

Integrate Temporal Graph Learning into LLM-based Temporal Knowledge Graph Model

He Chang

Communication University of China
hechangcuc@cuc.edu.cn

Jie Wu

Communication University of China
wujie@cuc.edu.cn

Zhulin Tao*

Communication University of China
taozhulin@gmail.com

Yunshan Ma

Singapore Management University
ysma@smu.edu.sg

Xianglin Huang

Communication University of China
huangxl@cuc.edu.cn

Tat-Seng Chua

National University of Singapore
dcscts@nus.edu.sg

ABSTRACT

Temporal Knowledge Graph Forecasting (TKGF) aims to predict future events based on the observed events in history. Recently, Large Language Models (LLMs) have exhibited remarkable capabilities, generating significant research interest in their application for reasoning over temporal knowledge graphs (TKGs). Existing LLM-based methods have integrated retrieved historical facts or static graph representations into LLMs. Despite the notable performance of LLM-based methods, they are limited by the insufficient modeling of temporal patterns and ineffective cross-modal alignment between graph and language, hindering the ability of LLMs to fully grasp the temporal and structural information in TKGs. To tackle these issues, we propose a novel framework TGL-LLM to integrate temporal graph learning into LLM-based temporal knowledge graph model. Specifically, we introduce temporal graph learning to capture the temporal and relational patterns and obtain the historical graph embedding. Furthermore, we design a hybrid graph tokenization to sufficiently model the temporal patterns within LLMs. To achieve better alignment between graph and language, we employ a two-stage training paradigm to finetune LLMs on high-quality and diverse data, thereby resulting in better performance. Extensive experiments on three real-world datasets show that our approach outperforms a range of state-of-the-art (SOTA) methods.

KEYWORDS

Temporal Knowledge Graph Forecasting, Large Language Model, Hybrid Prompt

ACM Reference Format:

He Chang, Jie Wu, Zhulin Tao, Yunshan Ma, Xianglin Huang, and Tat-Seng Chua. 2025. Integrate Temporal Graph Learning into LLM-based Temporal Knowledge Graph Model. In *Proceedings of ACM Conference (Conference'17)*. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/nnnnnnnnnnnnnn>

*Corresponding Author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Conference'17, July 2017, Washington, DC, USA

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
<https://doi.org/10.1145/nnnnnnnnnnnnnn>

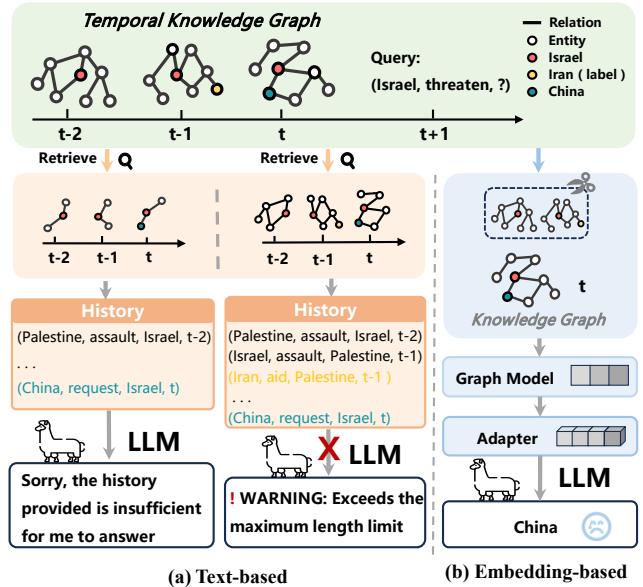


Figure 1: Illustration of the limitations in the current LLM-based TKGF methods. (a) Text-based: the cutting-edge retrieval-augmented solutions. (b) Embedding-based: incorporating the static knowledge graph into LLM.

1 INTRODUCTION

Temporal knowledge graph forecasting (TKGF) [4] aims to predict future facts by leveraging historical information, which plays a vital role in enhancing decision-making, resource allocation, and strategic planning across various domains [48]. Facts in temporal knowledge graphs (TKGs) represent real-world knowledge in a structured format, typically represented as a quadruple (s, r, o, t) , where s , r , o , and t denote the subject, relation, object, and timestamp, respectively. Previous research [13, 21, 22, 28] typically focuses on employing graph convolution networks (GCN) and recurrent neural networks (RNN) as the encoders to model the relational and temporal patterns hidden in historical facts. Despite their effectiveness, there are several limitations. First, these temporal graph neural network (TGNN) methods often fall short in being overly reliant on the training data, where the scope of knowledge is restricted when facing open-world data. Moreover, the errors induced by automatic information extraction systems bring substantial noise in TKGs [9], potentially resulting in suboptimal graph representation learning.

Finally, the prevailing existence of long-tail entities [32] with only a few observed links presents further challenges to the TGNN methods, which mostly follow the supervised learning paradigm and difficult to handle the long-tail issues.

Scrutinizing these limitations, the emerging large language models (LLMs) [15, 35, 39] demonstrate great potential and have been explored in some pioneering works [17, 23, 27, 44] on TKGF. These works can be broadly categorized into two branches: text-based and embedding-based. The text-based methods design a variety of retrieval strategies, such as temporal logical rules [23], semantic similarity [20], and certain heuristics [27], to transform the graph data into textual format for LLMs' reasoning. However, as illustrated in Figure 1, the varying retrieval strategies may inadvertently omit critical contextual details (*i.e.*, *(Iran, aid, Palestine, t-1)*), and the limited token length makes it impractical to incorporate all the historical contextual information into LLMs. To bridge the gap between structured graph data and natural language, researchers propose the embedding-based methods [46, 47] to encode the graphs into graph representations with a pre-trained graph model, followed by an adapter to align the graph representations with the language model. Compared to text-based methods, the aligned graph representation incorporates structural information into LLMs, providing valuable historical context for TKGF. However, existing embedding-based methods still face two unresolved limitations:

- **Insufficient modeling of temporal patterns.** Existing embedding-based methods [46, 47] primarily focus on reasoning over static knowledge graphs, which fails to fully exploit the temporal reasoning ability of LLM. For instance, as illustrated in Figure 1, due to the absence of temporal information, LLMs provide the incorrect answer '*China*' based on the historical facts at timestamp t . Therefore, the sufficient modeling of temporal patterns in LLMs remains an area that requires further exploration.
- **Ineffective cross-modal alignment between graph and language.** The inherent noise and long-tail issues present in TKGs result in suboptimal graph representations, which becomes increasingly challenging for the cross-modal alignment between graph and language. Furthermore, large-scale real-world knowledge exhibits a wide range of temporal and relational patterns, while only fine-tuning LLMs with similar patterns may degrade the performance in complex scenarios. Consequently, it is crucial to automatically identify high-quality and diverse graph data from the vast ocean of available datasets.

In order to advance the embedding-based approach by addressing the aforementioned limitations, we propose a novel LLM-based framework for TKGF, termed **TGL-LLM**, which integrates temporal graph learning into LLMs. To fully exploit the temporal reasoning ability of LLMs, we first introduce temporal graph learning to capture temporal and relational patterns. Rather than directly using the final representation in temporal graph learning, we extract recent historical graph embedding through GCNs to preserve temporal information of the evolving facts. More importantly, we design a hybrid graph tokenization to sufficiently model the temporal patterns in TKGs within LLMs. Specifically, we introduce a trainable temporal graph adapter to project the graph embedding into the language token space and concatenate the graph tokens in temporal

order as the description of the entity, thereby allowing LLMs to further explore complex temporal patterns. Additionally, to achieve effective cross-modal alignment between graphs and language, we employ a two-stage fine-tuning paradigm to fine-tune LLMs on high-quality and diverse data, resulting in the improvement of performance. To identify the high-quality and diverse data, we first introduce the influence function to evaluate the influence of removing each graph data point on the fine-tuning of LLM. A higher influence score indicates that the graph data is of higher quality for the LLM to effectively capture the temporal and relational patterns in graph embedding. Then we apply diversity sampling to ensure a variety of temporal and relation patterns exposed to LLMs. Extensive experiments demonstrate that our method outperforms both traditional approaches and state-of-the-art (SOTA) methods. In summary, our contributions can be outlined as follows:

- We advance embedding-based LLM solutions for TKGF by addressing two limitations, *i.e.*, insufficient modeling of temporal patterns and ineffective cross-modal alignment between graph and language.
- We propose a novel framework TGL-LLM, where the two key designs, hybrid graph tokenization and a two-stage training paradigm, pertinently alleviating the two limitations respectively.
- We conduct extensive experiments on three real-world datasets, and the results demonstrate the effectiveness of TGL-LLM.

2 RELATED WORK

We provide a brief literature review of methods for TKGF, including both conventional Non-LLM methods and LLM-based methods.

2.1 Temporal Knowledge Graph Forecasting

Temporal Knowledge Graph Forecasting is a rapidly evolving field that focuses on predicting the occurrence of future facts based on historical data. Traditional methods can generally be grouped into two categories: rule-based methods and GNN-based methods. Rule-based methods [3, 26] rely on predefined or learned temporal rules that capture common patterns or logical relationships in historical event data to guide the forecasting process. For instance, utilizes temporal random walks in TKGs to learn temporal logical rules without the need for trainable parameters, thereby enhancing the explainability of the reasoning process. TILP [42] proposes a differentiable framework for temporal logical rule learning by introducing a constrained walk mechanism and temporal operators, enabling symbolic learning in temporal knowledge graphs. TEILP [43] extends the framework of TILP to address the challenge of time prediction by constructing a temporal knowledge graph and developing a differentiable random walk approach for time prediction. However, these methods depend heavily on manually defined rules, which involve a significant time investment when applied to large-scale TKG datasets. On the other hand, TGNN-based methods [13, 22] employ graph neural networks to aggregate relational information from entity neighborhoods. Early works such as RGCN [37] and ConvTransE [38] treat event forecasting as a static knowledge graph completion task, updating entity and relation embeddings through GNNs. To capture the temporal evolution of entities and relations, methods like RENET and REGCN

incorporate recurrent neural networks to derive temporal representations. Addressing the absence of specific and informative facts, SeCoGD [28] uses textual topic modeling to disentangle subgraphs, while LoGo [29] applies text clustering to construct complex facts for forecasting using both local and global contexts. To incorporate textual information into representation learning, Glean [6] and CMF [8] fuse textual embeddings into graph edges. However, due to the limitations inherent in GNNs, these methods suffer from the inherent noise and long-tail entities in TKGs.

2.2 Large Language Models for TKGF

LLMs have recently been explored for event forecasting, leveraging their ability to process and understand contextual information. Several studies have highlighted the forecasting potential of in-context learning within large language models. Following the retrieval-augmented generation paradigm, several recent works employ different sampling strategies, flattening the structural graph data into textual descriptions to predict future facts. For instance, GPT-NeoX-ICL [17] retrieves relevant facts from the temporal knowledge graph and constructs prompt with a list of historical facts in quadruplet format. By utilizing an in-context learning framework, LLMs can comprehend the task of TKGF and the reason for the given historical information. Similar to TLogic [26], GENTKG [23] utilizes learned temporal logical rules to construct the relevant event histories. CoH [27] retrieves the results based on the well-designed rules to guide LLMs in making the final prediction. However, the limitations of the natural language format result in insufficient contextual information for effective reasoning on the temporal knowledge graph. In contrast to transforming graph data into textual descriptions, recent studies treat pre-trained graph embeddings as a new modality for LLMs, aligning them with language space. GraphTranslator [46] generates alignment data to aid the translator module in converting graph embeddings to token embeddings. TEA-LLM [41] aligns GNN representation with LLM token embeddings by a linear projector to make LLMs understand the structural information in various tasks. KoPA [47] introduces an adapter to obtain aligned structural embeddings, positioning them as a prefix to the input prompt. However, existing alignment methods assume pre-trained graph embeddings as ideal representations of structural information, overlooking the suboptimal representations caused by noise and long-tail entities in the training data. More importantly, all of the above embedding-based methods focus on static knowledge graph reasoning, neglecting the temporal dimension essential for event forecasting.

3 PRELIMINARY

We first present the classical formulation of TKGF. Then we present its adaption in the scenario of LLM-based solutions.

Classical TKGF Formulation. Given a set of entities \mathcal{E} and relations \mathcal{R} , we define a fact as a quadruple (s, r, o, t) , where $s \in \mathcal{E}$, $r \in \mathcal{R}$, and $o \in \mathcal{E}$ represent the subject, relation, object, and timestamp respectively. All facts occurring at the same timestamp t form a knowledge graph denoted as $\mathcal{G}_t = \{(s_n, r_n, o_n, t)\}_{n=1}^N$, where (s_n, r_n, o_n, t) is the n -th fact and $N = |\mathcal{G}_t|$ is the number of facts at timestamp t . Consequently, a TKG \mathcal{G} can be formalized as $\mathcal{G} = \{\mathcal{G}_0, \mathcal{G}_1, \dots, \mathcal{G}_t, \dots\}$. Given the historical knowledge

graphs prior to t , denoted as $\mathcal{G}_{\leq t} = \{\mathcal{G}_0, \mathcal{G}_1, \dots, \mathcal{G}_t\}$, and a query of the form $(s, r, ?, t)$, temporal knowledge graph forecasting aims to predict the missing object.

LLM-based TKGF Formulation. Compared to traditional methods that rank all entities within the graph, LLM-based methods face practical limitations due to the large size of the entity set and the constrained token length of LLMs, making it infeasible to input all entities into the model simultaneously. Therefore, we simplify the temporal knowledge graph forecasting settings as a Multi-Choice Questions(MCQ) problem, employing negative sampling strategies to construct a candidate set $\hat{\mathcal{O}} = \{\hat{o}_1, \hat{o}_2, \dots, \hat{o}_k, \dots\}$ for each query, where $K = |\hat{\mathcal{O}}|$ denotes the number of candidate entities. The reasons are as follows: First, it facilitates easier and more consistent quantitative evaluation for LLM-based methods. Second, the MCQ setting aligns closely with real-world scenarios since there are typically a limited number of highly probable candidate entities rather than an unrestricted range of possibilities. Third, even in a generative setting, the generated answers are more likely to be derived from the provided context rather than from independent reasoning based on historical information [12]. For more details, refer to Appendix A.3. Consequently, we argue that the MCQ setting is a reasonable and acceptable choice for this task. Notably, the MCQ setting differs from the typical Question and Answer(QA) task, which emphasizes factual retrieval and reasoning based on the current context, typically lacking the capacity to infer dynamic changes over time. Given the historical knowledge graphs prior to t , denoted as $\mathcal{G}_{\leq t} = \{\mathcal{G}_0, \mathcal{G}_1, \dots, \mathcal{G}_t\}$, a query of the form $(s, r, ?, t)$, and a candidate set $\hat{\mathcal{O}}$, temporal knowledge graph forecasting aims to predict the missing object from the candidate set.

4 METHOD

We present a novel framework TGL-LLM to integrate temporal graph learning into LLM-based temporal knowledge graph model, as illustrated in Figure 2. At the core of TGL-LLM are two key innovations: 1) a novel hybrid graph tokenization, to sufficiently model the temporal patterns, and 2) a two-stage training paradigm, to achieve effective cross-modal alignment between graph and language. In this section, we first present temporal graph learning, where we capture the temporal and relational patterns and obtain the temporal graph embedding. Then, we introduce hybrid graph tokenization to enable LLMs to capture the complex temporal patterns in TKGs. Finally, we introduce model training, which employs a data pruning method to identify high-quality graph data and diverse graph data, followed by a two-stage training paradigm.

4.1 Temporal Graph Learning

The representation learning of temporal graph has been continuously under development. To sufficiently model the temporal patterns, we introduce temporal graph learning to obtain the historical graph embedding. To efficiently leverage the temporal reasoning ability of LLMs, we utilize the historical graph embedding instead of the final representation to enable LLMs to preserve temporal information of the evolving facts. Meanwhile, in the training phase, we train the graph model with an RNN and a decoder to capture the temporal patterns within the recent historical graph representation.

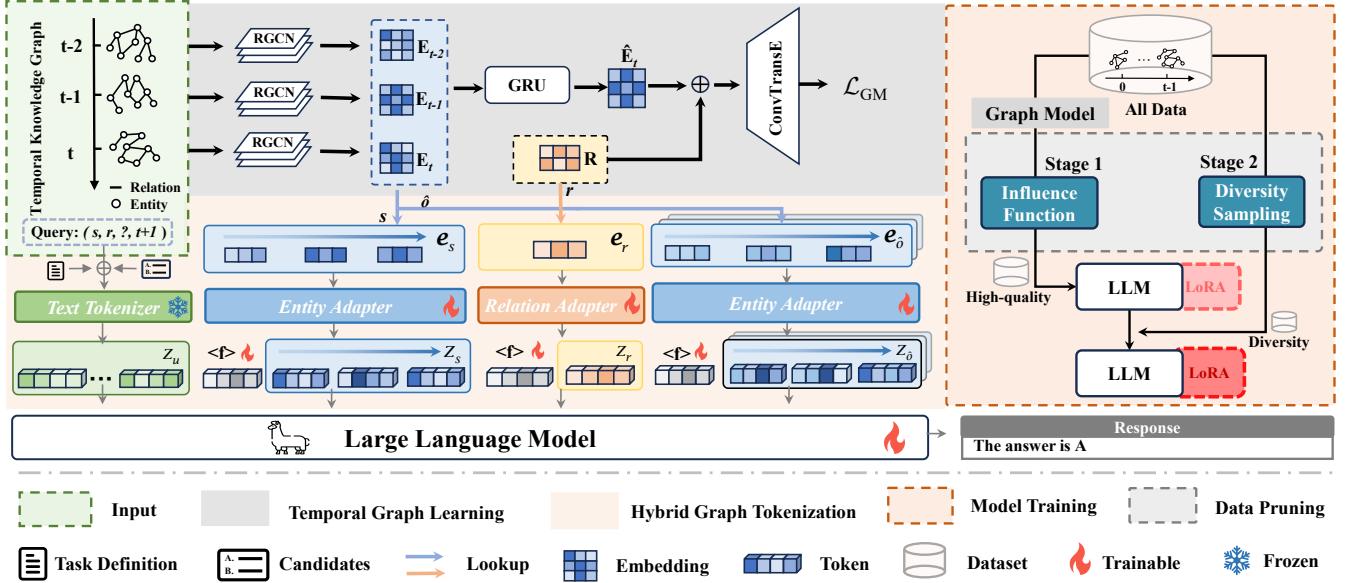


Figure 2: The overall framework of TGL-LLM consists of two key novel modules: 1) hybrid graph tokenization that sufficiently models the temporal patterns in LLMs (left), and 2) model training that follows a two-stage training paradigm (right).

Following the previous work REGCN [22], we employ RGCN as the graph model to capture the relational patterns among all facts at the same timestamp. Based on the \mathcal{G}_t at timestamp t , for each layer l of the graph propagation, the message $e_o^l \in \mathbb{R}^d$ obtained by each object o is defined as follows:

$$e_o^l = f\left(\frac{1}{|\mathcal{G}_t|} \sum_{(s,r,o) \in \mathcal{G}_t} \mathbf{W}_1^l (e_s^{l-1} + \mathbf{r}) + \mathbf{W}_2^l e_o^{l-1}\right), \quad (1)$$

where $\mathbf{W}_1^l, \mathbf{W}_2^l \in \mathbb{R}^d$ are the parameters of convolutional kernel in the $l-1$ -th layer, d is the dimension of the representation, $f(\cdot)$ is the activation function ReLU, and e_s^{l-1}, e_o^{l-1} , and \mathbf{r} denote the l -th embeddings of entities s , o and relation r . After executing multi-layer message passing, we aggregate the messages acquired from multi-layer propagation to derive the entity representation at timestamp t , formally defined as:

$$e_o = \sum_{l=0}^L e_o^l, \quad (2)$$

where L is the number of graph layers. Through the message-passing mechanism, the embedding for all entities at timestamp t and all relations are denoted as $\mathbf{E}_t \in \mathbb{R}^{|\mathcal{E}|*d}$ and $\mathbf{R} \in \mathbb{R}^{|\mathcal{R}|*d}$, respectively. It is noted that we adopt the recent T steps embedding of entities $\mathbf{E} = \{\mathbf{E}_{t-T}, \dots, \mathbf{E}_t\}$ and relations \mathbf{R} as our graph representation.

In order to capture the temporal evolving patterns in TKG, in the training phase, a GRU is employed to capture the temporal patterns, defined as follows:

$$\hat{\mathbf{E}}_t = \text{GRU}(\mathbf{E}_t, \mathbf{E}_{t-1}). \quad (3)$$

Subsequently, we utilize ConvTransE [38] as the decoder and apply the cross-entropy loss to optimize the model. The loss is defined as:

$$\mathcal{L}_{GM} = \sum_{(s,r) \in \mathcal{G}_{t+1}} y_{(s,r,t+1)} \log(p(\mathcal{E}|s, r, \mathcal{G}_{\leq t})), \quad (4)$$

where y denotes the ground truth. In this way, the pre-trained graph model can extract temporal information from TKG, thereby enhancing the capability of LLMs for more profound temporal reasoning.

4.2 Hybrid Graph Tokenization

4.2.1 Temporal Graph Adapter. To bridge the gap between graph and language, directly incorporating graph embedding into the prompt can not be interpreted by LLMs. Therefore, following previous embedding-based methods [41, 46], we first map the graph embedding into the language token space, allowing LLMs to understand the temporal and structural information distilled by the pre-trained graph model. To achieve this, we introduce a trainable Temporal Graph Adapter (TGA): Entity Adapter(EA) and Relation Adapter(RA).

Given the most recent T -step graph embedding of entity E and relation representation R , we can obtain the temporal graph representation for the subject $e_s = \{e_{t-T}^s, \dots, e_t^s\}$ and the candidate $e_{\hat{o}_k} = \{e_{t-T}^{\hat{o}_k}, \dots, e_t^{\hat{o}_k}\}$, along with relation representation r . Then we project the temporal graph representation into the language token space with the Entity Adapter and Relation Adapter:

$$z_{t-T}^s = EA(e_{t-T}^s; \Theta_{ea}) \quad (5)$$

$$z_r = RA(r; \Theta_{ra}) \quad (6)$$

$$z_{t-T}^{\hat{o}_k} = EA(e_{t-T}^{\hat{o}_k}; \Theta_{ea}) \quad (7)$$

where z_{t-T}^s and $z_{t-T}^{\hat{o}_k}$ denote the temporal graph tokens of subject and candidate at timestamp $t-T$, respectively; z_r is the graph token of relation; EA (\cdot) and RA (\cdot) denote the entity adapter and relation adapter; Θ_{ea} and Θ_{ra} are the parameters of entity adapter and relation adapter. Here, we employ a two-layer perception to serve as the adapter of entity and relation.

4.2.2 Hybrid Prompt Design. To help LLMs understand and capture complex temporal patterns, we propose a hybrid prompting method. Specifically, the designed hybrid prompt consists of three primary components:

Instruction. We first provide a clear task definition of TKGF. Then we give the query fact and the candidate set in natural language format. The instruction, which consists of task definition, query, and the candidate set, is then transformed into tokens represented as z_u . Figure 10 in Appendix A.5 illustrates the instructions we utilized.

Query with Graph Tokens. We concatenate the temporal graph tokens in temporal order as the description of the subject, denoted as z_s . Furthermore, we introduce a learned feature token $\langle f \rangle$ as the soft prompt to distinguish between textual tokens and various graph tokens, denoted as:

$$z_q = [\underbrace{z_f}_{\langle f \rangle}, \underbrace{z_{t-T}^s, \dots, z_t^s}_{\text{subject}}], \quad (8)$$

where z_q is the query subject with graph token and z_f denotes the learned feature token. Subsequently, The graph token of the query relation z_r is subsequently concatenated with z_f to form the description of the relation.

Candidate Set with Graph Tokens. Similar to the query subject, we concatenate the feature token z_f and temporal graph tokens z_{δ_k} to construct the description of candidates z_δ .

Notably, the hybrid prompt only utilizes temporal graph tokens as the description of the subject and the candidate, which facilitates the integration of temporal graph representation sourced from temporal graph learning. This design addresses the limitations of prompts that rely either on textual historical facts or static graph representation, thereby generating more accurate predictions.

4.3 Model Training

4.3.1 Data Pruning. To achieve more effective cross-modal alignment between the graph and the language, the key is the selection of high-quality, diverse data. Suboptimal graph representations caused by the noise and long-tail issue in TKGs increase the difficulty of alignment, while redundant samples with similar patterns restrict the capacity of LLMs to learn a wide range of domain knowledge. Therefore, first, we introduce the influence function [14] as the data pruning method to evaluate the influence of each training sample in TKGs.

Considering the heavy computational resources and time costs of LLMs, following previous studies [24], we utilize the pre-trained graph model as the surrogate model to evaluate the influence score. Given the parameters of the pre-trained graph model $\theta_G \in \Theta_G$, the train dataset \mathcal{D} , and a training sample d upweighted by a small ϵ , the empirical risk minimizer is given by:

$$\hat{\theta}_G = \underset{\theta_G \in \Theta_G}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^m \mathcal{L}(d_i, \theta_G) + \epsilon \mathcal{L}(d, \theta_G), \quad (9)$$

where $m = |\mathcal{D}|$ is the number of train dataset and $d_i \in \mathcal{D}$ is the training sample. Notably, removing a sample d is the same as assigning $-\frac{1}{m}$ to ϵ . Based on the classic result [25] and the chain rule, the influence of removing a sample d on the loss of an arbitrary

sample d' is given by:

$$\begin{aligned} I_{\text{remove,loss}}(d, d') &= \left. \frac{d\mathcal{L}(d', \hat{\theta}_G)}{d\epsilon} \right|_{\epsilon=0} \\ &= \nabla_{\theta_G} \mathcal{L}(d', \hat{\theta}_G)^T \left. \frac{d\hat{\theta}_G}{d\epsilon} \right|_{\epsilon=0} \\ &= \frac{1}{m} \nabla_{\theta_G} \mathcal{L}(d', \hat{\theta}_G)^T H_{\hat{\theta}_G}^{-1} \nabla_{\theta_G} \mathcal{L}(d, \hat{\theta}_G), \end{aligned} \quad (10)$$

where $H_{\hat{\theta}_G}^{-1} = \frac{1}{m} \sum_{i=1}^m \nabla_{\theta_G}^2 \mathcal{L}(d_i, \theta_G)$ is the Hessian and positive definite by assumption. Furthermore, we utilized the Hessian-vector products (HVP) [2] and symmetric property to obtain influence scores for each $d_i \in \mathcal{D}$ in train data:

$$I_{\text{remove,loss}}(d, \mathcal{D}) = \frac{1}{m} \nabla_{\theta_G} \mathcal{L}(d, \hat{\theta}_G)^T H_{\hat{\theta}_G}^{-1} \left(\sum_{i=1}^m \frac{1}{m} \nabla_{\theta_G} \mathcal{L}(d_i, \hat{\theta}_G) \right). \quad (11)$$

Benefiting from the auto-grad system, we only need to specify the loss to compute the influence score. Subsequently, we apply stratified sampling [49] to obtain a high-quality subset \mathcal{D}_h , ensuring a reduction in similar training samples.

In addition to quality, the effectiveness of LLM-based methods fundamentally depends on the diversity of training data [31], which enables LLMs to capture various temporal and relational patterns in TKGs effectively. In other words, diverse data is essential for instilling the general reasoning capabilities of LLMs for TKGF. Therefore, we selected a subset \mathcal{D}_p from the original dataset as the implementation of the high-quality subset, termed the diversity subset. Here, we employ random sampling for simplicity and efficiency to obtain the diversity subset. It is important to note that, to avoid impacting the representation learning on high-quality data, the size of the diversity subset is significantly smaller than that of the high-quality subset.

4.3.2 Prompt Tuning. Based on the aforementioned subset of train data, we train the model following a two-stage training paradigm. Compared to the random sample data, the two-stage training phase on the high-quality subset and diversity subset of train data better aligns the temporal and global graph representations with the token space and enhances the robustness of LLMs in TKGF.

Stage 1. Given the high-quality subset \mathcal{D}_h with the designed hybrid prompt, we adopt LoRA for the finetuning of LLM. The optimizing objective of LoRA can be formulated as follows:

$$\max_{\Theta} \sum_{d \in \mathcal{D}_h} \log(p(v_o | z_u, z_q, z_\delta; \Phi_0, \Phi(\Theta_{LoRA}), \Theta_{ea}, \Theta_{ra}, \Theta_f)), \quad (12)$$

where Φ_0 and $\Phi(\Theta_{LoRA})$ represent the frozen parameters of LLMs and the learnable parameters of LoRA, respectively; $d \in \mathcal{D}_h$ denote the training sample in the high-quality subset.

Stage 2. Based on the fine-tuned LLM on the high-quality subset, we aim to improve the generality reasoning ability of LLM. To this end, analogous to the training phase of stage 1, we continue to fine-tune the LLM on the diversity subset \mathcal{D}_p with the above parameters $\Phi(\Theta_{LoRA}), \Theta_{ea}, \Theta_{ra}, \Theta_f$.

5 EXPERIMENTS

We conduct experiments on three real-world datasets and compare the results of our method with several baselines, including conventional Non-LLM methods and LLM-based approaches. Additionally, we perform ablation studies to demonstrate the significant improvements resulting from the integration of temporal and global graph representations, as well as data pruning. Furthermore, we examine the effects of historical length, and the influence on long-tail entities, and present case studies to clearly illustrate our advantages over the baselines. To validate the superiority of our framework, we aim to answer the following research questions:

- **RQ1:** How does TGL-LLM perform compared to conventional TKGF models and LLM-based methods?
- **RQ2:** How does the integration of temporal and global graph representations affect performance? Are our data pruning methods superior and beneficial?
- **RQ3:** How does TGL-LLM perform under different settings, including varying historical lengths and long-tail entities?

5.1 Experiment Settings

5.1.1 Datasets. We utilize the recently proposed large-scale POLECAT dataset and crop three subsets to conduct experiments, *i.e.*, POLECAT-IR, POLECAT-IS, and POLECAT-EG. POLECAT is a temporal knowledge graph dataset in socio-political domain. Even though there are several widely-used TKG datasets, such as ICEWS [33], GDELT [19], WIKI [16], and YAGO [30], they were constructed quite long time ago, consequently, the knowledge in these datasets may have unintentionally leaked into the pre-training data of LLMs [11]. The training data leakage raise the concerns of unfair or unfaithful evaluation of the TKGF methods, because we cannot tell the performance improvement comes from better forecasting capability of the designed model or just because the LLMs already know the future facts. Although prior efforts have tried to mitigate leakage by anonymizing timestamps [18], LLMs can still recognize the semantics of entity names and event types, linking them to internal knowledge. To ultimately eliminate the data leakage concern, we adopted the POLECAT dataset [36], which was first released in April 2023 as an upgraded version of ICEWS. The biggest merit is that this is a live dataset which keeps updating new knowledge every week. And it introduces an upgraded schema, utilizes an enhanced information extraction pipeline with less noise, and provides a large-scale dataset enriched with various metadata, including location, context, *etc*. We use Llama2 as our default LLM backbone, and the training data cut-off date of Llama2 is July 2023. Regarding this cut-off date, we split the dataset along the timeline into training (January 2018 to May 2023), validation (June to October 2023), and test (November 2023 to April 2024), assuring all the test data and a part of the validation data are after the LLM cut-off date. Given the large scale of the original POLECAT dataset, we follow the previous approaches [7, 28] and crop three sub-datasets according to the country labels of each quadruple, *i.e.*, Iran, Israel, and Egypt, corresponding to POLECAT-IR, POLECAT-IS, and POLECAT-EG, respectively. Table 1 presents the statistics of the datasets, where $|\mathcal{R}|$ denotes the number of relations and $|\mathcal{E}|$ denotes the total number of entities, showing that they are large-scale in terms of the entity set and the total number of quadruples. And we also present

the detailed pre-processing steps in Appendix A.1 and extensive statistical analysis in Appendix A.2.

Table 1: Dataset statistics, including the number of facts, entities $|\mathcal{E}|$, and unique relations $|\mathcal{R}|$ for training, validation, and testing sets in each of the dataset.

Set	POLECAT-IR			POLECAT-IS			POLECAT-EG		
	Facts	$ \mathcal{E} $	$ \mathcal{R} $	Facts	$ \mathcal{E} $	$ \mathcal{R} $	Facts	$ \mathcal{E} $	$ \mathcal{R} $
Train	616,880	33,920	80	484,630	31,898	80	256,523	25,422	80
Valid	13,737	2,593	79	29,931	4,797	77	6,898	1,399	72
Test	11,053	2,174	72	56,750	6,364	77	3,812	1,096	69
Total	641,670	35,138	80	571,311	36,336	80	267,233	26,220	80

5.1.2 Evaluation Protocols. For each query event, we construct the candidate set by randomly selecting 4, 6, or 10 entities from the most recent three days, including one positive entity. We employ *Accuracy* as the primary metric to evaluate the ability of models to predict the positive item from this candidate set. A higher Acc@4, Acc@6, or Acc@10 indicates a superior capability for accurate temporal knowledge graph forecasting. To ensure a fair comparison across all baselines, we utilize the same number of candidates and evaluation methods.

5.1.3 Baselines. We compare our method with three types of baseline methods, *i.e.*, Non-LLM methods, and LLM-based methods including zero-shot and fine-tune approaches, which are widely used in the TKG formulation.

Non-LLM Methods convert TKG into embeddings, including DistMult [45], ConvTransE [38], RGCN [37], RENET [13], REGCN [22], and HisMatch [21].

Zero-Shot Methods use TKG information in text form in the LLMs prompt. We evaluate results using GPT-3.5-turbo [34] and GPT-4-mini [1].

Fine-tune Methods use TKG to train LLMs, including pure text form of TKG data (*e.g.*, GenTKG [23], CoH [27]), and direct use of graph data as input (*e.g.*, KoPA [47]).

5.1.4 Implementation Details. To ensure fairness of evaluation, all conventional Non-LLM methods employ the Adam optimizer, with the learning rate selected from {1e-2, 1e-3, 1e-4} and L2 regularization from {1e-4, 1e-5, 1e-6}. The embedding size is set to 200, and the number of propagation layers is set to 2. For all LLM-based methods, we utilize Llama2-7B-chat [40] as the backbone, and each experiment is trained on 100,000 training samples randomly selected from the training dataset, with a batch size of 128. Moreover, the sizes of the high-quality subset and diversity subset we utilized are set to 100,000 and 10,000, respectively. For the embedding-based method, the graph embedding size is set to 200. Notably, based on the split of the dataset, we adopt the real timestamp in all the LLM-based method instead of the index. All experiments are conducted on a Linux server with a GPU (NVIDIA RTX A40, Memory 40G). The training times of TGL-LLM are approximately 10 hours, while this may vary *w.r.t.* the size of the dataset and candidate set.

5.2 Performance Comparison (RQ1)

We compare the overall performance of our proposed method, TGL-LLM, with several state-of-the-art methods for temporal knowledge

Table 2: The results of TGL-LLM compared with Non-LLM methods and LLM-based methods. Bold and underlined indicate the best and the second-best performance, respectively.

Model Type	Model	Input	POLECAT-IR			POLECAT-IS			POLECAT-EG		
			Acc@4	Acc@6	Acc@10	Acc@4	Acc@6	Acc@10	Acc@4	Acc@6	Acc@10
Non-LLM	DistMult [45]	Graph	0.4901	0.4110	0.3291	0.5364	0.4718	0.4060	0.4100	0.3339	0.2442
	ConvTransE [38]		0.5169	0.4371	0.3563	0.5460	0.4846	0.4162	0.4562	0.3757	0.2736
	RGCN [37]		0.5125	0.4372	0.3661	0.5355	0.4731	0.4071	0.4318	0.3499	0.2558
	RENET [13]		0.5099	0.4274	0.3448	0.5399	0.4773	0.4077	0.4368	0.3539	0.2681
	REGCN [22]		0.4979	0.4264	0.3632	0.5353	0.4811	0.4227	0.4407	0.3568	0.2602
	HisMatch [21]		0.5291	0.4562	0.3841	0.5478	0.4887	0.4260	0.4344	0.3529	0.2647
Zero-shot	GPT-3.5-turbo [34]	Text	0.4447	0.3396	0.2409	0.4627	0.3624	0.2480	0.3893	0.2917	0.1873
	GPT-4o-mini [1]		0.4907	0.3972	0.3090	0.4874	0.3960	0.3107	0.4082	0.3130	0.2093
Fine-tune	GenTKG [23]	Text	0.5749	0.4806	0.3710	0.5863	0.4877	0.4129	0.4869	0.3911	0.2920
	CoH [27]		0.6037	0.5074	0.4060	0.6049	0.5158	0.4213	0.5092	0.4174	0.3153
	KoPA [47]	Graph	<u>0.6083</u>	<u>0.5083</u>	<u>0.4358</u>	<u>0.6110</u>	<u>0.5261</u>	<u>0.4336</u>	<u>0.5315</u>	<u>0.4397</u>	<u>0.3290</u>
	TGL-LLM(ours)		0.8514	0.8269	0.7407	0.8779	0.8459	0.7753	0.8109	0.7697	0.6810

graph forecasting. Table 2 presents the overall performance of our method alongside all the baselines. From these results, we can draw the following observations. First, as we expected, our method outperforms all the baselines of both Non-LLM and LLM-based TKGF methods by a large margin across three datasets. We further adopt ANOVA significance test [10] over the performances of all the baselines. All of the evaluated p-values are below 0.01, demonstrating the superior effectiveness of our method. Second, among the three types of baselines, LLM-based TKGF methods on the fine-tune setting generally perform better than the closed commercial LLMs and the Non-LLM method, indicating that supervised prompt tuning is crucial for enhancing the ability of LLMs to understand the task of TKGF. Furthermore, the embedding-based methods achieve better performance than the text-based methods across three datasets. We attribute this to the sufficient structural information provided by the graph embedding. Third, analyzing the results of Non-LLM methods, we observe that HisMatch performs best on the POLECAT-IR and POLECAT-IS datasets. However, for the POLECAT-EG dataset, the methods based on TKGs underperform compared to ConvTransE. The possible reasons are from two aspects: 1) insufficient modeling of temporal patterns, as evidenced by the poorer performance of REGCN compared to RGCN on POLECAT-IR and POLECAT-IS, and 2) smaller size of the POLECAT-EG dataset, particularly in the test set, which offers less useful temporal information.

5.3 Ablation Study (RQ2)

5.3.1 The impact of hybrid graph tokenization. To shed light on how insufficient modeling of temporal patterns affects the performance of LLMs, we compare four model variants: **Raw** (*i.e.*, without graph representation), **Static** (*i.e.*, with static graph embedding), **GRU** (*i.e.*, with the output of GRU), and **ConvTransE** (*i.e.*, with the output of ConvTransE). The results are presented in Table 3. Clearly,

Table 3: Performance comparison w.r.t. adopting different graph representations as graph tokens.

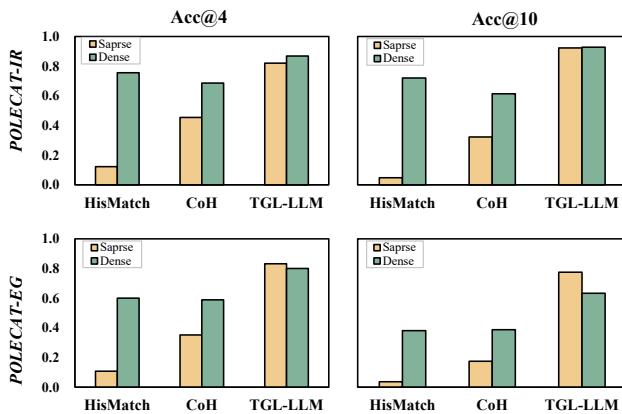
Model	POLECAT-IR		POLECAT-IS		POLECAT-EG	
	Acc@4	Acc@10	Acc@4	Acc@10	Acc@4	Acc@10
Raw	0.5688	0.3510	0.5898	0.3950	0.4969	0.2993
Static	0.6165	0.4084	0.6106	0.4329	0.5278	0.3308
GRU	0.7098	0.6404	0.8528	0.7426	0.6674	0.5168
ConvTransE	0.8080	0.6230	0.7841	0.6033	0.5039	0.3119
TGL-LLM	0.8514	0.7407	0.8779	0.7753	0.8109	0.6810

TGL-LLM achieves the best performance by utilizing recent historical graph embeddings as graph tokens, demonstrating the sufficient modeling of temporal patterns in our proposed method. Next, we observe that **Static** underperforms compared to the graph representation with temporal information. This indicates that there are significant temporal patterns within our datasets, and effectively capturing these temporal patterns is crucial for LLMs to make accurate forecasts. Additionally, By comparing the performance of **GRU**, **CovTransE**, and **TGL-LLM**, we find that deeper graph representations lead to a decline in the performance of LLMs. This decline may be attributed to the fact that the deeper graph representations in the pre-trained graph models are challenging for the LLMs to comprehend and struggle to align effectively with the token embedding space.

5.3.2 The impact of two-stage training paradigm. We conduct experiments to evaluate the impact of the two-stage training paradigm for TKGF. Here, we showcase three variants to analyze the effect: **Random** (*i.e.*, fine-tune on the randomly sampled training subset), **w/o-IF** (*i.e.*, replace the influence function with the random sampling), and **w/o-DS** (*i.e.*, without diversity sampling). As illustrated in Table 4, it is clear that the two-stage training paradigm outperforms the other three variants, which contributes to the effective cross-modal alignment between graph and language. Moreover,

Table 4: Performance comparison w.r.t. applying different sampling strategy.

Model	POLECAT-IR		POLECAT-IS		POLECAT-EG	
	Acc@4	Acc@10	Acc@4	Acc@10	Acc@4	Acc@10
Random	0.8467	0.6745	0.8631	0.7381	0.7823	0.6511
w/o-IF	0.8369	0.6871	0.8622	0.7388	0.7920	0.6605
w/o-DS	0.8354	0.6864	0.8765	0.7731	0.7673	0.6220
TGL-LLM	0.8514	0.7407	0.8779	0.7753	0.8109	0.6810

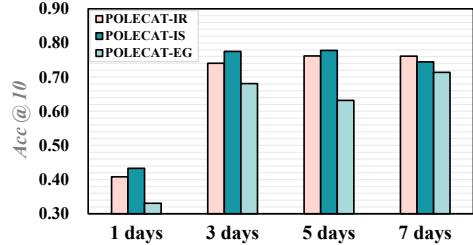
**Figure 3: The performance comparison across two datasets on the long-tail entities.**

we find that the absence of either the high-quality subset or the diversity subset can lead to worse performance compared to random sampling, demonstrating that high-quality and diverse data are equally important for LLMs. It is worth pointing out that the performance on the POLECAT-IS dataset remains strong without diversity sampling. This may be because the temporal and relational patterns in the testing set are more similar to those in the training set for POLECAT-IS.

5.4 Model Study (RQ3)

We conduct more experiments to study some key properties of our model.

5.4.1 Long-tail Issue. Several previous studies [5, 28] have identified that traditional methods still struggle with the prevalent issue of long-tail entities. To evaluate the performance of TGL-LLM on long-tail entities in TKGs, we re-split the test dataset into two subsets: dense and sparse. The dense subset includes queries where the query object appears over 100 times in the training dataset, while the remaining facts form the sparse subset. For instance, in the POLECAT-IR dataset, the dense subset has 353 objects and 7,099 facts, while the sparse subset contains 1,023 objects and 3,954 facts. The results across two datasets are shown in Figure 3. Benefiting from the open-world knowledge in LLMs, the LLM-based methods achieve better performance than Non-LLM methods. In contrast, with sufficient information from the training dataset, Non-LLM methods remain competitive. Notably, under no matter what

**Figure 4: Performance comparison of our method when using varying history length.****Table 5: The comparison of training costs w.r.t. applying different LLM-based methods. The statistics are based on NVIDIA RTX A40.**

Model	Avg.Token	GPU (GiB)	Time (h)	Acc@10
CoH	466	36.78	21.53	0.406
KoPA	223	22.98	10.23	0.4358
TGL-LLM	245	23.08	11.26	0.7407

specific settings, TGL-LLM demonstrates the strongest overall performance and has much smaller gaps, indicating that LLM can still effectively tackle the task of TKGF on long-tail entities with extensive knowledge and effective alignment between graph and language.

5.4.2 The impact of historical length. The scope of history plays a crucial role in effective forecasting. Driven by this hypothesis, we vary the historical length of graph embedding, and the result is shown in Figure 4, which illustrates that 5 is the best for POLECAT-IR and POLECAT-IS and 7 is the best for POLECAT-EG. Furthermore, we find that as the historical length increases, the performance starts to converge, which indicates that the expansion of history length comes along with more noisy information.

5.4.3 Training Costs. In order to evaluate the training costs of various LLM-based methods, we conduct experiments on POLECAT-IR, as shown in Table 5. **Avg.Token**, **GPU (GiB)**, and **Time(h)** represent the average token length of various LLM-based methods, the GPU memory usage, and the training time required for fine-tuning the LLMs, respectively. Compared to the text-based method, the training cost of the embedding-based method significantly decreases due to the integration of graph embedding. Although KoPA has the lowest training cost, TGL-LLM incurs a slightly higher training cost while endowing LLMs with enhanced reasoning capabilities over TKGs.

6 CONCLUSION AND FUTURE WORK

In this paper, we integrated temporal graph learning into LLM-based temporal knowledge graph model. We introduced a new framework, TGL-LLM, which consists of two critical components: hybrid graph tokenization and a two-stage training paradigm to address two major limitations of existing embedding-based methods: the insufficient modeling of temporal patterns and the ineffective cross-modal alignment between graph and language respectively.

Empirical results demonstrated that TGL-LLM significantly outperforms all baselines in TKGF, underscoring its effectiveness and superior performance.

Moving forward, we will shift our focus to: 1) developing more effective graph adapters instead of relying solely on a simple multi-layer perceptron (MLP); 2) endowing LLMs with capabilities for generative forecasting; 3) not limiting to global graph representations, but rather introducing more local graph representations to obtain more accurate contextual information.; and 4) improving the performance of LLMs across a broader range of subtasks, including relation prediction, time interval prediction, long-horizon prediction, and event distribution prediction, among others. And we hope that TGL-LLM can inspire the researchers to explore more powerful temporal patterns modeling within LLMs, paving the way to the development of temporal knowledge graph forecasting.

REFERENCES

- [1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).
- [2] Naman Agarwal, Brian Bullins, and Elad Hazan. 2016. Second-order stochastic optimization in linear time. *stat 1050* (2016), 15.
- [3] Luyi Bai, Wenting Yu, Die Chai, Wenjun Zhao, and Mingzhuo Chen. 2023. Temporal knowledge graphs reasoning with iterative guidance by temporal logical rules. *Information Sciences* 621 (2023), 22–35.
- [4] Li Cai, Xin Mao, Yuhao Zhou, Zhaoguang Long, Changxu Wu, and Man Lan. 2024. A Survey on Temporal Knowledge Graph: Representation Learning and Applications. *arXiv preprint arXiv:2403.04782* (2024).
- [5] He Chang, Chenchen Ye, Zhulin Tao, Jie Wu, Zhengmao Yang, Yunshan Ma, Xianglin Huang, and Tat-Seng Chua. 2024. A Comprehensive Evaluation of Large Language Models on Temporal Event Forecasting. *arXiv:2407.11638 [cs.CL]* <https://arxiv.org/abs/2407.11638>
- [6] Songgaojun Deng, Huzeфа Rangwala, and Yue Ning. 2020. Dynamic Knowledge Graph based Multi-Event Forecasting. In *KDD*. ACM, 1585–1595.
- [7] Songgaojun Deng, Huzeфа Rangwala, and Yue Ning. 2020. Dynamic knowledge graph based multi-event forecasting. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1585–1595.
- [8] Songgaojun Deng, Huzeфа Rangwala, and Yue Ning. 2021. Understanding Event Predictions via Contextualized Multilevel Feature Learning. In *CIKM*. ACM, 342–351.
- [9] Zifeng Ding, Zongyue Li, Ruoxia Qi, Jingpei Wu, Bailan He, Yunpu Ma, Zhao Meng, Shuo Chen, Ruotong Liao, Zhen Han, et al. 2023. Forecasttkgquestions: a benchmark for temporal question answering and forecasting over temporal knowledge graphs. In *International Semantic Web Conference*. Springer, 541–560.
- [10] Ellen R Girden. 1992. *ANOVA: Repeated measures*. Number 84. sage.
- [11] Danny Halawi, Fred Zhang, Chen Yueh-Han, and Jacob Steinhardt. 2024. Approaching Human-Level Forecasting with Language Models. *arXiv preprint arXiv:2402.18563* (2024).
- [12] Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. 2024. Large language models are zero-shot rankers for recommender systems. In *European Conference on Information Retrieval*. Springer, 364–381.
- [13] Woojeong Jin, Meng Qu, Xisen Jin, and Xiang Ren. 2020. Recurrent Event Network: Autoregressive Structure Inference over Temporal Knowledge Graphs. In *EMNLP (1)*. Association for Computational Linguistics, 6669–6683.
- [14] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning*. PMLR, 1885–1894.
- [15] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large Language Models are Zero-Shot Reasoners. In *NeurIPS*.
- [16] Julien Leblay and Melisachew Wudage Chekol. 2018. Deriving validity time in knowledge graph. In *Companion proceedings of the the web conference 2018*. 1771–1776.
- [17] Dong-Ho Lee, Kian Ahrabian, Woojeong Jin, Fred Morstatter, and Jay Pujara. 2023. Temporal Knowledge Graph Forecasting Without Knowledge Using In-Context Learning. In *EMNLP*. Association for Computational Linguistics, 544–557.
- [18] Dong-Ho Lee, Kian Ahrabian, Woojeong Jin, Fred Morstatter, and Jay Pujara. 2023. Temporal knowledge graph forecasting without knowledge using in-context learning. *arXiv preprint arXiv:2305.10613* (2023).
- [19] Kalev Leetaru and Philip A Schrodt. 2013. Gdelt: Global data on events, location, and tone, 1979–2012. In *ISA annual convention*, Vol. 2. Citeseer, 1–49.
- [20] Haoxuan Li, Zhengmao Yang, Yunshan Ma, Yi Bin, Yang Yang, and Tat-Seng Chua. 2024. MM-Forecast: A Multimodal Approach to Temporal Event Forecasting with Large Language Models. *arXiv preprint arXiv:2408.04388* (2024).
- [21] Zixuan Li, Zhongni Hou, Saiping Guan, Xiaolong Jin, Weihua Peng, Long Bai, Yajuan Lyu, Wei Li, Jiafeng Guo, and Xueqi Cheng. 2022. HiSMatch: Historical Structure Matching based Temporal Knowledge Graph Reasoning. *CoRR abs/2210.09708* (2022).
- [22] Zixuan Li, Xiaolong Jin, Wei Li, Saiping Guan, Jiafeng Guo, Huawei Shen, Yuanzhu Wang, and Xueqi Cheng. 2021. Temporal Knowledge Graph Reasoning Based on Evolutional Representation Learning. In *SIGIR*. ACM, 408–417.
- [23] Ruotong Liao, Xu Jia, Yunpu Ma, and Volker Tresp. 2023. GenTKG: Generative Forecasting on Temporal Knowledge Graph. *CoRR abs/2310.07793* (2023).
- [24] Xinyu Lin, Wenjie Wang, Yongqi Li, Shuo Yang, Fulí Feng, Yinwei Wei, and Tat-Seng Chua. 2024. Data-efficient Fine-tuning for LLM-based Recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 365–374.
- [25] Robert F Ling. 1984. Residuals and influence in regression.
- [26] Yushan Liu, Yunpu Ma, Marcel Hildebrandt, Mitchell Joblin, and Volker Tresp. 2022. Tlogic: Temporal logical rules for explainable link forecasting on temporal knowledge graphs. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 36. 4120–4127.
- [27] Ruilin Luo, Tianle Gu, Haoling Li, Junzhe Li, Zicheng Lin, Jiayi Li, and Yujiu Yang. 2024. Chain of History: Learning and Forecasting with LLMs for Temporal Knowledge Graph Completion. *CoRR abs/2401.06072* (2024).
- [28] Yunshan Ma, Chenchen Ye, Zijian Wu, Xiang Wang, Yixin Cao, and Tat-Seng Chua. 2023. Context-aware Event Forecasting via Graph Disentanglement. In *KDD*. ACM, 1643–1652.
- [29] Yunshan Ma, Chenchen Ye, Zijian Wu, Xiang Wang, Yixin Cao, Liang Pang, and Tat-Seng Chua. 2023. Structured, Complex and Time-complete Temporal Event Forecasting. *CoRR abs/2312.01052* (2023).
- [30] Farzaneh Mahdisoltani, Joanna Bięga, and Fabian M Suchanek. 2013. Yago3: A knowledge base from multilingual wikipedias. In *CIDR*.
- [31] Brando Miranda, Alycia Lee, Sudharsan Sundar, Allison Casasola, and Sammi Koyejo. 2023. Beyond Scale: The Diversity Coefficient as a Data Quality Metric for Variability in Natural Language Data. *arXiv e-prints* (2023). arXiv–2306.
- [32] Mehrnoosh Mirtaheri, Ryan A Rossi, Sungchul Kim, Kanak Mahadik, Tong Yu, Xiang Chen, and Mohammad Rostami. 2024. Tackling Long-Tail Entities for Temporal Knowledge Graph Completion. In *Companion Proceedings of the ACM on Web Conference 2024*. 497–500.
- [33] Sean P O'brien. 2010. Crisis early warning and decision support: Contemporary approaches and thoughts on future research. *International studies review* 12, 1 (2010), 87–104.
- [34] OpenAI. 2022. Introducing ChatGPT. <https://openai.com/blog/chatgpt>.
- [35] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinler, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*.
- [36] Grace I. Scarborough, Benjamin E. Bagozzi, Andreas Beger, John Berrie, Andrew Halterman, Philip A. Schrod, and Jevon Spivey. 2023. POLECAT Weekly Data. Harvard Dataverse. <https://doi.org/10.7910/DVN/AJGVIT>
- [37] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. In *ESWC (Lecture Notes in Computer Science, Vol. 10843)*. Springer, 593–607.
- [38] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. 2019. End-to-End Structure-Aware Convolutional Networks for Knowledge Base Completion. In *AAAI*. AAAI Press, 3060–3067.
- [39] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *CoRR abs/2302.13971* (2023).
- [40] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajwal Bhargava, Shruti Bhosale, Dan Bikell, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Eisbub, Jude Fernandes, Jeremy Fu, Wenjin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghaf Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madijan Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molbyog, Yixin Nie, Andrew Poultion, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov,

- Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288 [cs.CL] <https://arxiv.org/abs/2307.09288>
- [41] Duo Wang, Yuan Zuo, Fengzhi Li, and Junjie Wu. 2024. LLMs as Zero-shot Graph Learners: Alignment of GNN Representations with LLM Token Embeddings. *arXiv preprint arXiv:2408.14512* (2024).
- [42] Siheng Xiong, Yuan Yang, Faramarz Fekri, and James Clayton Kerce. 2022. TILP: Differentiable Learning of Temporal Logical Rules on Knowledge Graphs. In *The Eleventh International Conference on Learning Representations*.
- [43] Siheng Xiong, Yuan Yang, Ali Payani, James C Kerce, and Faramarz Fekri. 2024. Teip: Time prediction over knowledge graphs via logical reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 16112–16119.
- [44] Wenjie Xu, Ben Liu, Miao Peng, Xu Jia, and Min Peng. 2023. Pre-trained Language Model with Prompts for Temporal Knowledge Graph Completion. In *ACL (Findings)*. Association for Computational Linguistics, 7790–7803.
- [45] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *ICLR (Poster)*.
- [46] Mengmei Zhang, Mingwei Sun, Peng Wang, Shen Fan, Yanhu Mo, Xiaoxiao Xu, Hong Liu, Cheng Yang, and Chuan Shi. 2024. GraphTranslator: Aligning Graph Model to Large Language Model for Open-ended Tasks. In *Proceedings of the ACM on Web Conference 2024*. 1003–1014.
- [47] Yichi Zhang, Zhuo Chen, Wen Zhang, and Huajun Chen. 2021. Making large language models perform better in knowledge graph completion. In *MM. ACM*, 342–351.
- [48] Liang Zhao. 2022. Event Prediction in the Big Data Era: A Systematic Survey. *ACM Comput. Surv.* 54, 5 (2022), 94:1–94:37.
- [49] Haizhong Zheng, Rui Liu, Fan Lai, and Atul Prakash. 2022. Coverage-centric coreset selection for high pruning rates. *arXiv preprint arXiv:2210.15809* (2022).

A APPENDIX

A.1 Details on Dataset Construction

This section presents the sources and steps involved in constructing a new benchmark dataset for event prediction using large language models. To address the issue of information leakage in event prediction methods using LLMs—where the pre-training data of LLMs may already include events from the benchmark test set, making evaluation less reliable—we construct a new benchmark dataset based on POLECAT, a publicly available event dataset. POLECAT contains events extracted from internet news sources starting in 2018, including data from after the LLM’s training cutoff date, which can be used for evaluation. The construction pipeline consists of the following steps.

Filter by Country and Time. To reduce the size of the POLECAT, we applied filters based on both time and country. We downloaded data from January 1, 2018, to April 21, 2024, focusing on events related to Iran, Israel, and Egypt. The country filtering was applied to three fields: Actor Country, Recipient Country, and Country. Any event where at least one of these fields contained Iran, Israel, or Egypt was retained.

Remove Incomplete Events. Since POLECAT is extracted from news articles using event extraction algorithms, there are cases where the extraction process fails or results in missing information. To improve data quality, we removed events where both Actor Name and Actor Name Raw were listed as ‘None’, ‘np.NaN’, or empty strings. Actor Name Raw represents the original entity text as it appeared in the news sentence, while Actor Name is the result after entity mapping. When both fields are missing, it indicates a failure in the extraction process, with no fallback to the original entity name from the text. Similarly, we excluded events where both Recipient Name and Recipient Name Raw were missing.

Entity Expansion. The event extraction algorithm used by POLECAT often extracts multiple actors and recipients for a single event. Specifically, the Actor Name, Actor Name Raw, Recipient Name, and Recipient Name Raw fields may contain multiple entities within one data point, separated by semicolons. To maintain consistency with the previous steps in the dataset processing, we split these rows. For events with multiple actors and/or recipients, we create separate rows for each possible combination of actors and recipients. In this way, each unique actor-recipient pair is represented in its own row, while the event details, such as event type and other metadata, remain unchanged. If any Actor Name or Recipient Name is missing after expansion, we follow the same logic as before by filling it with the corresponding “Raw” fields or removing the row if no replacement is available.

Duplicate Removal. To eliminate redundant data, duplicates are identified based on five key fields: Actor Name, Recipient Name, Event Type, Event Mode, and Event Date. We removed any such duplicated entries to ensure that each unique event is represented only once in the dataset.

Entity Filtering. To improve the quality of the dataset, we removed irrelevant or incorrectly extracted entities, which may have resulted from algorithmic errors or noise. Using GPT-3.5-Turbo, we evaluated each unique entity in the dataset. The model was prompted to assess whether an entity was meaningful or not, responding with “yes” or “no.” Entities marked “yes” were kept. For entities marked “no,” we checked their frequency. If they appeared in 5 or fewer events, they were removed; otherwise, they were retained. The specific prompt used is shown in Figure 5.

Your task is to determine whether the input content is suitable as a subject or object in an evnet. Output YES if the input is a person's name, country, organization, institution, etc., and NO if the input is a number, date, verb, sentence, etc. Here are some examples:

Input: Israel
Output: YES

Input: Foreign minister
...

Input:
Output:

Figure 5: Specific prompt used for entity filtering.

A.2 Dataset Statistics

Overview. Our new dataset, constructed on the basis of POLECAT from filtered events extracted from news articles, spans from January 2018 to April 2024.

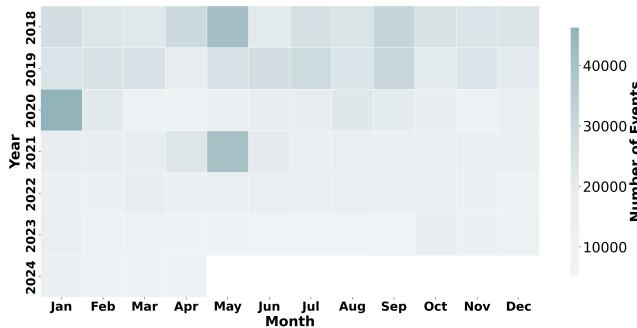


Figure 6: Heatmap illustration of event distribution by year and month.

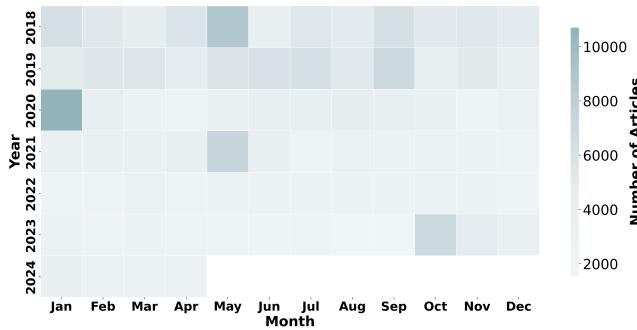


Figure 7: Heatmap illustration of article distribution by year and month.

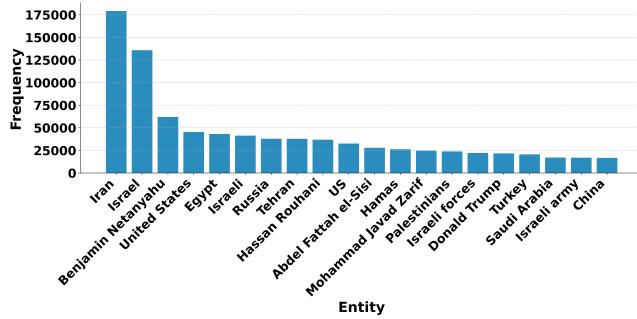


Figure 8: The top-20 most frequent entities in the dataset.

Time Distribution. To provide insights into the temporal distribution of the dataset, we analyzed the Event Date field to examine how the data is distributed over time. We drew two heatmaps: one showing the distribution of events by year and month, and the other showing the distribution of news articles by year and month. One news article may contain multiple events, providing additional context to the analysis. These visualizations help to identify temporal patterns, such as peaks or trends in event occurrences (see Figures 6 and 7).

Entity Distribution. In this section, we analyze the distribution of entities within the dataset. Entities are collected from two key fields: *Actor Name* and *Recipient Name*, representing the primary and secondary participants in each event, respectively. We ranked

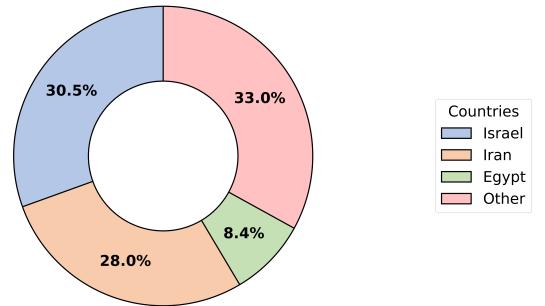


Figure 9: The event distribution across countries.

the top 20 most frequently occurring entities across the dataset. These entities represent the individuals, organizations, or groups that are most involved in or affected by the events in Iran, Israel, and Egypt during the given time period. The frequency of occurrence for each entity was calculated, and we visualized this distribution with a bar chart (see Figure 8).

Country Distribution. We analyzed the distribution of events related to Egypt, Iran, and Israel. Entities from these countries were identified from the *Actor Name* and *Recipient Name* fields. We counted the frequency of events for each country. A bar chart was generated to display the number of events involving Egypt, Iran, and Israel. Although we applied filtering, entity expansion during the process introduced other related entities, which are still relevant to the events (see Figure 9).

A.3 Statistical Analysis of Generative vs. MCQ Settings.

In this section, we explore the differences between the generative and MCQ settings for temporal knowledge graph forecasting tasks using LLM. We include this statistical analysis to demonstrate that the MCQ and generative settings mentioned in Section 3 are fundamentally equivalent when LLM is applied to TKG learning.

We conducted an empirical analysis using the GenTKG model, which uses rules to retrieve historical context for prompt construction in a generative setting, and then requires the LLM to make predictions using the generative setting instead of MCQ. We aim to demonstrate that historical information serves as an implicit MCQ, by analyzing the proportion of entities generated by the LLM that appear in the historical information provided in the prompt. This would show that the generative setting is essentially no different from the MCQ setting. We conducted experiments on two datasets, ICEWS and GDELT. The results are shown in Table 6.

Based on the results, we can see that in both datasets, the majority of entities generated by the LLM were already present in the prompt, indicating that the LLM rarely generates entirely new entities. This suggests that the LLM, when making predictions, tends to select entities from the provided context rather than truly generating new ones, meaning that the generative setting does not lead to genuine generation. Furthermore, the performance on the ICEWS dataset was significantly better than on GDELT. This could be because, during context prompt generation, the proportion of

Table 6: Statistics of Ground-truth (GT) and GenTKG Predictions from Context (Ctx). GT refers to ground-truth entities. Gen. Entities from Ctx refers to the percentage of generated entities that appear in the context.

Statistic	ICEWS	GDELT
Test Num	7,371	9,715
GT in Context (%)	63.60	45.58
Gen. Entities from Ctx (%)	96.97	94.85
Correct (Ctx, %)	37.59	11.83
Correct (Non-Ctx, %)	0.01	0.03
Wrong (%)	62.39	88.13

correct answers included in the context prompts was notably higher in ICEWS compared to GDELT. This indicates that even in the generative setting, the model’s performance is heavily influenced by the context prompt, making it no different from the MCQ setting.

Therefore, pure generation without any given context remains extremely challenging for large language models. Simply changing the problem setting does not result in genuine generation.

A.4 Baseline Implementation Details

This section provides a detailed explanation of the baseline mentioned in the Baseline section.

Non-LLM Methods including:

- DistMult [45] uses learned relation representations to model logical rules, primarily focusing on representing entity relationships in a simple multiplicative form.
- ConvTransE [38] applies convolution to concatenated entity and relation embeddings, helping to model more complex relationships and enhance link prediction capabilities.
- RGCN [37] employs relation-specific transformations with weight matrices for each relation. To handle many relations and prevent overfitting, it uses basis decomposition and block diagonal decomposition.
- RENET [13] builds on sequential and graph-based modeling to capture both temporal and structural interactions. It explicitly incorporates temporal sequences for dynamic knowledge graphs.
- REGCN [22] incorporates a recurrent unit to capture temporal dependencies in addition to structural information. It also uses a static graph to improve initial embeddings.
- HisMatch [21] reformulates temporal knowledge graph forecasting into a query-candidate matching problem, using a two-branch framework to efficiently match queries to candidate entities.

Zero-Shot Methods use TKG information in text form in the LLMs prompt. In our implementation, we used the approach described in the CoH [27] paper to construct text prompts and accessed GPT-3.5-turbo [34] and GPT-4o-mini [1] through the OpenAI API to obtain answers.

For fine-tune methods, we implement:

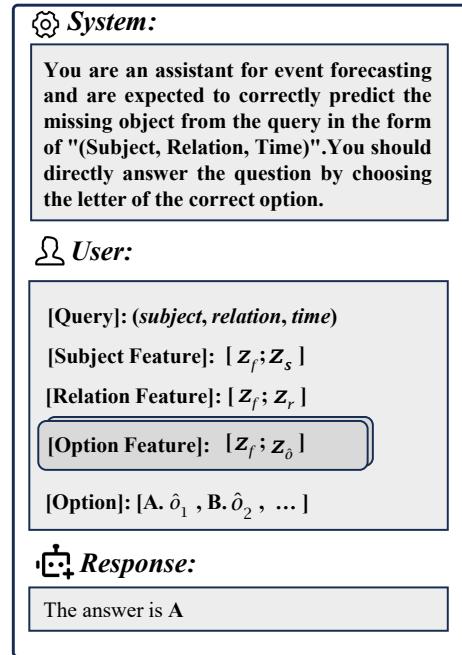


Figure 10: The instruction utilized in the hybrid graph tokenization.

- CoH [27] uses schema matching, entity-augmented, and relation-augmented historical facts to construct text prompts for fine-tuning LLMs.
- GenTKG [23] learns temporal logic rules to construct text prompts for fine-tuning LLMs.
- KoPA [47] employs structural embedding pre-training to capture knowledge graph (KG) information and transforms it into textual prefixes to guide LLMs.

A.5 Instruction

As illustrated in Figure 10, we present the instruction used for hybrid graph tokenization, which includes the textual task definition, as well as the format of the query and input.