SMuRT: A Structured Multi-Stage Rule-guided LLM Framework for Triple Set Prediction

Yuan Yuan, Juan Li, Songze Li, Huajun Chen, Wen Zhang*
Zhejiang University, Hangzhou, China
{yuanyuanyy, lijuan18, li.songze, huajunsir, zhang.wen}@zju.edu.cn

***Abstract*—Triple Set Prediction (TSP) aims to discover missing facts in Knowledge Graphs (KGs) from scratch, which is more challenging and realistic than conventional KG Completion (KGC) tasks (e.g., link prediction) that rely on partially known triple elements. While conventional embedding and rule-based methods lack semantic flexibility and require frequent retraining, Large Language Models (LLMs) offer a transformative paradigm for TSP with their advanced semantic reasoning capabilities, training-free adaptability, and ability to harness external knowledge beyond graph facts. However, direct application of LLMs to TSP is hindered by unguided discovery and factual hallucinations. To address these challenges, our key insight is to bridge the semantic-symbolic gap by transforming LLMs from unconstrained generators into rule-guided reasoners. We propose SMuRT, a Structured Multi-Stage Rule-guided LLM framework that implements a "plan-retrieve-validate" pipeline. SMuRT synergizes a Hybrid Rule Curation Engine, Structured Task Planning, Retrieval-augmented Prediction, and Dual-stage Validation to enforce logical grounding. This architecture allows LLMs to capture latent semantic patterns while remaining anchored in verifiable graph evidence, effectively mitigating hallucinations. Experiments on Wiki79k and CFamily benchmarks demonstrate that SMuRT achieves state-of-the-art performance for LLM-based TSP, effectively merging generative reasoning with specialized structural inference.**

***Index Terms*—Knowledge Graph, Knowledge Graph Completion, Triple Set Prediction, Large Language Models**

I. Introduction

Knowledge Graphs (KGs) represent real-world facts as structured triples (h, r, t), serving as foundational infrastructure for Web technologies such as search engines [1], question answering [2], and recommendation systems [3]. Despite their utility, KGs are inherently incomplete [4]–[7]. While conventional Knowledge Graph Completion (KGC) (e.g. link prediction) infers missing elements when part of a triple is known (like $(h, r, ?)$ and $(h, ?, t)$), this setting is too restrictive for real-world scenarios where missing knowledge is entirely unknown. Thus, Zhang et al. [7] introduced Triple Set Prediction (TSP), which aims to discover a set of missing triples from scratch based solely on the existing KG.

Current TSP methodologies [7] generally fall into three categories: rule-based (e.g., RuleTensor-TSP), embedding-based (e.g., KGE-TSP), and hybrid frameworks (e.g., GPHT). While pioneering, these methods rely heavily on the KG's intrinsic structural patterns and often require intensive retraining for new domains. Recent advances in LLMs offer a transformative paradigm for TSP. Leveraging vast pre-trained knowledge, LLMs generalize to data-scarce domains such as TSP without additional task-specific training, offering a distinct advantage over traditional data-dependent methods. More importantly, LLMs provide superior semantic understanding, capturing nuanced contexts that are often lost when entities are compressed into low-dimensional embedding vectors and enabling the mining of complex reasoning patterns.

However, applying LLMs to TSP remains challenging: (1) *Unguided Discovery*: Identifying missing facts in vast potential space of $|\mathcal{E}|^2 \times |\mathcal{R}|$ is needle-in-a-haystack. Lacking topological guidance, LLMs are prone to generate aimlessly between unrelated entities. (2) *Logical Inconsistency*: LLMs lack explicit mechanisms to enforce reasoning with the inherent structural constraints of KGs, yielding plausible but logically flawed results. (3) *Calibration Deficit*: As black-box models, LLMs lack reliable internal metrics to evaluate the validity of generated triples, hindering error tracing and filtering.

To address these challenges, we propose a **S**tructured **Mu**lti-Stage **R**ule-guided LLM framework for **T**riple set prediction (SMuRT). SMuRT implements a deliberate, multi-stage pipeline that synergizes rule-guided planning with retrieval-augmented reasoning. It consists of four key modules: (1) Hybrid Rule Curation Engine employs a dual-pass filtering strategy to select high-quality rules that satisfy both statistical significance and semantic reasonableness, effectively pruning potential space into a computationally tractable range. (2) Structured Task Planner decomposes the massive, complex prediction task into manageable reasoning steps, transforming open-ended prediction into executable operations. (3) Retrieval-augmented Triple Predictor grounds the reasoning process by integrating KG retrieval, ensuring predicted triples are supported by factual evidence. (4) Triple Validator-Decider applies a two-stage verification process that uses a multi-faceted scoring function and an LLM-based decider to filter out invalid outputs and provide interpretable decision traces.

Experimental results on Wiki79k and CFamily datasets [7] demonstrate that SMuRT significantly outperforms naive LLM applications and achieves competitive performance against state-of-the-art baselines, confirming that our multi-stage framework effectively harnesses LLMs' semantic capabilities while maintaining factual consistency and structural rigor.

The main contributions of this work are:

- We introduce SMuRT, a novel structured framework for TSP that integrates rule-guided planning and retrieval-augmented

*Corresponding author.

generation to ensure faithful, verifiable outputs.
- We design a task-planning mechanism that decomposes the complex TSP task into interpretable reasoning steps, significantly improving the controllability of the prediction.
- We develop a comprehensive multi-stage validation mechanism that effectively filters invalid predictions, enhancing both the precision and quality of the final triple set.

II. Related Work

KGC aims to infer missing facts within KGs. Traditional research primarily focuses on link prediction, which assumes one or two elements of a triple are known. These methods generally fall into three categories: Translation-based methods, such as TransE [4], regard a relation as a translation from the head entity to the tail entity; Semantic matching-based methods, such as DistMult [8] and ComplEx [5], measure the plausibility of triples by calculating the semantic similarity scores of entities and relations; Neural network-based methods employ GNNs [9] or CNNs [10] for KG completion. Additionally, instance completion methods, for example, RETA [6], predict relation-tail (r-t) pairs for a specific head entity using RETA-Filter and RETA-Grader components.

However, TSP represents a more challenging and realistic task where no elements of the missing triples are provided [7], which means the whole triple requires joint inference without any prior elements specified. While the pioneering GPHT framework [7] employs graph partitioning and entity-relation modeling for TSP, it requires extensive training and fails to exploit the rich textual semantics of entities. This limitation motivates the exploration of LLMs, which are training-free and possess superior semantic reasoning capabilities.

Recently, LLMs have shown prominent performance in KGC, particularly in link prediction. Some works consider re-ranking with LLMs, for example, KICGPT [11] uses an in-context learning strategy to reorder the top m candidate entities from an existing KGC method. KC-GenRe [12] tackles mismatch, misordering, and omission issues via a knowledge-constrained generative re-ranking method. FtG [13] presents using the filter-then-generate paradigm and formulates the KGC task into a multiple-choice question format. It devises an ego-graph prompt and introduces a structure-text adapter in instruction-tuning. DIFT [14] leverages discrimination instructions to finetune LLMs, where the instruction is constructed by a truncated sampling method. For the more open-ended TSP task, Yuan et al. [15] establish a rigorous evaluation framework LTSP to quantify both the capabilities and inherent limitations of LLMs when applied to the TSP task, including rule mining based on LLM, graph partitioning, and triple set prediction.

III. Preliminary

Knowledge Graph (KG) is defined as $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$, where $\mathcal{E}, \mathcal{R}$, and $\mathcal{T} = \{(h, r, t)|h, t \in \mathcal{E}, r \in \mathcal{R}\}$ denote the sets of entities, relations, and factual triples, respectively.

Triple Set Prediction (TSP) [7] aims to identify missing but correct triples within a given KG $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$. Once missing triples of $\mathcal{G}$ are predicted, denoted as $\mathcal{T}_{predict}$, $\mathcal{T}_{predict}$ is compared to the true missing triple set $\mathcal{T}_{missing} = \{(h, r, t)|h, t \in \mathcal{E}, r \in \mathcal{R}, (h, r, t) \notin \mathcal{T}\}$. $\mathcal{T}_{missing}$ could be considered the test set. We abbreviate the triplet in $\mathcal{G}$ as $\mathcal{T}_{\mathcal{G}}$.

Path-based Logical Rules refer to inference patterns extensively investigated in KG reasoning research [16], [17]. A rule p of length K is formulated as follows: $p : r_h(X, Y) \leftarrow r_1(X, Z_1) \wedge \cdots \wedge r_K(Z_{K-1}, Y)$, where the rule head $r_h(X, Y)$ denotes a derivable conclusion, and the rule body $r_1(X, Z_1) \wedge \cdots \wedge r_K(Z_{K-1}, Y)$ represents the conjunction of premises through intermediate entities $\{Z_1, \ldots, Z_{K-1}\}$. For brevity, we denote it as $p : r_h \leftarrow r_1 \wedge \cdots \wedge r_K$.

IV. Method

A. Overview

As illustrated in Figure 1, SMuRT comprises four components to bridge LLM reasoning with KG evidence: (1) Hybrid Rule Curation Engine: Discovers high-quality logical rules. (2) Structured Task Planner: Decomposes rules into executable steps. (3) Retrieval-augmented Triple Set Predictor: Generates predictions grounded in KG facts. (4) Triple Validator-Decider: Refines results via metric-based and LLM-driven verification.

B. Hybrid Rule Curation Engine

To address the challenge of ensuring logical consistency in rule-based reasoning, we introduce the Hybrid Rule Curation Engine. This module is designed to automatically generate and refine a high-quality set of logical rules by synergizing a classic rule mining algorithm for broad candidate discovery with an LLM-driven two-stage filtering process. Unlike purely statistical methods that may retain statistically prominent but logically flawed rules, our approach integrates semantic plausibility with empirical evidence to construct a robust rule set.

*1) **Initial Candidate Generation via AMIE+**:* We first employ AMIE+ [18] to mine Horn rules from $\mathcal{G}$. AMIE+ operates by treating rule mining as a series of SPARQL-like queries against the KG. It starts with short rules of length one and iteratively expands them by adding relations to the rule body, effectively exploring the rule space. To maintain efficiency, it prunes unpromising branches during the search. Specifically, given $\mathcal{G}$, AMIE+ outputs a set of path-based logical rules with their statistical metrics $\mathcal{R}_p = \{(p_1 : m_1), (p_2 : m_2), ..., (p_n : m_n)\}$, where m_i represents metrics. Three commonly used metrics are defined as [18], [19]:

Support quantifies the rule's prevalence by counting the number of entity pairs satisfying the rule: $Sup(p) = \#(X, Y) : \exists r_1(X, Z_1) \wedge \cdots \wedge r_L(Z_{L-1}, Y) \wedge r_h(X, Y) \in \mathcal{G}$.

Head Coverage normalizes support with the total number of head relation r_h facts, showing the proportion of existing r_h facts implied by the rule: $HC(p) = \frac{Sup(p)}{\#(X,Y):r_h(X,Y)\in\mathcal{G}}$.

PCA Confidence addresses incompleteness of KGs under the Partial Completeness Assumption (PCA). It is defined as the ratio of the number of facts that satisfy the rule p and the times the rule body is satisfied in incomplete KGs: $D(p) = \#(X, Y) : \exists r_1(X, Z_1) \wedge \cdots \wedge r_L(Z_{L-1}, \hat{Y}) \wedge r_h(X, \hat{Y}) \in \mathcal{G}$ and $PCA_{conf}(p) = \frac{Sup(p)}{D(p)}$.

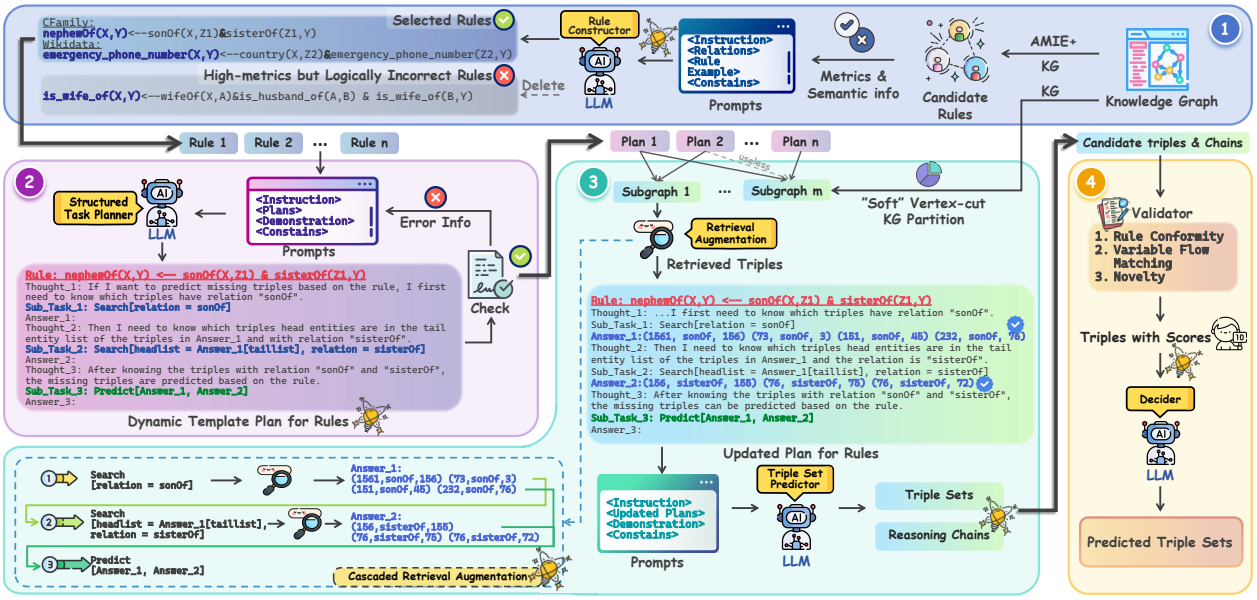

Fig. 1. Overview of our framework, including four parts: 1) Hybrid Rule Curation Engine, 2) Structured Task Planner with LLMs, 3) Retrieval-augmented LLM-based Triple Set Predictor, 4) Triple Validator-Decider.

*2) **Two-Stage LLM-based Rule Refinement**:* While statistical mining identifies frequent patterns, it often captures spurious correlations lacking causal logic. For example, the rule $playsFor(X, Z) \leftarrow bornIn(X, Y) \wedge hasTeam(Y, Z)$ may arise simply because many players are born in cities with major teams. However, presenting both metrics and rules to an LLM in a single pass can cause *anchoring bias*, where semantic judgment is disproportionately influenced by high statistical scores. To achieve independent semantic-statistical verification, we propose a coarse-to-fine refinement strategy.

This module employs a hierarchical pipeline to distill the candidate set $\mathcal{R}_p$ into a high-quality subset $\mathcal{R}_p^{llm}$:

- **Input:** The initial candidate set $\mathcal{R}_p = \{(p_i, m_i)\}_{i=1}^{n}$.
- **Stage 1 (Lightweight Quantitative Filtering):** To optimize efficiency, the LLM first acts as a coarse-grained filter. It performs a rapid scan of the rules and their metrics to prune those with marginal statistical support or obvious real-world irrelevance. This step significantly reduces the candidate pool size from N to K ($K \ll N$), minimizing the token cost for subsequent deep reasoning:
$$\mathcal{R}_p^{(1)} = \mathrm{LLM}_{\mathrm{filter}}\left(\mathcal{R}_p\right). \tag{1}$$
- **Stage 2 (Blind Semantic Ranking):** The remaining K rules are stripped of metrics to form $\mathcal{P}^{(1)}$. By withholding statistical scores, we force LLM to perform an unbiased "blind" assessment focusing strictly on *inferential soundness*. The LLM then ranks rules based on intrinsic logic:
$$\mathcal{P}^{(2)} = \mathrm{LLM}_{\mathrm{rank}}\left(\mathcal{P}^{(1)}\right). \tag{2}$$
- **Output:** The top-ranked rules in $\mathcal{P}^{(2)}$ are re-mapped to their original metrics, yielding the refined set $\mathcal{R}_p^{llm} \subseteq \mathcal{R}_p$.

Our filter-then-rank paradigm uses Stage 1 to prune noisy candidates, reducing computational overhead for expensive reasoning in Stage 2. Further, blind evaluation prevents statistical anchoring, ensuring the final rules possess both empirical support and intrinsic logical soundness.

C. Structured Task Planner with LLMs

While symbolic engines are efficient for exact matching, they are inherently semantically blind, failing on missing links or synonymous relations. We employ LLMs to capture these *latent semantic patterns* that symbolic logic overlooks. To prevent hallucinations and error propagation, our Planner acts as a semantic-to-symbolic bridge, decomposing rules into executable atomic operations instead of black-box inference, thereby merging semantic flexibility with symbolic precision.

We transforms each curated rule $p : r_h \leftarrow r_1 \wedge \cdots \wedge r_K$ into a structured execution plan $\mathcal{P}lan = \{\Phi_1, \ldots, \Phi_{K+1}\}$. Each step Φ_i is formulated as $\Phi_i = \langle \mathcal{T}_i, \mathcal{S}_i, \mathcal{A}_i \rangle$, where:

- $\mathcal{T}_i$ (*Thought*): The reasoning thought that decomposes the reasoning task through natural language.
- $\mathcal{S}_i$ (*Sub-task*): An atomic, verifiable operation (e.g., `Search`, `Predict`) that constrains LLM to follow KG's schema.
- $\mathcal{A}_i$ (*Answer*): The specific entities or triples retrieved or generated, serving as the state for the next step.

To further prevent error compounding, we utilize an iterative self-correction mechanism:

$$\mathcal{P}lan^{(t+1)} = \mathrm{LLM}(Ins_1 \oplus Dem_1 \oplus \mathcal{C}_1 \oplus \mathcal{E}^{(t)}), \mathcal{E}^{(t)} = \mathcal{F}(\mathcal{P}lan^{(t)}). \tag{3}$$

The refined plan $\mathcal{P}lan^{(t+1)}$ concatenates instructions Ins_1, few-shot demonstrations Dem_1, schema constraints $\mathcal{C}_1$, and error feedback $\mathcal{E}^{(t)}$ from symbolic validation $\mathcal{F}$, which identifies structural or logical flaws. This iterative process terminates when constraints are satisfied, mitigating hallucination and error propagation, yielding plans for each rule in $\mathcal{R}_p^{llm}$.

D. Retrieval-augmented LLM-based Triple Set Predictor

Directly applying LLMs to large-scale KGs leads to two primary issues: computational infeasibility due to context

TABLE I
COMPONENTS OF THE SCORING FUNCTION $\mathcal{S}(tri_p)$.

i	Component	Condition C_i	w_i
1	Rule Conformity ($\mathcal{S}_{\text{rule}}$)	Predicted triples adheres to relations of rules	0.2
2	Var-Flow Matching ($\mathcal{S}_{\text{var}}$)	Inference chain conforms to variable flow in rules	0.6
3	Novelty ($\mathcal{S}_{\text{nov}}$)	$tri_p \notin \mathcal{G}$ (novel prediction)	0.2

TABLE II
STATISTICS OF THE DATASETS IN EXPERIMENTS.

Datasets	#Entity	#Relation	#Triple	#$\mathcal{T}_{\mathcal{G}}$	#$\mathcal{T}_{missing}$	Assumption
Wiki79k	7983	85	79213	72876	15843	RS-POWA
CFamily	2378	12	22986	21147	4598	CWA

window limits, and hallucinations where LLMs generate ungrounded triples. Our approach utilizes graph partitioning to localize reasoning within manageable subgraphs and leverages cascaded retrieval to ground inference steps in verifiable facts.

- **Graph Partitioning:** We employ a "soft" vertex-cut method [7] to divide the KG $\mathcal{G}$ into overlapping subgraphs $\mathcal{G}_{part} = \{\mathcal{G}_1, \ldots, \mathcal{G}_k\}$. This phase groups entities and their L-hop neighbors, reducing the search space while preserving the structural connectivity necessary for multi-hop reasoning.
- **Cascaded Retrieval Augmentation:** For each `Search` operation in a plan, a unified retrieval function $\sigma(r, H_{\text{req}}, T_{\text{req}})$ queries the subgraph. The retrieval is cascaded as it uses tail entities from the previous step as head entity anchors (e.g.,$\texttt{Search}[headlist = \mathcal{A}_1[taillist], rel = sisterOf]$) according to the rule. To maintain efficiency, we utilize multi-threading for parallel subgraph traversal.
- **Chain-of-Thought Prediction:** The LLM synthesizes retrieved facts into new triples using a dynamic prompt:
$$Prompt = Ins_2 \oplus Dem_2 \oplus \mathcal{C}_2, \tag{4}$$
where Ins_2 defines the task, Dem_2 provides multi-granularity demonstrations, and $\mathcal{C}_2$ enforces semantic constraints. The LLM is explicitly required to output predicted triples alongside reasoning chains $\Lambda = \{\lambda_1, \ldots, \lambda_u\}$, which provides interpretable evidence for subsequent evaluation.

Grounding LLM synthesis in verified triples eliminates hallucinations. Graph partitioning and explicit reasoning chains ensure scalability and transparent, evidence-linked auditability.

E. Triple Validator-Decider

While the predictor generates reasoning chains, it cannot inherently distinguish between high-confidence inferences and spurious derivations that violate global KG constraints or replicate existing facts. The Triple Validator-Decider is introduced to provide a rigorous, two-stage verification that filters predictions by synthesizing quantitative metric scoring with LLM-based qualitative judgment. This ensures that the final output triples are not only logically consistent with the rules but also provide genuine novelty to the KG.

- **Metric-based Validation:** For each reasoning chain λ and its predicted triple tri_p, the validator computes a multi-faceted score:
$$\mathcal{S}(tri_p) = \mathcal{S}(\lambda, \rho, \mathcal{G}) = \sum_{i=1}^{3} \omega_i \cdot \mathcal{S}_i(\lambda, \rho, \mathcal{G}), \tag{5}$$
where $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3$ represent binary indicators for rule relation matching ($\mathcal{S}_{rule}$), entity variable alignment ($\mathcal{S}_{var}$), and triple novelty ($\mathcal{S}_{nov}$), respectively. A score of 1 indicates the chain satisfies all logical and novelty conditions. The conditions and weights are detailed in Table I.
- **LLM-based Decision:** The LLM acts as the final decider by taking the normalized scores, the associated logical rules, and current KG constraints as context. By employing LLM as a decider, we eliminate the need for a manual threshold and enable dynamic adaptive evaluation of reasoning quality.

V. EXPERIMENTS

A. Settings

*1) **Datasets and setting**:* We evaluate our framework on **CFamily** and **Wiki79k** [7]. CFamily is a rule-augmented relatively complete dataset used to assess performance under the Closed-World Assumption (CWA). Wiki79k, sourced from Wikidata, represents incomplete real-world knowledge and is used to test under Relation Similarity-based Partial-Open-World Assumption (RS-POWA). Statistical details are shown in Table II. For LLM versions, we abbreviate `GPT-3.5-turbo`, `GPT-4o`, `deepseek-r1`, and `deepseek-v3` as GPT3.5, GPT4o, DSR1, DSV3, respectively. The curated rule counts for Wiki79k are 298 (GPT4o) and 204 (DSV3), and for CFamily are 96 (GPT4o), 97 (DSV3), and 70 (DSR1). We also evaluate 25% and 50% rule subsets to analyze performance scaling and sensitivity to rule density.

*2) **Metrics**:* We utilize three specialized TSP metrics tailored to the world assumptions (WA). The recognized positive ($\mathcal{T}^{WA+}_{predict}$) and negative ($\mathcal{T}^{WA-}_{predict}$) triples are defined as:

- **Under CWA:**
$$\mathcal{T}^{CWA+}_{\text{predict}} = \mathcal{T}_{\text{predict}} \cap \mathcal{T}_{\text{missing}}, \ \mathcal{T}^{CWA-}_{\text{predict}} = \mathcal{T}_{\text{predict}} - \mathcal{T}^{CWA+}_{\text{predict}}, \quad \mathcal{T}^{CWA}_{\text{predict}} = \mathcal{T}^{CWA+}_{\text{predict}} \cup \mathcal{T}^{CWA-}_{\text{predict}} = \mathcal{T}_{\text{predict}}. \tag{6}$$
- **Under RS-POWA:**
$$\mathcal{T}^{POWA+}_{predict} = \mathcal{T}_{predict} \cap \mathcal{T}_{missing}, \tag{7}$$
$$\mathcal{T}^{POWA-}_{predict} = \{(h,r,t) | (h,r,t) \in \mathcal{T}_{predict}, (h,r,t) \notin \mathcal{T}_{missing}, \exists r' \in \mathcal{R} \ (h,r',t) \in (\mathcal{T}_{\mathcal{G}} \cap \mathcal{T}_{missing}) \wedge sim(r,r') < \theta\}, \tag{8}$$
$$\mathcal{T}^{POWA}_{predict} = \mathcal{T}^{POWA+}_{predict} \cup \mathcal{T}^{POWA-}_{predict}, \tag{9}$$
where $\theta = 0.8$ and $sim(r,r') = \max\left(\frac{|\mathcal{P}_r \cap \mathcal{P}_{r'}|}{|\mathcal{P}_r|}, \frac{|\mathcal{P}_r \cap \mathcal{P}_{r'}|}{|\mathcal{P}_{r'}|}\right)$ calculates relation similarity. $\mathcal{P}_r$ is the entity pairs set(e_1, e_2) with relation r, i.e. $(e_1, r, e_2) \in \{\mathcal{T}_{\mathcal{G}} \cup \mathcal{T}_{missing}\}$.

Based on these, we compute Joint Precision ($JPrecision$) to assess classification purity, Squared Test Recall ($STRecall$), and their harmonic mean $F_{TSP} = \frac{2 \times STRecall \times JPrecision}{STRecall + JPrecision}$:
$$JPrecision = \frac{1}{2}\left(\frac{|\mathcal{T}^{WA+}_{predict}|}{|\mathcal{T}^{WA}_{predict}|} + \frac{|\mathcal{T}^{WA+}_{predict}|}{|\mathcal{T}_{predict}|}\right), \ STRecall = \sqrt{\frac{|\mathcal{T}^{WA+}_{predict}|}{|\mathcal{T}_{missing}|}}. \tag{10}$$

*3) **Baselines**:* We compare SMuRT against four existing baselines: (1) **RuleTensor-TSP** [7]: a rule-based approach using tensor computation and logical rule mining. (2) **KGE-TSP** [7]: an embedding-based method scoring triples via HAKE and PairRE with dynamic thresholds. (3) **GPHT** [7]: a hybrid framework combining graph partitioning and meta-learning with KGE-based relation inference. (4) **LTSP** [15]: an LLM-based system that mines and applies logical rules through direct prompting.

TABLE III
TSP RESULTS ON WIKI79K (RS-POWA) AND CFAMILY (CWA) DATASETS.

Class	Models	Wiki79k: Number of Triples			Wiki79k: Metrics			Models	CFamily: Number of Triples			CFamily: Metrics		
		$\mathcal{T}_{predict}$	$\mathcal{T}_{predict}^{POWA}$	$\mathcal{T}_{predict}^{POWA+}$	$JPrecision$	$STRecall$	F_{TSP}		$\mathcal{T}_{predict}$	$\mathcal{T}_{predict}^{CWA}$	$\mathcal{T}_{predict}^{CWA+}$	$JPrecision$	$STRecall$	F_{TSP}
Non-LLM	RuleTensor-TSP	9188	1319	1292	**0.560**	0.128	0.208	RuleTensor-TSP	911	911	572	0.628	0.158	0.252
	HAKE-TSP	125	119	30	0.246	0.044	0.075	HAKE-TSP	3186	3186	1788	0.561	0.624	**0.591**
	PairRE-TSP	29742	10217	2901	0.191	**0.428**	0.264	PairRE-TSP	5732	5732	1747	0.305	**0.616**	0.408
	GPHT(HAKE)	209	191	44	0.220	0.053	0.085	GPHT(HAKE)	1896	1896	1222	**0.645**	0.516	0.573
	GPHT(PairRE)	12392	5866	2018	0.253	0.357	**0.296**	GPHT(PairRE)	3739	3739	1471	0.393	0.566	0.464
LLM	LTSP(GPT3.5)	1112	791	165	0.178	0.102	0.130	LTSP(GPT3.5)	3403	3403	96	0.028	0.144	0.047
	LTSP(GPT4o)	2031	403	176	0.262	0.105	0.150	LTSP(GPT4o)	1276	1276	179	0.140	0.197	0.164
	SMuRT(GPT4o)	13450	3197	1245	0.241	**0.280**	**0.259**	**SMuRT(GPT4o)**	2358	2358	657	0.279	**0.378**	**0.321**
	SMuRT(DSV3)	6983	2459	960	**0.264**	0.246	0.255	**SMuRT(DSV3)**	2249	2249	580	0.258	0.355	0.299
								SMuRT(DSR1)	1555	1555	437	**0.281**	0.308	0.294

TABLE IV
TSP RESULTS ON WIKI79K AND CFAMILY WITH DIFFERENT SETTINGS.

Dataset	Setting		Number of Triples			Classification Metrics		
	LLM	$Rule\%$	$\mathcal{T}_{predict}$	$\mathcal{T}_{predict}^{WA}$	$\mathcal{T}_{predict}^{WA+}$	$JPrecision$	$STRecall$	F_{TSP}
Wiki79k	GPT4o	25%	4091	1057	452	**0.269**	0.169	0.208
		50%	7172	2006	711	0.227	0.212	0.219
		100%	13450	3197	1245	0.241	**0.280**	**0.259**
	DSV3	25%	3973	1574	550	0.244	0.186	0.211
		50%	6983	2459	960	0.264	0.246	0.255
		100%	12948	4883	1177	0.166	0.273	0.206
CFamily	GPT4o	25%	1941	1941	526	0.271	0.338	0.301
		50%	2158	2158	599	0.278	0.361	0.314
		100%	2358	2358	657	0.279	**0.378**	**0.321**
	DSV3	25%	739	739	227	**0.307**	0.222	0.258
		50%	1247	1247	315	0.253	0.262	0.257
		100%	2249	2249	580	0.258	0.355	0.299
	DSR1	25%	595	595	174	0.292	0.195	0.234
		50%	904	904	255	0.282	0.235	0.257
		100%	1555	1555	437	0.281	0.308	0.294

B. Main Results

*1) **Results on Wiki79k under RS-POWA**:* Table III summarizes results on Wiki79k. Among non-LLM baselines, RuleTensor-TSP achieves the highest $JPrecision$ of 0.560, while GPHT(PairRE) maintains the best balanced performance with $F_{TSP} = 0.296$. Within the LLM category, our SMuRT framework achieves state-of-the-art results: SMuRT(GPT4o) and SMuRT(DSV3) reach F_{TSP} scores of 0.259 and 0.255, respectively, significantly outperforming LTSP. This remarkable improvement highlights the effectiveness of structured rule planning and retrieval augmentation in harnessing LLMs for TSP. While not yet surpassing specialized non-LLM baselines, our approach significantly narrows the performance gap, demonstrating that rule-guided and evidence-grounded LLMs can achieve high-quality triple prediction.

*2) **Results on CFamily under CWA**:* On the CFamily dataset, KGC models like HAKE-TSP show robust structural prediction capabilities, achieving the best F_{TSP} of 0.591, while GPHT(HAKE) and PairRE-TSP lead in $JPrecision$ and $STRecall$, respectively. In contrast, naive LLM application (LTSP) struggles with factual precision, yielding a meager F_{TSP} of 0.164. However, SMuRT(GPT4o) substantially improves F_{TSP} to 0.321, early doubling the baseline performance. This confirms its advantage in preserving logical coherence when predicting new triples.

It is crucial to interpret these findings within the context of our research objectives. Our primary goal is not to immediately outperform highly specialized non-LLM methods, but rather to explore and unlock the potential of LLMs for the formidable TSP task. Generating complete novel triples from scratch poses a fundamental reasoning challenge for LLMs that traditional KGC models circumvent with mathematical scoring functions. Therefore, the ability of SMuRT to dramatically narrow the performance gap and achieve a competitive score is a significant outcome. It confirms that with a deliberate framework of rule-guided planning, retrieval, and validation, the inferential power of LLMs can be effectively harnessed for complex KGC, marking a promising advancement.

C. Impact of Rule Density and LLM Backbones

Table IV analyzes the sensitivity of SMuRT to rule scale and LLM backbones. For rule density, a consistent positive correlation exists between the number of curated rules and overall performance. On CFamily, DSR1 shows steady F_{TSP} improvement from 0.234 (25% rules) to 0.294 (100% rules), confirming that our Hybrid Rule Curation Engine maintains high rule quality, allowing recall gains from expanded rule sets without noise. For LLM backbones, GPT4o is superior on CFamily ($F_{TSP} = 0.321$), which features explicit logical dependencies and a compact entity space, underscoring its robustness in transitive and hierarchical reasoning. Conversely, on the diverse, fact-dense Wiki79k, the gap narrows, with DSV3 (0.255) competitive with GPT4o (0.259). This suggests that for complex, open-world KGs, the bottleneck shifts from reasoning capabilities to the difficulty of grounding diverse facts, diminishing the advantage of superior LLMs.

D. Ablation Study

*a) **Effect of Hybrid Rule Curation Engine**:* Table V highlights the engine's balance of semantic validity and statistical reliability. Traditional statistical filtering often falters by either discarding logically sound rules with moderate metrics $\mathcal{R}^{+}_{\text{low}}$ like Rule 3 or retaining semantically erroneous rules with high metrics $\mathcal{R}^{-}_{\text{high}}$ like Rule 9. Our approach integrates LLM-based semantic evaluation with quantitative measures to prune statistically prominent yet logically flawed rules while preserving semantically valuable ones. This mechanism enhances curated rule set robustness, especially for sparse KGs where statistical significance often deviates from logical truth.

*b) **Effect of the Proposed Modules**:* To validate SMuRT, we conduct incremental ablation studies on CFamily using GPT4o with the full set of rules mined. For clarity and conciseness, the four key modules introduced in the previous section are denoted as M_1, M_2, M_3, and M_4. Due to the architectural dependency where M_3 operates based on dynamic plans from M_2, and M_2 and M_3 rely on the rules derived from M_1, we ablate modules in a top-down manner rather than independently to isolate their impact, where w/o M_3 means replacing the retrieval-augmentation with direct LLM prediction based on M_2's plans. As shown in Table VI, the baseline

TABLE V
COMPARISON OF RULE QUALITY BEFORE AND AFTER APPLYING THE HYBRID RULE CURATION ENGINE.

Class	Index	Rule	Supp.	HC	PCA Conf.
$\mathcal{R}^{+}_{\text{low}}$	1	$is\ uncle\ of(X,Y) \leftarrow is\ brother\ of(X,A) \wedge is\ brother\ of(A,B) \wedge is\ son\ of(Y,B)$	429	0.165	0.553
	2	$is\ uncle\ of(X,Y) \leftarrow is\ brother\ of(X,A) \wedge is\ wife\ of(A,B) \wedge is\ father\ of(B,Y)$	112	0.043	0.580
	3	$is\ son\ of(X,Y) \leftarrow is\ father\ of(A,X) \wedge is\ father\ of(A,B) \wedge is\ mother\ of(Y,B)$	189	0.199	0.600
	4	$is\ aunt\ of(X,Y) \leftarrow is\ sister\ of(X,A) \wedge is\ brother\ of(A,B) \wedge is\ daughter\ of(Y,B)$	353	0.142	0.629
	5	$is\ uncle\ of(X,Y) \leftarrow is\ brother\ of(X,A) \wedge is\ mother\ of(A,B) \wedge is\ brother\ of(B,Y)$	310	0.120	0.662
$\mathcal{R}^{-}_{\text{high}}$	6	$is\ daughter\ of(X,Y) \leftarrow is\ daughter\ of(X,A) \wedge is\ mother\ of(A,B) \wedge is\ mother\ of(Y,B)$	320	0.453	1.000
	7	$is\ husband\ of(X,Y) \leftarrow is\ father\ of(X,A) \wedge is\ son\ of(A,B) \wedge is\ husband\ of(B,Y)$	152	0.357	1.000
	8	$is\ wife\ of(X,Y) \leftarrow is\ mother\ of(X,A) \wedge is\ son\ of(A,B) \wedge is\ wife\ of(B,Y)$	141	0.324	1.000
	9	$is\ wife\ of(X,Y) \leftarrow is\ wife\ of(X,A) \wedge is\ husband\ of(A,B) \wedge is\ wife\ of(B,Y)$	368	0.846	0.995
	10	$is\ daughter\ of(X,Y) \leftarrow is\ mother\ of(Y,X)$	272	0.385	0.880

TABLE VI
ABLATION STUDY OF SMURT.

	$JPrecision$	$STRecall$	F_{TSP}
SMuRT	0.279	**0.378**	**0.321**
w/o M_3	0.266	0.307	0.285
w/o M_2, M_3	**0.365**	0.185	0.245
w/o M_2, M_3, M_4	0.119	0.364	0.179

(w/o M_2, M_3, M_4) performs direct LLM reasoning using only rules from M_1, yielding a low F_{TSP} of 0.179. Adding M_4 (w/o M_2, M_3) significantly spikes precision to 0.365 but sacrifices recall, as the validator filters noise but lacks structured guidance. Incorporating M_2 (w/o M_3) improves $STRecall$ to 0.307, confirming the effectiveness of structured task decomposition. Finally, the complete model integrates M_3 to ground predictions in factual evidence, achieving the optimal F_{TSP} of 0.321. These results demonstrate that synergistic integration of all modules is essential for balancing precision and recall.

VI. CONCLUSION

This paper presents a novel structured multi-stage LLM framework for triple set prediction that addresses the critical challenge of KG incompleteness. By integrating LLM capabilities with structured reasoning approaches, our framework can effectively identify missing triple sets in KGs. Experimental results on Wiki79k and CFamily datasets demonstrate the effectiveness of our approach in identifying missing triple sets in KGs, validating the effectiveness of our approach by combining automated rule mining, structured task planning, retrieval-augmented prediction, and comprehensive validation.

ACKNOWLEDGMENT

This work is funded by National Natural Science Foundation of China (No. NSFC62306276), Yongjiang Talent Introduction Programme (2022A-238-G), and Fundamental Research Funds for the Central Universities (226-2023-00138).

REFERENCES

[1] C. Rudnik, T. Ehrhart, O. Ferret, D. Teyssou, R. Troncy, and X. Tannier, "Searching news articles using an event knowledge graph leveraged by wikidata," in *Companion proceedings of the 2019 world wide web conference*, 2019, pp. 1232–1239.

[2] M. Yasunaga, H. Ren, A. Bosselut, P. Liang, and J. Leskovec, "Qa-gnn: Reasoning with language models and knowledge graphs for question answering," *arXiv preprint arXiv:2104.06378*, 2021.

[3] C.-M. Wong, F. Feng, W. Zhang, C.-M. Vong, H. Chen, Y. Zhang, P. He, H. Chen, K. Zhao, and H. Chen, "Improving conversational recommender system by pretraining billion-scale knowledge graph," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 2607–2612.

[4] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *Advances in neural information processing systems*, vol. 26, 2013.

[5] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *ICML*, ser. JMLR Workshop and Conference Proceedings, vol. 48. JMLR.org, 2016, pp. 2071–2080.

[6] P. Rosso, D. Yang, N. Ostapuk, and P. Cudré-Mauroux, "RETA: A schema-aware, end-to-end solution for instance completion in knowledge graphs," in *WWW*. ACM / IW3C2, 2021, pp. 845–856.

[7] W. Zhang, Y. Xu, P. Ye, Z. Huang, Z. Xu, J. Chen, J. Z. Pan, and H. Chen, "Start from zero: Triple set prediction for automatic knowledge graph completion," *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[8] B. Yang, W. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *ICLR (Poster)*, 2015.

[9] M. S. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *ESWC*, ser. Lecture Notes in Computer Science, vol. 10843. Springer, 2018, pp. 593–607.

[10] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *AAAI*. AAAI Press, 2018, pp. 1811–1818.

[11] Y. Wei, Q. Huang, Y. Zhang, and J. T. Kwok, "KICGPT: large language model with knowledge in context for knowledge graph completion," in *EMNLP (Findings)*. Association for Computational Linguistics, 2023, pp. 8667–8683.

[12] Y. Wang, M. Hu, Z. Huang, D. Li, D. Yang, and X. Lu, "Kc-genre: A knowledge-constrained generative re-ranking method based on large language models for knowledge graph completion," in *LREC/COLING*. ELRA and ICCL, 2024, pp. 9668–9680.

[13] B. Liu, J. Zhang, F. Lin, C. Yang, and M. Peng, "Filter-then-generate: Large language models with structure-text adapter for knowledge graph completion," in *COLING*. Association for Computational Linguistics, 2025, pp. 11 181–11 195.

[14] Y. Liu, X. Tian, Z. Sun, and W. Hu, "Finetuning generative large language models with discrimination instructions for knowledge graph completion," in *ISWC (1)*, ser. Lecture Notes in Computer Science, vol. 15231. Springer, 2024, pp. 199–217.

[15] Y. Yuan, Y. Xu, and W. Zhang, "Is large language model good at triple set prediction? an empirical study," in *2024 IEEE International Conference on Knowledge Graph (ICKG)*. IEEE, 2024, pp. 470–476.

[16] D. Stepanova, M. H. Gad-Elrab, and V. T. Ho, "Rule induction and reasoning over knowledge graphs," *Reasoning Web. Learning, Uncertainty, Streaming, and Scalability: 14th International Summer School 2018, Esch-sur-Alzette, Luxembourg, September 22–26, 2018, Tutorial Lectures 14*, pp. 142–172, 2018.

[17] S. Khan and M. Shaheen, "Wisrule: First cognitive algorithm of wise association rule mining," *Journal of Information Science*, vol. 50, no. 4, pp. 874–893, 2024.

[18] L. Galárraga, C. Teflioudi, K. Hose, and F. M. Suchanek, "Fast rule mining in ontological knowledge bases with amie +," *The VLDB Journal*, vol. 24, no. 6, pp. 707–730, 2015.

[19] L. Luo, J. Ju, B. Xiong, Y.-F. Li, G. Haffari, and S. Pan, "Chatrule: Mining logical rules with large language models for knowledge graph reasoning," in *Pacific-asia conference on knowledge discovery and data mining*. Springer, 2025, pp. 314–325.