in real-world settings. Depending on how LLMs interact with graph neural networks (GNNs), existing approaches can be broadly categorized into several lines of work. The first line of work (Liu et al., 2024a; Zhao et al., 2023; He et al., 2024; Chen et al., 2024b; Pan et al., 2024) adopts LLMs as feature augmenters, where textual information associated with nodes or edges is processed by LLMs to generate enriched node representations or pseudo-labels, which are then used to improve the training of GNNs. For example, Liu et al. (Liu et al., 2024a) proposed the OFA framework, which bridges LLMs and downstream graph tasks through a nodes-of-interest mechanism. Another line of work (Tang et al., 2024; Chen et al., 2024a; Zhang et al., 2024b; Wang et al., 2024a,b; He et al., 2025) focuses on encoding graph structures into token sequences that are compatible with LLM inputs. These methods typically perform alignment or instruction tuning to bridge graph structures and textual semantics, enabling LLMs to better comprehend and reason over structured graph data. A related line of study (Brannon et al., 2024; Li et al., 2023; Fang et al., 2025) leverages self-supervised learning to correlate textual semantics with graph structures. These approaches often employ graph-centric contrastive learning objectives to align embeddings produced by language models and graph neural networks. Finally, some studies (Wang et al., 2023; Fatemi et al., 2024; Dai et al., 2025; Zhang et al., 2024d; Tang et al., 2025) explore the feasibility of using LLMs alone to understand graph structures and perform inference on downstream tasks, without relying on dedicated GNNs.

LLMs for dynamic text-attributed graphs. Similar to static text-attributed graphs, the emergence of pioneering benchmarks (Zhang et al., 2024a) for dynamic text-attributed graphs has stimulated increasing interest in integrating large language models (LLMs) into dynamic graph modeling. One line of work (Zhang et al., 2025b; Xu et al., 2026), including CROSS (Zhang et al., 2025b) and DyGRASP (Wang et al., 2025), leverages the reasoning capabilities of LLMs to extract temporal semantics from textual information and introduces novel modules to fuse these semantics with structural and temporal graph information. Another line of research, such as GraphAgent-Dynamic (Lei et al., 2025), explores the use of LLMs as direct predictors for downstream tasks on dynamic text-attributed graphs through multi-agent systems. However, empirical results suggest that, on most tasks, LLM-only approaches tend to underperform traditional dynamic graph models.

3 Preliminary

Dynamic Text-attributed Graphs. Dynamic text-attributed graphs are represented as a sequence of non-decreasing chronological interactions. Formally, it can be represented as $\mathcal{G} = \{(u_1, v_1, t_1), (u_2, v_2, t_2), \dots\}$, with $t_1 \leq t_2 \leq \dots$, where $u, v \in \mathcal{N}$ signify the source node and destination node, respectively. Each node u is associated with a textual description $d_u \in \mathcal{D}$ and the

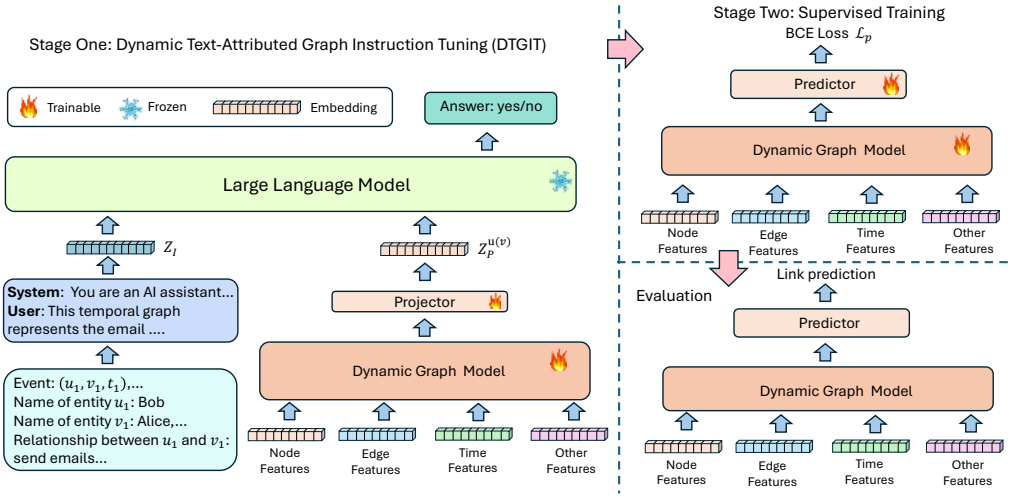


Figure 2: Illustration of our proposed two-stage training framework. In the first stage, dynamic graph model is guided to learn temporal patterns of the dynamic graph assisted by LLM. In the second stage, the model is further trained using standard supervised learning. Finally, the trained model is evaluated on downstream tasks.

System: You are an AI assistant tasked with modeling the evolution of a temporal graph.
User: This temporal graph captures email communications among employees. Each sequence of graph tokens begins with a specific employee, followed by tokens representing others they recently communicated with before timestamp 100.
Given the sequences for Bob and Alice, determine whether there is an email communication between them at timestamp 100.
<temporal_token><temporal_token>< temporal_token>...
<temporal_token><temporal_token>< temporal_token>...
Assistant: Yes/No.

Figure 3: An example of designed instructions from the Enron dataset. The instruction includes system information and user information. The LLM is tasked with determining if the given nodes represent positive or negative interactions.

relationship in the ith event is characterized by a textual description $r_i \in \mathcal{R}$. Note that there are no raw features associated with the nodes and edges in the dynamic text-attributed graphs.

Problem Formalization. Given a source node u, a destination node v, a timestamp t, and the historical interactions before t, i.e., $\{(u', v', t') \mid t' < t\}$, representation learning on dynamic text-attributed graphs aims to learn time-aware representations. The effectiveness of the learned representations is evaluated on downstream tasks such as future link prediction and destination node retrieval.

4 Methodology

4.1 Dynamic Text-Attributed Graph Instruction Tuning

During instruction tuning, given the dynamic graph context and embeddings produced by the dynamic graph model, we construct an instruction that prompts the LLM to determine whether a given node pair constitutes a positive example. Given a positive example (u, v, t), we retain the source node u and construct a negative example $(u, \tilde{v}, t)$ by sampling a destination node $\tilde{v}$ from the set of destination nodes that have no interaction with u at the current timestamp t. The instruction tuning stage equips the dynamic graph model with two key capabilities. First, the dynamic graph model learns to capture the temporal patterns among the nodes with the guidance of LLMs. Second, it learns to recognize the recurring patterns of interactions, as the destination node $\tilde{v}$ in the negative example may have interacted with the source node u in the past timestamps $t' < t$.

As shown in Figure 2, the instruction is constructed based on the rich textual information associated with each event. The instruction is structured in a chat format and the details of the instruction are illustrated in Figure 3. The instructions on other benchmarks can be found in Appendix Sec. C. Each instruction consists of system-level instructions and user-level instructions that define the task context (i.e., email communication between employees) and specify the target node pair (i.e., Bob and Alice). To align the node embedding with temporal patterns, a special token, <temporal_token>, is used to represent the source node (destination node) and its associated neighboring nodes. During training, this special token would be replaced by embeddings generated from the dynamic graph model. To bridge the gap between the output node embedding space and the input token space, we employ a linear projector, analogous to the connector used in multimodal instruction tuning (Qi et al., 2024b), to map the output node embedding

to the token space. The projected token embedding is given by $Z_P^{u(v)} = Z_{u(v)}^o W_P \in \mathbb{R}^{d_P}$, where $W_P \in \mathbb{R}^{d_o \times d_P}$ denotes the weight of the linear projector, $Z_{u(v)}^o \in \mathbb{R}^{d_o}$ denotes the output embedding from the dynamic graph model. d_o and d_P denote the dimensions of $Z_{u(v)}^o$ and $Z_P^{u(v)}$, respectively. The training objective in DTGIT is to maximize the probability of generating the correct answer $\max p(Z_A|Z_I, Z_P) = \prod_i p(z_{A,i}|Z_P, Z_I, Z_{A,<i})$, where Z_A is the token embedding of the answer, Z_I is the token embedding of the instruction. During our instruction tuning only the dynamic graph model and the projector are trainable, while the parameters of the LLM remain frozen.

4.2 Supervised Training

In the second stage, we perform supervised training to bridge the gap between the proposed instruction tuning and downstream tasks, effectively transitioning the alignment from natural language space to probability space. To achieve this, we introduce an additional predictor that is trained jointly with the dynamic graph model with the remaining training data. The weights of the dynamic graph model are initialized with the parameters learned from our proposed instruction tuning. The input to the predictor can be obtained using either $\tilde{Z}_{u(v)}^o = \text{Head}(Z_{u(v)}^o)$ or $\tilde{Z}_{u(v)}^o = \text{Mean}(Z_{u(v)}^o)$, depending on the original configuration of the dynamic graph model for aggregating output embeddings from neighboring nodes. The binary cross-entropy loss function $\mathcal{L}_p = -\frac{1}{M}\sum_{i=1}^{M}(y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i))$ is adopted to further optimize the model, where M is the total number of the positive and negative samples. y is the ground-truth label and $\hat{y}$ is the predictive result.

5 Experiments

5.1 Experimental Settings

Datasets and Baselines In our experiment, we adopt on five datasets from DTGB (Enron, GDELT, ICEWS1819, Googlemap_CT and Amazon_movies) (Zhang et al., 2024a). To evaluate the effectiveness of our method, we compare our approach with the previous methods, including DyRep (Trivedi et al., 2019), TGN (Rossi et al., 2020), TGAT (Xu et al., 2020), TCL (Wang et al., 2021a), GraphMixer (Cong et al., 2023), DyGFormer (Yu et al., 2023), RepeatMixer (Zou et al., 2024) and LKD4DyTAG (Roy et al., 2025). Besides, we also employ LLMs as predictors under a few-shot prompting setting as a baseline for comparison.

Evaluation Tasks and Metrics. Our method and the baselines are evaluated on future link prediction and destination node retrieval. For future link prediction, each task is conducted in both transductive and inductive settings. In the transductive setting, the testing nodes are observed during training, while in the inductive setting, some of the testing nodes are unseen during training (Zhang et al., 2026). Average Precision (AP) and Area Under the Receiver Operating Characteristic Curve (AUC-ROC) are adopted as the evaluation metrics. For destination node retrieval, Hit Rate@3 and Hit Rate@10 are adopted as the evaluation metrics. Following the previous works (Poursafaei et al., 2022; Zhang et al., 2024a), each dataset is chronologically split with the ratio of 70%/15%/15% for training/validation/testing.

Model Configuration. We adopt Llama-3-8B-Instruct as the large language model for both the temporal graph instruction tuning stage and the LLM-as-predictor baseline. To demonstrate the effectiveness and generality of our method, we use TCL (Wang et al., 2021a), DyGFormer (Yu et al., 2023) and RepeatMixer (Zou et al., 2024) as the base models. For both the base models and all baseline methods, we follow their original configurations.

Implementation Details. In our experiment, we adopt the BERT-base-uncased model (Devlin et al., 2019) to encode the node and edge texts as the initialization of the node and edge representations (Jiang et al., 2023). In our experiment, baseline methods are trained using standard supervised learning. The training data in our training framework is split into two parts. Approximately 5%-10% of the training interactions are used for instruction tuning, and the remaining interactions are used for supervised training. The model is trained for 2 epochs and 50 epochs for instruction tuning and second-stage supervised training, respectively. Each task is performed 5 times and the mean and standard deviation are reported. All experiments are conducted using an NVIDIA RTX A100 GPU for instruction tuning and an NVIDIA RTX A5000 GPU for supervised training. Additional experimental details are provided in Appendix Sec. B.

5.2 Main Results and Discussions

Table 1 shows the performance of our method and the baselines on the link prediction task in the trans-

Table 1: The comparison of our proposed method and the baselines on the link prediction task in the transductive setting. Random, historical and inductive negative sampling strategies are adopted. Each task is repeated 5 times. Bold and underlined values indicate the best and second-best performance, respectively. - denotes unavailable results due to out-of-memory errors.

NSS	Methods	Enron		GDELT		ICEWS1819		Googlemap_CT		Amazon_movies	
		AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC
rnd	DyRep (Trivedi et al., 2019)	96.24± 0.14	96.06± 0.16	93.69± 0.42	94.55± 0.03	94.92± 0.54	94.85± 0.38	76.98± 0.31	76.99± 0.44	-	-
	TGN (Rossi et al., 2020)	96.27± 0.13	96.97± 0.04	93.71± 0.05	94.41± 0.01	98.69± 0.11	98.63± 0.12	77.93± 1.58	78.47± 1.71	-	-
	TGAT (Xu et al., 2020)	95.59± 0.72	96.05± 0.49	95.01± 0.02	95.33± 0.01	98.99± 0.01	98.93± 0.01	81.69± 0.26	81.62± 0.48	90.40± 0.03	90.41± 0.03
	GraphMixer (Cong et al., 2023)	94.67± 0.16	94.65± 0.10	93.90± 0.05	93.84± 0.20	98.00± 0.09	97.76± 0.11	75.95± 0.59	77.00± 0.15	86.51± 0.35	86.29± 0.25
	LLM-as-predictor (few-shot)	55.93± 0.57	56.22± 0.67	65.48± 0.10	70.82± 0.11	67.62± 0.43	72.42± 0.53	58.98± 0.31	60.80± 0.42	55.99± 0.77	57.45± 0.66
	LKD4DyTAG (Roy et al., 2025)	**98.42**± 0.10	97.01± 0.10	96.26± 0.30	**97.22**± 0.10	**99.16**± 0.20	98.00± 0.10	**84.88**± 0.10	**83.29**± 0.10	-	-
	TCL (Wang et al., 2021a)	93.25± 0.67	94.18± 0.43	94.41± 0.12	94.85± 0.04	99.12± 0.04	99.06± 0.05	68.60± 0.22	70.76± 0.73	87.66± 0.21	87.12± 0.35
	TCL + DTGIT (Ours)	94.58± 0.30	94.84± 0.28	94.52± 0.10	94.93± 0.03	99.15± 0.03	99.07± 0.06	69.33± 0.20	71.10± 0.61	88.77± 0.24	88.62± 0.30
	DyGFormer (Yu et al., 2023)	97.51± 0.17	97.15± 0.20	96.43± 0.04	96.48± 0.04	98.98± 0.11	98.86± 0.14	80.73± 0.39	80.83± 0.40	89.95± 0.21	89.63± 0.26
	DyGFormer + DTGIT (Ours)	97.78± 0.08	**97.47**± 0.10	**96.51**± 0.05	96.59± 0.08	**99.16**± 0.03	**99.08**± 0.09	81.96± 0.33	82.22± 0.28	**90.54**± 0.20	**90.49**± 0.12
	RepeatMixer (Zou et al., 2024)	96.57± 0.03	96.27± 0.02	95.21± 0.03	95.63± 0.05	97.72± 0.05	97.39± 0.04	79.19± 0.02	79.40± 0.03	86.21± 0.40	85.81± 0.30
	RepeatMixer + DTGIT (Ours)	97.16± 0.01	96.90± 0.02	95.87± 0.02	95.93± 0.06	97.84± 0.01	97.47± 0.01	79.45± 0.01	79.64± 0.03	87.25± 0.34	87.30± 0.26
his	DyRep (Trivedi et al., 2019)	72.16± 0.75	71.16± 0.23	63.38± 2.92	65.46± 3.03	72.08± 3.09	74.09± 1.27	52.82± 0.40	50.41± 0.68	-	-
	TGN (Rossi et al., 2020)	68.34± 0.24	70.95± 0.41	62.93± 1.73	64.74± 0.19	72.43± 0.21	73.87± 0.18	51.79± 1.09	50.96± 1.34	-	-
	TGAT (Xu et al., 2020)	74.61± 0.32	74.20± 0.50	66.81± 0.51	68.28± 0.16	74.13± 0.21	74.23± 0.04	53.09± 0.14	53.19± 0.46	56.22± 0.26	54.60± 0.33
	GraphMixer (Cong et al., 2023)	74.12± 2.63	74.47± 0.84	66.26± 0.27	67.70± 0.26	72.23± 0.39	73.38± 0.41	53.65± 0.13	53.02± 0.30	57.20± 0.02	54.20± 0.01
	LLM-as-predictor (few-shot)	51.13± 0.60	51.90± 0.91	52.01± 0.04	53.80± 0.07	54.11± 0.25	57.25± 0.42	49.22± 0.32	48.32± 0.74	49.61± 0.31	49.17± 0.67
	LKD4DyTAG (Roy et al., 2025)	58.42± 0.31	56.19± 0.25	66.12± 0.71	67.40± 0.81	67.57± 0.48	66.47± 0.72	56.80± 0.83	54.24± 0.82	-	-
	TCL (Wang et al., 2021a)	75.05± 1.29	74.03± 1.82	67.25± 1.17	68.23± 0.55	74.56± 1.13	74.66± 1.36	53.27± 0.51	52.59± 0.37	57.80± 0.35	55.36± 0.30
	TCL + DTGIT (Ours)	78.47± 1.10	**79.10**± 1.70	68.81± 1.06	69.05± 0.55	75.59± 1.10	75.67± 1.21	**59.11**± 0.54	**56.88**± 0.40	58.66± 0.33	56.62± 0.28
	DyGFormer (Yu et al., 2023)	56.91± 0.92	54.35± 1.30	66.05± 0.06	67.50± 0.38	66.66± 0.78	65.40± 0.12	55.11± 1.51	53.96± 0.74	52.51± 1.38	46.57± 1.16
	DyGFormer + DTGIT (Ours)	59.57± 0.90	57.56± 0.85	67.55± 0.44	68.33± 0.15	68.91± 0.83	67.04± 0.52	56.47± 0.78	54.69± 0.49	60.25± 0.52	54.01± 0.60
	RepeatMixer (Zou et al., 2024)	78.63± 0.56	69.09± 0.25	82.74± 0.19	78.00± 0.23	83.56± 0.55	75.61± 0.67	46.21± 0.66	25.48± 0.63	71.12± 0.70	64.48± 0.54
	RepeatMixer + DTGIT (Ours)	**79.96**± 0.31	71.03± 0.23	**83.64**± 0.42	**79.59**± 0.03	**84.36**± 0.22	**77.15**± 0.37	56.04± 0.73	35.25± 0.45	**73.67**± 0.59	**67.22**± 0.60
ind	DyRep (Trivedi et al., 2019)	57.65± 0.05	56.56± 0.89	67.44± 4.25	68.85± 4.22	70.31± 5.11	70.42± 2.07	47.64± 0.21	46.00± 0.13	-	-
	TGN (Rossi et al., 2020)	64.26± 1.37	64.08± 0.52	66.70± 1.88	67.75± 0.13	70.38± 2.35	70.91± 2.11	46.28± 0.32	45.91± 0.42	-	-
	TGAT (Xu et al., 2020)	66.29± 0.56	64.92± 0.81	69.67± 0.16	70.61± 0.04	73.62± 0.06	72.00± 0.20	46.63± 0.26	42.83± 0.40	45.87± 0.12	40.66± 0.03
	GraphMixer (Cong et al., 2023)	64.40± 3.50	62.84± 2.19	70.61± 0.48	71.22± 2.35	73.99± 0.27	**74.13**± 0.29	48.47± 0.13	46.19± 0.16	49.85± 0.16	46.61± 0.18
	LLM-as-predictor (few-shot)	49.98± 0.02	49.95± 0.07	49.15± 0.27	40.80± 0.14	53.16± 0.13	55.32± 0.17	48.24± 0.55	47.00± 0.56	46.62± 0.04	44.59± 0.77
	LKD4DyTAG (Roy et al., 2025)	51.84± 0.37	45.65± 0.59	56.38± 0.82	54.05± 0.98	63.39± 0.42	56.87± 0.64	54.80± 0.90	52.09± 0.82	-	-
	TCL (Wang et al., 2021a)	67.48± 0.17	66.48± 0.85	71.28± 1.82	71.57± 0.71	68.69± 1.15	68.45± 1.38	48.96± 0.08	46.45± 0.26	50.10± 0.29	46.37± 0.25
	TCL + DTGIT (Ours)	70.12± 0.16	**69.95**± 0.85	72.38± 1.66	**71.96**± 0.88	71.89± 1.10	69.53± 1.25	53.71± 0.10	52.29± 0.30	50.95± 0.30	47.08± 0.25
	DyGFormer (Yu et al., 2023)	50.06± 0.85	43.71± 0.97	55.36± 1.30	53.82± 1.69	61.72± 1.03	56.63± 0.22	54.57± 0.87	51.90± 1.61	47.03± 1.31	39.72± 1.07
	DyGFormer + DTGIT (Ours)	53.60± 0.48	47.90± 0.85	57.86± 0.73	54.91± 0.69	65.24± 0.85	58.92± 0.13	**56.02**± 0.36	**53.20**± 0.38	53.70± 0.50	**47.84**± 0.69
	RepeatMixer (Zou et al., 2024)	68.60± 0.36	57.80± 0.03	75.49± 0.27	64.40± 0.87	76.67± 0.61	66.95± 0.69	45.10± 0.41	21.67± 0.98	54.48± 0.51	35.75± 0.37
	RepeatMixer + DTGIT (Ours)	**70.90**± 0.32	59.52± 0.73	**76.70**± 0.76	66.05± 0.65	**78.07**± 0.49	67.53± 0.48	52.20± 0.51	30.33± 0.28	**56.82**± 0.58	37.42± 0.39

ductive setting. More results in the inductive setting can be found in Table 4 in Appendix Sec. D. The results from random sampling, historical sampling and inductive sampling (Poursafaei et al., 2022) are reported. Performance under historical and inductive sampling declines compared with random sampling, indicating that current methods are less effective at identifying negative examples from historical or previously unobserved edges. This observation aligns with previous work (Poursafaei et al., 2022). Besides, directly applying LLMs also yields inferior performance compared to conventional dynamic graph models, which is consistent with previous findings (Zhang et al., 2025b; Lei et al., 2025).

Experimental results demonstrate that existing dynamic graph models, such as DyGFormer (Yu et al., 2023) and RepeatMixer (Zou et al., 2024), generally achieve improved performance when trained with our proposed two-stage framework compared with their counterparts trained using standard supervised learning alone, particularly under historical and inductive negative sampling. Specifically, our method generally outperforms these models when trained with standard supervised training alone across multiple datasets, particularly on tasks involving historical or inductive sampling. These findings highlight the effectiveness of instruction tuning in enhancing the ability of dynamic graph models to capture evolving patterns with natural language supervision.

However, under random sampling, our method provides only limited improvement. This is primarily due to the negative sample generation process during which nodes are randomly selected from the entire set without checking for collisions with positive examples (Poursafaei et al., 2022), which can introduce false negatives and reduce the effectiveness of instruction tuning. In contrast, historical sampling evaluates the model's ability to predict whether a past edge will recur in the future. Our instruction tuning approach enables the model to recognize such recurrence patterns, particularly when negative examples involve nodes that previously interacted. This contrasts with previous dynamic graph training routines, which typically use ran-

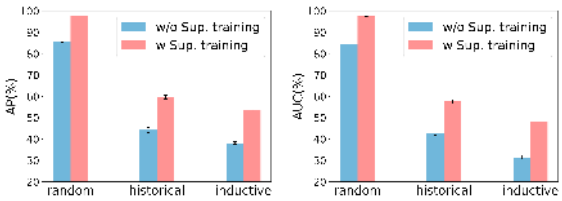


Figure 4: Performance comparison between DyGFormer trained solely with instruction tuning and complete framework for link prediction on Enron dataset.

System: You are an AI assistant tasked with modeling the evolution of a temporal graph.
User: You are tasked with analyzing a temporal graph representing email exchanges among employees. Every sequence begins with one employee, followed by tokens representing the employees they emailed prior to timestamp 100.
Examine the sequences for Bob and Alice and answer whether they communicated at timestamp100.
<temporal_token><temporal_token>< temporal_token>...
<temporal_token><temporal_token>< temporal_token>...
Assistant: True/False.

Figure 5: An example of a semantically different instruction from the previous version for the Enron dataset.

dom nodes for negative examples. The knowledge acquired through instruction tuning is successfully transferred to the dynamic graph model, resulting in more significant gains under historical and inductive negative sampling. The results on node retrieval can be found in Appendix Sec. D.

5.3 Ablation Study

We analyze the contribution of each training stage in our framework to the overall performance. We first examine the effect of instruction tuning by comparing it against a baseline trained solely with supervised learning. As shown in Tables 1 and 5, instruction tuning yields consistent performance gains across tasks, demonstrating its effectiveness in enhancing the temporal modeling of dynamic graph models.

We further assess the impact of supervised training within our framework by comparing the performance of the base model such as DyGFormer (Yu et al., 2023) trained with instruction tuning only. The evaluation is conducted on the link prediction task using the Enron dataset, with results shown in Figure 4.The results indicate that training solely with instruction tuning leads to substantial performance degradation. This is because the instruction tuning stage is designed as a warm-up process prior to supervised training and is conducted on only a limited subset of the training data, which is insufficient to comprehensively capture the complex temporal dependencies in dynamic graphs. Together, these findings demonstrate that supervised training remains an essential component of our framework.

Table 2: Comparison of the target model trained with different instructions on the link prediction task in the transductive setting using random, historical, and inductive negative sampling. Each task is repeated 5 times.

NSS	Methods	Enron		GDELT	
		AP	AUC	AP	AUC
rnd	DyGFormer (Yu et al., 2023)	97.51±0.17	97.15±0.20	96.43±0.04	96.48±0.04
	DyGFormer + DTGIT_v1 (Ours)	97.78±0.08	97.47±0.10	96.51±0.05	96.59±0.08
	DyGFormer + DTGIT_v2 (Ours)	**97.82**±0.07	**97.62**±0.90	**96.52**±0.02	**96.61**±0.04
	TCL (Wang et al., 2021a)	93.25±0.67	94.18±0.43	94.41±0.12	94.85±0.04
	TCL + DTGIT_v1 (Ours)	94.58±0.30	94.84±0.28	94.52±0.10	94.93±0.03
	TCL + DTGIT_v2 (Ours)	94.92±0.22	95.26±0.17	94.56±0.02	94.93±0.04
his	DyGFormer (Yu et al., 2023)	56.91±0.92	54.35±1.30	66.05±0.06	67.50±0.38
	DyGFormer + DTGIT_v1 (Ours)	59.57±0.90	57.56±0.85	67.55±0.44	68.33±0.15
	DyGFormer + DTGIT_v2 (Ours)	67.64±1.33	60.18±0.96	67.22±0.33	67.91±0.21
	TCL (Wang et al., 2021a)	75.05±1.29	74.03±1.82	67.25±1.17	68.23±0.55
	TCL + DTGIT_v1 (Ours)	**78.47**±1.10	**79.10**±1.70	**68.81**±1.06	**69.05**±0.55
	TCL + DTGIT_v2 (Ours)	78.22±0.92	77.34±1.45	69.05±0.86	68.45±0.64
ind	DyGFormer (Yu et al., 2023)	50.06±0.85	43.71±0.97	55.36±1.30	53.82±1.69
	DyGFormer + DTGIT_v1 (Ours)	53.60±0.48	47.90±0.85	57.86±0.73	54.91±0.69
	DyGFormer + DTGIT_v2 (Ours)	61.99±0.86	52.12±0.70	57.40±0.68	54.19±0.60
	TCL (Wang et al., 2021a)	67.48±0.17	66.48±0.85	71.28±1.82	71.57±0.71
	TCL + DTGIT_v1 (Ours)	70.12±0.16	**69.95**±0.85	72.38±1.66	**71.96**±0.88
	TCL + DTGIT_v2 (Ours)	**70.39**±0.29	69.71±0.65	**73.61**±1.50	71.91±0.97

Table 3: Comparison of the target model trained with different large language models on the link prediction task in transductive setting using random, historical, and inductive negative sampling.

NSS	Methods	Enron		GDELT	
		AP	AUC	AP	AUC
rnd	DyGFormer (Yu et al., 2023)	97.51±0.17	97.15±0.20	96.43±0.04	96.48±0.04
	DyGFormer + DTGIT_Llama3 (Ours)	97.78±0.08	97.47±0.10	96.51±0.05	96.59±0.08
	DyGFormer + DTGIT_Qwen3 (Ours)	**97.83**±0.05	**97.61**±0.09	**96.52**±0.02	**96.62**±0.03
	TCL (Wang et al., 2021a)	93.25±0.67	94.18±0.43	94.41±0.12	94.85±0.04
	TCL + DTGIT_Llama3 (Ours)	94.58±0.30	94.84±0.28	94.52±0.10	94.93±0.03
	TCL + DTGIT_Qwen3 (Ours)	94.62±0.32	95.16±0.26	94.54±0.03	94.92±0.02
his	DyGFormer (Yu et al., 2023)	56.91±0.92	54.35±1.30	66.05±0.06	67.50±0.38
	DyGFormer + DTGIT_Llama3 (Ours)	59.57±0.90	57.56±0.85	67.55±0.44	68.33±0.15
	DyGFormer + DTGIT_Qwen3 (Ours)	61.45±0.86	58.12±0.90	67.87±0.38	68.25±0.24
	TCL (Wang et al., 2021a)	75.05±1.29	74.03±1.82	67.25±1.17	68.23±0.55
	TCL + DTGIT_Llama3 (Ours)	**78.47**±1.10	**79.10**±1.70	68.81±1.06	**69.05**±0.55
	TCL + DTGIT_Qwen3 (Ours)	78.11±0.88	78.47±1.52	**68.95**±0.76	68.50±0.73
ind	DyGFormer (Yu et al., 2023)	50.06±0.85	43.71±0.97	55.36±1.30	53.82±1.69
	DyGFormer + DTGIT_Llama3 (Ours)	53.60±0.48	47.90±0.85	57.86±0.73	54.91±0.69
	DyGFormer + DTGIT_Qwen3 (Ours)	53.28±0.53	46.01±0.76	58.17±0.71	55.38±0.62
	TCL (Wang et al., 2021a)	67.48±0.17	66.48±0.85	71.28±1.82	71.57±0.71
	TCL + DTGIT_Llama3 (Ours)	70.12±0.16	**69.95**±0.85	72.38±1.66	71.96±0.88
	TCL + DTGIT_Qwen3 (Ours)	**70.38**±0.20	69.27±0.74	**73.52**±1.38	**71.98**±0.80

5.4 Analysis

5.4.1 Sensitivity Analysis

Our method introduces a dependency on the specific large language model and the prompt design used during instruction tuning. Prior work (Zhuo et al., 2024) has shown that LLM performance can vary significantly with subtle changes in prompt formatting or wording. To empirically assess the sensitivity of our approach, we conduct a prompt sensitivity analysis. Specifically, we generate semantically equivalent instructions by paraphrasing the original prompts while preserving the key task-related elements and keeping the remaining components unchanged. The new version of the instruction for Enron is shown in Figure 5. To distinguish between the two instructions, we refer to the previous version as "v1" and the new version as "v2".

We use DyGFormer (Yu et al., 2023) and TCL (Wang et al., 2021a) as the target models and train them with the new instruction within our training framework. We then compare their performance with that of the corresponding models

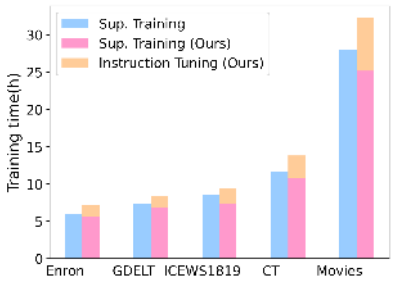


Figure 6: Comparison of training time between standard supervised training and the proposed two-stage training framework across multiple datasets. DyGFormer is used as the target model.

trained using the previous instruction. The results on link prediction are shown in Table 2. These results suggest that our method is robust to semantic variations in instruction tuning, and the model trained with different versions of the instruction generally outperforms the model trained with standard supervised training alone.

5.4.2 Effect of Different LLMs

To investigate the impact of different large language models in our instruction tuning framework, we compare Llama-3-8B-Instruct and Qwen3-4B-Instruct as the LLMs for semantic alignment. Specifically, the same instruction tuning and supervised training pipeline is adopted, while only the underlying LLM used for temporal semantic alignment is changed. The experiments are conducted on the Enron and GDELT datasets. From the results in Table 3, both Llama3 and Qwen3 generally improve the performance of the original dynamic graph models across most settings, demonstrating the effectiveness and robustness of our proposed instruction tuning framework. Overall, these results suggest that our framework is not limited to a specific LLM and can effectively transfer temporal semantic knowledge from different large language models to dynamic graph learning.

5.4.3 Computation Cost Analysis

To evaluate the computational overhead of the proposed framework, we measure the training time of each stage and compare the total training time with that of standard supervised learning. We adopt DyGFormer as the base model and conduct experiments on multiple benchmarks. For standard supervised training, all available training data are used. In our framework, approximately 5%-10% of the training data are used for instruction tuning, while the remaining data are used in the second-stage supervised training. All other settings follow the prior implementation details.

As shown in Figure 6, the proposed two-stage framework incurs a modest increase in training time compared to standard supervised training. Nevertheless, the overall computational cost remains comparable to that of standard supervised learning and scales well to large datasets. This efficiency can be attributed to two main factors: (1) Our proposed instruction tuning serves as a warm-up stage and requires only a small subset of the training data and (2) during instruction tuning, only the dynamic graph model and the projector are trainable, while the LLM remains frozen.

6 Visualization

To better understand the effectiveness of our proposed method, we present a t-SNE visualization of node embeddings from positive and negative examples obtained using standard supervised training and our proposed two-stage training framework. We adopt DyGFormer (Yu et al., 2023) as the target model. The visualizations are shown in Figure 11 and Figure 10. The quantitative clustering metrics are reported in Table 10. Compared to standard supervised training, the target model trained with our proposed method exhibits a more distinct separation between the embeddings of positive and negative examples. Since the primary difference between standard supervised training and our approach lies in the proposed dynamic graph instruction tuning, these results validate the effectiveness of instruction tuning in enhancing the ability of dynamic graph models to capture temporal dependencies.

7 Conclusion

In this study, we focus on dynamic text-attributed graph learning. To effectively leverage textual information for modeling temporal dependencies in dynamic graphs, we propose a novel two-stage training framework consisting of dynamic text-attributed graph instruction tuning followed by supervised training. Existing dynamic graph models can be seamlessly integrated into our approach, extending beyond traditional supervised training paradigms. Extensive experimental results demonstrate that dynamic graph models trained with our approach generally outperform those trained using standard supervised training alone.

Limitations

Despite the effectiveness of our proposed framework, several limitations remain. First, our method relies on textual information associated with temporal interactions, which may introduce privacy and security concerns in real-world applications. In many practical scenarios, textual attributes from communication networks, social platforms, or user interactions may contain sensitive or personally identifiable information. Directly leveraging such textual content for instruction tuning could therefore raise potential risks related to privacy leakage and data governance.

Second, the effectiveness of our framework depends on the quality and reliability of the textual information used for semantic alignment. When textual attributes are sparse, noisy, ambiguous, or weakly correlated with the underlying temporal dynamics, the transferred semantic knowledge may become less informative for downstream dynamic graph learning tasks. Developing privacy-preserving and more robust text integration strategies for dynamic graph instruction tuning remains an important direction for future work.

Acknowledgments

This work is supported in part by the U.S. Office of Naval Research Award under Grant Number N00014-24-1-2668, the National Science Foundation under Grants IIS-2316306 and CNS-2330215, the National Institutes of Health (NIH) under Grant R01EB293388, Merck BARDS Academic Collaboration Grant, and gifts from Adobe Research.

References

William Brannon, Wonjune Kang, Suyash Fulay, Hang Jiang, Brandon Roy, Deb Roy, and Jad Kabbara. 2024. Congrat: Self-supervised contrastive pretraining for joint graph and text embeddings. In *Proceedings of TextGraphs-17: Graph-based Methods for Natural Language Processing*, pages 19–39.

Runjin Chen, Tong Zhao, Ajay Kumar Jaiswal, Neil Shah, and Zhangyang Wang. 2024a. Llaga: Large language and graph assistant. In *International Conference on Machine Learning*, pages 7809–7823. PMLR.

Zhikai Chen, Haitao Mao, Hongzhi Wen, Haoyu Han, Wei Jin, Haiyang Zhang, Hui Liu, and Jiliang Tang. 2024b. Label-free node classification on graphs with large language models (llms). In *International Conference on Learning Representation*. OpenReview.net.

Weilin Cong, Si Zhang, Jian Kang, Baichuan Yuan, Hao Wu, Xin Zhou, Hanghang Tong, and Mehrdad Mahdavi. 2023. Do we really need complicated model architectures for temporal networks? In *International Conference on Learning Representations*. OpenReview.net.

Xinnan Dai, Haohao Qu, Yifei Shen, Bohang Zhang, Qihao Wen, Wenqi Fan, Dongsheng Li, Jiliang Tang, and Caihua Shan. 2025. How do large language models understand graph patterns? a benchmark for graph pattern comprehension. In *Proceedings of the International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Yi Fang, Dongzhe Fan, Sirui Ding, Ninghao Liu, and Qiaoyu Tan. 2025. Uniglm: Training one unified language model for text-attributed graphs embedding. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, pages 973–981.

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. 2024. Talk like a graph: Encoding graphs for large language models. In *International Conference on Learning Representation*. OpenReview.net.

Alessio Gravina, Giulio Lovisotto, Claudio Gallicchio, Davide Bacciu, and Claas Grohnfeldt. 2024. Long range propagation on continuous-time dynamic graphs. In *International Conference on Machine Learning*. OpenReview.net.

Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. In *First conference on language modeling*.

Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. 2024. Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning. In *International Conference on Learning Representation*. OpenReview.net.

Yufei He, Yuan Sui, Xiaoxin He, and Bryan Hooi. 2025. Unigraph: Learning a unified cross-domain foundation model for text-attributed graphs. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 448–459. ACM.

Shuo Ji, Mingzhe Liu, Leilei Sun, Chuanren Liu, and Tongyu Zhu. 2024. Memmap: An adaptive and latent memory structure for dynamic graph learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1257–1268.

Xiaodong Jiang, Ronghang Zhu, Pengsheng Ji, and Sheng Li. 2023. Co-embedding of nodes and edges with graph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(6):7075–7086.

Ming Jin, Yuan-Fang Li, and Shirui Pan. 2022. Neural temporal walks: Motif-aware representation learning on continuous-time dynamic graphs. In *Advances in Neural Information Processing Systems*.

Jongha Lee, Sunwoo Kim, and Kijung Shin. 2024. Slade: Detecting dynamic anomalies in edge streams without labels via self-supervised learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1506–1517.

Runlin Lei, Jiarui Ji, Haipeng Ding, Lu Yi, Zhewei Wei, Yongchao Liu, and Chuntao Hong. 2025. Exploring the potential of large language models as predictors in dynamic text-attributed graphs. *arXiv preprint*.

Dongyuan Li, Satoshi Kosugi, Ying Zhang, Manabu Okumura, Feng Xia, and Renhe Jiang. 2025a. Revisiting dynamic graph clustering via matrix factorization. In *Proceedings of the ACM on Web Conference 2025*, pages 1342–1352.

Dongyuan Li, Shiyin Tan, Ying Zhang, Ming Jin, Shirui Pan, Manabu Okumura, and Renhe Jiang. 2025b. Dyg-mamba: Continuous state space modeling on dynamic graphs. In *Proceedings of the Thirty-Ninth Annual Conference on Neural Information Processing Systems*. NeurIPS 2025 Poster/Presentation.

Yichuan Li, Kaize Ding, and Kyumin Lee. 2023. Grenade: Graph-centric language model for self-supervised representation learning on text-attributed graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2745–2757.

Hanmo Liu, Shimin Di, Haoyang Li, Xun Jian, Yue Wang, and Lei Chen. 2025a. A selective learning method for temporal graph continual learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 38332–38350. PMLR.

Hanwen Liu, Longjiao Zhang, Rui Wang, Tongya Zheng, Sai Wu, Chang Yao, and Mingli Song. 2025b. Salom: Structure aware temporal graph networks with long-short memory updater. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. 2024a. One for all: Towards training one graph model for all classification tasks. In *ICLR*. OpenReview.net.

Lingwen Liu, Guangqi Wen, Peng Cao, Jinzhu Yang, Weiping Li, and Osmar R. Zaïane. 2024b. Capturing temporal node evolution via self-supervised learning: A new perspective on dynamic graph learning. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 443–451. ACM.

Meng Liu, Ke Liang, Yawei Zhao, Wenxuan Tu, Sihang Zhou, Xinbiao Gan, Xinwang Liu, and Kunlun He. 2024c. Self-supervised temporal graph learning with temporal and structural intensity alignment. *IEEE Transactions on Neural Networks and Learning Systems*.

Meng Liu, Yue Liu, Ke Liang, Wenxuan Tu, Siwei Wang, Sihang Zhou, and Xinwang Liu. 2024d. Deep temporal graph clustering. In *International Conference on Learning Representations*. OpenReview.net.

Bo Pan, Zheng Zhang, Yifei Zhang, Yuntong Hu, and Liang Zhao. 2024. Distilling large language models for text-attributed graph learning. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 1836–1845.

Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B. Schardl, and Charles E. Leiserson. 2020. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Association for the Advancement of Artificial Intelligence*, pages 5363–5370. AAAI Press.

Farimah Poursafaei, Shenyang Huang, Kellin Pelrine, and Reihaneh Rabbany. 2022. Towards better evaluation for dynamic link prediction. In *Advances in Neural Information Processing Systems*.

Daiqing Qi, Handong Zhao, Xiaowei Jia, and Sheng Li. 2024a. Revealing an overlooked challenge in class-incremental graph learning. *Transactions on Machine Learning Research*.

Daiqing Qi, Handong Zhao, Zijun Wei, and Sheng Li. 2024b. Tag-grounded visual instruction tuning with retrieval augmentation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2008–2026.

Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. 2020. Temporal graph networks for deep learning on dynamic graphs. In *Proceedings of the ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+)*.

Amit Roy, Ning Yan, and Masood S. Mortazavi. 2025. Llm-driven knowledge distillation for dynamic text-attributed graphs. *CoRR*, abs/2502.10914.

Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. 2020. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *International Conference on Web Search and Data Mining*, pages 519–527. ACM.

Weili Shi, Dongliang Guo, and Sheng Li. 2026a. Revisiting dynamic graphs from the perspective of time series. *Transactions on Machine Learning Research*.

Weili Shi, Xueying Yang, Xujiang Zhao, Haifeng Chen, Zhiqiang Tao, and Sheng Li. 2023. Calibrate graph neural networks under out-of-distribution nodes via deep q-learning. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 2270–2279.

Weili Shi, Xueying Yang, Xujiang Zhao, Haifeng Chen, Zhiqiang Tao, and Sheng Li. 2026b. Optimized graph structures for calibrating graph neural networks with out-of-distribution nodes. *Transactions on Machine Learning Research*.

Junwei Su, Difan Zou, and Chuan Wu. 2024. PRES: toward scalable memory-based dynamic graph neural networks. In *International Conference on Learning Representations*. OpenReview.net.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2024. Graphgpt: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 491–500.

Jianheng Tang, Qifan Zhang, Yuhan Li, Nuo Chen, and Jia Li. 2025. Grapharena: Evaluating and exploring large language models on graph computation. In *Proceedings of the International Conference on Learning Representations*.

Yuxing Tian, Yiyan Qi, and Fan Guo. 2023. Freedyg: Frequency enhanced continuous-time dynamic graph model for link prediction. In *The Twelfth International Conference on Learning Representations*.

Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, and 1 others. 2021. Mlp-mixer: An all-mlp architecture for vision. In *Advances in Neural Information Processing Systems*, volume 34, pages 24261–24272.

Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. Dyrep: Learning representations over dynamic graphs. In *International Conference on Learning Representations*. OpenReview.net.

Duo Wang, Yuan Zuo, Fengzhi Li, and Junjie Wu. 2024a. Llms as zero-shot graph learners: Alignment of gnn representations with llm token embeddings. *Advances in Neural Information Processing Systems*, 37:5950–5973.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2023. Can language models solve graph problems in natural language? *Advances in Neural Information Processing Systems*, 36:30840–30861.

Jianing Wang, Junda Wu, Yupeng Hou, Yao Liu, Ming Gao, and Julian McAuley. 2024b. Instructgraph: Boosting large language models via graph-centric instruction tuning and preference alignment. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 13492–13510.

Lu Wang, Xiaofu Chang, Shuang Li, Yunfei Chu, Hui Li, Wei Zhang, Xiaofeng He, Le Song, Jingren Zhou, and Hongxia Yang. 2021a. TCL: transformer-based dynamic graph modelling via contrastive learning. *arXiv preprint*, abs/2105.07944.

Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. 2021b. Inductive representation learning in temporal networks via causal anonymous walks. In *International Conference on Learning Representation*. OpenReview.net.

Yunan Wang, Jianxin Li, and Ziwei Zhang. 2025. Global-recent semantic reasoning on dynamic text-attributed graphs with large language models. *arXiv preprint*.

Yuxia Wu, Yuan Fang, and Lizi Liao. 2024. On the feasibility of simple transformer for dynamic graph modeling. In *Proceedings of the ACM Web Conference*, pages 870–880. ACM.

Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive representation learning on temporal graphs. In *International Conference on Learning Representation*. OpenReview.net.

Yuanyuan Xu, Wenjie Zhang, Ying Zhang, Xuemin Lin, and Xiwei Xu. 2026. Unlocking multi-modal potentials for dynamic text-attributed graph representation. In *Proceedings of the 40th AAAI Conference on Artificial Intelligence*.

Le Yu, Leilei Sun, Bowen Du, and Weifeng Lv. 2023. Towards better dynamic graph learning: New architecture and unified library. *Advances in Neural Information Processing Systems*, 36:67686–67700.

Xingtong Yu, Zhenghao Liu, Xinming Zhang, and Yuan Fang. 2025. Node-time conditional prompt learning in dynamic graphs. In *The Thirteenth International Conference on Learning Representations*.

Haonan Yuan, Qingyun Sun, Xingcheng Fu, Ziwei Zhang, Cheng Ji, Hao Peng, and Jianxin Li. 2024. Environment-aware dynamic graph learning for out-of-distribution generalization. *Advances in Neural Information Processing Systems*, 36.

Guibin Zhang, Yiyan Qi, Ziyang Cheng, Yanwei Yue, Dawei Cheng, and Jian Guo. 2025a. Rationalizing and augmenting dynamic graph neural networks. In *The Thirteenth International Conference on Learning Representations*.

Jiasheng Zhang, Jialin Chen, Menglin Yang, Aosong Feng, Shuang Liang, Jie Shao, and Rex Ying. 2024a. DTGB: A comprehensive benchmark for dynamic text-attributed graphs. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Kexin Zhang, Shuhan Liu, Song Wang, Weili Shi, Chen Chen, Pan Li, Sheng Li, Jundong Li, and Kaize Ding. 2026. A survey of deep graph learning under distribution shifts: from graph out-of-distribution generalization to adaptation. *ACM Transactions on Knowledge Discovery from Data*, 20(2):1–38.

Mengmei Zhang, Mingwei Sun, Peng Wang, Shen Fan, Yanhu Mo, Xiaoxiao Xu, Hong Liu, Cheng Yang, and Chuan Shi. 2024b. Graphtranslator: Aligning graph model to large language model for open-ended tasks. In *Proceedings of the ACM on Web Conference 2024*, pages 1003–1014.

Siwei Zhang, Xi Chen, Yun Xiong, Xixi Wu, Yao Zhang, Yongrui Fu, Yinglong Zhao, and Jiawei Zhang. 2024c. Towards adaptive neighborhood for advancing temporal interaction graph modeling. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4290–4301.

Siwei Zhang, Yun Xiong, Yateng Tang, Xi Chen, Zian Jia, Zehao Gu, Jiarong Xu, Xuehao Zheng, and Jiawei Zhang. 2025b. Unifying text semantics and graph structures for temporal text-attributed graphs with large language models. In *Advances in Neural Information Processing Systems*.

Siwei Zhang, Yun Xiong, Yao Zhang, Xixi Wu, Yiheng Sun, and Jiawei Zhang. 2023a. ilore: Dynamic graph representation with instant long-term modeling and re-occurrence preservation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 3216–3225.

Yao Zhang, Yun Xiong, Yongxiang Liao, Yiheng Sun, Yucheng Jin, Xuehao Zheng, and Yangyong Zhu. 2023b. TIGER: temporal interaction graph embedding with restarts. In *Proceedings of the ACM Web Conference*, pages 478–488. ACM.

Zeyang Zhang, Xin Wang, Ziwei Zhang, Haoyang Li, Yijian Qin, and Wenwu Zhu. 2024d. Llm4dyg: can large language models solve spatial-temporal problems on dynamic graphs? In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4350–4361.

Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. 2023. Learning on large-scale text-attributed graphs via variational inference. In *International Conference on Learning Representation*. OpenReview.net.

Jingming Zhuo, Songyang Zhang, Xinyu Fang, Haodong Duan, Dahua Lin, and Kai Chen. 2024. Prosa: Assessing and understanding the prompt sensitivity of llms. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1950–1976.

Tao Zou, Yuhao Mao, Junchen Ye, and Bowen Du. 2024. Repeat-aware neighbor sampling for dynamic graph learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4722–4733.

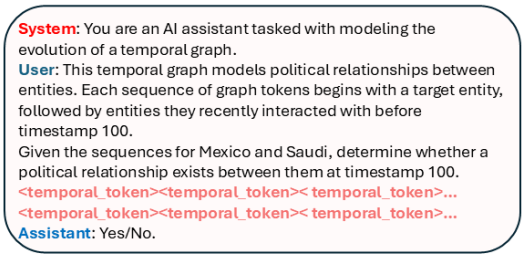


Figure 7: The instruction prompt for both GDELT and ICEWS1819 datasets. The instruction illustrates the political relationships between two entities.

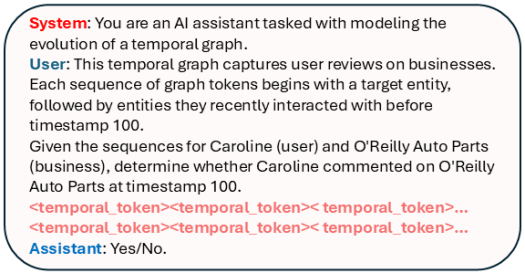


Figure 8: The instruction prompt for Googlemap_CT dataset. The instruction illustrates the reviews from the users on business entities.

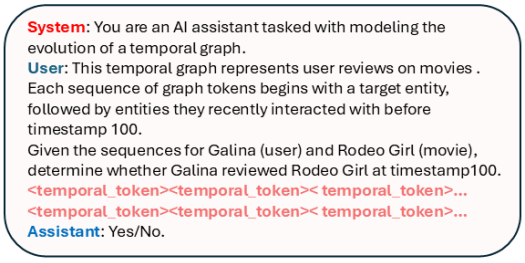


Figure 9: The instruction prompt for Amazon_movies dataset. The instruction illustrates the reviews from the users on movies.

A More Related Work

Dynamic graph learning. Dynamic graph learning has been extensively studied. The existing methods generally fall into two categories: discrete-time dynamic graph (DTDG) models and continuous-time dynamic graph (CTDG) models. DTDG methods (Zhang et al., 2025a; Liu et al., 2024c,b; Sankar et al., 2020; Pareja et al., 2020) treat dynamic graphs as a series of discrete snapshots that capture node connections up to specific time points, and adopt graph representation learning techniques to model temporal dependencies. However, this discrete segmentation can limit the model's ability to capture fine-grained dependen-

cies between nodes. In contrast, CTDG methods (Yu et al., 2025, 2023; Zhang et al., 2024c; Zou et al., 2024; Cong et al., 2023; Zhang et al., 2023b) view dynamic graphs as continuous events, represented by tuples, allowing for a more granular understanding of temporal evolution. These methods can be further divided into several subcategories. One approach models the flow of events as messages, using memory-based mechanisms to update node states based on these messages. However, memory-based methods (Liu et al., 2025b; Gravina et al., 2024; Su et al., 2024; Ji et al., 2024; Rossi et al., 2020; Xu et al., 2020) often suffer from temporal discontinuity, leading to information loss and performance degradation (Su et al., 2024). Another line of work (Li et al., 2025b; Yu et al., 2023; Tian et al., 2023; Zou et al., 2024; Wu et al., 2024; Zhang et al., 2023a; Wang et al., 2021a) employs mainstream models, such as Transformers, MLP-Mixers (Tolstikhin et al., 2021) and Mamba (Gu and Dao, 2024), to capture the temporal evolution within dynamic graphs. Additionally, temporal random walks (Jin et al., 2022; Wang et al., 2021b) have been adopted for dynamic graph learning. Beyond these approaches, some studies focus on related problems, such as temporal graph clustering (Li et al., 2025a; Liu et al., 2024d), anomaly detection within dynamic graphs (Lee et al., 2024), out-of-distribution generalization of dynamic graphs (Yuan et al., 2024), GNN calibration under out-of-distribution nodes (Shi et al., 2023, 2026b), temporal graph continual learning (Liu et al., 2025a), and class-incremental graph learning (Qi et al., 2024a).

B Experimental Settings

The statistics of the datasets are presented in Table 7. The descriptions of the datasets are as follows.

Enron. In this dataset, the nodes represent employees, while the edges correspond to emails exchanged between them. Each node's text attribute is derived from the employee's department and position, if available. The text attribute of each edge consists of the raw email content. To pre-process the text, we removed non-English statements, abnormal symbols, and tables, and applied length truncation to the emails. The edge categories were extracted from the email archiving of the original dataset, resulting in a total of 10 distinct categories.

GDELT. In this dataset, nodes represent political entities such as the United States, with their names directly used as textual attributes. Edges denote relationships between these entities, with the descriptions of these relationships serving as the textual attributes of edges. Each edge category corresponds to a specific type of political relationship or behavior.

ICEWS1819. This dataset is derived from the Integrated Crisis Early Warning System (ICEWS) project, which serves as a temporal knowledge graph for political events. We construct the dataset by extracting events that occurred between January 1, 2018, and December 31, 2019. Each political entity's textual attributes include its name, sector, and nationality, while edge text attributes consist of descriptions of political relationships. The edge categories represent different types of political relationships or behaviors.

Googlemap_CT. This dataset is derived from the Google Local Data project, which includes review information from Google Maps, along with user and business details up to September 2021 in the United States. To construct this dataset, we extract all business entities located in Connecticut State. Nodes represent users and business entities, while edges correspond to user reviews of businesses. Only business nodes are enriched with textual attributes, including the name, address, category, and self-introduction of the business entity. The edge text attributes consist of the raw text of user reviews, and edge categories are represented as integer ratings ranging from 1 to 5.

Amazon_movies. This dataset is extracted from the Amazon Review Data project, which contains product reviews and metadata from Amazon spanning from May 1996 to July 2014. To construct this dataset, we focus on products in the Movies and TV category along with their corresponding reviews. The text attribute of each product node includes its name, category, description, and rank score. The edge text attributes represent the review text provided by users for each product. Similar to the user-business review datasets, edge categories are represented as integer ratings ranging from 1 to 5.

In our experiment, we follow the model configurations from previous work. Specifically, the number of neighboring nodes for TGAT (Xu et al., 2020), TCL (Wang et al., 2021a) and GraphMixer (Cong et al., 2023) is 20. The number of neighboring nodes for TGN (Rossi et al., 2020) and DyRep (Trivedi et al., 2019) is 10. The maxi-

Table 4: The comparison of our proposed method and the baselines on the link prediction task in the inductive setting. The random, historical and inductive negative sampling are adopted. Each task is repeated 5 times. Bold and underline values represent the best and second best performance, respectively. - denotes unavailable results due to out-of-memory errors.

NSS	Methods	Enron		GDELT		ICEWS1819		Googlemap_CT		Amazon_movies	
		AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC
rnd	DyRep (Trivedi et al., 2019)	89.38± 0.25	88.60± 0.23	89.41± 0.66	89.10± 0.32	86.84± 0.48	85.20± 0.39	71.14± 0.55	69.52± 0.35	-	-
	TGN (Rossi et al., 2020)	85.20± 1.33	86.94± 0.92	80.71± 2.86	81.41± 3.21	95.23± 0.25	95.11± 0.20	72.58± 1.31	72.12± 1.40	-	-
	TGAT (Xu et al., 2020)	85.44± 2.15	86.09± 1.40	89.02± 0.00	88.96± 0.01	96.93± 0.08	96.84± 0.09	**78.79**± 0.08	**78.12**± 0.19	87.31± 0.01	86.81± 0.01
	GraphMixer (Cong et al., 2023)	84.60± 0.05	83.33± 0.02	88.33± 0.06	87.77± 0.08	94.75± 0.15	94.39± 0.20	71.68± 0.75	71.49± 0.46	82.50± 0.36	81.38± 0.22
	TCL (Wang et al., 2021a)	78.19± 1.06	79.48± 1.36	88.68± 0.46	88.29± 0.10	97.37± 0.03	97.21± 0.05	62.96± 1.73	63.31± 1.46	83.11± 0.31	81.34± 0.54
	TCL + DTGIT (Ours)	83.20± 0.80	82.85± 1.20	88.99± 0.40	88.40± 0.13	97.44± 0.04	97.25± 0.05	63.50± 1.30	64.21± 1.23	84.71± 0.30	83.64± 0.50
	DyGFormer (Yu et al., 2023)	93.18± 0.21	91.88± 0.40	92.51± 0.02	91.91± 0.04	97.31± 0.39	96.89± 0.56	76.30± 0.68	75.20± 0.79	86.71± 0.29	85.89± 0.23
	DyGFormer + DTGIT (Ours)	**93.43**± 0.10	**92.20**± 0.05	**92.62**± 0.05	**92.01**± 0.04	**97.54**± 0.02	**97.28**± 0.01	77.32± 0.64	76.62± 0.73	**87.61**± 0.28	**86.83**± 0.29
	RepeatMixer (Zou et al., 2024)	89.83± 0.42	87.84± 0.13	89.00± 0.02	89.26± 0.01	93.98± 0.19	92.42± 0.20	74.21± 0.02	72.86± 0.06	82.23± 0.56	80.60± 0.23
	RepeatMixer + DTGIT (Ours)	91.61± 0.40	90.18± 0.63	89.51± 0.01	89.69± 0.03	94.04± 0.13	92.53± 0.12	74.63± 0.02	73.28± 0.04	83.25± 0.36	81.99± 0.44
his	DyRep (Trivedi et al., 2019)	52.06± 1.03	48.54± 1.96	56.87± 2.85	58.27± 2.32	57.88± 1.06	57.23± 0.69	48.46± 0.38	46.61± 0.49	-	-
	TGN (Rossi et al., 2020)	54.60± 0.35	52.83± 0.17	56.33± 0.09	56.67± 1.02	56.93± 0.80	56.54± 1.08	47.58± 0.39	45.86± 0.46	-	-
	TGAT (Xu et al., 2020)	60.17± 1.40	58.24± 1.00	59.81± 0.08	60.97± 0.13	60.76± 0.16	59.37± 0.21	46.94± 0.22	43.24± 0.27	45.91± 0.12	40.65± 0.01
	GraphMixer (Cong et al., 2023)	57.88± 3.04	55.95± 1.96	60.87± 0.24	62.05± 0.24	60.11± 0.39	61.05± 0.32	49.08± 0.23	46.90± 0.24	49.85± 0.16	46.50± 0.10
	TCL (Wang et al., 2021a)	61.88± 1.08	60.61± 1.51	61.45± 1.11	62.53± 0.29	58.25± 1.57	57.62± 1.28	49.32± 0.11	46.76± 0.39	50.15± 0.40	47.40± 0.30
	TCL + DTGIT (Ours)	**64.76**± 0.86	**63.95**± 1.15	62.16± 1.60	**62.93**± 0.33	63.37± 1.52	**64.22**± 1.16	52.67± 0.21	47.91± 0.45	51.22± 0.42	**48.47**± 0.28
	DyGFormer (Yu et al., 2023)	45.57± 0.81	36.13± 0.59	53.37± 1.06	52.97± 1.15	52.79± 1.03	47.77± 0.44	53.85± 1.49	50.41± 1.23	47.19± 1.31	39.64± 1.40
	DyGFormer + DTGIT (Ours)	47.82± 0.74	39.45± 0.77	55.18± 0.61	53.92± 0.84	54.69± 0.86	49.37± 0.86	**55.62**± 0.34	**51.60**± 0.49	53.74± 0.31	46.74± 0.39
	RepeatMixer (Zou et al., 2024)	57.97± 0.66	44.07± 0.33	70.82± 0.43	61.20± 0.87	66.34± 0.46	54.22± 0.25	45.82± 0.51	22.67± 0.41	54.36± 0.42	36.14± 0.44
	RepeatMixer + DTGIT (Ours)	59.48± 0.51	45.41± 0.33	**71.35**± 0.06	62.85± 0.02	**67.01**± 0.31	55.54± 0.68	52.98± 0.51	31.46± 0.45	**56.54**± 0.39	37.72± 0.43
ind	DyRep (Trivedi et al., 2019)	52.05± 1.01	48.54± 1.94	56.88± 2.86	58.28± 2.32	57.85± 1.08	57.19± 0.72	48.06± 0.55	46.54± 0.08	-	-
	TGN (Rossi et al., 2020)	54.55± 0.38	52.80± 0.19	56.33± 0.12	56.67± 1.03	56.87± 0.80	56.50± 1.07	47.50± 0.39	45.82± 0.46	-	-
	TGAT (Xu et al., 2020)	60.16± 1.48	58.24± 1.04	59.80± 0.09	60.98± 0.12	60.76± 0.15	59.36± 0.22	46.95± 0.22	43.25± 0.27	45.90± 0.12	40.68± 0.01
	GraphMixer (Cong et al., 2023)	57.87± 2.05	55.96± 1.98	60.85± 0.24	62.03± 0.24	60.13± 0.36	**61.07**± 0.30	49.00± 0.23	46.90± 0.24	49.82± 0.15	46.50± 0.11
	TCL (Wang et al., 2021a)	**61.87**± 1.08	**60.62**± 1.52	61.44± 1.11	62.51± 0.29	58.29± 1.57	57.63± 1.28	49.31± 0.11	46.75± 0.39	50.14± 0.40	46.40± 0.30
	TCL + DTGIT (Ours)	**64.75**± 0.89	**63.94**± 1.30	61.69± 1.20	**62.85**± 0.24	62.38± 1.40	60.95± 0.90	52.68± 0.16	47.92± 0.43	51.36± 0.36	**47.57**± 0.22
	DyGFormer (Yu et al., 2023)	45.56± 0.80	36.80± 1.12	53.36± 1.05	52.96± 1.15	52.77± 1.04	47.75± 0.44	53.80± 1.13	50.40± 1.56	47.19± 1.48	39.63± 1.20
	DyGFormer + DTGIT (Ours)	47.82± 1.02	39.46± 0.94	55.18± 0.93	53.91± 0.64	54.68± 0.86	49.36± 0.86	**54.62**± 0.46	**51.60**± 0.51	53.75± 0.26	47.15± 0.40
	RepeatMixer (Zou et al., 2024)	57.98± 0.65	44.09± 0.71	**70.85**± 0.42	61.23± 0.86	65.30± 0.99	53.23± 0.27	45.83± 0.50	22.67± 0.36	54.37± 0.50	36.14± 0.40
	RepeatMixer + DTGIT (Ours)	59.13± 0.52	45.17± 0.32	**71.40**± 0.17	62.61± 0.12	**66.54**± 0.32	54.46± 0.68	52.98± 0.51	31.45± 0.25	**56.54**± 0.29	37.73± 0.43

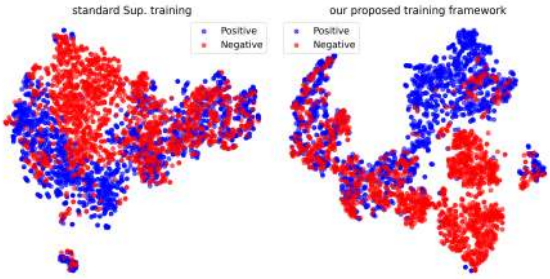


Figure 10: t-SNE visualization of positive and negative examples from the target model trained with standard supervised learning (left) and with our proposed method (right) from the Amazon_movies dataset.

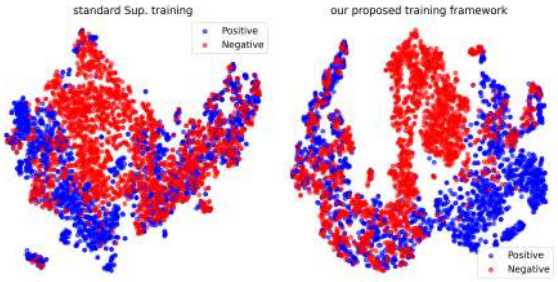


Figure 11: t-SNE visualization of positive and negative examples from the target model trained with standard supervised learning (left) and with our proposed method (right) from the Googlemap_CT dataset.

mum input sequence lengths for DyGFormer (Yu et al., 2023) and RepeatMixer (Zou et al., 2024) are 48 and 32, respectively. The dropout rate is set to 0.1. The batch size for supervised training and instruction tuning is 256 and 24, respectively. The initial learning rate is 1e-4 and weight decay is 0.0. Adam is used for optimization during training. In the LLM-as-predictor baseline, we use 5-shot prompting.

C Methodology

The designed instruction prompts for the other datasets are illustrated in Figures 7–9. The instructions are designed to prompt LLMs to identify the positive/negative examples.

D Main Results

The performance of target dynamic graph models and the baselines on the link prediction task in the inductive setting is presented in Table 4. The results demonstrate that our instruction tuning approach enables existing dynamic graph models to achieve better generalization to unseen nodes and improves performance compared to training with supervised learning alone. Moreover, the target dynamic graph models can also outperform the baselines on most

Table 5: The comparison of our method and the baselines on the node retrieval task in the transductive setting. Random, historical and inductive negative sampling strategies are adopted. Each task is repeated 5 times. Bold and underlined values indicate the best and second-best performance, respectively. - denotes unavailable results due to out-of-memory errors.

Methods	Enron		GDELT		ICEWS1819		Googlemap_CT		Amazon_movies	
	HR@3	HR@10	HR@3	HR@10	HR@3	HR@10	HR@3	HR@10	HR@3	HR@10
DyRep (Trivedi et al., 2019)	77.85± 1.51	94.10± 0.56	56.49± 1.96	83.36± 0.73	74.89± 6.88	88.44± 3.79	20.02± 0.07	42.23± 0.36	-	-
TGN (Rossi et al., 2020)	80.95± 1.34	91.11± 1.21	56.26± 0.06	83.74± 0.18	91.31± 0.46	96.77± 0.37	22.44± 0.66	44.72± 0.82	-	-
TGAT (Xu et al., 2020)	69.27± 4.06	84.09± 3.31	61.51± 0.50	86.00± 0.19	91.22± 0.08	97.04± 0.07	27.48± 0.08	50.95± 0.40	46.59± 0.19	70.65± 0.06
GraphMixer (Cong et al., 2023)	67.98± 0.01	83.74± 0.61	60.04± 0.03	84.59± 0.03	85.40± 0.24	93.97± 0.13	23.79± 0.48	46.57± 0.32	42.31± 0.34	66.80± 0.53
TCL (Wang et al., 2021a)	62.88± 1.26	82.33± 1.41	59.96± 0.06	84.52± 0.02	92.29± 0.22	97.43± 0.07	22.72± 0.28	44.36± 1.32	41.50± 0.18	64.93± 0.02
TCL + DTGIT (Ours)	64.91± 1.05	82.81± 1.18	60.22± 0.03	85.50± 0.03	93.02± 0.24	**98.08**± 0.06	23.36± 0.32	44.96± 1.26	43.16± 0.15	67.07± 0.04
DyGFormer (Yu et al., 2023)	86.68± 0.90	94.42± 0.94	70.24± 0.11	89.52± 0.33	92.21± 0.92	96.47± 0.46	26.37± 0.73	49.84± 1.07	46.19± 0.55	68.65± 0.79
DyGFormer + DTGIT (Ours)	**88.29**± 1.00	**95.68**± 1.14	**70.77**± 0.07	**90.14**± 0.04	**93.25**± 0.12	97.44± 0.08	**28.13**± 0.17	**53.80**± 0.48	**47.33**± 0.35	**70.81**± 0.60
RepeatMixer (Zou et al., 2024)	81.47± 0.41	87.30± 0.28	69.68± 0.18	89.46± 0.27	84.92± 0.50	92.04± 0.03	22.92± 0.33	45.82± 0.03	40.91± 0.32	64.32± 0.59
RepeatMixer + DTGIT (Ours)	83.27± 0.43	88.88± 0.09	70.76± 0.35	90.11± 0.08	86.51± 0.05	92.62± 0.05	23.24± 0.06	46.07± 0.11	41.58± 0.30	65.03± 0.48

Table 6: The comparison of our method and the baselines on the node retrieval task in the inductive setting. The random, historical and inductive negative sampling are adopted. Each task is repeated 5 times. Bold and underlined values indicate the best and second-best performance, respectively. - denotes the unavailable results due to out-of-memory errors.

Methods	Enron		GDELT		ICEWS1819		Googlemap_CT		Amazon_movies	
	HR@3	HR@10	HR@3	HR@10	HR@3	HR@10	HR@3	HR@10	HR@3	HR@10
DyRep (Trivedi et al., 2019)	54.19± 3.01	77.25± 1.28	48.07± 3.36	72.18± 0.59	51.15± 8.41	71.11± 8.46	16.11± 0.48	32.69± 0.71	-	-
TGN (Rossi et al., 2020)	51.04± 1.06	71.98± 1.22	22.60± 0.10	48.76± 0.35	73.90± 0.55	87.80± 0.40	17.48± 0.60	34.40± 1.02	-	-
TGAT (Xu et al., 2020)	42.36± 5.28	62.86± 4.32	44.11± 0.88	70.51± 0.40	77.53± 0.15	90.68± 0.30	22.08± 0.02	43.71± 0.05	40.10± 0.29	62.87± 0.18
GraphMixer (Cong et al., 2023)	39.17± 1.22	61.24± 0.09	44.97± 0.06	69.93± 0.04	67.97± 0.58	84.24± 0.56	19.94± 0.08	39.26± 0.14	36.70± 0.10	58.12± 0.29
TCL (Wang et al., 2021a)	30.96± 0.80	54.18± 0.71	45.20± 0.03	69.92± 0.05	80.39± 0.01	91.79± 0.10	18.13± 0.13	36.29± 0.99	34.86± 0.29	55.94± 0.03
TCL + DTGIT (Ours)	32.73± 0.69	56.40± 0.58	45.88± 0.04	70.10± 0.04	80.78± 0.03	91.96± 0.12	18.83± 0.10	36.75± 0.97	36.53± 0.22	58.25± 0.04
DyGFormer (Yu et al., 2023)	73.41± 0.84	83.58± 1.13	56.16± 0.53	77.82± 0.12	81.77± 0.29	91.25± 0.15	21.82± 0.91	42.05± 1.42	39.51± 0.28	61.23± 0.50
DyGFormer + DTGIT (Ours)	**74.08**± 0.75	**84.07**± 0.81	**56.82**± 0.27	**78.24**± 0.04	**82.78**± 0.64	**92.67**± 0.33	**22.82**± 0.37	**44.76**± 0.48	**40.49**± 0.18	**62.94**± 0.33
RepeatMixer (Zou et al., 2024)	72.49± 0.39	77.77± 0.67	55.54± 0.27	76.79± 0.02	72.15± 0.38	81.09± 0.42	18.65± 0.22	37.75± 0.19	34.78± 0.48	55.88± 0.66
RepeatMixer + DTGIT (Ours)	73.29± 0.27	80.54± 0.81	56.44± 0.15	**78.24**± 0.04	73.86± 0.59	83.30± 0.08	19.04± 0.05	38.19± 0.02	35.24± 0.38	56.56± 0.56

Table 7: The statistics of datasets

Dataset	#Nodes	#Edges	#Edge Categories	#Timestamps	#Domain
Enron	42,711	797,907	10	1,006	Email
GDELT	6,786	1,339,245	237	2,591	Knowledge graph
ICEWS1819	31,796	1,100,071	266	730	Knowledge graph
Googlemap_CT	111,168	1,380,623	5	55,521	E-commerce
Amazon_movies	293,566	4,217,324	5	7,287	E-commerce

Table 8: Comparison of the target model trained with different instructions on the link prediction task in transductive setting using random, historical, and inductive negative sampling. Each task is repeated 5 times.

NSS	Methods	Enron		GDELT	
		AP	AUC	AP	AUC
rnd	DyGFormer (Yu et al., 2023)	93.18± 0.21	91.88± 0.40	92.51± 0.02	91.91± 0.04
	DyGFormer + DTGIT_v1 (Ours)	93.43± 0.10	92.20± 0.05	92.62± 0.05	92.01± 0.04
	DyGFormer + DTGIT_v2 (Ours)	**93.96**± 0.32	**93.08**± 0.06	**92.65**± 0.07	**92.05**± 0.06
	TCL (Wang et al., 2021a)	78.19± 1.06	79.48± 1.36	88.68± 0.46	88.29± 0.10
	TCL + DTGIT_v1 (Ours)	83.20± 0.80	82.85± 1.20	88.99± 0.40	88.40± 0.13
	TCL + DTGIT_v2 (Ours)	83.31± 0.62	83.19± 1.12	89.19± 0.50	88.44± 0.14
his	DyGFormer (Yu et al., 2023)	45.57± 0.81	36.13± 0.59	53.37± 1.06	52.97± 1.15
	DyGFormer + DTGIT_v1 (Ours)	47.82± 0.74	39.45± 0.77	55.18± 0.61	53.92± 0.84
	DyGFormer + DTGIT_v2 (Ours)	52.44± 0.78	41.45± 0.73	54.78± 0.43	53.63± 0.61
	TCL (Wang et al., 2021a)	61.88± 1.08	60.61± 1.51	61.45± 1.11	62.53± 0.29
	TCL + DTGIT_v1 (Ours)	64.76± 0.86	63.95± 1.15	62.16± 1.60	**62.93**± 0.33
	TCL + DTGIT_v2 (Ours)	**66.58**± 0.72	**64.83**± 1.20	**62.51**± 0.99	62.30± 0.58
ind	DyGFormer (Yu et al., 2023)	45.56± 0.80	36.80± 1.12	53.36± 1.02	52.96± 1.15
	DyGFormer + DTGIT_v1 (Ours)	47.82± 1.02	39.46± 0.94	55.18± 0.93	53.91± 0.64
	DyGFormer + DTGIT_v2 (Ours)	52.40± 0.77	41.38± 0.97	54.77± 0.81	53.62± 0.56
	TCL (Wang et al., 2021a)	61.87± 1.08	60.62± 1.52	61.44± 1.11	62.51± 0.29
	TCL + DTGIT_v1 (Ours)	64.75± 0.89	63.94± 1.30	61.69± 1.20	**62.85**± 0.24
	TCL + DTGIT_v2 (Ours)	**66.46**± 0.92	**64.79**± 1.21	**62.51**± 1.24	62.31± 0.47

tasks, when trained with our two-stage framework.

The performance of our method and the baselines on the node retrieval task in both transductive and inductive settings is presented in Tables 5 and 6, respectively. Overall, the target dynamic graph models trained with our two-stage framework generally outperform both the original models trained solely with standard supervised learning and other baseline methods across most datasets and evaluation settings. In particular, integrating DTGIT with DyGFormer achieves the best performance on the majority of datasets under both transductive and inductive settings, demonstrating the effectiveness of our instruction tuning framework in enhancing temporal representation learning for retrieval tasks.

Compared with standard supervised training, our framework generally improves HR@3 and HR@10 across different negative sampling strategies, including random, historical, and inductive sampling. The improvements are especially noticeable in more challenging inductive settings, indicating that the semantic knowledge acquired during instruction tuning can improve the generalization ability of dynamic graph models toward unseen nodes and evolving interaction patterns. Furthermore, the gains achieved by integrating DTGIT with different backbone models, including TCL, DyGFormer, and RepeatMixer, demonstrate the general appli-

Table 9: Comparison of the target model trained with different large language models on the link prediction task in transductive setting using random, historical, and inductive negative sampling. Each task is repeated 5 times.

NSS	Methods	Enron		GDELT	
		AP	AUC	AP	AUC
rnd	DyGFormer (Yu et al., 2023)	93.18± 0.21	91.88± 0.40	92.51± 0.02	91.91± 0.04
	DyGFormer + DTGIT_Llama3 (Ours)	93.43± 0.10	92.20± 0.05	92.62± 0.05	92.01± 0.04
	DyGFormer + DTGIT_Qwen3 (Ours)	**93.74**± 0.12	**92.87**± 0.04	**92.61**± 0.03	**92.02**± 0.02
	TCL (Wang et al., 2021a)	78.19± 1.06	79.48± 1.36	88.68± 0.46	88.29± 0.10
	TCL + DTGIT_Llama3 (Ours)	83.20± 0.80	82.85± 1.20	88.99± 0.40	88.40± 0.13
	TCL + DTGIT_Qwen3 (Ours)	83.26± 0.59	83.00± 0.75	89.14± 0.55	88.42± 0.10
his	DyGFormer (Yu et al., 2023)	45.57± 0.81	36.13± 0.59	53.37± 1.06	52.97± 1.15
	DyGFormer + DTGIT_Llama3 (Ours)	47.82± 0.74	39.45± 0.77	55.18± 0.61	53.92± 0.84
	DyGFormer + DTGIT_Qwen3 (Ours)	48.24± 0.83	39.47± 0.80	55.25± 0.39	54.03± 0.43
	TCL (Wang et al., 2021a)	61.88± 1.08	60.61± 1.51	61.45± 1.11	62.53± 0.29
	TCL + DTGIT_Llama3 (Ours)	64.76± 0.86	63.95± 1.15	**62.16**± 1.60	**62.93**± 0.33
	TCL + DTGIT_Qwen3 (Ours)	**66.14**± 0.78	**64.15**± 0.82	62.13± 0.80	62.25± 0.68
ind	DyGFormer (Yu et al., 2023)	45.56± 0.80	36.80± 1.12	53.36± 1.02	52.96± 1.15
	DyGFormer + DTGIT_Llama3 (Ours)	47.82± 1.02	39.46± 0.94	55.18± 0.93	53.91± 0.64
	DyGFormer + DTGIT_Qwen3 (Ours)	48.25± 0.95	39.49± 1.04	55.27± 0.77	54.04± 0.65
	TCL (Wang et al., 2021a)	61.87± 1.08	60.62± 1.52	61.44± 1.11	62.51± 0.29
	TCL + DTGIT_Llama3 (Ours)	64.75± 0.89	63.94± 1.30	61.69± 1.20	**62.85**± 0.24
	TCL + DTGIT_Qwen3 (Ours)	**66.17**± 0.96	**64.16**± 0.99	**62.10**± 1.14	62.23± 0.57

Table 10: Comparison of quantitative clustering metrics for positive and negative examples between standard supervised training and the proposed method on the Googlemap_CT dataset. The bold values indicate the best performance.

Methods	Silhouette Score ↑	Davies-Bouldin Index ↓	Calinski-Harabasz ↑
Supervised Training	0.0344	7.2363	74.9258
Two-stage training (Ours)	**0.0648**	**5.0497**	**154.7174**

cability and robustness of our framework across various dynamic graph architectures.

E Analysis

To investigate the sensitivity of our framework to different instruction formulations, we evaluate two semantically equivalent instruction variants, denoted as DTGIT_v1 and DTGIT_v2, on the link prediction task under random, historical, and inductive negative sampling settings. Specifically, the alternative instructions are generated through paraphrasing while preserving the original task semantics and key contextual information.

As shown in Table 8, both instruction variants generally improve the performance of the original dynamic graph models across different datasets and evaluation settings, demonstrating the robustness of our framework to prompt variations. In particular, the gains are more substantial under the historical and inductive negative sampling settings, where capturing temporal dependencies is more challenging. The relatively stable performance across different instruction variants suggests that our framework is not overly sensitive to prompt wording and can generally transfer temporal semantic knowledge to dynamic graph models.

To further investigate the impact of different large language models in our framework, we compare Llama-3-8B-Instruct and Qwen3 as the teacher models used during the instruction tuning stage. Specifically, the same two-stage training framework and experimental settings are adopted, while only the underlying LLM for temporal semantic alignment is changed.

As shown in Table 9, both Llama3 and Qwen3 generally improve the performance of the original dynamic graph models across different datasets and negative sampling strategies, demonstrating the effectiveness and robustness of our proposed framework. In particular, the improvements are more significant under the historical and inductive negative sampling settings, where modeling complex temporal dependencies and generalizing to evolving interaction patterns are more challenging. Overall, these results demonstrate that our framework is not restricted to a specific large language model and can effectively transfer temporal semantic knowledge from different LLMs to dynamic graph models.

F LLM Usage Statement

Large language models are used in this work for both method design and auxiliary language-related tasks. Specifically, our proposed framework leverages LLMs during the instruction tuning stage to provide temporal semantic supervision for dynamic graph models. In addition, we employ LLMs to generate semantically equivalent paraphrases of instruction prompts for the prompt sensitivity analysis. The paraphrased instructions preserve the original task semantics while varying the wording and expression style. No experimental results, analyses, or conclusions in this paper are automatically generated by LLMs. We also use AI tools for language editing and improving the clarity of portions of the manuscript. The authors reviewed and take full responsibility for all content.