KGV-Agent: Schema-Aware Planning and Hybrid Knowledge Toolset for Reliable Knowledge Graph Triple Verification

Anonymous ACL submission

Abstract

Knowledge Graphs (KGs) serve as a critical foundation for AI systems, yet their automated construction inevitably introduces noise, compromising data trustworthiness. Existing triple verification methods, relying on graph embeddings or pre-trained language models, often suffer from single-source bias—ignoring either internal structural constraints or external semantic context—and adhere to a static inference paradigm. Consequently, they struggle to verify complex or long-tail facts and lack interpretability. To address these limitations, we propose KGV-Agent, a training-free autonomous agent that reframes triple verification as a dynamic process of strategic planning, active investigation, and evidential reasoning. Specifically, KGV-Agent mitigates reasoning instability and cold-start issues via a Memory-Augmented Mechanism and Schema-Aware Strategic Planning. It then executes an enhanced ReAct loop equipped with a Hybrid Knowledge Toolset, dynamically fusing internal structural logic with external textual evidence for robust cross-verification. Empirical results across multiple datasets demonstrate that KGV-Agent establishes new state-of-the-art performance, while providing transparent, fact-based evidence chains for every judgment.

1 Introduction

Knowledge Graphs, which encapsulate human knowledge in the form of structured triples (h, r, t), serve as the critical cornerstone for Question Answering systems (Lan et al., 2022), Recommender Systems (Guo et al., 2020), and mitigating hallucinations in Large Language Models (LLMs) (Agrawal et al., 2024; Liu et al., 2025). Despite the colossal scale of existing KGs (e.g., Freebase, Wikidata), their construction predominantly relies on automated extraction techniques, which inevitably introduce noise and errors. Such erroneous triples not only compromise data credibility (Bian, 2025) but also severely undermine the reliability of downstream applications through error propagation (Liu et al., 2024). Consequently, achieving accurate and interpretable Triple Verification has emerged as a paramount and urgent research challenge (QUDUS et al., 2023).

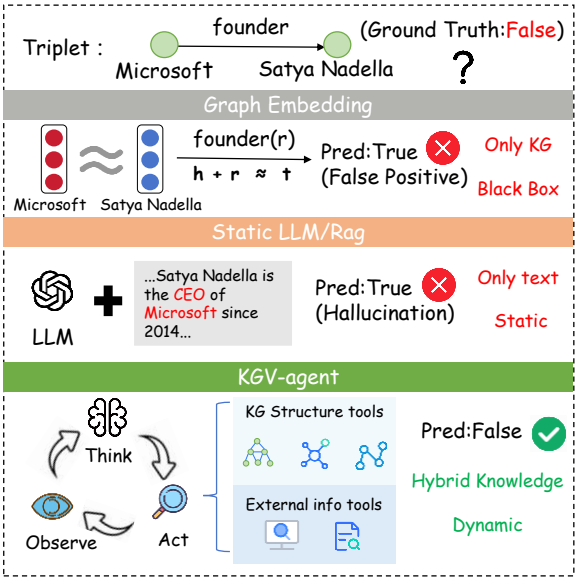

Figure 1: Comparisons between previous methods and our proposed KGV-Agent. Traditional graph embedding and static LLM methods are limited by a single perspective and static reasoning, leading to verification failures (False Positives). However, KGV-Agent successfully identifies errors by coordinating internal structure and external text within a dynamic reasoning loop.

Traditionally, triple verification and link prediction tasks have relied on Graph Embedding-based methods (e.g., TransE (Bordes et al., 2013), RotatE (Sun et al., 2019)), which map structural information into vector spaces via geometric transformations to compute plausibility, yet they inherently ignore the rich textual semantics of the open world. Conversely, Pre-trained Language Model (PLM)-based methods (e.g., KG-BERT (Yao et al., 2019)) utilize models like BERT for semantic encoding and classification but overlook the strong

logical constraints and structural information (e.g., Schema, paths) inherent in Knowledge Graphs (KGs), leading to a dichotomy of information sources (Pan et al., 2024). Although with the advent of powerful Large Language Models (LLMs), Retrieval-Augmented Generation (RAG) methods have introduced external evidence (Shami et al., 2025), they predominantly adhere to an inference paradigm of "single-step reasoning" and "static retrieval", failing to conduct the "multi-step investigation" required for complex logic (Sun et al., 2023). Furthermore, these methods often operate as black boxes, lacking interpretability (Luo et al., 2024). As shown in Figure 1, regarding the erroneous triple (Microsoft, Founder, Satya Nadella), existing methods fail due to high vector similarity or semantic hallucination, whereas KGV-Agent accurately detects the error via a dynamic "active investigation and evidential reasoning" process.

To address these limitations, drawing inspiration from research on LLM Agents, we propose KGV-Agent, a verification agent driven by schema-aware planning and hybrid knowledge fusion. We reframe the triple verification task from a traditional "classification problem" to a cognitive process of "Strategic Planning – Dynamic Investigation" – Evidential Reasoning."

Specifically, KGV-Agent simulates the cognitive workflow of a human expert conducting fact-checking. First, via Memory-Augmented Mechanism, the agent utilizes analogical reasoning to derive guidance from historical successful paths and generates a global plan based on triple schema characteristics. Subsequently, the agent enters a Think-Act-Observation loop, deploying a Hybrid Toolset that encompasses both internal KG structures and external knowledge to dynamically gather evidence and refine its perspective. This mechanism breaks information silos, achieving complementarity between structural constraints and external unstructured semantic evidence.

Empirical evaluations on the FB15k-237 and Wikidata5M benchmarks demonstrate that KGV-Agent significantly outperforms state-of-the-art (SOTA) baselines, achieving substantial accuracy gains of 4.2% and 12.9%, respectively. As a generic "plug-and-play" solution, it not only exhibits superior verification precision but also ensures interpretability by providing transparent, fact-based evidence chains for every judgment.

To summarize, the main contributions of this paper are as follows: (1) We propose KGV-Agent, a novel framework that reframes triple verification as dynamic reasoning. By integrating Schema-Aware Planning with a Memory-Augmented mechanism, it effectively mitigates reasoning instability. (2) We design a Hybrid Knowledge Toolset that synergizes internal KG structural logic with external unstructured semantics, resolving single-source bias through robust knowledge complementarity. (3) Results show that our method achieves state-of-the-art performance across multiple benchmarks, providing transparent evidence chains that validate its superiority and interpretability in handling complex, long-tail tasks.

2 KGV-Agent

In this section, we first provide a formal definition of the Knowledge Graph Triple Verification task. Subsequently, we present an overview of the KGV-Agent framework. Finally, we elaborate on the three core components of our framework: trajectory-based initialization, the iterative reasoning mechanism, and the hybrid knowledge toolset.

2.1 Problem Definition

A Knowledge Graph is denoted as $G = (E, R, T)$, where E is the set of entities, R is the set of relations, and $T \subseteq E \times R \times E$ represents the set of known triples (Hogan et al., 2021). Each triple is represented as (h, r, t), indicating that a head entity $h \in E$ is connected to a tail entity $t \in E$ via a relation $r \in R$. The task of this study is to verify the plausibility of a given query triple $\tau = (h, r, t)$. Formally, our objective is to construct an agent $\mathcal{A}$ that interacts with an environment comprising both the internal graph structure G and external world knowledge W, to output a binary verdict $y \in \{\text{True}, \text{False}\}$ alongside a corresponding chain of evidence $\mathcal{E}$. The agent $\mathcal{A}$:$\mathcal{A}(h, r, t, G, W)$ is formulated as follows:

$$\mathcal{A}(h, r, t, G, W) = \begin{cases} \text{True}, & \text{if } (h, r, t) \in T_{\text{true}} \\ \text{False}, & \text{if } (h, r, t) \notin T_{\text{true}} \end{cases}$$

2.2 Framework Overview

We formulate the triple verification task as a sequential decision-making process driven by hybrid knowledge fusion. As illustrated in Figure 2, KGV-Agent is built upon Large Language Models (LLMs) and adopts the "Think-Act-Observation" ReAct paradigm. To address the cold-start dilemma in complex logic handling and bridge the gap between structural and semantic information, the in-

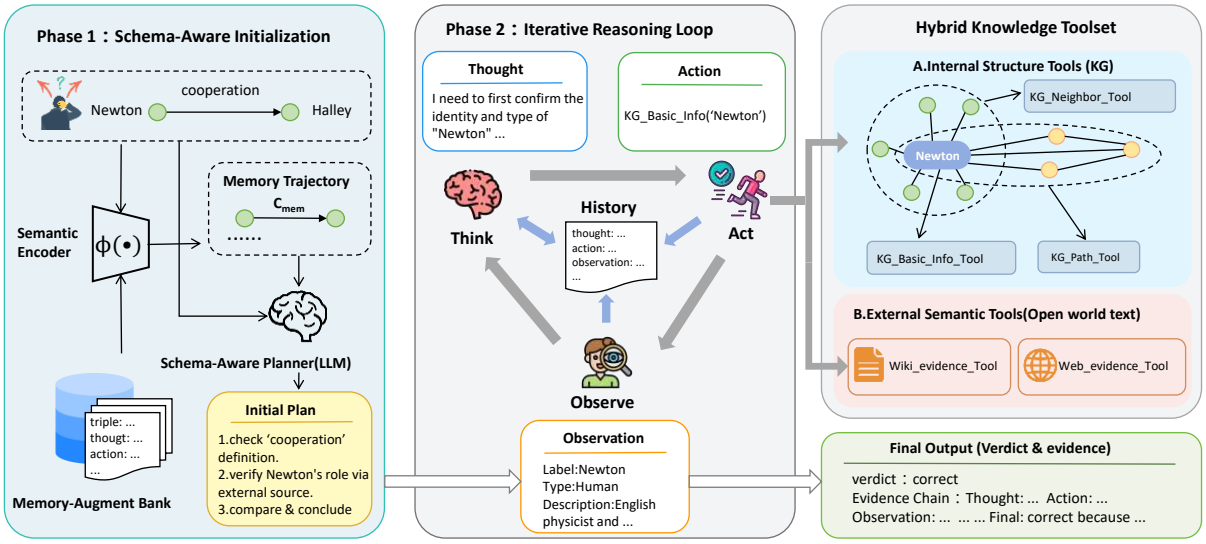

Figure 2: The overall architecture of KGV-Agent. The framework operates in three synergistic phases: (Phase 1) Schema-Aware Initialization: The agent retrieves historical reasoning trajectories via a semantic encoder and generates an Initial Plan based on schema constraints to address the cold-start problem. (Phase 2) Iterative Reasoning Loop: Guided by the plan, the agent executes an enhanced ReAct loop, dynamically adjusting its strategy based on real-time observations. During the act process in (Phase 2), the agent primarily utilizes a Hybrid Knowledge Toolset: A suite of complementary tools is employed to probe both Internal KG Structure (e.g., Neighbors, Paths) and External Semantics (e.g., Wiki, Web Search), achieving deep fusion of heterogeneous knowledge.

ference process of KGV-Agent is decomposed into three synergistic phases: (1) **Schema-Aware Initialization:** By analyzing the structural and semantic characteristics of (h, r, t), the agent leverages a **Memory-augmented mechanism** to retrieve analogous historical reasoning trajectories and generates an initial strategic plan to prime the reasoning process. (2) **Iterative Reasoning Loop:** The agent executes a dynamic ReAct cycle, continuously refining investigation strategies and verifying evidence through a recursive "Think-Act-Observation" loop. (3) **Hybrid Knowledge Acquisition:** A toolset comprising five atomic capabilities is employed to synergistically acquire both internal structural constraints and external semantic evidence.

2.3 Schema-Aware Initialization

2.3.1 Memory-Augmented Mechanism

To empower the agent with expert-level reasoning logic and ensure consistent guidance throughout the investigation process, we discard the traditional fixed few-shot strategy in favor of a holistic Memory-Augmented Mechanism. First, we construct an external memory bank $\mathcal{M} = \{(x_i, traj_i)\}_{i=1}^{N}$ rich in expert knowledge, curated through a rigorous process of agent testing followed by manual screening and refinement. To ensure robustness, this repository covers 50 common triple relation types and comprises 200 high-quality reasoning trajectories from wikidata. Each trajectory $traj_i$ meticulously records the complete inference process (i.e., the "Think-Act-Observation" sequence) executed for the specific verification task x_i.Trajectory examples can be found in Appendix C. For an incoming query triple τ, we employ a semantic encoder $\phi(\cdot)$ to retrieve the top-k most similar trajectories, denoted as $\mathcal{C}_{mem}$, from $\mathcal{M}$:

$$\mathcal{C}_{mem} = \{traj_j \mid j \in \text{Top-}k(\text{sim}(\phi(\tau), \phi(x_i)))\}$$

Crucially, $\mathcal{C}_{mem}$ serves not merely as a cold-start prompt but as a Persistent Reference within the agent's working memory. During reasoning, the agent continuously consults these demonstrations to dynamically calibrate its investigation direction and optimize tool selection strategies, thereby preventing deviation during long-chain inference.

2.3.2 Schema-Aware Strategic Planning

To circumvent aimless exploration within complex graph structures, the model first performs a meta-cognitive step. Synthesizing its understanding of tool functionalities, the schema characteristics of the triple (e.g., relation domain and range), and the retrieved memory $\mathcal{C}_{mem}$, the agent generates an

initial verification plan $\mathcal{P}_{init}$:

$$\mathcal{P}_{init} = \text{LLM}(\tau, \mathcal{C}_{mem}, \text{Instruction}_{plan})$$

This plan explicates the logical steps for verification (e.g., "Step 1: Check entity definitions to verify type constraints; Step 2: Retrieve 1-hop neighbors to verify direct connectivity; Step 3: Search external news to confirm event timing"). Crucially, we inject $\mathcal{P}_{init}$ into the agent's working memory as the Initial Observation (Obs_0). This not only provides macro-level guidance but also acts as an "Anchor," effectively mitigating Goal Drifting—a common issue in long-chain reasoning.

2.4 Reasoning with ReAct Paradigm

We enhance the standard ReAct paradigm by introducing a Plan Adherence and Correction Mechanism. The reasoning process is modeled as a sequence of discrete time steps $t = 1, \ldots, T$.

At step t, the agent operates on the current global context $\mathcal{H}_t$, which comprises four distinct components: the system instruction I, the retrieved expert demonstrations $\mathcal{C}_{mem}$, the initial strategic plan $\mathcal{P}_{init}$, and the accumulated interaction history. Formally, $\mathcal{H}_t = (I, \mathcal{C}_{mem}, \mathcal{P}_{init}, h_{0:t-1})$.As outlined in Algorithm 1, the agent executes the following recursive sub-processes:

Think The agent derives the subsequent strategy Th_t based on the combination of the initial plan $\mathcal{P}_{init}$ and the accumulated observations Obs. This process ensures adherence to the original objectives while enabling dynamic plan refinement in response to evolving empirical evidence.

Act Based on the thought trace th_t, the agent selects the optimal tool a from the toolset $\mathcal{T}$ and generates the corresponding execution arguments p. This policy is formally represented as:

$$ac_t = (a, p) \sim \pi_\theta(ac_t \mid \mathcal{H}_t, th_t)$$

where π_θ denotes the action policy distribution parameterized by the LLM.

Observe The environment (i.e., the Knowledge Graph interface or external search engine) receives and executes the action ac_t, yielding the observation result $obs_t = Env(ac_t)$. This new observation is appended to the interaction history, updating the global context to $\mathcal{H}_{t+1}$.

This loop persists until the agent issues a termination action Finish[Answer] or reaches the maximum step count T_{max}. To prevent infinite loops,

Algorithm 1 KGV-Agent Reasoning Algorithm

Require: Triple τ; Knowledge Graph $\mathcal{G}$; Toolset $\mathcal{T}$; Memory $\mathcal{M}$
Ensure: Verdict y, Evidence $\mathcal{E}$
$\mathcal{C}_{mem} \leftarrow \text{Retrieve}(\tau, \mathcal{M})$
$\mathcal{P}_{init} \leftarrow \text{LLM}_{plan}(\tau, \mathcal{C}_{mem})$
$\mathcal{H} \leftarrow \emptyset$; $t \leftarrow 0$
while $t < T_{max}$ **do**
  $th_t, ac_t \sim \pi_\theta(\cdot \mid \tau, \mathcal{P}_{init}, \mathcal{C}_{mem}, \mathcal{H})$
  if ac_t is **Finish then**
    return Result parsed from ac_t
  end if
  $obs_t \leftarrow \text{ExecuteTool}(ac_t, \mathcal{G}, \text{Web})$
  $\mathcal{H} \leftarrow \mathcal{H} \cup \{th_t, ac_t, obs_t\}$
  $t \leftarrow t + 1$
end while
return $\text{LLM}_{judge}(\mathcal{H})$

we implement a Mandatory Judgment Mechanism at $t = T_{max}$, compelling the agent to perform probabilistic reasoning based on currently available information and output a final verdict. This mechanism ensures an effective synergy between structured macro-planning (guided by $\mathcal{P}_{init}$) and dynamic micro-execution (driven by ReAct). The complete prompt templates for planning, reasoning, and judgment are detailed in Appendix B.

2.5 Hybrid Knowledge Toolset

To bridge the gap between internal structural information and external textual semantics, we design a hybrid toolset $\mathcal{T}$ comprising five atomic capabilities, categorized into two distinct groups:

A. KG Internal Structure Tools These tools are designed to leverage the inherent logical constraints and topological structures within the KG.

- **KG Definition Tool:** Taking an entity or relation as input, this tool queries the underlying metadata of the KG,, including entity Labels, Descriptions, and Aliases, as well as relation semantics, Domains, and Ranges. This serves as the foundation for the verification logic chain, ensuring clear definitions prior to reasoning.
- **KG Neighbor Tool:** Given an input entity e and a target relation r, this tool retrieves the Top-k 1-hop neighbor triples of e that are most semantically relevant to r based on vector similarity. This facilitates the agent's understanding of the entity's background context in a specific relational direction.
- **KG Path Tool:** Taking an entity pair (h, t) as input, this tool employs graph algorithms to retrieve 1-Hop, 2-Hop, and 3-Hop paths from h

to t. This provides critical structural evidence for verifying implicit relationships (e.g., inferring a "Grandfather" relation via intermediate "Father" links). The Multi-hop path connections expression is as follows:

$$\text{Paths}(h,t) = \{(h,r_1,x_1),\ldots,(x_{n-1},r_n,t) \mid \exists x_1,\ldots,x_n \in \text{KG},\ n \leq 3\}$$

B. External Semantic Tools These tools are designed to utilize open-world unstructured text to mitigate the inherent sparsity of KGs.

- **Wiki Evidence Tool:** This tool supports a dual-mode strategy: (1) **Entity Mode:** Queries a single entity to fetch its abstract and key attributes; (2) **Co-occurrence Mode:** Queries an entity pair (h,t) to retrieve sentences where both entities co-occur, providing authoritative encyclopedic evidence.
- **Web Evidence Tool:** Taking a natural language query as input, this tool utilizes a search engine API to return Top-k web snippets. It serves as a final fallback verification mechanism, addressing long-tail knowledge or time-sensitive facts

$$\text{Paths}(h,t) = \{(h,r_1,x_1),\ldots,(x_{n-1},r_n,t) \mid \exists x_1,\ldots,x_n \in \text{KG},\ n \leq 3\}$$

Notably, we implement a multi-level retrieval mechanism across all tools. By integrating keyword-based exact matching with vector-based semantic retrieval (e.g., utilizing semantic vectors to recall fuzzy entities or re-rank search results), this hybrid strategy overcomes the coverage limitations of traditional keyword search while mitigating the precision bias of pure vector retrieval. This design ensures the efficient complementarity of heterogeneous knowledge, realizing a robust cross-verification process where structure guides semantics, and semantics interprets structure.

Table 1: Details for each dataset

Dataset	Entity	Relation	Triplet	Source
Wikidata5M-Ind	4,594,458	822	20,510,107	Wikidata
FB15K-237	14,541	237	310,116	Freebase

3 Experiments and Results

3.1 Datasets and Negative Sampling

We evaluate KGV-Agent on two benchmarks with distinct challenges: FB15k-237 (Toutanova and Chen, 2015), a reasoning-intensive dataset with inverse relations removed, requires multi-hop inference. Wikidata5M-Inductive (Wang et al., 2021), derived from Wikidata, is introduced as an inductive benchmark with disjoint entity sets, characterized by extreme sparsity and long-tail distributions. Detailed information is provided in Table 1.

Type-Constrained Negative Sampling Since the original datasets consist exclusively of positive triples, we employ negative sampling to construct erroneous triples. To avoid generating trivial negatives (e.g., "Obama born in Banana") and to foster a high-difficulty testing scenario, we implement a Type-Constrained Negative Sampling strategy. Specifically, given a positive triple $\tau = (h,r,t)$, we generate a negative set τ^- by replacing the head or tail entity with an entity e' that shares the same fine-grained type constraint:

$$\tau^- = \{(h,r,e') \mid e' \in \mathcal{E}, e' \neq t, T(e') = T(t)\},$$

where $\mathrm{T}(e)$ denotes the semantic type. These hard negatives (e.g., replacing a "place of birth" with another "city" rather than a "fruit") force the model to perform deep semantic discrimination.

3.2 Baselines

To demonstrate the superiority of KGV-Agent, we compare it against representative models covering three mainstream paradigms:

- **Embedding-based KGE**:TransE (Bordes et al., 2013), DistMult (Yang et al., 2015), and RotatE (Sun et al., 2019),which rely on structural embedding information.
- **PLM-based Methods**: KG-BERT (Yao et al., 2019) and SimKGC (Wang et al., 2022a), which utilizes pre-trained parameter.
- **LLM-based Approaches**: (1) **Zero-shot Inference** using GPT-3.5-Turbo (Ouyang et al., 2022), GPT-4o (Achiam et al., 2023), and Qwen3-max (Yang et al., 2025) relying on parametric knowledge; and (2) **Inference-Augmented Frameworks** including Chain-of-Thought(CoT) (Wei et al., 2022), Self-Consistency(s-c) (Wang et al., 2022b), and the verification-specific KGValidator (Boylan et al., 2024).

Results for all baseline models are cited from their original papers or reproduced using official open-source implementations under identical settings.

Table 2: The result of our method and other baseline methods on FB15k-237 and Wikidata5M-Inductive benchmarks. Entries marked with '-' denote results not reported in original papers or where official implementations were unavailable for faithful reproduction.The best results are highlighted in bold.

Method	FB15k-237				Wikidata5M-Ind			
	Accuracy	**F1**	**Precision**	**Recall**	**Accuracy**	**F1**	**Precision**	**Recall**
Embedding-based								
TransE	66.4	73.7	60.5	94.2	49.6	49.2	49.6	48.8
DistMult	61.2	70.9	56.7	94.4	50.2	50.7	51.4	50.1
RotatE	66.6	74.4	60.3	**97.0**	50.8	51.3	50.5	52.2
PLM-based								
KG-BERT	66.3	72.0	62.2	85.5	-	-	-	-
SimKGC	69.8	72.5	66.6	79.4	78.4	78.9	76.9	81.1
LLM(zero-shot)								
GPT-3.5-turbo	69.7	73.4	65.4	83.4	73.8	72.6	76.1	69.4
GPT-4o	77.3	73.9	86.8	64.4	77.6	72.6	93.1	59.6
Qwen3-max	76.6	73.7	84.3	65.4	74.5	67.7	92.4	53.4
LLM(inference-Augmented)								
Qwen3-max(CoT)	81.4	80.1	86.0	75.0	80.5	77.4	92.0	66.8
Qwen3-max(S-C)	80.8	79.5	85.2	74.6	80.8	77.9	91.4	68.0
KGValidator	83.0	81.0	-	-	-	-	-	-
Our Methods								
KGV-Agent	**87.2**	**86.6**	**91.2**	82.4	**93.7**	**93.4**	**98.7**	**88.6**

In KGV-Agent, we employ Qwen3-max (Yang et al., 2025) as the LLM driving the agent. The temperature is set to 0, and the maximum interaction round is defined as $T_{max} = 10$. Detailed parameter settings for both the baseline and KGV-Agent can be found in Appendix A.

3.3 Evaluation metrics

We report Accuracy and F1-score. For KGE baselines outputting continuous scores (e.g., TransE), we search for the optimal threshold δ on the validation set to binarize predictions. For LLM-based and Agent methods, we parse their explicit text responses. To ensure rigorous evaluation, any refusal to answer or formatting error is strictly penalized as an incorrect prediction.

3.4 Main Results

As shown in Table 2, KGV-Agent outperforms all baseline models across all evaluation metrics on both the FB15k-237 and Wikidata5M-Ind datasets, achieving new State-of-the-Art results. The detailed comparative analysis is as follows:

First, compared to traditional graph embedding models, KGV-Agent improves accuracy and F1 score by an average of 22.5% and 13.6% respectively on FB15k-237, and by 43.5% and 43.0% on Wikidata5M-Ind. This is because KGV-Agent integrates external tools, overcoming the limitations of models that rely solely on internal triplet structural information.

Second, compared to Pre-trained Language Model (PLM) baselines, our method demonstrates a significant advantage. We observe average improvements of 19.2% in accuracy and 14.4% in F1 on FB15k-237, along with average gains of 15.3% (Accuracy) and 14.5% (F1) on Wikidata5M-Ind. This performance gain is primarily attributed to KGV-Agent transcending the Closed World Assumption. Instead of relying on static triplet text, our framework actively retrieves information and reasons via the Agent architecture to form effective multi-hop reasoning chains.

Furthermore, in comparison with general LLM-based reasoning methods, KGV-Agent achieves average improvements of 9.1% in accuracy and 9.7% in F1 on FB15k-237, as well as 16.3% and 19.8% on Wikidata5M-Ind. These results indicate that static reasoning relying on parametric knowledge or simple external context is prone to hallucination.

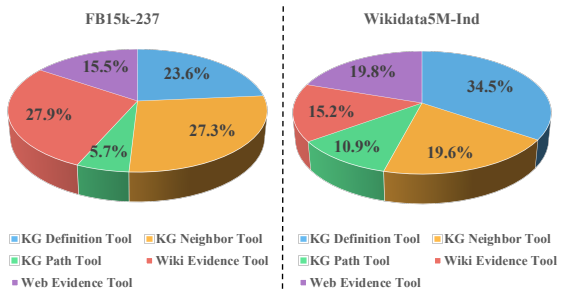

Figure 3: Statistics of tool usage on FB15k-237 and Wikidata5M-Ind

In contrast, our Schema-Aware Planning mechanism effectively mitigates noise through dynamic retrieval and constrained reasoning paths, substantially enhancing verification performance.

Notably, KGV-Agent achieves exceptional Precision while maintaining a high F1 score, reaching a remarkable 98.7% on Wikidata5M-Ind. This implies an extremely high confidence level in positive predictions, highlighting the immense potential of our model for application in high-stakes domains such as medicine and law, where stability is paramount. In addition, we conducted a Case Study and Error Analysis in Appendix D

3.5 Tool Usage Analysis

Figure 3 shows the frequency distribution of tool utilization for correctly predicted samples by KGV-Agent on the FB15k-237 and Wikidata5M-Ind datasets. Statistically, the average number of tool invocations per triple verification is 9.8 on FB15k-237 and 6.6 on Wikidata5M-Ind. The overall distribution reveals that all pre-defined tools are actively employed with notable diversity. This confirms that a single knowledge source—whether solely structural or textual—is insufficient for complex verification tasks, necessitating the fusion of multi-source information for optimal performance.

Specifically, the distinct characteristics of the datasets drive a shift in tool preferences. Given the prevalence of long-tail entities in Wikidata5M-Ind, the Agent most frequently invokes the KG Definition Tool (34.5%) to acquire fundamental semantic descriptions. In contrast, FB15k-237 involves more complex multi-hop reasoning, prompting the Agent to prioritize the Wikipedia Evidence Tool (27.9%) and KG Neighbor Tool (27.3%) to aggregate external textual evidence and internal neighborhood structures. This pattern not only reflects KGV-Agent's capability to flexibly schedule multi-source knowledge according to verification needs but also strongly demonstrates the effective, context-aware guidance provided by our Memory enhancement mechanism and Schema-Aware planning strategy.

3.6 Efficiency Analysis

To optimize efficiency, we implemented a multi-threaded concurrent framework (50 threads), maintaining system stability with average inference latencies of 1.86 s on FB15k-237 and 1.26 s on Wikidata5M-Ind. While marginally higher than lightweight embedding models, KGV-Agent distinguishes itself with a Plug-and-Play capability. Unlike baselines requiring extensive fine-tuning, our training-free approach significantly reduces deployment barriers and computational costs, offering superior cost-effectiveness in real-world scenarios.

3.7 Ablation Study

To investigate the effectiveness of key components within KGV-Agent, we designed the following three variants for ablation studies:

- **w/o Schema-Aware Init**: Removes the Schema-Aware Initialization module, which includes the Memory enhancement mechanism and Schema-Aware planning.
- **w/o KG Tools**: Removes all KG-related tools, including the KG Definition Tool,KG Neighbor Tool, and KG Path Tool.
- **w/o External Tools**: Removes all external semantic tools, including the Wikipedia Evidence Tool and Web Evidence Tool.

Table 3 presents the ablation results. Overall, removing any component leads to significant performance degradation, validating the necessity of the full framework. The detailed analysis is as follows:

Impact of Schema-Aware Init. Removing this module results in accuracy drops of 14.6% on Wikidata5M-Ind and 9.3% on FB15k-237. This indicates that without the planning provided by schema information and the guidance of the Memory enhancement mechanism, the agent is prone to getting trapped in invalid reasoning loops or being misled by irrelevant noise.

Table 3: Ablation Experiments Results. The values in red parentheses indicate the performance drop compared to the full KGV-Agent model.

Method	FB15k-237		Wikidata5M-Ind	
	Accuracy	F1	Accuracy	F1
KGV-Agent	**87.2**	**86.6**	**93.7**	**93.4**
w/o Schema-Aware Init	77.9 (↓ 9.3%)	76.7 (↓ 9.9%)	79.1 (↓ 14.6%)	76.7 (↓ 16.7%)
w/o KG Tools	78.4 (↓ 8.8%)	77.2 (↓ 9.4%)	83.7 (↓ 10.0%)	81.9 (↓ 11.5%)
w/o External Tools	66.8 (↓ 20.4%)	53.6 (↓ 33.0%)	80.5 (↓ 13.2%)	76.1 (↓ 17.3%)

Impact of KG Tools Removing KG tools causes an accuracy loss of approximately 10% across both datasets. This confirms that internal structures and path information provide necessary contextual constraints for external evidence, effectively preventing the LLM from hallucinating during reasoning.

Impact of External Tools Removing external tools leads to accuracy declines of 13.2% on Wikidata5M-Ind and 20.4% on FB15k-237. This demonstrates that relying solely on closed-world structural information is insufficient for complex triple verification.

4 Related Work

Triple Verification and Completion Traditional approaches primarily fall into two categories: embedding-based models (e.g., TransE (Bordes et al., 2013), TransH (Wang et al., 2014), DistMult (Yang et al., 2015), RotatE (Sun et al., 2019)) and PLM-based methods (e.g., KG-BERT (Yao et al., 2019), SimKGC (Wang et al., 2022a)). While these methods have advanced the field, they often suffer from single-source limitations: embedding models are constrained by the closed-world assumption (Shi and Weninger, 2018), while PLM-based approaches tend to overlook global structural constraints. Unlike these single-modality methods, KGV-Agent transcends such limitations by dynamically synthesizing internal structural logic with external open-world evidence for deep reasoning.

LLM-based Fact Checking LLMs have been extensively adapted for fact-checking tasks, such as error identification (Self-CheckGPT (Manakul et al., 2023)), granular evaluation (FActScore (Min et al., 2023)), and KG quality assessment (KG-Validator (Boylan et al., 2024)). These methods generally adhere to a static "single-pass" inference paradigm, limiting their efficacy on complex, long-tail knowledge. KGV-Agent addresses this by reframing verification as a dynamic process of strategic planning, active investigation, and evidential reasoning via an autonomous framework.

Autonomous Agents Agents represent a paradigm shift from static reasoning towards dynamic reasoning via tool use (Schick et al., 2023). While the ReAct paradigm (Yao et al., 2022) has successfully empowered textual fact-checking (e.g., FactAgent (Li et al., 2024), SAFE (Wei et al., 2024)), existing agents remain predominantly parameter- and text-centric, lacking the capability to perceive KG topological structures or exploit explicit logical constraints (Jiang et al., 2023). To bridge this gap, KGV-Agent introduces a hybrid toolset and schema-aware planning, endowing the agent with structure-aware capabilities and achieving a robust synergy between structural constraints and semantic retrieval.

5 Conclusion

In this paper, we propose KGV-Agent, a framework that reframes triple verification as a dynamic "planning-retrieval-reasoning" process. By integrating Schema-Aware Planning with a Memory-Augmented Mechanism, we mitigate reasoning instability and achieve deep synergy between internal structural logic and external semantics via a Hybrid Knowledge Toolset. Empirical results on FB15k-237 and Wikidata5M-Ind confirm that KGV-Agent significantly outperforms SOTA baselines in a training-free manner. Crucially, our approach provides transparent evidence chains alongside high accuracy, showing superior robustness and interpretability for complex verification tasks.

Limitation

Despite the KGV-Agent's superior performance and transparency in triple verification, certain limitations remain for future exploration. First,

our framework currently focuses on atomic-level triples, whereas real-world facts often manifest as complex claims or long-form documents. Extending KGV-Agent to claim-level fact-checking represents a key direction for our future work. Second, while multi-threading mitigates latency, the computational overhead of multi-step reasoning and tool invocation remains higher than that of static models. Future research will focus on balancing verification precision and inference efficiency through knowledge distillation or policy optimization.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Garima Agrawal, Tharindu Kumarage, Zeyad Alghamdi, and Huan Liu. 2024. Can knowledge graphs reduce hallucinations in llms?: A survey. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3947–3960.

Haonan Bian. 2025. Llm-empowered knowledge graph construction: A survey. *arXiv preprint arXiv:2510.20345*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Jack Boylan, Shashank Mangla, Dominic Thorn, Demian Gholipour Ghalandari, Parsa Ghaffari, and Chris Hokamp. 2024. Kgvalidator: a framework for automatic validation of knowledge graph construction. *arXiv preprint arXiv:2404.15923*.

Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3549–3568.

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, and 1 others. 2021. Knowledge graphs. *ACM Computing Surveys (Csur)*, 54(4):1–37.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*.

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Complex knowledge base question answering: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(11):11196–11215.

Xinyi Li, Yongfeng Zhang, and Edward C Malthouse. 2024. Large language model agent for fake news detection. *arXiv preprint arXiv:2405.01593*.

Runxuan Liu, Luobei Luobei, Jiaqi Li, Baoxin Wang, Ming Liu, Dayong Wu, Shijin Wang, and Bing Qin. 2025. Ontology-guided reverse thinking makes large language models stronger on knowledge graph question answering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15269–15284.

Xiangyu Liu, Yang Liu, and Wei Hu. 2024. Knowledge graph error detection with contrastive confidence adaption. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 8824–8831.

L Luo, YF Li, G Haffari, and S Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *ICLR 2024: The Twelfth International Conference on Learning Representations*. ICLR.

Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 9004–9017.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3580–3599.

UMAIR QUDUS, MICHAEL RÖDER, MUHAMMAD SALEEM, and AXEL-CYRILLE NGONGA. 2023. Fact checking over knowledge graphs–a survey.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023.

Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

Farzad Shami, Stefano Marchesin, and Gianmaria Silvello. 2025. Fact verification in knowledge graphs using llms. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3985–3989.

Baoxu Shi and Tim Weninger. 2018. Open-world knowledge graph completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M Ni, Heung-Yeung Shum, and Jian Guo. 2023. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. *arXiv preprint arXiv:2307.07697*.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, pages 57–66.

Liang Wang, Wei Zhao, Zhuoyu Wei, and Jingming Liu. 2022a. Simkgc: Simple contrastive knowledge graph completion with pre-trained language models. *arXiv preprint arXiv:2203.02167*.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022b. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, volume 28.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Jerry Wei, Chengrun Yang, Xinying Song, Yifeng Lu, Nathan Hu, Jie Huang, Dustin Tran, Daiyi Peng, Ruibo Liu, Da Huang, and 1 others. 2024. Long-form factuality in large language models. *Advances in Neural Information Processing Systems*, 37:80756–80827.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Bishan Yang, Scott Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the International Conference on Learning Representations (ICLR) 2015*.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Kg-bert: Bert for knowledge graph completion. *arXiv preprint arXiv:1909.03193*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

A Implementation Details

A.1 Core Agent Configuration

We employ the full version of Qwen3-max as the backbone controller for KGV-Agent, selected for its superior performance in instruction following and complex reasoning capabilities. The model is accessed via the official OpenAI Python SDK. To ensure reproducibility and eliminate randomness in generation, we set the decoding temperature to 0. The stop sequence is defined as "[Observation]" to strictly adhere to the ReAct interaction format. Furthermore, the maximum number of interaction turns is limited to $T_{max} = 10$.

A.2 Vector Retrieval and Indexing

For all vector-based retrieval tasks—including trajectory retrieval from the Memory Bank and semantic re-ranking of web snippets—we utilize the all-MiniLM-L6-v2 model for efficient text encoding. We employ the FAISS library to construct dense vector indexes, enabling high-speed similarity search.

A.3 Knowledge Environment and Tool Implementation

Knowledge Retrieval Environment. Given that the raw data for both FB15k-237 and Wikidata5M are aligned with the Wikidata knowledge base, we utilize the Wikidata API as the unified interface. Natural language queries are converted into SPARQL statements to retrieve precise structural information.

Entity/Relation Disambiguation. To map natural language terms to Wikidata IDs (PID/QID), we adopt a two-stage strategy: (1) **Keyword Matching:** We first attempt to retrieve IDs via exact keyword search; (2) **Semantic Matching:** If keyword retrieval fails, we fallback to vector-based matching using the aforementioned encoder to identify the ID with the closest semantic meaning.

Anti-Leakage Mechanism. Crucially, during the execution of the KG Neighbor Tool and KG Path Tool, we explicitly filter out the test triple $\tau = (h, r, t)$ itself from the retrieval results. This strict decontamination step prevents the agent from seeing the ground truth directly (Data Leakage), forcing it to reason based on context rather than rote memorization.

Tool Return Limits. To balance information density and context window usage, the KG tools (Neighbor and Path) return the Top-20 most relevant triples or paths, while the Web Evidence Tool (utilizing the Google Search API) returns the Top-5 text snippets.

A.4 Baseline Reproduction

To ensure a fair comparison, all baselines are reproduced using standard libraries or official implementations:

- **Embedding Methods:** TransE, DistMult, and RotatE are implemented using the OpenKE library with random initialization for embeddings.
- **PLM-based Methods:** KG-BERT and SimKGC are reproduced using their official open-source codebases.
- **LLM-based Methods:** All LLM baselines are implemented via the official OpenAI API with default parameters.

All hyperparameters for these baselines strictly follow the configurations reported in their original papers.

A.5 Evaluation metrics

We adopt four fundamental metrics widely adopted in classification tasks to quantify model performance: Accuracy, Precision, Recall, and F1-score. Their formal definitions and mathematical formulations are as follows, where: TP (true positives) denotes the count of positive instances correctly predicted; TN (true negatives) denotes the count of negative instances correctly predicted; FP (false positives) denotes the count of negative instances incorrectly predicted as positive; and FN (false negatives) denotes the count of positive instances incorrectly predicted as negative. Accuracy measures the overall proportion of correct predictions among all instances:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision quantifies the reliability of positive predictions, i.e., the proportion of predicted positives that are actually positive:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall quantifies the completeness of positive predictions, i.e., the proportion of actual positives that are correctly identified:

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1-score is the harmonic mean of Precision and Recall, balancing their trade-off to provide a single metric for overall performance:

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

B Prompt Templates

To ensure the reproducibility of KGV-Agent, we provide the full prompts used in our experiments. The **Schema-Aware Planning Prompt** (Figure 4) is used in Phase 1 to generate the initial verification strategy. The **Reasoning Instruction Prompt** (Figure 5) serves as the system instruction for the core ReAct loop in Phase 2, defining tool signatures and interaction protocols. Finally, the **Mandatory Judgment Prompt** (Figure 6) is triggered in Phase 3 if the agent exceeds the maximum step limit, forcing a probabilistic verdict.

C Memory-Augmented trajectory case

To demonstrate the quality of the expert knowledge stored in our Memory Bank, we present a complete reasoning trajectory in Figure 7. This example illustrates how KGV-Agent validates the triple **(Elon Musk, CEO, Tesla)**. The trajectory showcases the agent's systematic approach: (1) **Schema Grounding**: Verifying the precise definitions of entities and relations in the KG (Steps 1 & 2); (2) **Structural Verification**: Exploring internal connectivity within the KG (Step 3); (3) **External Cross-Validation**: seeking real-world evidence via web search when internal paths are implicit (Step 4). Such high-quality trajectories serve as few-shot demonstrations to guide the agent in handling similar complex queries.

D Case Study and Error Analysis

In this section, we provide a detailed analysis of two representative reasoning trajectories of KGV-Agent to illustrate its decision-making process and typical error modes. These cases highlight how the agent synthesizes internal knowledge graph structures with external semantic verification.

Success Case Analysis (Biological Kingdom Boundary). As shown in Figure 8, the agent correctly refuted a triple claiming a rodent species (Kemp's Thicket Rat) belongs to a plant genus (Gliricidia). The agent demonstrated high-level ontological reasoning by first identifying the disparate biological kingdoms: Animalia for the mammal and Plantae for the plant genus. By confirming the absence of structural paths in the KG and finding no contradicting external evidence, the agent successfully leveraged the principle of taxonomic hierarchy to provide a robust, interpretable rejection.

Failure Case Analysis (Temporal Contradiction). Figure 9 illustrates a failure mode related to temporal constraints. While the agent correctly identified the head entity Dat Nguyen) and his affiliation with Texas A&M football, it failed to account for the dataset's specific ground truth logic regarding time-sliced entities. The agent identified a chronological impossibility—the player was born in 1975, while the triple specified the 1930s—and thus labeled it Incorrect. However, this case was marked True in the original dataset, likely due to a misalignment between the general relationship and the specific temporal tail entity. This failure reveals the agent's tendency to prioritize strict logical consistency over potential data noise or relaxed temporal matching.

Prompt 1: Schema-Aware Planning

System Instruction:
You are a specialized Knowledge Graph Verification Strategist. Your goal is to provide a decisive Initial Impression and a Verification Plan for a given triple (Head, Relation, Tail).
[Tool Definitions]
(Note: The full tool definitions are identical to those in the Reasoning Prompt and are omitted here for brevity.)
Rule 1: Strict Initial Impression (Internal Knowledge Only)
You MUST explicitly judge the triple as [Correct] or [Incorrect] immediately based on your internal parametric knowledge.

- **Forbidden words:** You are strictly forbidden from using "Uncertain", "Maybe", "Likely", "Unclear", or "Need verification".
- **Force a decision:** If the triple contradicts common sense (e.g., a place used as a person), output [Incorrect]. If you strictly don't know, please analyze the possibility based on the entity type and triplet content to make the best guess.

Rule 2: Strategic Plan (Must use Tools)
List a few concrete steps. **Requirement:** You must mention which specific tool to use in each step.
Output Format
Please strictly follow this format:
=== Initial Impression ===
[Correct/Incorrect] Because [Reasoning...]
=== Strategic Plan ===
[Step 1 description using ToolName]
[Step 2 description using ToolName]
...
User Input:
Target Triple: {triple}

Figure 4: The prompt template for the Schema-Aware Planning phase (Phase 1).

Prompt 2: Reasoning Instruction

System:
Role Definition
You are an expert Knowledge Graph Verification Agent (KGV-Agent). Your goal is to verify the factual correctness of a given triple (Head, Relation, Tail). You must follow the "Think-Act-Observation" ReAct paradigm, synthesizing Internal Structural Information (KG) and External Semantic Information (Text) for cross-verification.
Tool Definitions
You have access to the following tools. Invoke them strictly according to their signatures:
1. Internal Structural Tools(KG)

- **KG_Basic_Info_Tool(entity: str/relation: str):** Query the schema definitions of the entity or the relation in KG (such as entity labels/aliases/descriptions, relation domain values). Usage: To understand the basic information.
- **KG_Neighbor_Tool(entity: str, relation: str):** Retrieves the top-20 target entities that are most strongly correlated with the input entity under the specified relation. Usage: To understand the local subgraph structure and co-occurrence patterns.
- **KG_Path_Tool(entity_a: str, entity_b: str):** Retrieves explicit 1-hop, 2-hop and 3-hop paths between two entities. Usage: To verify structural connectivity and explore implicit relationships (e.g., subsidiary, competitor paths).

2. External Semantic Tools

- **Wiki_Basic_Info_Tool(entity: str):** Retrieves the Wikipedia summary and key attributes. Usage: For background knowledge of a single entity.
- **Wiki_Relation_Tool(entity_a: str, entity_b: str):** Retrieves sentences explicitly mentioning both entities from Wikipedia. Usage: To find explicit textual evidence linking two entities to discover implicit relationships.
- **Web_Evidence_Tool(question: str):** Validates a natural language statement via Google Search, returning Top-5 snippets. Usage: For long-tail knowledge or as a final fallback verification. As a guarantee mechanism for external queries.

Interaction Protocol (ReAct)
You must strictly follow this loop:

- **Thought:** Analyze the current observation and the Initial Plan. What evidence is missing?
- **Action:** Select a tool. Format: ToolName(value, ...)
- **STOP:** Do NOT generate the Observation! Stop immediately after the Action. Wait for system feedback.
- **Observation:** (The system will provide the tool output here)

... (Repeat the loop) ...
Task Completion:
When sufficient evidence is gathered, output the final verdict:
Final Answer: [Correct/Incorrect] Because [Detailed explanation based on evidence]
Judgment criteria

- Please note that this verification does not consider timeliness, and any time this triplet is valid will be judged as [Correct].
- If there is clear evidence to prove/refute the triplet, judge accordingly.
- If there is no clear evidence, please make reasonable inferences based on the retrieved information. Note: Lack of evidence does not necessarily mean [Incorrect]. Please make the most likely judgment based on known information (e.g., semantic plausibility).
- When you believe there is sufficient evidence, you can output the final answer without getting stuck in a long-term loop.

User Context:
Strategic Plan: {plan}
Related Reasoning Demonstrations: {trajectory case}
Target Triple: {triple}
Execution History: ...

Figure 5: The prompt template for the Reasoning phase (Phase 2).

Prompt 3: Mandatory Judgment

SYSTEM ALERT: The verification process has reached the maximum step limit. You MUST now make a final decision.
Reference Information
Your Initial Impression (Internal Knowledge): {impression}
Original Plan: {plan}
Execution History (Evidence Collected): {history}
Decision Logic

- If the History contains clear evidence to prove or refute the triplet (e.g., explicit KG paths or Web search results), prioritize the History.
- If the History is useless (e.g., tools failed, no data found), please make the most likely judgment of the triplet based on your Initial Impression and History.
- Note: Even if History has no evidence to prove the triplet, it does not necessarily mean that the triplet is wrong. Please make the most likely judgment based on known information.
- The probability of each provided triplet being either [Correct] or [Incorrect] is 50%.

Result MUST be strictly:
Final Answer: [Correct/Incorrect] Because [Detailed explanation based on evidence or fallback knowledge]
Target Triple: {triple}

Figure 6: The mandatory judgment prompt triggered at the maximum step limit (Phase 3).

Memory Trajectory Case: (Elon Musk, CEO, Tesla)

Target Triple: "Elon Musk, CEO, Tesla"

- **Thought 1:** I need to first confirm the identity and type of "Elon Musk" to ensure we are referring to the correct person.
- **Action 1: KG_Basic_Info_Tool**(entity='Elon Musk')
- **Observation 1:**
  Entity Profile: Q317521
  Label: Elon Musk;
  Description: businessman and entrepreneur (born 1971);
  Type: human;
  Aliases: Elon Reeve Musk.
- **Thought 2:** The entity 'Elon Musk' correctly refers to the businessman. Next, I should search for the definition of the "CEO" relationship to ensure my understanding is accurate.
- **Action 2: KG_Basic_Info_Tool**(relation='CEO')
- **Observation 2:**
  Relation Profile: P169
  Label: chief executive officer;
  Description: highest-ranking corporate officer appointed as the CEO within an organization;
  Aliases: executive director, chief executive, CEO;
  Subject Constraint (Domain): None defined (Open Domain / Any);
  Object Constraint (Range); presidential term; fictional facility; facility; editorial collection; fictional organization; event; project; organization; fictional company; branch office
- **Thought 3:** The relation 'CEO' means the highest-ranking corporate officer. Next, I will verify if there is a direct or 2-hop connection between Elon Musk and Tesla regarding the "CEO" relationship.
- **Action 3: KG_Path_Tool**(entity_a='Elon Musk', entity_b='Tesla')
- **Observation 3:**
  (Elon Musk) –> [employer] –> (Tesla, Inc.)
  (Elon Musk) <– [owned by] <– (Tesla, Inc.)
  (Elon Musk) –[unmarried partner]–> (Shivon Zilis) –[employer]–> (Tesla, Inc.)
  .....
- **Thought 4:** The knowledge base indicates a direct relationship: (Elon Musk) <– [owner] <– (Tesla), suggesting that the candidate triple is likely correct. However, to ensure factual accuracy, I will conduct cross-validation using external network evidence.
- **Action 4: Web_Evidence_Tool**(question='Is Elon Musk the CEO of Tesla?')
- **Observation 4:**
  Snippet 1: "...Tesla CEO Elon Musk Gets a Long Do-Not-Tweet List From the SEC..." (Source: Barron's)
  Snippet 2: "...Elon Musk is the co-founder and CEO of Tesla..." (Source: Official Bio) .....
- **Thought 5:** Based on the current evidence, it is sufficient to prove the triplet.
- **Final Answer:** [Correct] Because the knowledge base explicitly states that Elon Musk is the employer/owner of Tesla, Inc., and this is corroborated by multiple credible web sources confirming his role as CEO.

Figure 7: A representative reasoning trajectory from the Memory Bank. This case demonstrates the "Think-Act-Observation" loop where the agent fuses KG structural info with Web evidence.

Successful Verification Trajectory

Target Triple: (kemp's thicket rat, parent taxon, gliricidia) **Ground Truth:** False **Verdict:** [Incorrect]

- **Thought 1:** I need to first confirm the biological classification and taxonomic rank of "kemp's thicket rat" to understand what kind of entity it is and whether it could logically have a "parent taxon" relationship with "gliricidia".
- **Action 1: KG_Basic_Info_Tool**(entity: "kemp's thicket rat")
- **Observation 1:** Label: Kemp's Thicket Rat; Description: species of mammal; Type: taxon; Aliases: Thamnomys kempi.
- **Thought 2:** The entity "Kemp's Thicket Rat" is confirmed as a taxon and specifically a species of mammal. Next, I need to verify what "Gliricidia" refers to.
- **Action 2: KG_Basic_Info_Tool**(entity: "gliricidia")
- **Observation 2:** Label: Gliricidia; Description: genus of plants; Type: taxon.
- **Thought 3:** Both are confirmed as taxa, but they belong to entirely different biological kingdoms: Animalia (mammal) vs. Plantae (plant genus). A parent taxon relationship cannot cross kingdom boundaries. I will check for any structural paths.
- **Action 3: KG_Path_Tool**(entity_a: "kemp's thicket rat", entity_b: "gliricidia")
- **Observation 3:** No direct,2-hop or 3-hop paths found between Q1761125 and Q12549487.
- **Thought 4:** The knowledge graph shows no structural paths, which aligns with biological expectations. I will perform a final web search to rule out any obscure claims.
- **Action 4: Web_Evidence_Tool**(question: "Is Kemp's thicket rat taxonomically related to Gliricidia?")
- **Observation 4:** No relevant webpages found (API returned empty).
- **Final Answer:** [Incorrect] Because "Kemp's thicket rat" is a mammalian species (Animalia), while "Gliricidia" is a genus of plants (Plantae). The "parent taxon" relationship strictly applies within a lineage and cannot cross kingdom boundaries. The KG confirms no structural path exists, consistent with biological classification principles.

Figure 8: Success Case: Cross-Kingdom Ontological Reasoning

Error Verification Trajectory

Target Triple: (Nguyễn Tán Đạt, member of sports team, texas a&m aggies football(1930-1939))
Ground Truth: True **Agent Prediction:** [Incorrect] **(Error)**

- **Thought 1:** I need to first verify the identity and background of "Nguyễn Tán Đạt" to determine if this person could plausibly have been associated with a U.S. college football team in the 1930s.
- **Action 1: KG_Basic_Info_Tool**(entity: "Nguyễn Tán Đạt")
- **Observation 1:** Label: Dat Nguyen; Description: All-American college football player; Aliases: Dat Tan Nguyen.
- **Thought 2:** This refers to Dat Nguyen, a modern player. I need to check the team entity "texas a&m aggies football, 1930-1939" to see if it exists as a distinct time-sliced entity.
- **Action 2: KG_Basic_Info_Tool**(entity: "texas a&m aggies football, 1930-1939")
- **Observation 2:** Label: Texas A&M Aggies football; Description: University American football team.
- **Thought 3:** The KG only contains the general team entity. I will check the player's birth date and team membership to verify the 1930s timeframe.
- **Action 3: KG_Neighbor_Tool**(entity: "Nguyễn Tán Đạt", relation: "member of sports team")
- **Observation 3:** [member of sports team] -> Texas A&M Aggies football; [date of birth] -> 1975-09-25.
- **Thought 4:** Dat Nguyen was born in 1975, making it chronologically impossible for him to have played in the 1930s. I will perform a final web search for any historical namesake.
- **Action 4: Web_Evidence_Tool**(question: "Was there a Vietnamese player named Dat Nguyen on Texas A&M in the 1930s?")
- **Observation 4:** No relevant webpages found (API returned empty).
- **Final Answer:** [Incorrect] Because Nguyễn Tán Đạt (Dat Nguyen) was born in 1975 and played for Texas A&M in the 1990s, making it chronologically impossible for him to be a member between 1930 and 1939.

Figure 9: Failure Case: Chronological Impossibility and Data Noise