Verify-in-the-Graph: Entity Disambiguation Enhancement for Complex Claim Verification with Interactive Graph Representation

Anonymous ACL submission

Abstract

Claim verification is a long-standing and challenging task that demands not only high accuracy but also explainability of the verification process. This task becomes an emerging research issue in the era of large language models (LLMs) since real-world claims are often complex, featuring intricate semantic structures or obfuscated entities. Traditional approaches typically address this by decomposing claims into sub-claims and querying a knowledge base to resolve hidden or ambiguous entities. However, the absence of effective disambiguation strategies for these entities can compromise the entire verification process. To address these challenges, we propose Verify-in-the-Graph (VeGraph), a novel framework leveraging the reasoning and comprehension abilities of LLM agents. VeGraph operates in three phases: (1) Graph Representation - an input claim is decomposed into structured triplets, forming a graph-based representation that integrates both structured and unstructured information; (2) Entity Disambiguation -VeGraph iteratively interacts with the knowledge base to resolve ambiguous entities within the graph for deeper sub-claim verification; and (3) Verification - remaining triplets are verified to complete the fact-checking process. Experiments using Meta-Llama-3-70B (instruct version) show that VeGraph achieves competitive performance compared to baselines on two benchmarks HoVer and FEVEROUS, effectively addressing claim verification challenges. Our source code and data are available for further exploitation[1].

1 Introduction

In the era of rapidly advancing large language models (LLMs), the widespread dissemination of misinformation, combined with the increasing presence of AI-generated content, has made it significantly harder for individuals to assess the reliability of information. Consequently, claim verification, leveraging advanced natural language processing (NLP) techniques to automatically determine the veracity of claims, has emerged as a critical research topic (Guo et al., 2022; Dmonte et al., 2024).

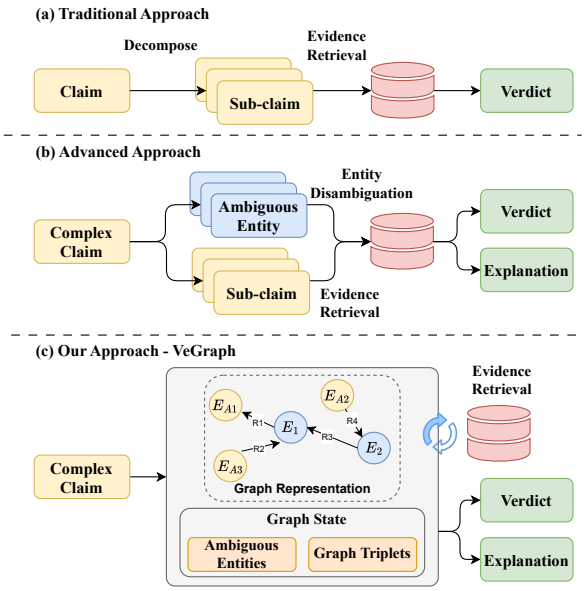

Figure 1: Conceptual analysis of previous works and VeGraph: a) Traditional approaches use IR to retrieve evidence and then verify sub-claims; b) Advanced approaches use IR to resolve ambiguous entities and then verify sub-claims; c) Our approach represents claims with graph triplets, then iteratively interacts with IR for entity disambiguation and sub-claims verification.

Traditional approaches typically begin by decomposing a given claim (e.g., at the sentence or passage level) into sub-claims, often using methods such as chain-of-thought (CoT) prompting (Wei et al., 2022a). Subsequently, each sub-claim is evaluated in a step-by-step reasoning process, incorporating knowledge sources (e.g., information retrieval systems) to determine the truthfulness of the overall claim (Krishna et al., 2022; Zhang and Gao, 2023), as shown in Figure 1(a). Recently, more advanced methods have enhanced this process by

[1]https://github.com/anonymous

incorporating multi-step reasoning to resolve ambiguous entities before verifying sub-claims (Wang and Shu, 2023; Pan et al., 2023a; Zhao et al., 2024), as illustrated in Figure 1(b). These improvements, particularly the incorporation of entity interaction, have made such methods more promising for explainable and interpretable claim verification systems.

However, despite the advances achieved by multi-step reasoning approaches, several critical challenges remain: i) *Ambiguous entity interactions*: Ambiguous entities continue to pose a major challenge to the performance of fact verification systems (Sedova et al., 2024). This issue becomes even more pronounced in multi-step reasoning, where entity disambiguation must occur across the entire verification process, rather than relying on external tools for each individual sub-claim as done in prior works; ii) *LLM-based multi-step reasoning agents*: Most prior works addressing multi-step reasoning rely on static, single-path veracity prediction, failing to take full advantage of recent multi-agent frameworks (Zhang et al., 2024).

In response to these challenges, this study introduces a controllable agent-based framework, named Verify-in-the-Graph (VeGraph), for automatic fact verification. Our approach, illustrated in Figure 1(c), consists of three interconnected stages: an LLM agent first constructs a graph-based representation by decomposing the input claim into sub-claim triplets. The agent then interacts with a knowledge base to resolve ambiguous entities in triplets, iteratively updating the graph state. Finally, the agent verifies triplets, completing the fact-checking process. Overall, the primary contributions of this work are as follows:

(1) We propose a novel multi-step reasoning approach for claim verification using an LLM agent framework with interactive graph representation (VeGraph). To the best of our knowledge, this is the first study to leverage multi-step reasoning in conjunction with an interactive entity disambiguation process to enhance claim verification performance.

(2) The proposed method, by integrating interactive graph representations with LLM agent frameworks, enhances explainability and interpretability by exploiting both structured and unstructured information—key elements for advancing multi-step reasoning tasks.

(3) We evaluate and show the effectiveness of our approach on two widely recognized benchmark datasets in this field: HoVer (Jiang et al., 2020) and FEVEROUS (Aly et al., 2021).

2 Related Work

Claim verification is a long-standing and challenging task that seeks to determine the veracity of a claim by retrieving relevant documents, selecting the most salient evidence, and making a veracity prediction. In the era of large language models (LLMs), LLM-based claim verification has evolved to generate sub-claims from input claims using the chain-of-thought (CoT) approach, and to retrieve evidence by augmenting the LLM with external knowledge sources for verification (Guo et al., 2022). ProgramFC (Pan et al., 2023b) enhances this process by leveraging in-context learning alongside the CoT method, decomposing the original claim into program-like functions to guide the verification steps. Similarly, FOLK (Wang and Shu, 2023) translates the claim into First-Order-Logic (FOL) clauses, where each predicate corresponds to a sub-claim that requires verification. FOLK then performs FOL-guided reasoning over a set of knowledge-grounded question-answer pairs to predict veracity and generate explanations, justifying its decision-making process. Furthermore, (Zhao et al., 2024) leveraging LLM Agent concept which incorporates self-reflection technique and global planning to enhance performance.

Despite the advancement of these methods, which exploit LLM reasoning capabilities to interact with external knowledge bases, they are limited to a single interaction with the knowledge base for a ambiguous entity. If the knowledge base fails to identify the requested entity in the query, the entire verification process may collapse. In light of these limitations, our proposed method similarly leverages LLM reasoning in conjunction with external knowledge retrieval systems. However, we extend this by incorporating agent-based LLM reasoning, enabling iterative interactions with the knowledge base to resolve ambiguous entities and execute multi-step reasoning for more robust and in-depth claim verification.

3 Methodology

The main target of this study is to predict the veracity of a complex input claim C through automated reasoning using an LLM agent with interpretability, incorporating both structured and unstructured information via graph representation. Figure 2 shows the architecture of our proposed framework. Specif-

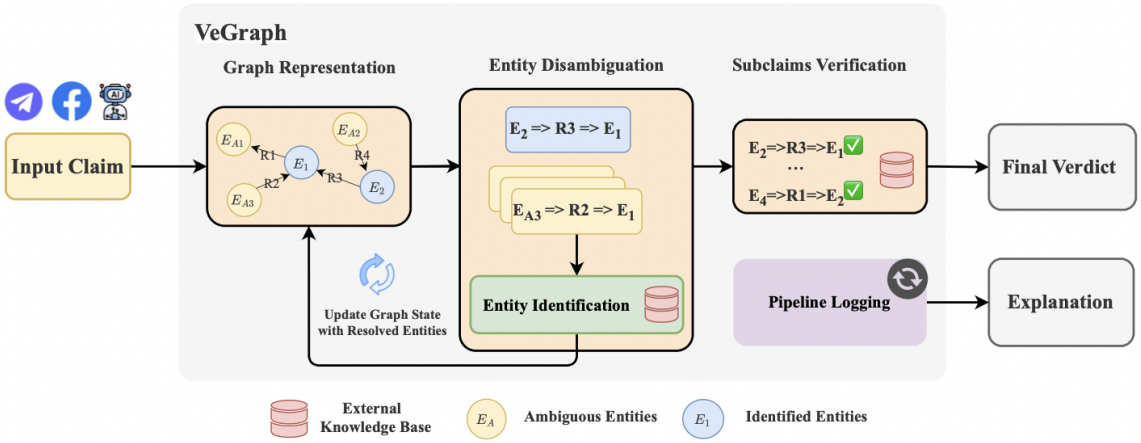

Figure 2: Three key components of VeGraph: (i) **Graph Representation**, which decomposes the complex input claim into graph triplets; (ii) **Entity Disambiguation**, ambiguous entities are resolved through iterative interactions with the knowledge base (KB); and (iii) **Sub-claim Verification**, which evaluates each triplet by delegating the verification process to the sub-claim verification function. The logging module records the whole process.

ically, VeGraph consists of three stages: (i) agent represent the claim C with graph triplets, each corresponding to a sub-claim; (ii) agent interacts with an external knowledge base to resolve ambiguous entities; and (iii) once all ambiguities are addressed, the agent verifies sub-claims corresponding to the remaining triplets. The veracity of the input claim is determined by the veracity of all the graph triplets, if all the graph triplets are verified with the information in the knowledge base then the claim C is $Supported$, if one of the triplets can not be verified then the claim C is $Refuted$. During processing through stages, the logging module records the activities of the agent for explanability.

3.1 Graph Representation

Input claims often contain complex sentence structures that challenge LLMs to grasp their semantic meaning. To address this, we transform each claim into a graph representation composed of triplets, with each triplet capturing a sub-claim within the original claim (illustrated in Figure 3). This semantic graph construction is grounded in techniques from the information extraction field, utilizing a joint approach for entity and relation extraction (Li et al., 2013; Miwa and Bansal, 2016) in an end-to-end fashion. Entities (nodes) are defined as spans of text representing objects, events, or concepts mentioned in the claim. Unlike traditional Named Entity Recognition (NER) systems, which rely on fixed categories, this approach accommodates a more diverse set of entity types. For relation extraction (edges), we apply methods from Open Information Extraction (OpenIE) (Fader et al., 2011). Instead of restricting relations to predefined categories (e.g., OWNERSHIP, LOCATED), this method extracts relations expressed in natural language, allowing for the capture of detailed document-level interactions. For instance, in a semantic graph, a relation like "is based on the life of" accurately represents the relationship between two entities within the claim.

Task: Construct a graph that captures entities and relationships from a given claim, including hidden, ambiguous or implicit entities. Only use information from the claim, do NOT repeat similar triplets in the graph and return the graph in the following format:
...
Examples:
<input_claim> Radha started her career in a 1964 Kannada film. The film was based on the life of the creator of the music form Geetam, who was born in 1484.
<guidance_for_graph_construction>
The claim mentions "a 1964 film" without specific information, so it will be marked as an ambiguous entity X_0
The claim also mentions "the creator of the music form Geetam" without specific information so it will be marked as an ambiguous entity X_1
<graph>
Radha || started career in || X_0
X_0||is a||1964 Kannada film
X_0||is based on the life of||X_1
X_1||createda music form||Geetam
X_1||was born in||1484
...
Actual claim
<input_claim> {{claim}}

Graph Representation

Figure 3: Prompt to make LLM construct the Graph Representation

Formally, in **VeGraph**, the graph construction process leverages in-context learning (Wei et al., 2022a) to prompt the LLM to generate graph

$G = \{T_1, T_2, ..., T_N\}$ consist of N triplets, each triplet $T_i = (E_{1i}, R_i, E_{2i})$ corresponds to a sub-claim extracted from the original claim C. Here, E_{1i} and E_{2i} denote the head and tail entities, respectively, while R_i captures the semantic relation between them. Complex claims often contain implicit or ambiguous entities that need to be resolved to facilitate claim verification. For example, in the claim shown in Figure 3, the entity "a 1964 Kannada film" is not explicitly named, necessitating a disambiguation process. To address this, we categorize entities into two types: explicitly stated entities are marked as standard nodes, while ambiguous entities are tagged as X_i to signal the need for further clarification. This disambiguation process of these entities, detailed in Section 3.3, ensures a comprehensive representation of claim semantics. With this graph-based representation, the LLM can more effectively capture the claim's semantic intricacies, thereby enhancing its reasoning capabilities and supporting improved performance of claim verification. (Refer to Figure 3 in Appendix for the detailed prompt)

3.2 Knowledge Base Interaction Functions

To facilitate interaction with the knowledge base in an open-book setting, we implement two core functions: *Entity Identification* and *Claim Verification*. Both functions utilize information retrieval techniques to retrieve relevant documents enabling context-aware decision-making. During execution, all the retrieved documents are recorded for thoroughness and explainability.

Entity Identification. This function acts as a question-answering module that extracts a specific entity. Formally, for a given question Q, a set of top-k relevant documents D are retrieved from the knowledge base using an information retrieval system. The question Q and the retrieved documents D are processed jointly by the LLM to identify the target entity requested in the question. This allows the system to leverage external knowledge to resolve ambiguities and produce informed answers. (Refer to Figure 11 in Appendix for the prompt)

Sub-claim Verification. The *Sub-claim Verification* function is designed to assess the truthfulness of a given claim C. Upon receiving a claim as input, the system retrieves a set of top-k documents D relevant to C from the knowledge base. These documents are then processed alongside the claim by the LLM, which determines whether the information supports or refutes the claim. The output is a binary decision—either *True* or *False*—that indicates the veracity of the sub-claim (Refer to Figure 10 in Appendix for the detailed prompt).

3.3 Entity Disambiguation Process

Following the transformation of the claim into a graph representation, the next step is identifying and resolving ambiguous entities. The disambiguation process is described in Algorithm 1 and illustrated step-by-step in Figure 4.

Algorithm 1: Entity Disambiguation

```
Input  : Claim C, Input graph G, Max iteration k
Output : Clarified graph G, Verified triplets
         VTriplets
Initialize:
  Agent Attempt Logs: logs = ∅;
  Verified Triplets: VTriplets = ∅;
Function Main(C, G, k):
    // Logic of the disambiguation process
    for i = 1 to k do
        groups = GroupTriplets(G);
        foreach (ae, g) in groups do
            GenQuesAndResEntity(ae, g)
    if Clarified(G) then
        // check if all ambiguous entities is identified
        return "Successful";
    return "Failed";
Function GenQuesAndResEntity(ae, g):
    // Agent try to generate question q to identify the
      ambiguous entity ae of the group g
    r, q, ids = GenQues(C, g, log[ae]);
    e = QA(q);
    if e ≠ None then
        VTriplets.add(ids);
        UpdateState(G, ae, e);
        // Update verified triplets and the state of the
          graph when identified a new entity
    else
        logs[ae].add((r, q));
        // Log the reasoning trace and question
          when the agent failed
Function GroupTriplets(G):
    // Group triplets for ambiguous entities
    groups = ∅;
    entities = AmbiguousEntities(G);
    foreach ae in entities do
        group = ∅;
        foreach triplet in G do
            if ae ∈ triplet then
                group.add(triplet);
        groups.add((ae, group));
    return groups;
```

Triplet Grouping. To effectively address entity ambiguities, we organize the extracted triplets from the graph G into distinct groups based on shared ambiguous entities. Each group consists of triplets containing the same ambiguous entity. For instance,

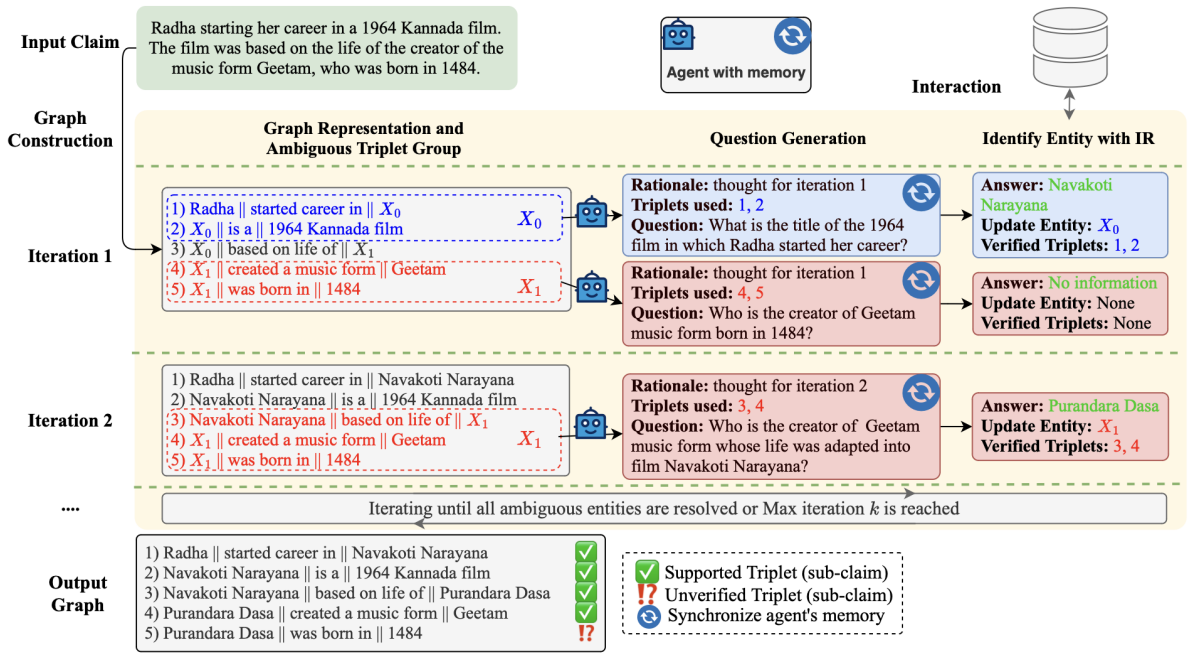

Figure 4: Illustration of the entity disambiguation process

in Figure 4, the triplets are grouped according to two ambiguous entities, X_1 and X_2. This method isolates each ambiguous entity along with relevant information, facilitating a more focused resolution.

Interaction with Knowledge Base. Once the triplets are grouped, the LLM interacts with each group to generate clarifying questions for the ambiguous entities. A major challenge arises when entity-related information in the knowledge base is often fragmented across multiple documents or sections, leading to that if we combine all the information or aspects related to an entity to find it from a specific partition of the knowledge base can be difficult. To address this, we adopt an iterative question refinement approach where the LLM uses the triplet information to narrow down ambiguities. In each iteration, the LLM processes a group g of triplets, producing the following outputs: i) a rationale **r**, which outlines the reasoning for selecting specific triplet information to construct the question; ii) a set of triplet identifiers **ids**, denoting the triplets used in formulating the question; and iii) a targeted question **q**, designed to clarify the ambiguous entity. The rationale r guides the LLM in filtering relevant triplets (ids) for constructing a precise question q. This dynamic and self-controlled process enables the LLM to consider various aspects of the triplet group, ensuring comprehensive coverage of the information. The question q is then processed by the function ***Entity Identification*** to resolve the ambiguous entity. If a question fails to resolve the ambiguity, the reasoning trace and the failed question are fed back into the LLM at the next iteration to generate a refined question that incorporates alternative triplet aspects.

As the process iterates, if an ambiguous entity in a group is clarified, after the iteration, the graph G is updated accordingly. Other groups that share the same ambiguous entity benefit from this update, improving question refinement for those groups in subsequent iterations. For example, in Figure 4, after the first iteration entity X_0 is identified as "Navakoti Nrayana", this information is then used to update other triplets (e.g. triplet id 3). At the next iteration, this resolved entity adds more information that is related to the X_1 group. The iteration continues until either: i) all ambiguous entities are resolved; or ii) a maximum iteration limit **k** is reached. The iterative refinement provides opportunities for the system to interact with the knowledge base and resolve the required ambiguous entity (Detailed prompts confer Figures 13 and 14).

Verified Information and Outcome. When a question resolves an entity's ambiguity, the corresponding triplets (with ids) are marked as containing verified information. The disambiguation process concludes when all ambiguous entities are resolved. If an entity remains ambiguous after k iterations, the entire claim associated with that entity is classi-

fied as "REFUTES", indicating insufficient information for verification. Once all ambiguities are resolved, the disambiguation process outputs an updated graph with: i) **Verified triplets**: Triplets that contributed to the process of resolving ambiguities; and ii) **Remaining triplets**: Triplets that did not participate in the disambiguation process.

3.4 Verification of Remaining Sub-claims

After entity disambiguation, some triplets remain unverified, while others were not initially grouped for the disambiguation process. These remaining triplets require further verification. To achieve this, we employ a large language model (LLM) to generate full-text sub-claim questions based on the unverified triplets. For example, consider the triplet from Figure 4: *"Purandara Dasa || was born in || 1484"*. The LLM transforms this triplet into a full-text subclaim, such as *"Purandara Dasa is the person who was born in 1484"*. This subclaim is then used in conjunction with the knowledge base for verification, facilitated by the ***Subclaim Verification*** function. Once all remaining sub-claims are verified, the original claim C is classified. If all sub-claims are supported, C is categorized as $Supported$; otherwise, if any sub-claim fails, C is categorized as $Refuted$.

4 Experiments

4.1 Datasets and Evaluation Metric

Dataset. We conduct our experiments using an open-book setting, simulating a real-world scenario where the system must interact with an external knowledge base to verify claims. We evaluate the proposed *VeGraph* on two widely-used benchmark datasets for complex claim verification: HoVer and FEVEROUS. Both datasets contain intricate claims that require multi-hop reasoning and evidence gathering from various information sources. Due to the unavailability of public test sets, we rely on validation sets for evaluation. The HoVer dataset (Jiang et al., 2020) is a multi-hop fact verification benchmark designed to validate claims using evidence across multiple sources, including 2-hop, 3-hop, and 4-hop paths. It is based on the introductory sections of the October 2017 Wikipedia dump. The multi-hop nature of HoVer challenges the system to retrieve and aggregate information from several interrelated documents. The FEVEROUS dataset (Aly et al., 2021) addresses complex claim verification using both structured and unstructured data. Each claim is annotated with evidence derived from either sentences or table cells within Wikipedia articles of the December 2020 dump. For consistency with prior work (Aly et al., 2021), we evaluate FEVEROUS claims on three key partitions: Multi-hop Reasoning, Entity Disambiguation, and Numerical Reasoning. As our research focuses on textual fact-checking, we exclusively select claims that require sentence-based evidence, discarding those involving table cells or other structured data. To manage computational costs, specifically for the HoVer dataset, we sample 200 claims from each partition while ensuring balanced label distributions.

Metrics. Following practices in the field, we use the Macro-F1 as the primary evaluation metric.

4.2 Baselines

For the comparison, we selected recent modern methods using LLM for multi-step reasoning veracity prediction, which are related to our work, as the baselines. Specifically, the baselines are sequentially described as follows:

CoT-Reasoning (Wei et al., 2022b) is a popular approach that demonstrates chains of inference to LLMs within an in-context prompt. Accordingly, for the claim verification task, the input claim is decomposed by asking LLMs to generate the necessary query (sub-claims) that needs to be verified. The sub-claims are then verified step-by-step using prompt LLMs incorporating knowledge sources via information retrieval systems.

ProgramFC (Pan et al., 2023b) is one of the first claim verification models in the era of LLMs with the explainable capability for multi-step reasoning of veracity prediction. Specifically, the model decomposes complex claims into simpler sub-tasks and then solves the sub-tasks by using specialized functions with program-guided reasoning.

FOLK (Wang and Shu, 2023) improve the explainable claim verification by introducing the first-order-logic (FOL) clause as the guided claim decomposition to make veracity predictions and generate explanations to justify step-by-step the verification decision-making process.

PACAR (Zhao et al., 2024) is a recent method based on the LLM agent technology to improve the performance of the verification. Specifically, the model utilizes the capability of strong LLMs for dynamic planning and execution of actions, in which a self-reflection module is designed in order to verify the veracity of prediction with multi-path

Method	IR System	HoVer			FEVEROUS		
		2hop	3hop	4hop	Multi-hop	Disambiguation	Numerical
Underlying LLM: GPT-3.5 Turbo (175B) or Codex; Different Experimental Setup							
ProgramFC* (Pan et al., 2023b)	BM25	69.36	60.63	59.16	-	-	-
FOLK* (Wang and Shu, 2023)	SERP API	66.26	54.8	60.35	-	-	-
PACAR* (Zhao et al., 2024)	Serper API	73.13	64.07	63.82	-	-	-
Underlying LLM: Meta-Llama-3 (70B); Same Experimental Setup							
CoT-Reasoning	Bi-Encoder	67.97	62.45	46.21	57.81	60.51	50.56
ProgramFC	Bi-Encoder	68.00	62.26	53.96	**64.32**	67.11	72.01
FOLK	Bi-Encoder	67.74	58.49	53.47	60.96	61.00	47.44
VeGraph (ours)	Bi-Encoder	**69.70**	**66.13**	**58.59**	59.39	**73.89**	**82.60**
VeGraph (ours)	BM25	69.22	63.10	56.68	53.29	72.46	82.06

Table 1: Report results of Macro-F1 score on HoVer and FEVEROUS datasets. * are taken from respective papers. Both texts indicate the best score for the same experimental setup.

reasoning decision-making.

4.3 Experimental Setup

Since the baselines have different configurations including input data, information retrieval systems, and underlying LLM in their respective papers, therefore, we try to reproduce the baseline with the unified configuration, following their available source codes[2][3]. Note that, since the source code of PACAR is not release yet and the main contributions of this work mainly depend on the prompt-based framework, which is not available for reproducibility. In this regard, we report the original results that are provided by the respective paper, without re-constructing this model regarding our experiment.

4.4 Backbone LLM and Prompting Strategy

In our experiments, we employ Meta-Llama-3-70B-Instruct[4] as the underlying LLM. To construct graph representations, we leverage in-context learning by providing the model with human-crafted examples to guide the LLM to perform the required tasks. For other tasks, we use zero-shot prompting leveraging existing LLM reasoning capability.

Retrieval System: Focusing on open-book settings, we utilize the corresponding Wikipedia corpora constructed specifically for the HOVER and FEVEROUS as knowledge sources. To simulate real-world systems, we implement a two-layer retrieval system. The first layer employs BM25 (Robertson et al., 1994) as the sparse retrieval algorithm. The second layer combines a Bi-Encoder model (bge-m3) with a Reranker (bge-reranker-v2-m3) (Chen et al., 2024), refining the search results by filtering out irrelevant documents. When interacting with the two functions described in Section 3.2, we set a constraint of a maximum of 15 retrieved documents or a maximum of 6000 tokens, adhering to the model's maximum input length.

4.5 Main Results

The overall performance of VeGraph and the baselines is presented in Table 1. The results are organized into two sections: the first section outlines the performance of the baselines, as reported in their respective papers. These models employ varying configurations, including differences in the number of examples used for inference and the retrieval systems implemented. The second section presents the results of our proposed VeGraph model, alongside the reproduced baselines, which are evaluated under identical configurations. From these experiments, we derive several key insights:

VeGraph effectively verifies complex claims: VeGraph consistently outperformed most previous models across test cases. Notably, on the HoVer dataset, where input claims exhibit significant complexity, VeGraph demonstrated a performance improvement of 4% and 5% on three-hop and four-hop claims, respectively. These findings indicate that VeGraph is particularly well-suited for verifying complex claims that demand more advanced reasoning mechanisms.

Enhanced entity disambiguation leads to gaining in performance: Through the integration of interactive graph representations and the agent-based LLM framework, VeGraph achieves substantial performance gains across multiple benchmark datasets. For instance, in the FEVEROUS dataset, VeGraph surpassed baselines by 6.8% in the Disambiguation category and 10.5% in the Numerical

[2]https://github.com/teacherpeterpan/ProgramFC

[3]https://github.com/wang2226/FOLK

[4]https://huggingface.co/meta-llama/Meta-Llama-3-70B-Instruct

category. However, for the Multi-hop category of FEVEROUS, VeGraph exhibited slightly lower performance. We hypothesize that the claims in this test case are less complex than others which do not require multiple interactions with the knowledge base, making the symbolic reasoning capabilities of the ProgramFC more effective in this context.

4.6 Ablation Study

To evaluate the contribution of each component in the proposed VeGraph framework, we conducted an ablation study on the HoVer dataset. Specifically, we analyzed the impact of graph representation for disambiguating entity interactions and the role of multi-path reasoning in decision-making within the LLM-agent framework. The configurations and results are presented in Table 2. The

Method	2hop	3hop	4hop
VeGraph - w/o Interactive Graph	64.71	56.68	43.16
VeGraph - 1 step	69.09	62.34	54.83
VeGraph - 2 steps	69.70	63.82	57.33
VeGraph - 5 steps	69.70	66.13	58.59

Table 2: Ablation studies on the number of disambiguation steps and the effectiveness of graph representation on Hover dataset.

results demonstrate that removing graph representation severely degrades performance, especially on more complex claims (e.g., 3-hop and 4-hop). This highlights the importance of graph-based reasoning in VeGraph. Additionally, increasing the number of reasoning steps improves performance, indicating that multi-path decision-making is crucial for verifying complex claims.

4.7 Interpretability and Error Analysis

Our proposed VeGraph framework not only enhances the performance of claim verification systems but also offers a high degree of interpretability, which is essential for human comprehension and trust. Examples of these generated reasoning traces are provided in Figure 7 of Appendix B. To evaluate the quality of the reasoning processes and the generated graphs, we conducted a human analysis on 50 failed predictions for each partition (2-hop, 3-hop, 4-hop) of the HOVER dataset, focusing on instances where VeGraph incorrectly predicted the claim's veracity. Human annotators categorized the errors into three primary types, corresponding to different stages of the framework: i) **Graph Representation Errors:** These occur when VeGraph fails to accurately capture the semantic structure of the claim, resulting in flawed graph representations; ii) **Entity Resolution Errors:** These arise when the system either fails to disambiguate entities or struggles to correctly identify the entities relevant to the claim; iii) **Subclaim Errors:** These involve incorrect predictions at the level of individual sub-claims leading to erroneous final verdicts. As shown in Table 3, the error distribution varies across the 2-hop, 3-hop, and 4-hop partitions of the HOVER dataset. Despite few-shot in-context learning strategies being employed, the LLM occasionally encounters challenges in constructing accurate graph representations, particularly when processing complex claims. The increasing complexity of multi-hop claims (e.g., 3-hop and 4-hop) further exacerbates issues in entity disambiguation, as a larger number of ambiguous entities complicates the retrieval of relevant documents. Even after multiple interaction cycles, entity disambiguation may remain incomplete, affecting the overall reasoning process. These limitations in both graph construction and entity resolution propagate through the framework, leading to reduced accuracy in the final verdicts, particularly in multi-hop scenarios. Additionally, another source of error comes from failed interactions with the knowledge base, where unresolved triplets mislead the retrieval system, underscoring the critical importance of retrieval performance.

Error Types	2hop	3hop	4hop
Graph Representation	29%	15%	17%
Entity Disambiguation	37%	53%	45%
Subclaims Verification	34%	32%	38%

Table 3: Evaluation for incorrectly predicted examples across partitions in the HOVER dataset.

5 Conclusion

This study presents VeGraph, a novel claim verification framework using the concept of interactive graph representation incorporating LLM agent technology to identify ambiguous entities in terms of multi-path multi-step reasoning of veracity predictions. Specifically, the input claim first is decomposed into a set of triplets. These triplets are then identified with ambiguous entities and verified of fact interactively using the proposed agent LLM pipeline. The experiment on two well-known benchmark claim verification datasets indicates promising results of VeGraph for claim verification tasks, especially in the case of complex claims.

Limitations

While the proposed framework enhances performance in disambiguating entities and verifying subclaims, it imposes computational overhead due to its frequent reliance on large language models. This increased demand for computational resources can introduce latency, posing challenges for real-world applications that require rapid response times.

Despite their advanced reasoning capabilities, LLMs are prone to errors and may exhibit biases toward certain types of content. This highlights the need for careful deployment, especially in fact-checking systems, where biased or incorrect outputs could lead to misinformation. Developing effective mechanisms to detect, control, and mitigate these biases remains an open challenge for future research.

Another limitation lies in the dataset used for our experiments, which predominantly focuses on explicit reasoning. Although the framework incorporates self-analysis and structured representation, real-world claims often require processing implicit information, adding complexity beyond the current design. Addressing this gap will be a crucial direction for future work, enabling the framework to better manage nuanced reasoning and improve its practical applicability.

References

Rami Aly, Zhijiang Guo, Michael Sejr Schlichtkrull, James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Oana Cocarascu, and Arpit Mittal. 2021. FEVEROUS: fact extraction and verification over unstructured and structured information. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *Preprint*, arXiv:2402.03216.

Alphaeus Dmonte, Roland Oruche, Marcos Zampieri, Prasad Calyam, and Isabelle Augenstein. 2024. Claim verification in the age of large language models: A survey. *CoRR*, abs/2408.14317.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1535–1545. ACL.

Zhijiang Guo, Michael Sejr Schlichtkrull, and Andreas Vlachos. 2022. A survey on automated fact-checking. *Trans. Assoc. Comput. Linguistics*, 10:178–206.

Yichen Jiang, Shikha Bordia, Zheng Zhong, Charles Dognin, Maneesh Kumar Singh, and Mohit Bansal. 2020. Hover: A dataset for many-hop fact extraction and claim verification. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 3441–3460. Association for Computational Linguistics.

Amrith Krishna, Sebastian Riedel, and Andreas Vlachos. 2022. Proofver: Natural logic theorem proving for fact verification. *Trans. Assoc. Comput. Linguistics*, 10:1013–1030.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 73–82. The Association for Computer Linguistics.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Liangming Pan, Xiaobao Wu, Xinyuan Lu, Anh Tuan Luu, William Yang Wang, Min-Yen Kan, and Preslav Nakov. 2023a. Fact-checking complex claims with program-guided reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 6981–7004. Association for Computational Linguistics.

Liangming Pan, Xiaobao Wu, Xinyuan Lu, Anh Tuan Luu, William Yang Wang, Min-Yen Kan, and Preslav Nakov. 2023b. Fact-checking complex claims with program-guided reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 6981–7004. Association for Computational Linguistics.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994*, volume 500-225 of *NIST Special Publication*, pages 109–126. National Institute of Standards and Technology (NIST).

Anastasiia Sedova, Robert Litschko, Diego Frassinelli, Benjamin Roth, and Barbara Plank. 2024. To

know or not to know? analyzing self-consistency of large language models under ambiguity. *Preprint*, arXiv:2407.17125.

Haoran Wang and Kai Shu. 2023. Explainable claim verification via knowledge-grounded reasoning with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 6288–6304. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022a. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

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

Xuan Zhang and Wei Gao. 2023. Towards llm-based fact verification on news claims with a hierarchical step-by-step prompting method. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, IJCNLP 2023 -Volume 1: Long Papers, Nusa Dua, Bali, November 1 - 4, 2023*, pages 996–1011. Association for Computational Linguistics.

Yadong Zhang, Shaoguang Mao, Tao Ge, Xun Wang, Adrian de Wynter, Yan Xia, Wenshan Wu, Ting Song, Man Lan, and Furu Wei. 2024. LLM as a mastermind: A survey of strategic reasoning with large language models. *CoRR*, abs/2404.01230.

Xiaoyan Zhao, Lingzhi Wang, Zhanghao Wang, Hong Cheng, Rui Zhang, and Kam-Fai Wong. 2024. PACAR: automated fact-checking with planning and customized action reasoning using large language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation, LREC/COLING 2024, 20-25 May, 2024, Torino, Italy*, pages 12564–12573. ELRA and ICCL.

A Examples

Figure 5, 6, and 7 show three types of errors specified in the main section. Figure 8 and 9 are the correct samples showing reasoning traces and output of each stage in the framework.

B Prompt Templates

For better reproducibility, we present all prompt templates in the appendix. Below is a quick reference list outlining the prompt templates and their usages:

- Figure 10: Verify a claim based on the information within a set of documents.
- Figure 11: Extract an entity within a set of documents that satisfies a question.
- Figure 12: Construct a graph representation of the input claim.
- Figure 13: Generate a question to resolve the ambiguous entity from the given graph triplets and claim.
- Figure 14: Refine failed questions and generate a new question to resolve the ambiguous entity from the given graph triplets and claim.
- Figure 15: Generate sub-claims each corresponding to a graph triplet.

All prompts are zero-shot, except for the prompt in Figure 12, which uses few-shot demonstrations to better guide the LLM to perform the task. These prompts were chosen because they perform effectively in practice.

Error in Constructing the Graph

Input Claim: Wild About Harry stars an actor born September 28, 1992, who appeared in Kill Your Darlings in 2006.

Wrong Graph:
X_0||stars in||Wild About Harry
X_0||was born on||September 28, 1992
X_0||appeared in||X_1
X_1||is a||Kill Your Darlings

Correct Graph
X_0||stars in||Wild About Harry
X_0||was born on||September 28, 1992
X_0||appeared in||||Kill Your Darlings

Figure 5: Wrong Example 1

Error in Verifying the Remaining Triplets

Input Claim: The institution, where Eddie George earned an MBA from, and Middlebury College are both private schools.

Graph:
X_0||is a||private school
Eddie George||earned an MBA from||X_0
Middlebury College||is a||private school

Entity Disambiguation Process
- Iteration 1:
- For X_0
+ Grouped Triplet:
    X_0||is a||private school
    Eddie George||earned an MBA from||X_0
+ Rationale: Using the information that Eddie George earned an MBA from the hidden entity, I generated a question to find the institution where Eddie George got his MBA.
+ Question: What institution did Eddie George earn his MBA from?
+ Resolved Entity: Northwestern Universitys Kellogg School of Management

Verify Remaining Triplets:
Triplet: Northwestern Universitys Kellogg School of Management||is a||private school
Subclaim: Northwestern Universitys Kellogg School of Management is a private school ==> False (Wrongly verified)

Predicted Verdict: False
Gold Verdict: True

Figure 6: Wrong Example 2

Error in Identity Ambiguous Entity

Input Claim: The actress who made her drama movie debut in Friday Night Lights also starred in a 2014 French-American crime-thriller directed by McG.

Graph:
X_0||made her drama movie debut in||Friday Night Lights
X_0||starred in||X_1
X_1||is a||2014 French-American crime-thriller
X_1||was directed by||McG

Entity Disambiguation Process
- Iteration 1:
- For X_0
+ Grouped Triplet:
    X_0||made her drama movie debut in||Friday Night Lights

+ Rationale: Using the information the actress made her drama movie debut in Friday Night Lights, I generated a question to find this actress.
+ Question: Who made her drama movie debut in Friday Night Lights?
+ Resolved Entity: Connie Britton (Must be Amber Heard)

.....

Figure 7: Wrong Example 3

Correct Example

Input Claim: Little Big Girl was a Simpsons episode directed by an American animator and artist.

Graph:
Little Big Girl||was directed by||X_0
X_0||is an||American animator
X_0||is an||artist
Little Big Girl||is a||Simpsons episode

Entity Disambiguation Process
Iteration 1:
- For X_0
+ Grouped Triplet:
    Little Big Girl||was directed by||X_0
    X_0||is an||American animator
    X_0||is an||artist
+ Rationale: Using the information that the hidden entity is an American animator and artist, and that it directed a Simpsons episode, I generated a question to find the specific person who matches this description.
+ Question: Who is the American animator and artist who directed a Simpsons episode?
+ Resolved Entity: David Silverman

Verify Remaining Triplets:
Triplet: Little Big Girl||is a||Simpsons episode
Subclaim: Little Big Girl is a Simpsons episode ==> True

Predicted Verdict: True
Gold Verdict: True

Figure 8: Correct Example Output 1

Correct Example

Input Claim: The TV show starring Pāvels Gumennikovs was written by the creator of "The Borowitz Report". This show also starred Joseph Marcell as the butler.

Graph:
X_0||stars||Pāvels Gumennikovs
X_0||was written by||X_1
X_1||is the creator of||The Borowitz Report
X_0||stars||Joseph Marcell
Joseph Marcell||plays the role of||butler

Entity Disambiguation Process
Iteration 1:
- For X_0
+ Group Triplet:
    X_0||stars||Pāvels Gumennikovs
    X_0||stars||Joseph Marcell
+ Rationale: Using the information that the hidden entity stars Pāvels Gumennikovs and Joseph Marcell, I generated a question to find the TV show that features these two actors.
+ Question: What TV show stars Pāvels Gumennikovs and Joseph Marcell?
+ Resolved Entity: None

- For X_1
+ Group Triplet:
    X_1||is the creator of||The Borowitz Report
+ Rationale: Using the information that the hidden entity is the creator of The Borowitz Report, I generated a question to find the creator of The Borowitz Report to resolve the hidden entity.
+ Question: Who is the creator of The Borowitz Report?
+ Resolved Entity: Andy Borowitz

Iteration 2:
- For X_0
+ Group Triplet:
    X_0||stars||Pāvels Gumennikovs
    X_0||was written by||Andy Borowitz
    X_0||stars||Joseph Marcell
+ Rationale: Using the information that the hidden entity was written by Andy Borowitz, who is the creator of The Borowitz Report, I generated a question to find the TV show written by Andy Borowitz.
+ Question: What TV show was written by Andy Borowitz?
+ Resolved Entity: The Fresh Prince of Bel-Air

Verify Remaining Triplets:
Triplet: The Fresh Prince of Bel-Air||stars||Pāvels Gumennikovs
Subclaim: The Fresh Prince of Bel-Air stars Pāvels Gumennikovs ==> False

Triplet: The Fresh Prince of Bel-Air||stars||Joseph Marcell
Subclaim: The Fresh Prince of Bel-Air stars Joseph Marcell == False

Triplet: Joseph Marcell||plays the role of||butler
Subclaim: Joseph Marcell plays the role of butler ==> True

Predicted Verdict: False
Gold Verdict: False

Figure 9: Correct Example Output 2

FACT_CHECK_WITH_DOCS

Task: Verify the following claim based on the information in the provided documents
Guidelines:
1) Use only the content from the provided documents to verify the claim. Do NOT rely on outside information or generate knowledge yourself. Avoid making implications.
2) To verify the claim:
- Return true if the claim is supported by the documents.
- Return false if the documents provide information that contradicts the claim.
3) Return in the following format:
```
{
    "rationale": "A short rationale with supported or contradicted evidence to guide the verifying process.",
    "veracity": "true or false"
}
```
4) If the claim cannot be answered due to insufficient information, return:
{"rationale": null, "veracity": null}

Documents:
{{context}}

Claim:
{{claim}}

Figure 10: Prompt template to find related section content from articles.

QA_WITH_DOCS

Task: Based only on the information provided in the given documents, answer the question.
Guidelines:
1) Use only the content from the provided documents to verify the claim. Do NOT rely on outside information or generate knowledge yourself. Avoid making implications.
2) Return one specific entity requested in the question in the following format:
{"answer": "the one entity you identified"}
3) If the entity is not found in the documents, return
{"answer": null}

Documents:
{{context}}

Question:
{{question}}

Figure 11: Prompt template to find related section content from articles.

FEW_SHOT_CONSTRUCT_GRAPH

Task: Construct a graph that captures entities and relationships from a given claim. Extract triplets with entities and relations between them, including hidden, ambiguous or implicit entities
Guidelines:
1) Only use information from the claim, do NOT include external knowledge
2) Do NOT repeat similar triplets in the graph

Examples:
-- Example 1 --
<input_claim> One of the hosts of the 2022 KBS Drama Awards is a Korean actor. He is well known for his role in a 2016 South Korean television soap opera and starred alongside Han Hyo-joo. Kim Eui-sung also appeared in the series.
<guidance_for_graph_construction>
The claim mentions "One of the hosts of the 2012 KBS Drama Awards" without specifying the name, so it will be marked as X_0. The claim mentions a 2016 South Korean television soap opera without specifying the name, so it will be marked as X_1
<graph>
X_0||is a||Korean actor
X_0||hosted||2012 KBS Drama Awards
X_0||is well known for his role in||X_1
X_1||is a||2016 South Korean television soap opera
X_1||starred||Han Hyo-joo
Kim Eui-sung||appeared in||X_1

... Other examples ...

Claim:
{{question}}

Figure 12: Prompt template to find related section content from articles.

GENERATE_QUESTION

Task: You will be given a claim and a graph via triplet form. In the graph, there will be an entity that is hidden (marked as X) that needs to be resolved via searching in an external knowledge base. Your job is to generate the search question to resolve this entity
Guidelines:
1) Graph will provided with triplets following the form: triplet_id||entity_1||relation||entity_2
2) You do NOT need to combine all the information of the triplets to form the question. Try one or more aspects corresponding to triplets at a time that is enough to form the question to identify that entity

Return format:
```
{
    "rationale": "a short rationale explaining how you use the information to generate the query",
    "question": "generated search question to resolve the entity",
    "triplet_ids": "a list containing ids of the triplets with information used to generate the query"
}
```

Input
Claim: {{claim}}
Graph:
{{graph}}

Figure 13: Prompt template to find related section content from articles.

REFINE_QUESTION

Task: You will be given a claim and a graph via triplet form. In the graph, there will be an entity that is hidden (marked as X) that needs to be resolved via searching in an external knowledge base.
You have already made some trials with thoughts and questions to try to get the entity but failed
Your job is to generate the search question to resolve this entity

Guidelines:
1) Graph will provided with triplets following the form: triplet_id||entity_1||relation||entity_2
2) You MUST generate one new question to resolve the X entity in the graph
3) You do NOT need to combine all the information of the triplets to form the question. Try one or more aspects corresponding to triplets at a time that is enough to form the question to identify that entity

Return format:
```
{
    "rationale": "a short rationale explaining how you use the information to generate the query",
    "question": "new search question to resolve the entity",
    "triplet_ids": "a list containing ids of the triplets with information used to generate the query"
}
```

Input
Claim: {{claim}}
Graph:
{{graph}}

Failed rationales and questions:
{{failed_trials}}

Figure 14: Prompt template to find related section content from articles.

GENERATE_SUBCLAIMS

Task: You will be given a claim and a graph represented as triplets. The graph is divided into two parts: verified and unverified triplets. Each triplet represents a sub-claim from the original claim. For each triplet in the unverified triplets, convert this triplet to a text claim to verify. You can use additional information from the claim to generate proper sub-claims.

Return format:
{"sub_claims": [list of generated sub-claims to verify]}

Claim: {{claim}}
Verified Triplets:
{{verified_triplets}}

Unverified Triplets:
{{unverified_triplets}}

Figure 15: Prompt template to find related section content from articles.