LLM-driven Knowledge Distillation for Dynamic Text-Attributed Graphs

Amit Roy[1*] **, Ning Yan**[2]**, Masood Mortazavi**[2]

[1]Purdue University, [2]Futurewei Technologies Inc.
roy206@purdue.edu, {yan.ningyan, masood.mortazavi}@futurewei.com

Abstract

Dynamic **T**ext-**A**ttributed **G**raphs (**DyTAG**s) have numerous real-world applications, e.g., social, collaboration, citation, communication, and review networks. In these networks, nodes and edges often contain text descriptions, and the network structure can evolve over time. Future link prediction, edge classification, relation generation, and other downstream tasks on DyTAGs require powerful representations that encode structural, temporal, and textual information. Graph Neural Networks (GNNs) are adept at managing structural data, yet encoding temporal information in dynamic graphs has proven challenging. In this work, we propose **L**LM-driven **K**nowledge **D**istillation for **Dy**namic **T**ext **A**ttributed **G**raph (**LKD4DyTAG**) with temporal encoding to address those challenges. We first use a simple yet effective approach to encode temporal information in edges so that graph convolution can simultaneously capture both temporal and structural information in the hidden representations. To leverage LLM's text processing capabilities to learn richer representations on DyTAGs, we distill knowledge from LLM-driven edge representations (based on a neighborhood's text attributes) into saptio-temporal representations learned by a lightweight GNN model that encodes temporal and structural information. Our knowledge distillation objective enables the GNN to learn representations that more effectively encode available *structural*, *temporal* and *textual* information in DyTAGs. Extensive experimentation conducted on six real-world dynamic text-attributed graph datasets prove the efficacy of our approach LKD4DyTAG for the future link prediction and edge classification task. The results show that our approach significantly improves the performance of downstream tasks compared to baseline models.

Introduction

Dynamic **T**ext-**A**ttributed **G**raph (**DyTAG**) structures change over time, incorporating text descriptions in both nodes and edges. DyTAGs have a wide range of applications across various domains (Tang et al. 2023; Luo et al. 2023; Zhang, Shao, and Cui 2023; Cai et al. 2022; Skarding, Gabrys, and Musial 2021; Kazemi et al. 2020), such as email networks, political event networks, online Q&A forums, and review networks for products and movies. For instance, in an email network, nodes can represent email users with their profile information, while edges can represent the emails exchanged between them at specific times. Similarly, in a movie or item review network, nodes can describe users, movies, or items, while edges may denote reviews between users and movies or items at different times. Again, in a political event network nodes can denote the political figures, places, institution and the edges may indicate the interrelationships between those entities with temporal information attached. Handling both the dynamically changing graph structure and the semantic information in nodes and edges described in natural language is a challenging task since structure and text have different modality. When designing deep learning models for various downstream tasks for such text attributed dynamic graph data, it is crucial to carefully consider structural, temporal, and textual information.

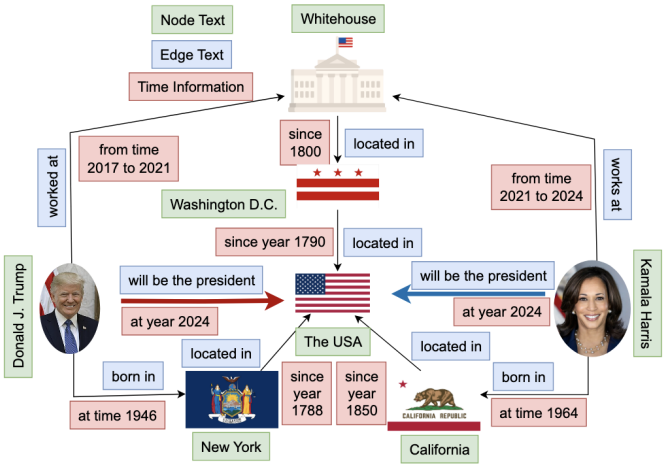

Figure 1: Example Dynamic Text-Attributed Graph in the context of 2024 Presidential Election in USA. Each entity is described with text, while the relations between them are represented by edges including temporal information.

Dynamic networks are often represented as snapshots with discrete timestamps. Recent works propose various methods to capture temporal information in dynamic networks, such as anonymous random walk-based

*Corresponding Author — Work done as an intern at Futurewei Technologies Inc.

approach (Wang et al. 2021b) and window-based approach (Alomrani et al. 2023). However, when the number of timestamps is extensively large and the downstream task depends on earlier timestamps, it becomes challenging to efficiently encode the complete temporal information into the latent representation. Additionally, as the graph structure evolves over time, it is also essential to include temporal information in graph convolution as well as structural information. In addition to temporal and structural information found in dynamic graphs, Dynamic Text-Attributed Graphs(DyTAGs) also incorporate textual information. This added text provides a more comprehensive representation of real-world systems compared to dynamic graphs that only contain structural and temporal data.

With the increasing capability of LLMs to produce semantic representations of text, numerous works have emerged that utilize them for graph tasks (Fang et al. 2024b; Xu et al. 2024). Most of these works rely on prompt engineering to describe the graph structure and the downstream task in order to "tune" the LLMs, expecting the LLMs to generate the desired results for the given task. However, despite their advanced capabilities, LLMs face challenges in scalability and are limited in their ability to process input prompts of longer sequence lengths. Consequently, their applicability to diverse downstream tasks is constrained. Furthermore, despite LLMs versatility in capturing semantic information from text, they often try to memorize the common structural motifs/patterns and are not effective in comprehending the underlying evolving structure and the complex temporal dynamics of Dynamic Text-Attributed Graphs (DyTAGs). To address these challenges, one can deploy the capabilities of LLMs for scalable processing of all locally-scoped text attributes in DyTAGs and transfer representations thus learned by heavyweight large language models to lightweight graph neural networks models (GNNs) for downstream tasks involving DyTAGs. Here, text-based edge representation generated by the LLM serves as a guide for the spatio-temporal edge representation produced by temporally encoded GNNs. This approach ensures that, among the three types of information present in DyTAGs, temporal and structural information are captured by the temporally encoded GNN, while the LLMs are utilized to encode the textual information of edges for distillation purposes.

This approach leads to our design of a novel LLM-driven knowledge distillation framework for processing Dynamic Text-Attributed Graphs (DyTAGs) with temporal encoding. The structure and temporal information of edges are transformed into spatio-temporal edge representations using GNNs with a simple yet efficient encoding of time. For the textual edge representation, we describe the neighbors of adjacent nodes to the LLMs one edge at a time and sum these descriptions to obtain neighborhood textual embeddings. Next, we add up the neighborhood textual embeddings of adjacent node with the LLM output for the corresponding edge's description to obtain a text-based edge representation. Next, we bring into alignment the spatio-temporal edge representation and the text-based edge representation. In short, we are using knowledge distillation method to transfer knowledge from the text-based edge representation from a teacher LLM model to the spatio-temporal representation of a student GNN model with temporal encoding for dynamic text attributed graph. Our contributions in this work are as follows:

- We propose a novel framework **LKD4DyTAG** which integrates time, structure, and text information in Dynamic Text-Attributed Graphs to produce suitable representations for downstream tasks.
- We designed a simple yet effective approach to produce spatio-temporal edge representations of DyTAG that capture the temporal and structural information using temporal edge encoding with GNNs.
- To make the best utilization of textual information from DyTAG and the text processing capability from LLMs, we perform knowledge distillation from text-based edge representation to spatio-temporal edge representation.
- We perform extensive experiments on six real-world benchmark dynamic text-attributed graph datasets and show that our proposed method LKD4DyTAG can outperform state-of-the-art approaches in certain downstream tasks such as *future link prediction* and *edge classification*.

Notation and Problem Formulation

Dynamic Text-Attributed Graph (DyTAG). A Dynamic Text-Attributed Graph (DyTAG) can be defined as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ denotes the set of nodes or entities and $\mathcal{E}$ denotes the set of edges or relations among those entities, each with associated timestamp, relation, and label information. The terms node-edge and entity-relation are used in regular graphs and knowledge graphs respectively. We will use both terminologies interchangeably. Each entity $u \in \mathcal{V}$ has an associated text description u_{text} that describes the entity. Each edge $e_{uv} \in \mathcal{E}$ between two adjacent entities u and v can be described as a three-tuple $\{r_{uv}, t_{uv}, l_{uv}\}$, where r_{uv} describes the relationship between u and v in texts as r_{text}, t_{uv} describes the timestamp information when u and v are connected, and l_{uv} is the label of the edge e_{uv}. Each timestamp $t_{uv} \in \mathcal{T}$, and labels $l_{uv} \in \mathcal{L}$, where $\mathcal{T}$ and $\mathcal{L}$ denote the set of timestamps and edge labels, respectively. We denote the DyTAG until timestamp T as $\mathcal{G}_T = (\mathcal{V}_T, \mathcal{E}_T)$. Additionally, we describe the 1-hop neighborhood of a node u as $\mathcal{N}_u$. In this work, we aim to design a framework that effectively utilize the temporal, structural, and textual information of DyTAGs to address two tasks, *future link prediction* and *edge classification*.

Future Link Prediction. Given a DyTAG $\mathcal{G}_T$ containing the interactions between nodes and edges with their text descriptions until timestamp T, the task of future link prediction aims at predicting the existence of an edge e_{uv} between node u and node v at the next k timestamps $T + k$. For example, given a email network among the users, the task of future link prediction aims to determine whether two users will exchange emails in the near future based on their past email interactions and the contents of the emails.

Edge Classification. Given a DyTAG $\mathcal{G}_T$ with the interactions between entities and relations up to timestamp T, the

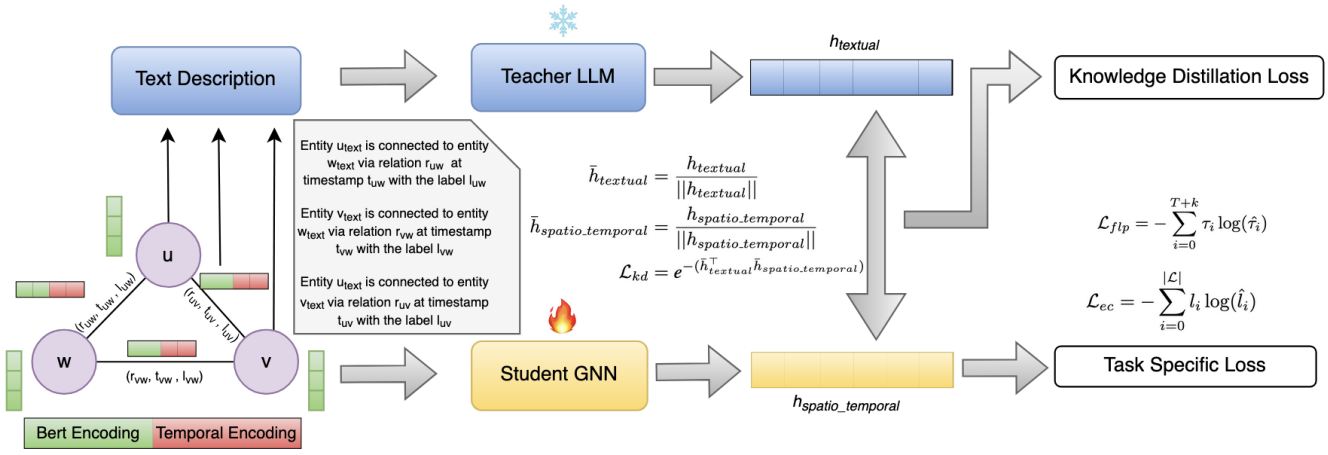

Figure 2: **LKD4DyTAG : LLM-driven Knowledge Distillation for Dynamic Text-Attributed Graph with temporal encoding.** First, temporal encodings (red) are added to the graph edges along with their BERT encodings (green). The graph information is then transformed into textual information by incorporating the adjacent node's 1-hop neighbors to encode the semantic context, as illustrated in the textbox. The derived text description is fed into a pretrained teacher LLM model to obtain the textual representation, while the graph is fed into a trainable student GNN to obtain the spatio-temporal representation. These two representations are brought closer in latent space by minimizing the cosine similarity, which defines the knowledge distillation loss. Additionally, the student model's spatio-temporal edge representation is utilized for the downstream task, which is trained using the task-specific loss.

task of edge classification focuses on determining the category of an edge between two candidate entities u and v at the future timestamp $T+k$. For example, in an item-review network containing user and item nodes with edges describing reviews between them, the edge classification task involve determining the type of review a user has given for an item, such as a product or a movie.

Methodology

Motivations. A Dynamic Text-Attributed Graph encompasses structural, temporal, and textual information to describe the dynamic inter-relationships between entities. We design LKD4DyTAG utilizing knowledge distillation approach to encode the semantic understanding of a teacher LLM regarding edge creation and edge types from the text description of network elements of an DyTAG into the representation of a lightweight student GNN model, which captures both structural and temporal information. First, we learn two different representations for each edge, one from the student GNN which includes structural and temporal information and the other from LLM that covers textual information. Then, we distill the semantic understanding of the LLM representation into lightweight GNN representation by adopting knowledge distillation mechanism. We illustrate the model architecture of LKD4DyTAG in Figure 2 and provide the pseudocode in Algorithm 1 in the Appendix.

Spatio-temporal Edge Embedding (Student Model). To simultaneously encode structural and temporal information into edge representations, we propose a simple yet effective temporal edge encoding method. For each edge e_{uv}, we use a vector $\tau \in \mathbb{R}^{T+k}$ to represent the status of an edge—its non-existence up to timestamp t_{uv} and its existence thereafter—by setting $\tau[0 : t_{uv}] = 0, \tau[t_{uv} : T] = 1$. Intuitively, we incorporate temporal information into the edges by using 0 and 1 that denote the non-existence and existence of an edge, respectively. Furthermore, we mask the future timestamps, T to $T+k$ during the training time, i.e., $\tau[T : T+k] = -1$. With the temporal information encoded in the edges, we can apply message-passing Graph Neural Networks (GNNs) to learn edge representations that capture both structural and temporal information. We denote such embedding as $h_{spatio\_temporal}$.

First, we obtain the node and edge representations using a multi-layer perceptron over the BERT embeddings of the node and edge text.

$$h_u = \phi_{mlp\_student}(BERT(u_{text})) \quad (1)$$
$$h_v = \phi_{mlp\_student}((BERT(v_{text}))) \quad (2)$$
$$h_{uv} = \phi_{mlp\_student}(BERT(r_{text})) \quad (3)$$

Here, $\phi_{mlp\_student} : \mathbb{R}^{d_{BERT}} \rightarrow \mathbb{R}^{d_{student}}$ is a multi-layer perceptron that transforms the output embeddings of node text description u_{text}, v_{text}, and edge text description r_{text} from a frozen BERT encoder into node representations $h_u \in \mathbb{R}^{d_{student}}$ and $h_v \in \mathbb{R}^{d_{student}}$, as well as edge representation $h_{uv} \in \mathbb{R}^{d_{student}}$, which represents the relation r_{uv} between node u and v at timestamp t_{uv}. Here d_{bert} and $d_{student}$ denote the dimensionalities of the BERT embeddings and student model embeddings, respectively.

Next, we concatenate h_{uv} with the temporal encoding τ to incorporate temporal information into edges, resulting in $\hat{h}_{uv}$. We use a 1-layer message passing GNN ψ, which considers edge attributes to capture structural and temporal information through neighborhood aggregation, producing the node representations $\bar{h}_u$ and $\bar{h}_v$. Here, $AGG(\cdot)$ can be any aggregation function that combines the neighborhood node and edge embeddings, while $UPDATE(\cdot)$ function integrates the aggregated neighborhood embedding with the target node embedding. After that, we perform element-wise

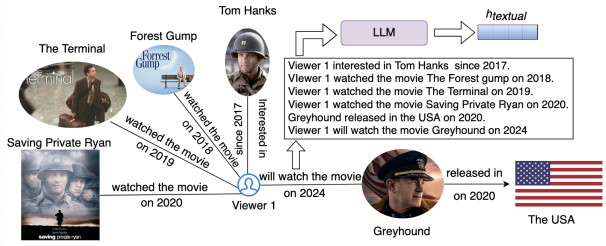

Figure 3: **Textual edge representation from the LLM using the descriptions of adjacent nodes' neighbors.**

multiplication (Hadamard product) between adjacent node representations $\bar{h}_u$ and $\bar{h}_v$ after the graph convolution to obtain $h_{spatio\_temporal}$ for each edge.

$$\hat{h}_{uv} = h_{uv}||\tau \tag{4}$$

$$\bar{h}_u = \text{UPDATE}(h_u, \text{AGG}(\{(h_w, \hat{h}_{uw}) : w \in \mathcal{N}_u\})) \tag{5}$$

$$\bar{h}_v = \text{UPDATE}(h_v, \text{AGG}(\{(h_w, \hat{h}_{vw}) : w \in \mathcal{N}_v\})) \tag{6}$$

$$h_{spatio\_temporal} = \bar{h}_u \odot \bar{h}_v \tag{7}$$

With the edge representation $h_{spatio\_temporal}$ learnt from the temporal encoded graph structure by the message passing GNN, we predict the temporal edge encoding $\hat{\tau}$ as follows:

$$\hat{\tau} = \sigma(\phi_{flp}(h_{spatio\_temporal})) \tag{8}$$

where ϕ_{flp} is a multi-layer perceptron which maps the spatio-temporal representation into a vector of size equal to the total number of timestamps of interest, i.e., $\phi_{flp} : \mathbb{R}^{d_{student}} \rightarrow \mathbb{R}^{T+k}$ and σ can be an activation function, such as sigmoid.

We learn the GNN parameters by minimizing the binary cross entropy (BCE) loss $\mathcal{L}_{flp}$ that denote the difference between the original temporal edge encoding and predicted one for the future link prediction task.

$$\mathcal{L}_{flp} = -\sum_{i=0}^{T+k} \tau_i \log(\hat{\tau}_i) \tag{9}$$

For the edge classification task, we utilize a multi-layer perceptron ϕ_{ec} to transform the $h_{spatio\_temporal}$ into the predicted class probabilities of the edges ,i.e., $\phi_{ec} : \mathbb{R}^{d_{student}} \rightarrow \mathbb{R}^{|\mathcal{L}|}$.

$$\hat{l}_{uv} = \sigma(\phi_{ec}(h_{spatio\_temporal})) \tag{10}$$

We minimize the BCE loss between the one hot encoding of the true labels of the edge class label l and the predicted class probabilities $\hat{l}$.

$$\mathcal{L}_{ec} = -\sum_{i=0}^{|\mathcal{L}|} l_i \log(\hat{l}_i) \tag{11}$$

Textual Edge Embedding (Teacher Model). Compared to traditional dynamic graphs, DyTAGs are rich in textual information associated with their nodes and edges, necessary for completely describing real-world dynamically changing systems. Therefore, in addition to structural and temporal information, the semantics from the textual information from edge relations, adjacent nodes, and their neighbors is valuable for predicting future links or classifying the label of an edge.

Recently, large language models (LLMs) (Brown et al. 2020) have shown impressive capabilities in understanding text semantics. However, their high storage and computational requirements challenge their direct use in real-time systems like DyTAGs. Assuming an LLM can effectively encode edge context into a text-based representation, we aim to use the LLM as a teacher model to distill this knowledge into a spatio-temporal representation by aligning the two in the latent space. This approach leverages the LLM's text processing strengths to efficiently encode semantic understanding into dynamic text-attributed graph representations, in a computationally and resource-efficient manner. Modern LLMs, such as Llama2 (Touvron et al. 2023), ChatGPT (Brown et al. 2020), Gemini AI (Reid et al. 2024), excel at generating embeddings that capture text semantics – crucial for predicting future edge existence and types in link prediction. We incorporate these capabilities by distilling the LLM's textual edge representation into our spatio-temporal representation. We initially obtain the representation of an edge by the describing text of the 1-hop neighborhood of the adjacent nodes and also the edge itself, as illustrated in Figure 3.

For each true edge or fake sampled edge e_{uv} between two adjacent entities u and v in a particular dynamic text-attributed graph, we design a text prompt by describing the 1-hop neighborhood of the adjacent nodes u and v. We add a decision sentence to denote whether the edge is true or false for future link prediction between entity u and v at timestamp t via relation r. Label information for the link prediction task is also included. Note that this latent distilled knowledge objective is not available at test time.

LLM-driven Knowledge Distillation. The LLM's representation can grasp the semantics and context of edge creation and the adjacent nodes. However, obtaining such representation from an LLM using text prompt is resource-intensive and computationally demanding. Therefore, we employ knowledge distillation to transfer the LLM's capabilities to our spatio-temporal representation, which can be obtained in a resource-efficient and scalable manner.

We distill the knowledge from LLM-based representation into spatio-temporal encoding from GNN-based representation by making $h_{spatio\_temporal}$ to be as close as $h_{textual}$ in the latent space.

First, we compute the neighborhood textual embedding $h_{\mathcal{N}(u)} \in \mathbb{R}^{d_{teacher}}$ for each node u by describing each neighboring edge e_{uw} in the node's 1-hop neighborhood to the LLM, where $d_{teacher}$ is the dimensionality of the MLP-transformed textual edge representation.

$$\bar{h}_{\mathcal{N}(u)} = \sum_{w \in \mathcal{N}(u)} LLM(neighbor\_prompt(e_{uw})) \tag{12}$$

$$h_{\mathcal{N}(u)} = \phi_{mlp\_teacher}(\bar{h}_{\mathcal{N}(u)}) \tag{13}$$

Here, $\phi_{mlp\_teacher}$ is an MLP for the teacher model which transforms the combined frozen LLM embeddings into textual representation for each edge, i.e., $\phi_{mlp\_teacher} : \mathbb{R}^{d_{LLM}} \rightarrow \mathbb{R}^{d_{teacher}}$ where $\mathbb{R}^{d_{LLM}}$ is the dimensionality of LLM output embeddings.

Next, for each edge e_{uv} we sum up the neighborhood textual embedding $h_{\mathcal{N}(u)}$ and $h_{\mathcal{N}(v)}$ of the adjacent nodes u and v with the $\tilde{h}_{uv}$, the textual link representation for edge e_{uv}. We use an/no term in the edge description to differentiate between true and fake edges for the link prediction task during the training. This serves as a mechanism for the LLM to contrast positive and negative edges in the hidden representation.

$$\tilde{h}_{uv} = \phi_{mlp\_teacher}(LLM(link\_prompt(e_{uv}))) \quad (14)$$

$$h_{textual} = h_{\mathcal{N}(u)} + h_{\mathcal{N}(v)} + \tilde{h}_{uv} \quad (15)$$

Our assumption is that LLM can produce an appropriate representation of each edge in a DyTAG by comprehending the semantics of edge creation and edge labels since the node and edge texts describe actual relationships between adjacent entities in DyTAGs. We aim to align the temporal edge encoding-based spatio-temporal representation from the GNN with the textual edge encoding from the LLM. This is achieved by minimizing the negative exponential of the similarity between the two embeddings, thereby bringing $h_{textual}$ and $h_{spatio\_temporal}$ closer by reducing the angle between these representations in the latent space.

We normalize both embeddings $h_{textual}$ and $h_{spatio\_temporal}$. Then, we define our knowledge distillation loss as the negative exponential of the dot product between $\bar{h}_{textual}$ and $\bar{h}_{spatio\_temporal}$, intuitively the similar $\bar{h}_{textual}$ and $\bar{h}_{spatio\_temporal}$ are in the latent space the smaller $\mathcal{L}_{kd}$ is. Note that the LLM is frozen and knowledge distillation occurs through the latent space.

$$\bar{h}_{textual} = \frac{h_{textual}}{||h_{textual}||} \quad (16)$$

$$\bar{h}_{spatio\_temporal} = \frac{h_{spatio\_temporal}}{||h_{spatio\_temporal}||} \quad (17)$$

$$\mathcal{L}_{kd} = e^{-(\bar{h}_{textual}^{\top} \bar{h}_{spatio\_temporal})} \quad (18)$$

Here, $\lambda_{flp}, \lambda_{ec}, \lambda_{kd}$ are the hyperparameters that denote the importance of the future link prediction, edge classification, and knowledge distillation loss.

Experiments

To comprehensively verify the effectiveness of LKD4DyTAG , we conduct extensive experiments and attempt to answer the following questions:

- How well LKD4DyTAG , utilizing knowledge distillation technique from teacher LLM to student GNN model, comprehend the semantics of link creation and edge types for the future link prediction and edge classification tasks?
- How does knowledge distillation impacts the performance of LKD4DyTAG in future link prediction task under both transductive and inductive setting?
- What is the impact of the knowledge distillation loss on future link prediction and edge classification task?

Experimental Setup

Dataset and Baseline. To validate the efficacy of LKD4DyTAG on future link prediction and edge classification task, we compare with the performance of LKD4DyTAG with seven baseline from the DTGB (Zhang et al. 2024) paper for six Dynamic Text-Attributed Graphs. We present the statistics of the dynamic text-attributed graphs used in the benchmark in Table 1. Among the tested baseline methods, we include the RNN-based approach JODIE (Kumar, Zhang, and Leskovec 2019), DyReP (Trivedi et al. 2019), self-attention based approach TGAT (Xu et al. 2020), causal anonymous random-walk based approach CAWN (Wang et al. 2021b), transformer based approach TCL (Wang et al. 2021a), fixed time-encoding and summarization based approach GraphMixer (Cong et al. 2023), and patching with transformer based DyGFormer (Yu et al. 2023). A detailed description of the datasets is provided in the Appendix.

Experimental Settings. In our experimental setting, we first obtain the node and edge embeddings from a comparatively lightweight language model BERT (Devlin et al. 2018). Also, for each edge we add a zero-vector of shape equal to the total number of timestamps. We set it as 1 from the timestamp when an edge has occurred until the end to denote the existence of an edge, and use -1 to mask the future part. We do not consider edge drop which can be denoted using 0. We concatenate the BERT embedding with the temporal edge encoding. For obtaining the spatio-temporal embedding, we use GATConv (Kipf and Welling 2016) and TransformerConv (Shi et al. 2020) from Pytorch geometric package[1] which utilizes edge features during graph convolution to produce node features. For all datasets, we split the data into training, validation, and test sets using a *70/15/15* ratio based on the timestamps of the edges. Additionally, during training we sample an equal number of fake edges not originally present in the dataset for the future link prediction task. This helps train the student model to distinguish between true and fake edges through the knowledge distillation process. The temporal encoding for the fake edges is kept as a zero vector. We run each experiment on every dataset five times for 50 epochs and report the mean and standard deviation for difference metrics.

To obtain textual representations from the text prompt efficiently, we use superfastllm [2] from Huggingface (Wolf et al. 2019) as our LLM model. We compute the BERT embeddings and the textual edge embedding as a preprocessing step for training of each epoch. For the future link prediction task, we adopt the AUC-ROC metric to compare the performance between LKD4DyTAG and the baseline

[1] https://pytorch-geometric.readthedocs.io/en/latest/

[2] https://huggingface.co/power-greg/super-fast-llm

Dataset	Nodes	Edges	Edge Categories	Timestamps	Domain	Text Attributes
Enron	42,711	797,907	10	1,006	E-mail	Node & Edge
GDELT	6,786	1,339,245	237	2,591	Knowledge graph	Node & Edge
ICEWS1819	31,796	1,100,071	266	730	Knowledge graph	Node & Edge
Googlemap CT	111,168	1,380,623	5	55,521	E-commerce	Node & Edge
Stack elec	397,702	1,262,225	2	5,224	Multi-round dialogue	Node & Edge
Stack ubuntu	674,248	1,497,006	2	4,972	Multi-round dialogue	Node & Edge

Table 1: Dataset statistics of Dynamic Text-Attributed Graph from DTGB benchmark (Zhang et al. 2024).

Metric	Task Type	Datasets	JODIE	DyRep	TGAT	CAWN	TCL	GraphMixer	DyGFormer	LKD4DyTAG
ROC-AUC	*tr.*	Enron	0.9731 ± 0.0052	0.9274 ± 0.0026	0.9681 ± 0.0026	0.9740 ± 0.0007	0.9618 ± 0.0025	0.9567 ± 0.0013	0.9779 ± 0 .0014	**0.9887 ± 0.0011**
		ICEWS1819	0.9741± 0.0113	0.9632 ± 0.0027	0.9904 ± 0.0039	0.9857 ± 0.0018	0.9923 ± 0.0012	0.9863 ± 0.0024	0.9888 ± 0.0015	**0.9950 ± 0.0014**
		Googlemap CT	*OOM*	*OOM*	0.9049 ± 0.0071	0.8687 ± 0.0063	0.8348 ± 0.0094	0.8095 ± 0.0014	0.8207 ± 0.0018	**0.9127 ± 0.0037**
		GDELT	0.9533 ± 0.0020	0.9453 ± 0.0018	0.9595 ± 0.0033	0.9600 ± 0.0061	0.9619 ± 0.0008	0.9552 ± 0.0018	0.9662 ± 0.0003	**0.9998 ± 0.0001**
	in.	Enron	0.8732 ± 0.0037	0.7901 ± 0.0047	0.8650 ± 0.0032	0.9091 ± 0.0014	0.8512 ± 0.0062	0.8347 ± 0.0039	0.9316 ± 0.0015	**0.9367 ± 0.0011**
		ICEWS1819	0.9285 ± 0.0065	0.9030 ± 0.0097	0.9706 ± 0.0054	0.9774 ± 0.0039	**0.9778 ± 0.0012**	0.9605 ± 0.0025	0.9630 ± 0.0027	0.9543 ±. 0.0013
		Googlemap CT	*OOM*	*OOM*	0.8791 ± 0.0028	0.7058 ± 0.0047	0.7895 ± 0.0046	0.7895 ± 0.0046	0.7648 ± 0.0052	**0.8857 ± 0.2327**
		GDELT	0.8921 ± 0.0065	0.8917 ± 0.0007	0.9012 ± 0.0011	0.8899 ± 0.0082	0.9099 ± 0.0022	0.8942 ± 0.0035	0.9206 ± 0.0003	**0.9577 ± 0.0012**
AP	*tr.*	Enron	0.9553 ± 0.0051	0.9066 ± 0.0076	0.9668 ± 0.0026	0.9756 ± 0.0008	0.9603 ± 0.0018	0.9559 ± 0.0027	0.9804 ± 0.0015	**0.9943 ± 0.0001**
		ICEWS1819	0.9752 ± 0.0037	0.9676 ± 0.0026	0.9908 ± 0.0032	0.9886 ± 0.0025	0.9927 ± 0.0012	0.9871 ± 0.0034	0.9901 ± 0.0018	**0.9970 ± 0.0012**
		Googlemap CT	*OOM*	*OOM*	0.9002 ± 0.0019	0.8721 ± 0.0027	0.8335 ± 0.0018	0.8072 ± 0.0010	0.8183 ± 0.0038	**0.9144 ± 0.0013**
		GDELT	0.9466 ± 0.0032	0.9416 ± 0.0017	0.9572 ± 0.0029	0.9582 ± 0.0053	0.9601 ± 0.0011	0.9523 ± 0.0020	0.9653 ± 0.0003	**0.9776 ± 0.0034**
	in.	Enron	0.8761 ± 0.0023	0.7734 ± 0.0044	0.8589 ± 0.0031	0.9223 ± 0.0011	0.8560 ± 0.0024	0.8328 ± 0.0034	0.9409 ± 0.0025	**0.9550 ± 0.0013**
		ICEWS1819	0.9333 ± 0.0026	0.9134 ± 0.0041	0.9716 ± 0.0033	0.9631 ± 0.0034	**0.9789 ± 0.0022**	0.9625 ± 0.0030	0.9688 ± 0.0018	0.9634 ± 0.0010
		Googlemap CT	*OOM*	*OOM*	0.8750 ± 0.0015	0.8012 ± 0.0021	0.7936 ± 0.0009	0.7633 ± 0.0013	0.7735 ± 0.0031	**0.8890 ± 0.1134**
		GDELT	0.9019 ± 0.0023	0.8928 ± 0.0011	0.9023 ± 0.0010	0.8986 ± 0.0077	0.9151 ± 0.0045	0.8925 ± 0.0048	0.9263 ± 0.0009	**0.9369 ± 0.0056**

Table 2: Performance comparison of *ROC-AUC* and *AP* for **future link prediction** for transductive (*tr.*) and inductive (*in.*) setting. (*OOM* means Out-Of-Memory.)

Datasets	Metrics	JODIE	DyRep	TGAT	CAWN	TCL	GraphMixer	DyGFormer	LKD4DyTAG
Enron	Precision	0.6568 ± 0.0043	0.6635 ± 0.0052	0.6148 ± 0.0012	0.6076 ± 0.0070	0.5530 ± 0.0079	0.6313 ± 0.0024	0.6601 ± 0.0067	**0.6685 ± 0.0013**
	Recall	0.6472 ± 0.0039	0.6390 ± 0.0089	0.5530 ± 0.0001	0.5783 ± 0.0094	0.5394 ± 0.0061	0.5735 ± 0.0015	0.5802 ± 0.0071	**0.6567 ± 0.0013**
	F1	0.6478 ± 0.0065	0.6432 ± 0.0062	0.5519 ± 0.0028	0.5685 ± 0.0132	0.5177 ± 0.0044	0.5507 ± 0.0019	0.5604 ± 0.0063	**0.6675 ± 0.0014**
GDELT	Precision	0.1361 ± 0.0036	0.1451 ± 0.0071	0.1241 ± 0.0056	0.1781 ± 0.0011	0.1229 ± 0.0021	0.1293 ± 0.0026	0.1775 ± 0.0041	**0.2139 ± 0.0012**
	Recall	0.1338 ± 0.0013	0.1365 ± 0.0013	0.1321 ± 0.0012	0.1545 ± 0.0001	0.1235 ± 0.0047	0.1320 ± 0.0008	0.1580 ± 0.0052	**0.2159 ± 0.0016**
	F1	0.0992 ± 0.0009	0.1039 ± 0.0012	0.0967 ± 0.0010	0.1340 ± 0.0012	0.0987 ± 0.0051	0.1014 ± 0.0017	0.1291 ± 0.0068	**0.2234 ± 0.0023**
ICEWS1819	Precision	0.3106 ± 0.0023	0.3270 ± 0.0025	0.3013 ± 0.0007	0.3451 ± 0.0023	0.3212 ± 0.0096	0.2999 ± 0.0022	0.3297 ± 0.0034	**0.3743 ± 0.0043**
	Recall	0.3494 ± 0.0018	0.3636 ± 0.0020	0.3512 ± 0.0006	0.3676 ± 0.0034	0.3517 ± 0.0009	0.3502 ± 0.0001	0.3632 ± 0.0026	**0.3823 ± 0.0013**
	F1	0.2965 ± 0.0008	0.3097 ± 0.0006	0.2908 ± 0.0008	0.3156 ± 0.0057	0.2939 ± 0.0022	0.2903 ± 0.0008	0.3079 ± 0.0027	**0.3243 ± 0.0013**
Googlemap CT	Precision	0.6163 ± 0.0032	0.6073 ± 0.0019	0.6160 ± 0.0001	0.6166 ± 0.0023	0.6213 ± 0.0087	0.6171 ± 0.0020	0.6166 ± 0.0003	**0.6305 ± 0.0057**
	Recall	0.6871 ± 0.0002	0.6827 ± 0.0006	0.6862 ± 0.0002	0.6870 ± 0.0001	0.6875 ± 0.0001	0.6872 ± 0.0003	0.6877 ± 0.0002	**0.6905 ± 0.0002**
	F1	0.6189 ± 0.0016	0.6134 ± 0.0006	0.6225 ± 0.0015	0.6187 ± 0.0003	0.6230 ± 0.0003	0.6185 ± 0.0005	0.6196 ± 0.0008	**0.6354 ± 0.0012**
Stack elec	Precision	*OOM*	*OOM*	0.6265 ± 0.0046	0.6167 ± 0.0094	0.6325 ± 0.0023	0.6074 ± 0.0039	0.6026 ± 0.0471	**0.6565 ± 0.0056**
	Recall	*OOM*	*OOM*	0.7205 ± 0.0094	0.6313 ± 0.0462	0.7474 ± 0.0004	0.7412 ± 0.0061	0.5891 ± 0.2747	**0.7556 ± 0.0056**
	F1	*OOM*	*OOM*	0.6496 ± 0.0032	0.6209 ± 0.0216	0.6420 ± 0.0003	0.6412 ± 0.0005	0.4860 ± 0.2686	**0.6796 ± 0.0024**
Stack ubuntu	Precision	*OOM*	*OOM*	0.6858 ± 0.0047	0.6921 ± 0.0040	0.6915 ± 0.0118	**0.6930 ± 0.0028**	0.6789 ± 0.0490	0.6824 ± 0.0054
	Recall	*OOM*	*OOM*	**0.7921 ± 0.0012**	0.5650 ± 0.1015	0.7880 ± 0.0026	0.7902 ± 0.0130	0.7494 ± 0.0991	0.7554 ± 0.1224
	F1	*OOM*	*OOM*	0.7201 ± 0.0013	0.6002 ± 0.0738	**0.7219 ± 0.0046**	0.7214 ± 0.0014	0.7033 ± 0.0294	0.7156 ± 0.0056

Table 3: Performance of dynamic graph learning methods on **edge classification** task. (*OOM* means Out-Of-Memory.)

models. For the edge classification task, we report precision, recall, and F1 score to compare the performance difference.

Hyperparameter Tuning. To tun the performance of our models, we perform grid search of the hyperparameters of LKD4DyTAG as follows: 1) Dimensionalities of student and teacher models $d_{student}, d_{teacher} \in \{16, 32, 64, 128, 256\}$; 2) Number of MLP layers of $\phi_{mlp\_student}$ and $\phi_{mlp\_teacher} \in \{1, 2, 3\}$; 3) Number of layers in message passing GNN $\psi \in \{1, 2, 3\}$; 4) Future Link Prediction, Edge Classification, Knowledge Distillation hyperparameter $\lambda_{flp}, \lambda_{ec}, \lambda_{kd} \in \{0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0\}$; 5) GNN Type $\in$ {GATConv, TransformerConv}; 6) Batch Size $\in \{32, 64, 128, 256, 512, 1024, 2048\}$; 7) learning rate $\in \{0.01, 0.001, 0.0001\}$. Also, we choose Adam optimizer (Kingma and Ba 2014) to update the parameters of the student model using the task-specific and knowledge distillation loss.

Hardware. All experiments are conducted on a Linux server equipped with a 2.20 GHz Intel Xeon E5-2650 v4 processor and four NVIDIA Tesla V100 GPUs, each with 32 GB of VRAM.

Results

Future Link Prediction and Edge Classfication Results. We present the performance comparsion of LKD4DyTAG with baseline approaches for the future link prediction task in the Table 2 and for the edge classification task in the Table 3. From Table 2, we observe that LKD4DyTAG outperforms the baseline approaches in the future link prediction task for all four datasets in the transductive setting. From this we can conclude that LKD4DyTAG, with the temporal encoding and knowledge distillation loss, effectively captures both the temporal dynamics and the semantics of edge creation for future link prediction in the transductive setting. However, for inductive future link prediction, we observe that the performance of LKD4DyTAG on the ICEWS1819 is not superior. Note that ICEWS1819 dataset represents the edges between political entities where the timestamp has much sparser granularity, which makes it challenging to grasp the semantics behind edge creation

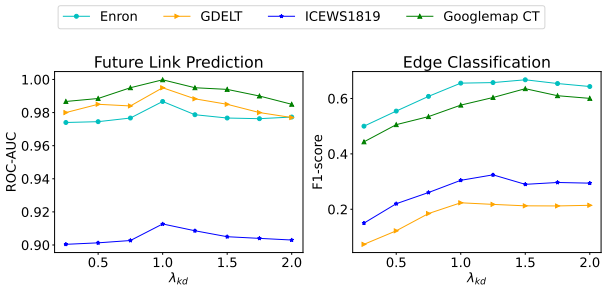

Figure 4: The impact of λ_{kd} for **future link prediction** and **edge classification** tasks.

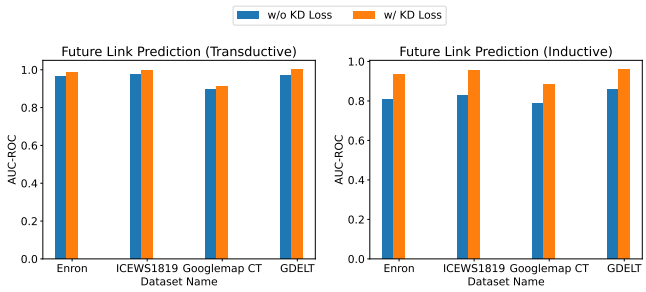

Figure 5: The impact of knowledge distillation on the **future link prediction** task for the inductive setting (right) and the transductive setting (left).

when new nodes appear in the test timestamps. This may explain LKD4DyTAG's performance degradation compared to state-of-the-art approaches in inductive link prediction on the ICEWS1819 dataset. For the edge classification task, we present the performance comparison on precision, recall, and F1 score in the Table 3. We observe that LKD4DyTAG can outperform the baseline approaches in all the datasets in all three metrics except for the Stack ubuntu dataset. In stack ubuntu, the text information of the answer edges include code as well as plain text between users and questions while the edge label is whether the answer is useful or not useful. The mixture of code and natural language might make it difficult to understand the semantics and thus the performance of LKD4DyTAG for edge classification is not superior compared to baseline models.

Impact of Knowledge distillation for future link prediction and edge classification. To understand how knowledge distillation process impact future link prediction and edge classification tasks, we vary the hyperparameter $\lambda_{kd} \in \{0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0\}$ to analyze the impact on both tasks and present the results in Figure 4. Intuitively, the creation of a future edge should primarily rely on the text information of neighboring nodes, with structural and temporal information also playing a significant role. On the other hand, the edge classification should mostly depend on the semantic meaning of the text description of the edge to determine the edge label. From Figure 4, we can observe a clear trend on the edge classification task, as λ_{kd} increases, the performance improves across all datasets up to a certain point and then stabilizes. For future link prediction task, the performance change is less pronounced when the impact of knowledge distillation loss increases. Therefore, semantic information has a greater impact on LKD4DyTAG 's edge classification performance than on future link prediction. From this ablation study of knowledge distillation, we conclude that semantic information is more critical for edge classification task than the future link prediction task in Dynamic Text-Attributed Graphs (DyTAGs).

Knowledge Distillation for transductive and inductive link prediction. To analyze the impact of semantic information from textual edge representation on the future link prediction task for transductive and inductive settings, we conduct experiments with and without the knowledge distillation loss. From Figure 5, for different datasets, the performance difference with knowledge distillation loss is more significant for the inductive setting than for the transductive setting. We observe that adding knowledge distillation impacts the performance of future link prediction task more on the inductive setting. In inductive link prediction, new nodes are introduced during inference, and knowledge distillation enhances the student GNN model's ability to understand the semantic context of edge creation. In the inductive setting, where new nodes are introduced during inference, the impact of knowledge distillation is more pronounced compared to the transductive setting, where the same nodes are present during both training and inference. Therefore, the ability to understand the semantics of edge creation with new nodes lead to better future link predictions for LKD4DyTAG .

Conclusion

We propose a simple yet efficient framework that encodes time in edges to integrate temporal and structural information together into the spatio-temporal representation. For the text component, we use knowledge distillation from LLM-driven textual representations of edges in dynamic text-attributed graphs. We show that distilling knowledge from text-based edge representations of the teacher LLM to the student GNN enables us to achieve state-of-the-art performance on benchmark datasets of dynamic text-attributed graphs for both future link prediction and edge classification tasks. Future directions of this work may possibly include exploration of very large dynamic text attributed graphs.

References

Alomrani, M.; Biparva, M.; Zhang, Y.; and Coates, M. 2023. DyG2Vec: Efficient Representation Learning for Dynamic Graphs. *TMLR*.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry; et al. 2020. Language models are few-shot learners. *NeurIPS*.

Cai, B.; Xiang, Y.; Gao, L.; Zhang, H.; Li, Y.; and Li, J. 2022. Temporal knowledge graph completion: A survey. *arXiv preprint arXiv:2201.08236*.

Cong, W.; Zhang, S.; Kang, J.; Yuan, B.; Wu, H.; Zhou, X.; Tong, H.; and Mahdavi, M. 2023. Do we really need complicated model architectures for temporal networks? *arXiv preprint arXiv:2302.11636*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Fang, Y.; Fan, D.; Ding, S.; Liu, N.; and Tan, Q. 2024a. UniGLM: Training One Unified Language Model for Text-Attributed Graphs. *arXiv preprint arXiv:2406.12052*.

Fang, Y.; Fan, D.; Zha, D.; and Tan, Q. 2024b. Gaugllm: Improving graph contrastive learning for text-attributed graphs with large language models. *KDD*.

Hsieh, C.-Y.; Li, C.-L.; Yeh, C.-K.; Nakhost, H.; et al. 2024. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *ACL*.

Huang, Y.; Mao, X.; Guo, S.; Chen, Y.; Lin, Y.; and Wan, H. 2024. STD-LLM: Understanding Both Spatial and Temporal Properties of Spatial-Temporal Data with LLMs. *arXiv preprint arXiv:2407.09096*.

Kazemi, S. M.; Goel, R.; Jain, K.; Kobyzev, I.; Sethi, A.; Forsyth, P.; and Poupart, P. 2020. Representation learning for dynamic graphs: A survey. *JMLR*.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kumar, S.; Zhang, X.; and Leskovec, J. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *KDD*.

Liu, C.; He, S.; Zhou, Q.; Li, S.; and Meng, W. 2024a. Large language model guided knowledge distillation for time series anomaly detection. *arXiv preprint arXiv:2401.15123*.

Liu, C.; Yang, S.; Xu, Q.; Li, Z.; Long, C.; Li, Z.; and Zhao, R. 2024b. Spatial-temporal large language model for traffic prediction. *arXiv preprint arXiv:2401.10134*.

Liu, L.; Yu, S.; Wang, R.; Ma, Z.; and Shen, Y. 2024c. How can large language models understand spatial-temporal data? *arXiv preprint arXiv:2401.14192*.

Luo, R.; Gu, T.; Li, H.; Li, J.; Lin, Z.; Li, J.; and Yang, Y. 2024. Chain of history: Learning and forecasting with llms for temporal knowledge graph completion. *arXiv preprint arXiv:2401.06072*.

Luo, X.; Yuan, J.; Huang, Z.; Jiang, H.; Qin, Y.; Ju, W.; Zhang, M.; and Sun, Y. 2023. Hope: High-order graph ode for modeling interacting dynamics. In *ICML*. PMLR.

Pan, B.; Zhang, Z.; Zhang, Y.; Hu, Y.; and Zhao, L. 2024. Distilling large language models for text-attributed graph learning. *arXiv preprint arXiv:2402.12022*.

Reid, M.; Savinov, N.; Teplyashin, D.; Lepikhin, D.; Lillicrap, T.; et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Shi, Y.; Huang, Z.; Feng, S.; Zhong, H.; Wang, W.; and Sun, Y. 2020. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*.

Skarding, J.; Gabrys, B.; and Musial, K. 2021. Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey. *iEEE Access*.

Tang, H.; Wu, S.; Xu, G.; and Li, Q. 2023. Dynamic graph evolution learning for recommendation. In *SIGIR*.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Trivedi, R.; Farajtabar, M.; Biswal, P.; and Zha, H. 2019. Dyrep: Learning representations over dynamic graphs. In *ICLR*.

Wang, L.; Chang, X.; Li, S.; Chu, Y.; Li, H.; Zhang, W.; He, X.; Song, L.; Zhou, J.; and Yang, H. 2021a. Tcl: Transformer-based dynamic graph modelling via contrastive learning. *arXiv preprint arXiv:2105.07944*.

Wang, Y.; Chang, Y.-Y.; Liu, Y.; Leskovec, J.; and Li, P. 2021b. Inductive representation learning in temporal networks via causal anonymous walks. *arXiv preprint arXiv:2101.05974*.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Xu, D.; Ruan, C.; Korpeoglu, E.; Kumar, S.; and Achan, K. 2020. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962*.

Xu, J.; Wu, Z.; Lin, M.; Zhang, X.; and Wang, S. 2024. LLM and GNN are Complementary: Distilling LLM for Multimodal Graph Learning. *arXiv preprint arXiv:2406.01032*.

Yu, L.; Sun, L.; Du, B.; and Lv, W. 2023. Towards better dynamic graph learning: New architecture and unified library. *NeurIPS*, 36: 67686–67700.

Zhang, J.; Chen, J.; Yang, M.; Feng, A.; Liang, S.; Shao, J.; and Ying, R. 2024. DTGB: A Comprehensive Benchmark for Dynamic Text-Attributed Graphs. *NeurIPS 2024 Datasets and Benchmarks Track*.

Zhang, J.; Shao, J.; and Cui, B. 2023. StreamE: Learning to Update Representations for Temporal Knowledge Graphs in Streaming Scenarios. In *SIGIR*.

Zhang, Q.; Ren, X.; Xia, L.; Yiu, S. M.; and Huang, C. 2023. Spatio-temporal graph learning with large language model.

Appendix

Notations

We present the notations used to describe our model LKD4DyTAG in Table 4.

Psedocode

Algorithm 1: **LKD4DyTAG** : LLM-driven Knowledge Distillation for Dynamic Text-Attributed Graph with temporal encoding

Preprocessing:
function $neighbor\_prompt(e_{uv})$:
    prompt = "Entity u_{text} is connected to entity v_{text} via relation r_{uv} at timestamp t_{uv} with label l_{uv}"
    return prompt
end function
function $link\_prompt(e_{uv})$:
    *prompt = "There is **an/no** edge between entity u_{text} and entity v_{text} via relation r_{uv} at timestamp t_{uv} with label l_{uv}"*
    return prompt
end function
function $compute\_textual\_representation(V, E)$
    for each node $u \in V$ **do**
        $h_u = BERT(u_{text})$
        $h_{\mathcal{N}(u)} = \sum_{w \in \mathcal{N}(u)} LLM(neighbor\_prompt(e_{uw}))$
    end for
    for each edge $e_{uv} \in E$ **do** ▷ $e_{uv} = (r_{uv}, t_{uv}, l_{uv})$
        $h_{uv} = BERT(r_{uv})$
        $\tilde{h}_{uv} = LLM(link\_prompt(e_{uv}))$
        $h_{textual} = h_{\mathcal{N}(u)} + h_{\mathcal{N}(v)} + \tilde{h}_{uv}$
    end for
end function

LKD4DyTAG :
Input: DyTAG $G = (V, E)$ with timestamp and label set $\mathcal{T}, \mathcal{L}$ for each edge
compute_textual_representation (V,E)
for each edge $e_{uv} \in E$ **do** ▷ $e_{uv} = (r_{uv}, t_{uv}, l_{uv})$
    $\tau[: t_{uv}] = 0, \tau[t_{uv} : T + 1] = 1$
    $\tau[T : T + k + 1] = -1, \tau \in \mathbb{R}^{T+k}$
    $\hat{h}_{uv} = [h_{uv}||\tau]$ ▷ Temporal Encoding : Concatenate h_{uv} and τ
    $\bar{h}_u = \text{UPDATE}(h_u, \text{AGG}(\{(h_w, \hat{h}_{uw}) : w \in \mathcal{N}_u\}))$
    $\bar{h}_v = \text{UPDATE}(h_v, \text{AGG}(\{(h_w, \hat{h}_{vw}) : w \in \mathcal{N}_v\}))$
    $h_{spatio\_temporal} = \bar{h}_u \odot \bar{h}_v$
    $\hat{\tau} = \sigma(\phi_{flp}(h_{spatio\_temporal}))$
    $\hat{l}_{uv} = \sigma(\phi_{ec}(h_{spatio\_temporal}))$
    $\bar{h}_{textual} = \frac{h_{textual}}{||h_{textual}||}$
    $\bar{h}_{spatio\_temporal} = \frac{h_{spatio\_temporal}}{||h_{spatio\_temporal}||}$
    $\mathcal{L}_{flp} = -\sum_{i=0}^{T+k} \tau_i \log(\hat{\tau}_i)$
    $\mathcal{L}_{ec} = -\sum_{i=0}^{|\mathcal{L}|} l_i \log(\hat{l}_i)$
    $\mathcal{L}_{kd} = e^{-(\bar{h}_{textual}^{\top} \bar{h}_{spatio\_temporal})}$
    $\mathcal{L}_{flp\_with\_kd} = \lambda_{flp}\mathcal{L}_{flp} + \lambda_{kd}\mathcal{L}_{kd}$ ▷ Future Link Prediction
    $\mathcal{L}_{ec\_with\_kd} = \lambda_{ec}\mathcal{L}_{ec} + \lambda_{kd}\mathcal{L}_{kd}$ ▷ Edge Classification
end for

Time Complexity of LKD4DyTAG

As a pre-processing step, we use BERT encoder to obtain the embeddings for the node and edge text. Let us consider the number of layers in the BERT encoder is L_{BERT}, the input text length is n_{text} and the hidden dimension of BERT is d_{BERT}.

Then, to compute the self-attention part of BERT, the time complexity will be $O(n_{text}^2.d_{BERT})$ since each token depends on every other token while the feed-forward layers will take $O(n_{text}.d_{BERT}^2)$ For each layer of BERT, the total time complexity will be $O(n_{text}^2.d_{BERT} + n_{text}.d_{BERT}^2)$ and for L_{BERT} layers it will be $O(L_{BERT}.(n_{text}^2.d + n_{text}.d_{BERT}^2))$. Also, since we use text prompts for each edge in the teacher LLM to obtain teacher embeddings as a pre-processing step the time complexity will be similar i.e. $O(|E|L_{LLM}.(n_{text}^2.d_{teacher} + n_{text}.d_{teacher}^2))$ where $d_{teacher}$ is the dimension of teacher LLM and L_{LLM} is the dimension of the teacher LLM and $|E|$ is the total number of edges in the dynamic text attributed graph.

To obtain the edge encoding the time complexity will be $O(|E|.|T|)$ where $|E|$ denotes the total edges in the dynamic text-attributed graph while $|T|$ denotes the total number of timestamps.

We run student GNN to obtain the representation for the dynamic text-attributed graphs which can be parallelized and take the time complexity $O(L_{GNN}(n^2 + |E|d_{student}))$ where L_{GNN} is the number of layers in the GNN, n is the number of nodes in the dynamic text attributed graph and $|E|$ is the total number of edges in the dynamic text attributed graph and $d_{student}$ is the dimension of the student GNN model.

Related Works

Dynamic Graphs. JODIE (Kumar, Zhang, and Leskovec 2019) uses coupled recurrent neural networks to forecast embedding trajectories for entities, enabling predictions of their temporal interactions. DyRep (Trivedi et al. 2019) integrates recurrent node state updates with a deep temporal point process and temporal-attentive network to model evolving graph dynamics nonlinearly. TGAT (Xu et al. 2020) utilizes self-attention for aggregating temporal-topological neighborhood features and captures temporal patterns with a functional time encoding method based on Bochner's theorem. CAWN (Wang et al. 2021b) employs an anonymization strategy using sampled walks to explore network causality and generate node identities, which are encoded and aggregated using a neural network model to produce the final node representation. TCL (Wang et al. 2021a) utilizes a two-stream encoder for temporal neighborhoods of interaction nodes, integrating a graph-topology-aware Transformer with cross-attention to learn node representations considering both temporal and topological information. GraphMixer (Cong et al. 2023) demonstrates the effectiveness of fixed-time encoding for dynamic interactions using a simple architecture with components for link summarization, node summarization, and link prediction. DyGFormer (Yu et al. 2023) learns node representations from historical first-hop interactions using a neighbor co-occurrence encoding scheme and a patching technique to

Notation	Description
$\mathcal{G}$	Dynamic Text Attributed Graph
$\mathcal{V}, \mathcal{E}$	Set of Node and Edges
e_{uv}	Edge between node u and node v
r_{uv}	Relation between node u and v
t_{uv}	Timestamp of the edge between node u and v
l_{uv}	Label of the edge between node u and node v
u_{text}, v_{text}	Text Description of node u and v
r_{text}	Text description of an edge relation r
$\mathcal{T}$	Set of timestamps
$\mathcal{L}$	Set of edge labels
$\mathcal{N}_u$	Neighbor set of node u
$\mathcal{G}_T = (\mathcal{V}_T, \mathcal{E}_T)$	Dynamic Graph until timestmp T
k	Future edge timestamps to be predicted
τ	Temporal Edge Encoding
h_u, h_v	Embedding for nodes u and v (Student Model)
h_{uv}	Embedding for an edge e_{uv} (Student Model)
$\hat{h}_{uv}$	Edge embedding concatenated with temporal encoding
$\phi_{mlp\_student}$	MLP to transform the frozen BERT embedding to node representation (Student Model)
ψ	Message passing GNN (Student Model)
$h_{spatio\_temporal}$	Spatio-temporal Representation of an edge (Student Model)
d_{BERT}	Dimensionality of the BERT embeddings
$d_{student}$	Embedding for the spatio-temporal representation from student model
ϕ_{flp}	MLP for future link prediction (Student Model)
ϕ_{ec}	MLP for edge classification (Student Model)
$\hat{\tau}$	Student Model predicted Temporal Encoding
$\hat{l}$	Student Model predicted label vector
$h_{\mathcal{N}_u}, h_{\mathcal{N}_v}$	Neighborhood Textual Embedding of node u and node v
$\phi_{mlp\_teacher}$	MLP to transform the neighborhood textual representation
d_{LLM}	Dimensionality of the LLM embeddings
$d_{teacher}$	Embedding for the textual representation from teacher model
$\tilde{h}_{uv}$	LLM transformed output of link prompt of an edge e_{uv}
$h_{textual}$	Textual edge representation (Teacher Model)
$\bar{h}_{spatio\_temporal}$	Normalized Spatio-Temporal Edge Representation (Teacher Model)
$\bar{h}_{textual}$	Normalized Textual Edge Representation (Teacher Model)
L_{flp}	Future Link Prediction loss
L_{flp}	Edge Classification loss
L_{kd}	Knowledge Distillation loss
$BERT(.)$	Function call to BERT model with Text to obtain embedding
$LLM(.)$	Function call to any LLM with Text to obtain embedding

Table 4: Notations used in LKD4DyTAG

effectively leverage longer historical sequences. To include textual information into the nodes and edges of dynamic graphs, a recent benchmark, DTGB (Zhang et al. 2024) formally defines Dynamic Text-Attributed Graph and performs baseline comparisons on future link prediction and edge classification tasks.

LLMs for Temporal Data. STD-LLM (Huang et al. 2024) proposes spatial and temporal tokens and hypergraph learning module to effectively capture non-pairwise and higher-order spatial-temporal correlations, which also helps to understand the capabilities of large language models (LLMs) for spatio-temporal forecasting and imputation tasks. STLLM (Zhang et al. 2023) proposes to use spatio-temporal prompts to obtain representations from LLMs, which is further aligned with GNNs representations using the InfoNCE loss. In another work (Liu et al. 2024b), spatio-temporal embeddings are input into both frozen and unfrozen transformer blocks to produce representations for traffic prediction. STGLLM (Liu et al. 2024c) proposes a tokenizer and a LLM-based adapter for performing traffic prediction.

Knowledge Distillation involving LLMs and GNNs. There are several recent works focus on using knowledge distillation to transfer the text processing capabilities of LLMs to lightweight models (Hsieh et al. 2024) like GNNs for text-attributed graphs. LinguGKD (Xu et al. 2024) proposes a model for distilling knowledge from k-hop prompt representations in an LLM to k-hop GNN representations.

However, their approach describe the entire k-hop neighborhood which is not well-suited for graphs with long text in nodes and edges. LLM4TAG (Pan et al. 2024) proposes to learn text-attributed graph by knowledge distillation from LLMs to GNNs. Another recent work (Luo et al. 2024) utilizes LLMs for temporal knowledge graph completion by leveraging LLMs' capabilities of reasoning with particular attention to reverse logic. GAugLLM (Fang et al. 2024b) proposes an approach for improving the contrastive learning for Text-Attributed Graph by mixture of prompt experts. AnomalyLLM (Liu et al. 2024a) proposes a method for time series anomaly detection by knowledge distillation. UniGLM (Fang et al. 2024a) proposes an unified graph language model that generalizes both in-domain and cross-domain TAGs. GALLON (Xu et al. 2024) proposes a multi-modal knowledge distillation strategy to transfer the text and structure processing capability of LLMs and GNNs into MLPs for molecular property prediction.

Dataset Description

Data Format: For each of the Dynamic Text-Attributed Graph (DyTAG) dataset, we have three different files *edge_list.csv*, *entity_text.csv*, *relation_text.csv*. The *edge_list.csv* file contains the dynamicallly evolving grpah structure, where each line describe the source node id, the destination node id, the relation id, timestamp and label of the edge. The *entity_text.csv* and *relation_text.csv* contains the mapping of node id and relation id to the rich text description. We download the datasets from the github repository [3] of the benchmark paper DTGB (Zhang et al. 2024). We present a short description of each of the six DyTAG dataset below.

- **Enron** [4]: This dataset is derived from the email interactions between the employees of ENRON energy corporation from 1999 to 2002. The nodes contain the employee profiles and their position and the edges include the truncated emails between them without non-English statement, abnormal symbols, tables from the raw text. The edge categories are the type of the emails e.g. calendar, notes, deal communication etc. The edges of this dataset are ordered by the sending time of the email.
- **GDELT** [5]: This dataset is based on the political behavior across all countries in the world, derived by the Global Database of Events, Language, and Tone project. The nodes denote the political entities e.g. "United States", "Donald Trump" whereas the edges denote the relationship between these nodes e.g. "born in", "president of". Edge categories are based on the political relationship/-type of behavior and the edges are ordered based on the timestamp of the occuring events.
- **ICEWS1819** [6]: This is another dynamic text-attributed graph dataset based on political events from 2018-01-01 to 2019-12-31 derived by the Integrated Crisis Early Warning System project. The name, sector and nationality of each political entity is used as the text attribute and the edges denote the political relationship. The edge categories represents the political relationship/behavior and the edges are sorted based on the timestamp of the occurring events. The difference between GDELT and ICEWS1819 is that GDELT describes the event in a more fine-grained way (15 minutes of time interval) whereas the events of ICEWS1819 are more sparse grained (24 hours of time interval). The number of nodes of ICEWS1819 is 4 times more than the GDELT dataset, therefore represents more sparse scenario.
- **Stack elec** [7]: Stack exchange data contains anonymized stack exchange data with question, answer, comments, tags etc. The nodes denote the question and user whereas the edges denote the answer and comments between the user node and question node, describing an dynamic bipartite graph which allow multi-round dialogue between users and questions. The stack elec dataset contains stack exchange data where the questions are related with electronic techniques and their corresponding answers and comments. For user nodes the self-introduction and the name of the technical areas of users expertise are used as the text whereas for the question nodes the title and body of each question are used as the text attributes. For the edges the text of the answer is used as the text attribute.Two types of edge categories are defined based on the answers, *useful* if the number of votes on the answer is greater than 1 else *useless*. The edges are ordered based on the answering timestamp from users.
- **Stack ubuntu** [8]: Stack ubuntu dataset is another stack exchange dataset containing the questions related to ubuntu system. The answers contains mixture of codes and natural language which makes the understanding of semantic context of interactions more challenging.
- **Googlemap CT** [9]: This dataset is a review dataset extracted from the Google Local Data project, containing review information on google map between users and business entities of the Connecticut state up to September 2021. Nodes are users and business entities whereas the edges are reviews of from user nodes to business nodes. Only the business nodes contains the text descriptions which is the name, address, category, and self-introduction of the business entity. The edge text attributes are the raw text of user reviews without the emojis and meaningless characters. The edge categories are review ranting ranging from 1 to 5. The edges in the dataset are ordered based on the timestamp of the review.

Impact of Temporal Encoding on Future Link Prediction and Edge Classification: To understand the impact of temporal encoding, we run experiments including and excluding the temporal encoding as the edge attribute into the input of student-model. We observe that the performance of LKD4DyTAG on both future link prediction and the edge

[3] https://github.com/zjs123/DTGB

[4] https://www.cs.cmu.edu/ enron/

[5] https://www.gdeltproject.org/

[6] https://dataverse.harvard.edu/dataverse/icews

[7] https://archive.org/details/stackexchange

[8] https://archive.org/details/stackexchange

[9] https://datarepo.eng.ucsd.edu/mcauley_group/gdrive/googlelocal/

Dataset	Metrics	w/o Temporal Encoding	with Temporal Encoding
Enron	Precision	0.6456 ± 0.0076	**0.6685 ± 0.0013**
	Recall	0.6212 ± 0.0056	**0.6567 ± 0.0013**
	F1	0.6345 ± 0.0073	**0.6675 ± 0.0014**
GDELT	Precision	0.1733 ± 0.0034	**0.2139 ± 0.0012**
	Recall	0.1843 ± 0.0076	**0.2159 ± 0.0016**
	F1	0.1834 ± 0.0123	**0.2234 ± 0.0023**
ICEWS1819	Precision	0.3212 ± 0.0056	**0.3743 ± 0.0043**
	Recall	0.3357 ± 0.0076	**0.3823 ± 0.0013**
	F1	0.3057 ± 0.0034	**0.3243 ± 0.0013**
Googlemap CT	Precision	0.6299 ± 0.0987	**0.6305 ± 0.0057**
	Recall	0.6845 ± 0.0003	**0.6905 ± 0.0002**
	F1	0.6256 ± 0.0034	**0.6354 ± 0.0012**
Stack elec	Precision	0.6176 ± 0.1345	**0.6565 ± 0.0056**
	Recall	0.6956 ± 0.1575	**0.7556 ± 0.0056**
	F1	0.6345 ± 0.1356	**0.6796 ± 0.0024**
Stack ubuntu	Precision	0.6257 ± 0.0045	**0.6824 ± 0.0054**
	Recall	0.7133 ± 0.0023	**0.7554 ± 0.1224**
	F1	0.6894 ± 0.0876	**0.7156 ± 0.0056**

Table 5: Ablation experiment on the temporal encoding for the edge classification performance of LKD4DyTAG

Metric	Task Type	Datasets	w/o Temporal Encoding	with Temporal Encoding
ROC-AUC	tr.	Enron	0.9112 ± 0.0563	**0.9887 ± 0.001**
		ICEWS1819	0.9224 ± 0.4568	**0.9950 ± 0.0014**
		Googlemap CT	0.8643 ± 0.0654	**0.9127 ± 0.0037**
		GDELT	0.9123 ± 0.1455	**0.9998 ± 0.0001**
	in.	Enron	0.8765 ± 0.0674	**0.9367 ± 0.0011**
		ICEWS1819	0.8854 ± 0.0545	**0.9543 ±. 0.0013**
		Googlemap CT	0.8223 ± 0.1676	**0.8857 ± 0.2327**
		GDELT	0.8675 ± 0.0664	**0.9577 ± 0.0012**
AP	tr.	Enron	0.9254 ± 0.0854	**0.9943 ± 0.0001**
		ICEWS1819	0.9346 ± 0.0056	**0.9970 ± 0.0012**
		Googlemap CT	0.8534 ± 0.0885	**0.9144 ± 0.0013**
		GDELT	0.9245 ± 0.0878	**0.9776 ± 0.0034**
	in.	Enron	0.8956 ± 0.0945	**0.9550 ± 0.0013**
		ICEWS1819	0.8984 ± 0.0044	**0.9634 ± 0.0010**
		Googlemap CT	0.8445 ± 0.0845	**0.8890 ± 0.1134**
		GDELT	0.8456 ± 0.0876	**0.9369 ± 0.0056**

Table 6: Ablation experiment on the temporal encoding for the future link prediction performance of LKD4DyTAG

classification task drops by the exclusion of temporal encoding. However, the performance degradation on the future link prediction task is more significant than the edge classification task. Without the presence of time information affects the proper prediction of future edges more compared to the edge classification where the semantic information from the knowledge distillation term might help LKD4DyTAG to perform comparatively better.