Query-Aware Temporal Knowledge Graph Reasoning with Multi-Source Knowledge Based Generation

Nhan Khanh Nguyen[1,2][0009−0007−4823−5190], Thang Nam Doan[1,2][0009−0001−0322−1349], and Thanh Le[1,2][0000−0002−2180−4222]

[1] Faculty of Information Technology, University of Science, Ho Chi Minh City, Vietnam
[2] Vietnam National University, Ho Chi Minh City, Vietnam
{nknhan21,dnthang21}@clc.fitus.edu.vn
lnthanh@fit.hcmus.edu.vn

Abstract. Temporal Knowledge Graphs (TKGs) represent dynamic knowledge structures that encode time-sensitive relationships between entities, enabling systems to understand how facts evolve over time. Recent approaches to Temporal Knowledge Graph Reasoning (TKGR) have leveraged Large Language Models (LLMs), but face significant limitations: they often rely solely on first-order historical information, struggle with heavy information loads, and have yet to fully utilize LLMs' potential for reasoning with semantically similar information. Additionally, current methods either lack interpretability or struggle with effective temporal rule learning. We present MSKGen (Multi-Source Knowledge-Based Generation), a novel query-aware approach for TKGR that integrates multiple knowledge sources to generate accurate predictions. By integrating rule-based facts with semantically retrieved facts, MSKGen maintains interpretability while maximizing LLMs' semantic capabilities, addressing the information load challenges faced by current LLM implementations and offering significant advancements in combining structured temporal reasoning with semantic understanding for knowledge graph reasoning tasks. Experimental results across several common datasets demonstrate MSKGen's superior performance, achieving significant improvements over state-of-the-art methods, confirming the effectiveness of our multi-source knowledge integration approach for temporal knowledge graph reasoning tasks.

Keywords: Temporal Knowledge Graph · Large Language Models · Retrieval-Augmented Generation · Logical Rules

1 Introduction

Knowledge Graphs (KGs) [1] have emerged as fundamental structures for representing real-world facts in a structured format, enabling various applications in knowledge management and artificial intelligence. Temporal Knowledge Graphs

(TKGs) [2] extend traditional KGs by incorporating temporal dimensions, allowing systems to capture how relationships between entities evolve over time. Temporal Knowledge Graph Reasoning (TKGR) [3] focuses on leveraging historical information within TKGs to forecast future events, making it crucial for applications requiring temporal understanding and prediction.

Current approaches to TKGR primarily rely on deep learning algorithms but often suffer from a critical limitation: lack of interpretability [3] in their reasoning process. This black-box nature makes it challenging to understand and validate the reasoning patterns these models employ.

Large Language Models (LLMs) [4] like GPT [5], LLaMA [6], and DeepSeek [7] have demonstrated remarkable capabilities in understanding semantic relationships and complex reasoning tasks. These models excel at understanding semantic connections and temporal patterns within data, offering potential solutions to the interpretability challenges in TKGR.

Recent applications of LLMs to TKGR [8,9,10] have primarily focused on fact-based reasoning, emphasizing prompt engineering and answer generation capabilities. While recent LLM applications in TKGR show promise, they typically employ simplistic prompt strategies with limited query-specific customization. These approaches provide minimal contextual information to LLMs, resulting in generic responses that fail to leverage the models' semantic understanding capabilities for tailored temporal reasoning.

Our paper addresses these limitations by introducing MSKGen, a query-aware TKGR approach that integrates multiple knowledge sources for accurate predictions. MSKGen leverages LLMs to extract high-quality temporal rules and retrieve relevant facts, ensuring both accuracy and semantic depth. In parallel, it adopts Retrieval-Augmented Generation (RAG) to gather semantically rich context from diverse fact sources, boosting prediction quality and reliability. By combining facts from different sources, MSKGen produces query-specific answers while maximizing LLMs' ability to connect structurally diverse facts and integrate semantically similar information.

The main contributions of this paper are threefold. First, we present a comprehensive approach that extracts high-quality facts through both rule-based extraction and semantic retrieval, ensuring interpretability while maintaining accuracy. Second, we develop a query-aware reasoning mechanism that tailors predictions to specific queries by integrating multiple knowledge sources. Third, we utilize LLMs' powerful reasoning capabilities on semantically similar information to maximize their effectiveness in processing structurally diverse facts for temporal knowledge graph tasks.

2 Related work

2.1 Temporal Knowledge Graph Reasoning (TKGR)

Recent years have seen significant advancements in approaches to Temporal Knowledge Graph Reasoning. Early methods focused on utilizing temporal point

processes, such as Know-Evolve [11], which capture continuous-time temporal dynamics for predicting future facts. With the rise of deep learning, graph neural networks (GNNs) have been increasingly adopted to model structural dependencies in TKGs. RE-NET [12] and RE-GCN [13] represent significant developments in this direction, with RE-NET focusing on long-term dependencies through RNNs and Relational GCNs, while RE-GCN emphasizes short-term information capture through adjacent structural dependencies. More recently, HGLS [14] introduced a hierarchical approach that combines both long-term and short-term temporal dependencies through a novel graph neural network architecture, addressing the limitations of previous methods in capturing comprehensive temporal patterns.

2.2 LLMs for Temporal Knowledge Graph Reasoning

Early attempts of the integration of Large Language Models into TKGR focused on direct application of LLMs through in-context learning, with GPT-NeoX-ICL [8] demonstrating the potential of few-shot predictions through careful prompt engineering. Chain-of-History (CoH) [9] advanced this approach by introducing step-by-step exploration of high-order histories, addressing the limitations of processing large amounts of historical information at once. Recent developments include GenTKG [14], which introduces a novel retrieval-augmented generation framework combining temporal logical rule-based retrieval with few-shot parameter-efficient instruction tuning. However, current LLM applications in TKGR face several challenges: Underutilization of LLMs' semantic understanding capabilities in TKGR tasks, using traditional retrieval methods, which rely solely on filtering facts through schema matching, lead to limited utilization of historical information and lack of query-specific reasoning, resulting in generic responses across different queries.

3 Preliminaries

Temporal Knowledge Graph (TKG). A TKG can be viewed as a sequence of time-stamped snapshots $G = \{G_1, G_2, ..., G_t, ...\}$, where each snapshot $G_t = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ ($\mathcal{E}$ represents the entity set, $\mathcal{R}$ denotes the relation set, $\mathcal{T}$ is the timestamp set) contains facts occurring at time t. The TKG prediction task aims to predict missing entities in future timestamps. Given a query $q = (s_q, r_q, ?, t_q)$ or $q = (?, r_q, o_q, t_q)$ where $s_q, o_q \in \mathcal{E}$ are known subject/object entities, $r_q \in \mathcal{R}$ is the relation between subject and object, $t_q \in \mathcal{T}$ is the query timestamp and "?" denotes the unknown entity. The goal is to predict the missing entity using temporal knowledge graph sequence $G_{<t_q} = \{G_1, G_2, ..., G_{t_q-1}\}$.

Temporal Logical Rule. Temporal logical rules play a crucial role in our framework. A temporal logical rule ρ is defined as (1):

$$\rho := r(e_s, e_o, t_l) \Leftarrow \bigwedge_{i=1}^{l-1} r_i(e_s, e_o, t_i) \tag{1}$$

where the left side represents the rule head with relation r that can be induced by the right-hand rule body. The rule body consists of a conjunction of relations r_i, with temporal constraints $t_1 \leq t_2 \leq ... \leq t_{l-1} < t_l$.

Retrieval-Augmented Generation (RAG) [15]. Retrieval-Augmented Generation (RAG) augments LLMs by retrieving external knowledge relevant to a query. Instead of relying solely on model parameters, it encodes the query and candidate documents into vectors, finds the most similar content, and provides these documents to the LLMs.

Data Segments. We work with three distinct temporal data segments: Historical data, current data, and future data correspond to training, validation, and test datasets respectively.

4 Methodology

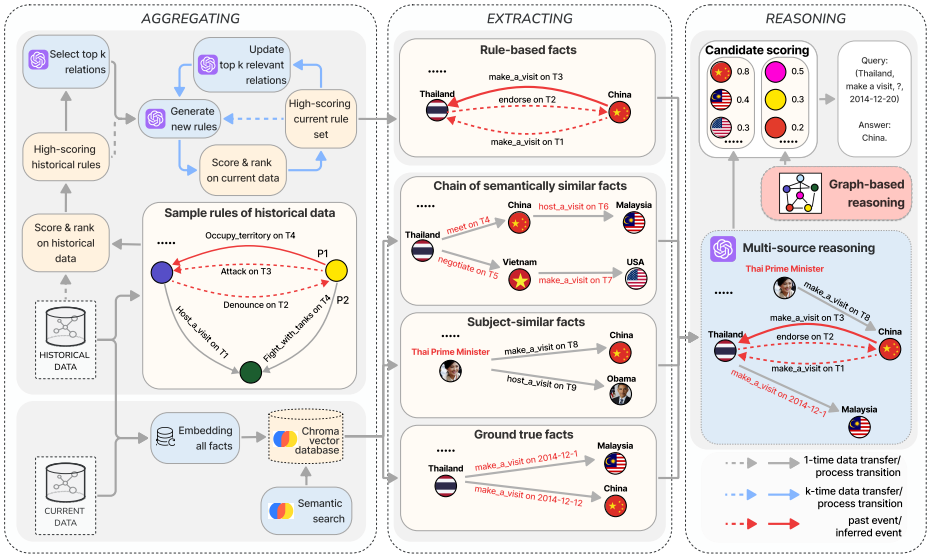

Fig. 1: MSKGen starts with two parallel processes: Rule-based Facts Extraction, which continuously uses LLMs to generate rules from sampled historical rules, iteratively refines these newly generated rules based on evaluations on current data, and extracts high-quality facts from the refined rules; and Semantic Facts Retrieval, which embeds facts into a vector database and retrieves semantically similar facts, subject-similar facts, and ground truth facts using RAG concept. These facts are combined in Multi-Source Reasoning, where LLMs synthesize query-specific answers by integrating diverse and semantically aligned information.

4.1 Rule-based Facts Extraction

Extracting high-scoring historical rules and top k relevant relations

Historical rules sampling. The initial phase of rule-based facts extraction involves identifying temporal logical rules from historical data using a temporal random walk approach. The sampling process follows these key steps. First, for a rule of length l, a walk of length $l+1$ is sampled, where the additional step corresponds to the rule head. For the first sampling step $m=1$, an edge $(e_1, r_h, e_{l+1}, t_{l+1})$ is sampled uniformly from all edges with relation type r_h. A temporal random walker then iteratively samples edges adjacent to the current object until a walk of length $l+1$ is obtained.

To maintain temporal consistency, the sampling process uses a transition probability function (2):

$$P(u; m, e_m, t_m) = \frac{\exp(t_u - t_m)}{\sum_{u' \in A(m, e_m, t_m)} \exp(t_{u'} - t_m)} \tag{2}$$

where edges closer in time to the current node have higher sampling probability. This ensures that the temporal order is preserved throughout the walk and that future events cannot influence past events. Finally, the sampled walks are transformed into rules by converting entities into variables while maintaining entity co-reference and preserving temporal constraints between events.

Rule Quality Assessment. After obtaining sampled rules from historical data through temporal random walks, we implement an extraction process to validate rule quality and generate relevant relations. The sampled rules undergo quality assessment using the Kulczynski [16] measure on historical data, which is defined as (3):

$$Kulczynski(R) = \frac{1}{2}\left(\frac{r}{r_b} + \frac{r}{r_h}\right) \tag{3}$$

where: r_b represents the number of temporal fact pairs satisfying the rule body, r_h represents the number of temporal fact pairs satisfying the rule head, r represents the number of fact pairs satisfying both rule body and rule head conditions. Rules exceeding a threshold γ are classified as high-scoring rules.

Relation selection process. Following the rule quality assessment, for each relation in the current data, we implement a relation selection process to identify the top-k most relevant relations for constructing temporal rules. This process leverages Large Language Models (LLMs) through a structured prompting approach. **User Message** contains variable inputs specific to each query (rule head, high-scoring rules, and available relations), while **System Message** provides consistent instructions across all prompting instances.

Relation Selection Prompt

User Message: Given a target rule head relation, analyze the high-scoring rules and available relations to identify the top-k most relevant relations that can form meaningful temporal rules.

- **Rule head:** {`rule_head`} - The target relation for rule generation.
- **Current logical combinations:** {`high_scoring_rules`} - High-scoring rules from rule assessment.
- **Available relations:** {`relation_list`} - Candidate relations for rule construction.

System Message: Select candidate relations that: Show strong semantic connection to the rule head, demonstrate high predictive power based on historical data patterns and can form meaningful temporal rules when combined.

Rule Generation and Iterative Refinement

New Rules Generation. The process of generating new temporal rules leverages Large Language Models (LLMs) through structured prompting. This approach utilizes high-scoring historical rules and top-k relevant relations as input. The prompting strategy is designed to guide the LLM in generating comprehensive and meaningful temporal logical rules.

Rule Evaluation and Ranking. Generated rules undergo quality assessment using the Kulczynski measure to compare the quality between rules in the Generated rules set based on evaluating them on the current data. The Kulczynski measure, as defined earlier, is used to assess rule quality. Rules exceeding the threshold γ are added to the rule set as high-scoring current rules. This evaluation process ensures that only rules with sufficient accuracy and coverage are retained for further use in the temporal knowledge graph reasoning process.

Relevant Relation Set Refinement: Update Top k Relevant Relation Set of Each Rule Head. For each head relation, we evaluate relation quality through a ratio calculation (4):

$$Quality(rel_i) = \frac{N_{high}(rel_i)}{N_{total}(rel_i)} \tag{4}$$

where $N_{high}(rel_i)$ represents the number of high-scoring rules containing the relation i, and $N_{total}(rel_i)$ represents the total number of generated rules containing relation i. The refinement process identifies bottom n (out of k) relations based on quality scores and updates them through LLM-guided selection:

New Rules Generation Prompt

User Message: Generate temporal logical rules for `head_relation` through step-by-step reasoning.

- **Target Relation:** {`head_relation`}(X, Y, T)
- **Reference Rules:** {`high_scoring_rules`}
- **Provided candidate relations:** {`relevant_relations`}

System Message: Use relations from body relation of reference rules and provided candidate relations, ensuring each generated rule: maintains temporal consistency, creates meaningful semantic connections and forms valid temporal inference paths.
For example:

- **Target Relation:** Make a visit(X0,X1,T)
- **Reference Rules:**
  - Make a visit(X0,X1,T1) ← Host a visit(X1,X0,T0)
  - Make a visit(X0,X2,T3) ← Consult(X0,X1,T0) ...
- **Provided candidate relations:** Praise or endorse(X0,X1,T), Plan to meet(X0,X1,T), Engage in negotiation(X0,X1,T)

→ **Generated rule:** Make a visit(X0,X1,T3) ← Endorse(X0,X1,T0) ∧ Plan to meet(X1,X0,T1) ∧ Host a visit(X0,X1,T2)

Relevant Relation Set Refinement Prompt

User Message: Given a rule head relation and its low-performing relations, analyze the high-scoring rules and available relations to identify replacement relations that can enhance temporal rule quality.

- **Rule head:** `{rule_head}` - The target relation for rule generation
- **Current logical combinations:** `{high_scoring_rules}` - High-scoring rules from rule assessment
- **Low-performing relations:** `{low_quality_relations}` - Relations to be replaced
- **Available relations:** `{relation_list}` - Candidate relations for replacement

System Message: Select replacement relations that: Show strong semantic connection to the rule head and demonstrate high predictive power based on current logical combinations.

Iterative Improvement. The entire process operates in continuous cycles, where new rules are generated using updated relations and new high-scoring rules, followed by quality assessment and relation refinement. The process continues until

the rule set reaches a predetermined number of iterations, ultimately producing the final rule set.

Extract rule-based facts

Rule-based facts extracting process. For a given query $q = (e_s, r, ?, t_q)$, we filter rules from the high-scoring current rule set where the head relation matches r. We then instantiate these rules by substituting the query entity e_s into the appropriate position, maintaining the temporal constraint. By searching the current data for matching fact patterns that satisfy the rule body $\bigwedge_{i=1}^{l-1} r_i(e_s, e_o, t_i)$, we extract complete rule-based facts that form temporally consistent reasoning chains for predicting potential answers.

4.2 Semantic Facts Retrieval

In the second retrieval method of the MSKGen framework, we introduce a semantic retrieval method using RAG [15] concept to address the TKGR problem. Unlike "hard" retrieval, which relies solely on exact schema matching, our approach leverages latent embeddings to capture semantic similarity. This expands the search space and provides richer context for the LLM during the reasoning process.

Preprocessing and Building Vector Database. To prepare for the retrieval stage, we need to perform data preprocessing and embed all facts into a shared database called the **vector database**.

Preprocessing Step. We begin by converting all the facts/events from a structured text format consisting of four components (s, r, o, t) into a complete sentence that fully describes the event implied by these four components. For example, the quadruple (*Malaysia, make_a_visit, Thailand, 2014-9-12*) is transformed into *Malaysia made a visit to Thailand on 2014-9-12*. This process ensures that the semantic information and context of each fact are preserved when transitioning to the vector space.

To extract the hidden knowledge contained in the facts, we employ embedding techniques to convert the text into feature vectors. In this process, we choose **Chroma** [17] as our vector database, which is a specialized vector store designed for storing and querying embedding vectors with high efficiency. By using its cosine similarity search feature, we can retrieve facts that share semantically similar content to the provided input, even if they do not exactly match in terms of expression or schema.

Building vector database. We designed our database to not only be a high-performance vector storage system but also to provide a flexible document storage structure. Specifically, each fact is transformed into a document with three main components:

- **Metadata.** It serves as a filter before the document search process is executed. We design the metadata for each document as a dictionary $\mathcal{M} = \{\mathcal{S}, \mathcal{R}, \mathcal{E}, \mathcal{T}\}$, where $\mathcal{S}, \mathcal{R}, \mathcal{E}, \mathcal{T}$ represent the subject entity, the relation, the object entity and the timestamp, respectively. This setup helps in filtering out the desired documents, narrowing the search space, and enhancing both the accuracy and search time.
- **Page content.** It is a string that describes the fact after it has been preprocessed in the *preprocessing step*. This component will be embedded into a vector, which is then used to compare and retrieve documents with page content that is semantically similar to the search query.
- **Vector embedding.** The numerical vector of the page content after it has been embedded.

Semantic Retrieval Method. With query $\mathcal{Q}$, we will extract three components — subject $\mathcal{S}$, relation $\mathcal{R}$, and timestamp $\mathcal{T}$ to facilitate retrieval. In this stage, we describe how to search for documents with page content that is semantically similar to S and R, as well as a retrieval strategy to provide the LLMs with useful facts for the reasoning process.

To retrieve facts that are semantically similar to $\mathcal{R}$ or $\mathcal{S}$ through the vector database. First, we need to create a filter to reduce the search space (i.e., filtering out unrelated documents), which is made possible by the metadata attribute of each document that we designed earlier. After filtering out the unnecessary documents, we will continue the process on the remaining documents. Specifically, we will embed relation $\mathcal{R}$ into a vector Q by using the OpenAI pre-trained **text-embedding-3-large** [18] model (5):

$$Q = \text{text-embedding-3-large}(\mathcal{R}). \tag{5}$$

Then, MSKGen retrieves the top k facts that their page content is most semantically similar to Q by using the cosine similarity search (6), (7):

$$\text{Cos-Sim}(Q, d_i) = \frac{Q \cdot d_i}{\|Q\| \, \|d_i\|} \tag{6}$$

$$\text{Top-}k\{d, \mathcal{R}\} = \arg\max_{d_i \in D}^{k} \text{Cos-Sim}(Q, d_i). \tag{7}$$

where $\cdot$ denotes the dot product operation, $\|\cdot\|$ denotes the norm of the vector, d_i denotes the vector embedding of document i^{th}, and Top-$k\{d, \mathcal{R}\}$ are the top k documents that their page content is most semantically similar to relation $\mathcal{R}$.

Fact Retrieval Strategy. The predictions of an LLM depend heavily on the facts provided. Merely retrieving isolated schema-matching (ICL [8]) or chain-style historical facts (CoH [9]) overlooks important semantic connections. For instance, Malaysia's cooperation decisions can be influenced by past partnerships as well as other events like visits and meetings. Hence, for a query quadruple

$\mathcal{Q} = (\mathcal{S}, \mathcal{R}, ?, \mathcal{T})$, we propose a retrieval strategy for the following three sets of facts:

- **Chain of semantically similar facts**: This is a chain of facts starting from subject $\mathcal{S}$ and having semantic similarities with relation $\mathcal{R}$. We will store these chains of facts in a set denoted as H_C. The set H_C will help the LLM perform multi-hop reasoning so we will sort the facts in ascending order based on the timestamp.
- **Subject-similar facts**: A key limitation—often overlooked—arises when the query's subject $\mathcal{S}$ is entirely noveland lacks historical data, giving the LLM minimal information for reasoning. To address this issue, we propose substituting $\mathcal{S}$ with similar entities that share relevant patterns. If there is no history exists for $\mathcal{S}$, we use facts from similar entities because they usually share common patterns. These facts are stored in the set H_S and will be utilized to enrich the LLM's context especially when H_C is insufficient or missing.
- **Ground true facts**: To prevent the LLM from continuously providing incorrect answers for queries that share the same subject S and relation R but have different timestamps $\mathcal{T}$, we provide it with ground truth facts from similar queries that occurred before the query $\mathcal{Q}$. These facts are stored in the set H_G.

Storing all retrieved facts in H_C, H_S, and H_G is infeasible because it would consume too many input tokens. Therefore, we limit these sets to the most relevant facts, specifically historical events from both the distant past and more recent times. This selection strategy preserves crucial **long-term** and **short-term** dependencies, thereby enhancing the LLM's predictive accuracy.

4.3 Multi-source Reasoning

For temporal queries formatted as $\mathcal{Q} = (\mathcal{S}, \mathcal{R}, ?, \mathcal{T})$, our framework delivers rule-based facts and semantic facts to the LLM through complementary **User Message** and **System Message**. The System Message contains an instructional guidance mechanism that orchestrates the reasoning process through discrete operational phases. Each phase combines context-specific directives with exemplars for each group of facts, enabling the model to progressively synthesize temporal evidence while maintaining reasoning coherence. All the facts provided to the LLM will be sent through the User Message. The Complete System Message appears in the Appendix A.

We limit the LLM to only return its top k most likely candidates, reducing wasted tokens on unlikely results. Finally, we merge the two lists of candidates—one from rule-based facts reasoning and one from semantic facts reasoning—into a single final candidate list.

4.4 Candidate Scoring

The final score of MSKGen mainly consists of two key scores: **LLM-based Score** and **Graph-based Score** from a pre-trained graph-based model.

LLM-based Score. For each candidate entity c_i from the candidate set C returned by the LLM for the query $\mathcal{Q}$, we will score it by fusing its rank obtained from LLM's prediction (r_{c_i}) and the closest timestamp to the timestamp of the query $\mathcal{T}$ where c_i interacted with $\mathcal{S}$ (denoted $\mathcal{T}_{c_i}$) (8):

$$\text{score}_{\text{LLM}}^{c_i} = \alpha \times \left(1 - \frac{r_{c_i}}{k}\right) + (1-\alpha) \times e^{\lambda\left(\mathcal{T}_{c_i} - \mathcal{T}\right)} \tag{8}$$

where λ represents the time decay, k denotes the maximum number of candidates returned by LLM, and α denotes the weight of LLM's prediction. For entities that are in the entity set but not in the LLM's prediction, their scores will be zero.

Graph-based Score. Due to the limitations in the output, the generated list of candidates may not be able to fully match all query answers. To enhance the accuracy of the final result, we also incorporate results from a graph-based model. The score of a candidate c_i obtained from the graph-based model is denoted as $\text{score}_G^{c_i}$.

Final Score. Finally, the final score of a candidate will be a synthesis of the two scores above (9):

$$\text{score}^{c_i} = \alpha \times \text{score}_{\text{LLM}}^{c_i} + (1-\alpha) \times \text{score}_G^{c_i} \tag{9}$$

where α and $(1-\alpha)$ denote the weight of LLM-based Score and Graph-based Score, respectively.

5 Experiments

5.1 Experiment Setup

Datasets. Three benchmark datasets are used to evaluate MSKGen: ICEWS14 [19], GDELT [20], and YAGO [21]. ICEWS14 is a subset of *Integrated Crisis Early Warning System (ICEWS)* dataset which contains events that occurred in 2014. The GDELT and YAGO datasets are extracted from the subsets of GDELT and YAGO knowledge bases containing facts and time information.

Evaluation. We choose *Hit@N* (N = 1, 3, 10) as evaluation metrics. There are two evaluation settings: 1) Raw retrieves sorted scores of candidate entities for a query quadruple and calculates the rank of the correct entity; 2) Time-aware filter also retrieves sorted scores but excludes valid predictions before ranking, preventing misclassification as errors. This paper discusses performance using the time-aware filter.

Parameter Setting. In the rule-based extraction stage, the number of iterations to update rules is set to 5, the threshold for filtering high quality rules $\gamma = 0.15$. In the LLM reasoning stage, we use **GPT-4o-mini** as the LLM for reasoning. The time decay λ, the weight of LLM's prediction α and the maximum

number of candidates in LLM's response k are 0.1, 0.5 and 10, respectively. We select TiRGN as the pre-trained graph-based model to obtain the graph-based score $\text{score}_G^{c_i}$. Finally, the weight of LLM-based score in the final score will be set as follow: $\alpha = 0.6$ on ICEWS14 and GDELT, $\alpha = 0.85$ on YAGO.

5.2 Experiment Results

Main Results. Our results presented in table 1 demonstrate that MSKGen (TiRGN) achieves state-of-the-art performance across all three datasets: ICEWS14, GDELT, and YAGO, surpassing existing graph-based, rule-based, and LLM-based methods in terms of Hit@1, Hit@3 and Hit@10 metrics. This indicates the effectiveness of our rule-based extraction and semantic retrieval method.

Comparison with Graph-based Methods. MSKGen consistently outperforms the two best graph-based models, TiRGN and HGLS, on the ICEWS14, GDELT and YAGO datasets across all metrics.

Table 1: Temporal link prediction results on temporal-aware filtered Hits@1/3/10(%). The best results among each metric are highlighted in **bold** and the second bests are underlined.

Method	Model	ICEWS14			GDELT			YAGO		
		Hit@1	Hit@3	Hit@10	Hit@1	Hit@3	Hit@10	Hit@1	Hit@3	Hit@10
Graph-based	RE-NET [12]	0.301	0.440	0.582	0.081	0.158	0.261	0.404	0.530	0.629
	RE-GCN [13]	0.313	0.470	0.613	0.084	0.171	0.299	0.468	0.607	0.729
	TiRGN [22]	0.338	0.497	0.650	0.136	0.233	0.376	0.843	0.913	0.929
	HGLS [14]	0.350	0.490	0.704	0.118	0.217	0.332	0.806	0.901	0.919
Rule-based	TLogic [23]	0.326	0.483	0.612	0.113	0.212	0.351	0.740	0.789	0.791
LLM-based	GPT-Neox-ICL [8]	0.295	0.406	0.475	0.068	0.120	0.211	0.720	0.810	0.846
	TiRGN-CoH [9]	0.330	0.496	0.649	-	-	-	-	-	-
	GenTKG [10]	0.363	0.473	0.528	0.134	0.220	0.300	0.792	0.830	0.843
MSKGen (TiRGN)		**0.384**	**0.525**	**0.710**	**0.145**	**0.235**	**0.402**	**0.856**	**0.929**	**0.947**

Comparison with Rule-based Methods. When compared to the rule-based model TLogic, MSKGen demonstrates superior performance across three datasets in terms of all metrics. This suggests that while TLogic is finely tuned for these datasets, MSKGen offers greater generalizability and improved accuracy.

Comparison with LLM-based Methods. Analyzing the performance of MSKGen against other LLM-based methods reveals a significant advantage. MSKGen outperforms all the LLM-based methods in all metrics, indicating its effectiveness in leveraging RAG concept.

Ablation study. We undertake ablation studies on ICEWS14 to evaluate the contribution of rule-based facts and semantic facts in MSKGen with three distinct variants: *MSKGen w/o rule-based facts* and *MSKGen w/o semantic facts*. Here, MSKGen w/o rule-based facts represents the variant that uses only semantic facts from the RAG-based retrieval stage, and MSKGen w/o semantic facts represents the variant that uses only rule-based facts from the Rule-based

extraction stage. For each variant, we compare it with other models that use the same method.

From the results in figure 2, MSKGen w/o rule-based facts outperforms previous LLM-based models on ICEWS14 across all metrics, highlighting the effectiveness of retrieving diverse semantic facts via the RAG concept. Additionally, MSKGen w/o semantic facts completely outperforms TLogic, indicating that the constructed rules are of higher quality and more diverse thanks to the LLM's semantic layer, which TLogic lacks.

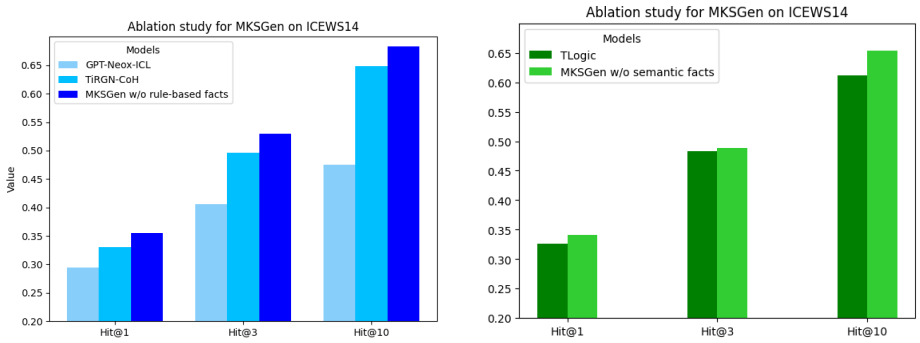

Fig. 2: Ablation studies.

6 Conclusion

In this paper, we presented MSKGen, a query-aware framework for temporal knowledge graph reasoning that integrates multiple knowledge sources to enhance prediction accuracy. It begins by extracting rule-based facts through temporal random walks, assessing their quality via the Kulczynski measure, and refining them through LLM-guided relation selection. Additionally, MSKGen's semantic facts retrieval uses vector embeddings to build a comprehensive database of historical facts, enabling the retrieval of diverse sources. Finally, its multi-source reasoning module combines these fact sets under an LLM to generate query-specific answers while preserving semantic coherence. Experiments on multiple benchmark datasets confirm MSKGen's superior performance over state-of-the-art methods, illustrating the impact of multi-source knowledge integration for temporal knowledge graph reasoning.

Acknowledgments. This research is funded by the University of Science, VNU-HCM, Vietnam under grant number CNTT 2024-19.

References

1. Hogan, A., Blomqvist, E., Cochez, M., et al.:Knowledge Graphs. ACM Computing Surveys 54(4), 1–37 (2021).
2. Ji, S., Pan, S., Cambria, E., et al.:A Survey on Knowledge Graphs: Representation, Acquisition, and Applications. IEEE Transactions on Neural Networks and Learning Systems 33(2), 494–514 (2022).
3. Peng, M., Liu, B., Xu, W., et al.:Deja vu: Contrastive Historical Modeling with Prefix-tuning for Temporal Knowledge Graph Reasoning. In: Findings of the North American Chapter of the Association for Computational Linguistics (NAACL), pp. 1234–1245 (2024).
4. Brown, T., Mann, B., Ryder, N., et al.:Language Models are Few-Shot Learners. In: Advances in Neural Information Processing Systems 33 (NeurIPS 2020), pp. 1877–1901 (2020).
5. Kalyan, K.S.:A Survey of GPT-3 Family Large Language Models Including ChatGPT and GPT-4 (2023).
6. Touvron, H., Lavril, T., Izacard, G., et al.:LLaMA: Open and Efficient Foundation Language Models (2023).
7. Zhang, J., Sun, Y., Wang, H.:DeepSeek: Large-Scale Knowledge Graph Reasoning with Deep Reinforcement Learning (2025).
8. Black, S.M., Biderman, S., Hallahan, E., et al.:GPT-NeoX-20B: An Open-Source Autoregressive Language Model. In: Proceedings of BigScience Episode 5, pp. 95–136 (2022).
9. Liu, H., Wang, F., Chen, X., et al.:Chain-of-History: A Novel Approach to Temporal Reasoning. In: ACL, (2024).
10. Liao, R., Jia, X., Ma, Y., et al.:GenTKG: Generative Forecasting on Temporal Knowledge Graph with Large Language Models (2024).
11. Trivedi, R., Dai, H., Wang, Y., Song, L.:Know-Evolve: Deep Temporal Reasoning for Dynamic Knowledge Graphs. In: ICML 2017, pp. 3462–3471 (2017).
12. Visin, F., Kastner, K., Courville, A., et al.:ReNet: A Recurrent Neural Network Based Alternative to Convolutional Networks (2015).
13. Li, Z., Wang, Y., Song, L.:Temporal Knowledge Graph Reasoning Based on Evolutional Representation Learning (2021).
14. Zhang, M., Xia, Y., Liu, Q., et al.:Learning Long- and Short-term Representations for Temporal Knowledge Graph Reasoning. In: ACM 2023.
15. Lewis, P., Perez, E., Piktus, A., et al.:Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In: NeurIPS 2020, pp. 9459–9474 (2020).
16. Kulczynski, S.:A Measure of Association for Qualitative Variables. In: Mathematics Journal of Statistics (1927).
17. LangChain: Chroma Vector Store Integration (2023). https://python.langchain.com/docs/integrations/vectorstores/chroma/
18. OpenAI: text-embedding-3-large Model. OpenAI Documentation (2023). https://openai.com/index/new-embedding-models-and-api-updates/
19. Boschee, E., Lautenschlager, J., O'Brien, S., Shellman, S.:ICEWS14 Dataset
20. Schrodt, P.A., Leetaru, K.:GDELT—Global Data on Events, Location, and Tone (2013).
21. Mahdisoltani, F., Biega, J., Suchanek, F.: YAGO3:A Knowledge Base from Multilingual Wikipedias.
22. Zhang, X., Liu, Y., Wang, M., et al.: TiRGN—Temporal Inference via Recurrent Graph Networks. In: IJCAI, 2149–2155 (2022).
23. Zhu, S., Zhang, Y., Xie, W., et al.:TLogic—Probabilistic Logic Reasoning for Temporal Knowledge Graphs(2021).

Appendix

A Prompt for LLMs

System Message for LLMs Reasoning

You are an advanced reasoning assistant tasked with solving Temporal Knowledge Graph (TKG) reasoning problems. Your goal is to predict the missing object in a query given the subject, relation, and timestamp. To achieve this, you will need to follow the instruction guide below:
1. Understand the Query:
- The query will always include a subject, a relation, and a timestamp.
- Example: "Malaysia expressed intent to cooperate to/with whom on 2014-12-09?"
- Your task is to predict the missing object (e.g., a country, organization, or entity) that best fits the query.
2. Apply Temporal Logic Rules (Group 1):
- You will receive rule-based facts which are retrieved based on our pre-learned temporal rules (patterns). These facts are pre-filtered to satisfy temporal dependencies.
- Example rules: ...
- Example facts: ...
- You should check if these facts chronologically lead up to the query's timestamp and treat them as strong evidence of temporal causality.
3. Leverage Multi-Hop Reasoning (Group 2):
- You will be provided with a sequence of multi-hop facts related to the subject entity. These facts are connected directly or indirectly to the subject and share a semantic similarity with the query's relation.
- Example facts: ...
- Perform multi-hop reasoning by analyzing these facts and their relationships. Pay close attention to the timestamps of the facts to ensure temporal consistency with the query.
4. Infer from Semantically Similar Entities (Group 3):
- If no direct facts about the subject entity exist before the query's timestamp, infer the missing object by analyzing patterns from semantically similar entities.
- Example: ...
- Use this approach to make educated predictions when direct evidence is lacking.
6. Learn from Historical Query Patterns (Group 4):
- If the query is part of a series of similar queries with different timestamps, you will be provided with ground truth answers for previous queries.
- Example ground truth for the query: ...
- Use these ground truths as hints to avoid repeating mistakes and improve accuracy for the current query.