Self-Resolve: Resolving Consistent Reasoning Structures for Fact Verification

Anonymous ACL submission

Abstract

Language models often generate non-factual statements, especially when handling complex queries that require synthesizing information from multiple sub-queries. Verifying the factuality in such cases poses significant challenges and often demands the use of large language models, which can be computationally expensive. In this work, we focus on one such scenario: addressing non-factual statements within a multi-hop question-answering setup using a smaller model. We propose a novel approach (Self-Resolve) inspired by the self-discovery and self-check prompting techniques, enabling language models to construct their own reasoning structures for fact verification and then resolve the final answer based on a majority voting mechanism. This integrated framework outperforms closed-source models like GPT-4 by 9% in F1 score for 2-hop query-answer verification using Llama3-8B while achieving competitive results in 3-hop and 4-hop settings. These results underscore the effectiveness of our approach and provide valuable insights into the challenges and potential of fact-checking in language models.

1 Introduction

Factual consistency is a critical aspect of evaluating the reliability of outputs generated by language models. Despite their impressive language generation capabilities, these models frequently produce non-factual content, especially in response to complex queries requiring compositional reasoning. Compositional reasoning involves tasks where the meaning of a complex expression is determined by its individual components and the rules governing their combination. Such scenarios are common in daily life and involve the challenge of efficiently orchestrating the use of appropriate tools in the correct sequence. For instance, consider the query: 'Which airline offers the cheapest flight from New York to Mumbai that also provides an option for vegetarian meals?' Answering this requires a multi-step (2-hop) process: first invoking an API to retrieve flight details and then using another API to check meal options for vegetarian availability.

In recent years, significant progress has been made in leveraging large language models (LLMs) to address complex reasoning tasks. However, as highlighted by (Augenstein et al., 2024), these tasks often expose the limitations of LLMs' reasoning capabilities, emphasizing the critical role of fact verification in ensuring the reliability of generated outputs. While substantial efforts have focused on addressing factual inconsistencies in general NLP tasks, the domain of fact verification in complex reasoning scenarios remains relatively underexplored. Fact Verification involves assessing the factual correctness of a statement based on reference evidence. In this work, we aim to advance research in this area by focusing on detecting factual correctness in setups where the input query comprises of multiple interconnected sub-queries. Although approaches like those discussed in Manakul et al. (2023); Arora et al. (2022); Dhuliawala et al. (2023) utilize large LLMs such as GPT (Achiam et al., 2023) for hallucination detection, these methods often suffer from high computational costs and limited transparency posing significant challenges for practical adoption.

Most studies have focused on using large language models (LLMs) such as GPT-4 (Achiam et al., 2023) for fact verification. These models, although powerful, are computationally expensive and often lack interpretability in their reasoning processes. Current studies often leverage these large models but fail to explore the potential of smaller language models (under 10 billion parameters) for such tasks. Smaller models, if appropriately guided, can offer more transparent and cost-effective solutions for detecting hallucinations and verifying facts.

Direct prompting methods, which rely on ex-

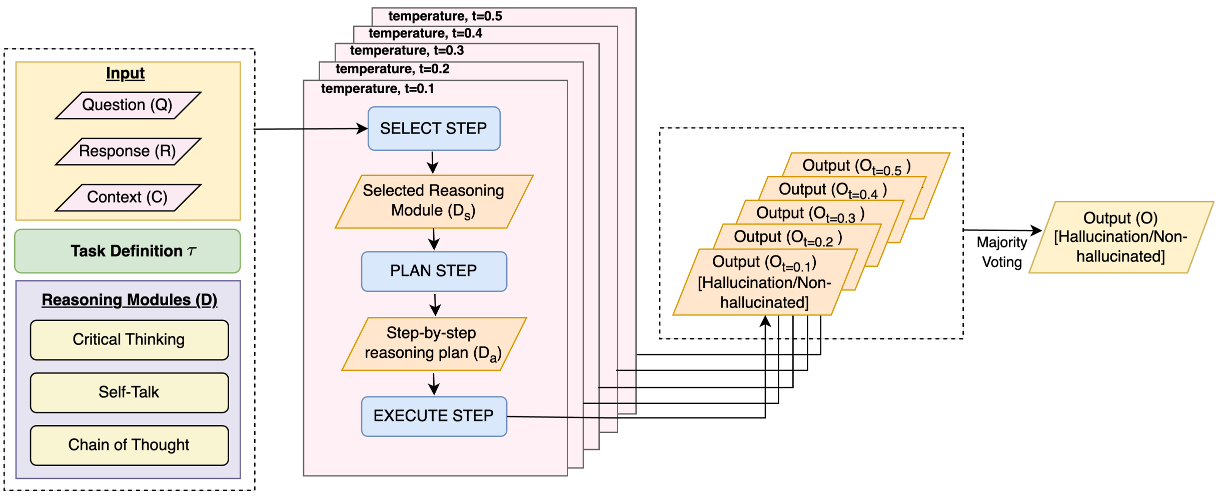

Figure 1: Self-Resolve (proposed framework): We begin by providing the model with initial heuristics (reasoning modules), alongside the input and task definition. The Select module is then invoked to identify the most suitable heuristics for the given task. Next, the Plan module generates a detailed, step-by-step reasoning plan tailored to the specific query, incorporating both the Response and Context. Following this, the Execute module carries out the reasoning plan, working towards the goal outlined in the Task Definition. This process is repeated across five different temperature settings to introduce diversity in the reasoning paths. Finally, a majority voting mechanism is applied to aggregate and resolve the outputs and determine the final result.

plicit cues in reference documents, have shown limitations in handling complex reasoning tasks. To address this, we draw inspiration from meta-prompting approaches (Zhou et al., 2024; Fernando et al., 2023), which prompt language models to generate their own prompts for reasoning tasks. While these techniques are primarily designed for large LLMs and focus on world knowledge, they often neglect the intricacies of compositional reasoning.

In this work, we extend the meta-prompting paradigm to smaller models, adapting it to handle multi-hop queries by emphasizing reasoning over world knowledge. Our proposed approach involves finding the optimal solution by generating multiple variants of the reasoning plan and resolving the final answer through majority voting. The core idea is inspired by how humans tackle complex problems: by considering various possible solutions and prioritizing the one supported by the majority of reasoning paths.

Our contributions are:

- An evaluation of meta-prompting capabilities of smaller language models (less than 10 Billion) for hallucination detection (See Table 2). To the best of our knowledge, this is the first such study for smaller models.
- A novel approach for fact-verification inspired by the thinking style of humans. The proposed framework outperforms closed-source models like GPT-4 by 9% in F1 score for 2-hop query-answer verification using Llama3-8B while achieving competitive results in 3-hop and 4-hop settings (See Table 2).
- An extension of the MusiQue dataset with 600 instances containing hallucinated question-answer pairs to facilitate the study of factual inconsistency in multi-hop reasoning tasks (See Table 1).

The motivation for this work stems from the need to improve the reliability of smaller models in handling complex natural language understanding tasks. By enhancing the interpretability of hallucination detection, we aim to increase transparency in the reasoning processes underlying predictions. Furthermore, this research has broader implications for navigating complex tasks requiring multi-aspect reasoning, such as diagnosing diseases based on diverse symptoms.

Despite its promise, hallucination detection in compositional reasoning setups presents significant challenges. Smaller models face inherent limitations in reasoning capabilities, making it difficult to accurately detect factual inconsistencies in multi-hop queries. This work addresses these challenges by combining the cost-efficiency of smaller models with innovative reasoning-driven techniques, paving the way for more reliable and interpretable solutions in NLP.

2 Background

Recent advancements in hallucination detection in natural language processing (NLP) have led to a deeper understanding of the challenges and methodologies in this domain. Huang et al. (2023) provides a comprehensive survey of recent developments in hallucination detection, while Wang et al. (2024) and Augenstein et al. (2024) explore these advancements in the context of factuality. Notably, Wang et al. (2024) distinguished hallucination from fact verification, emphasizing that generated text can diverge from the specifics of the input prompt (hallucination) without necessarily being factually incorrect if the content remains accurate.

Prompt-Based Approaches for Hallucination Detection: Prompt-based methods have emerged as popular strategies for hallucination detection, particularly when reference documents are unavailable. However, they often lack frameworks for incorporating reference text for verification. For instance, Manakul et al. (2023) proposed a method that stochastically samples multiple outputs (e.g., 20 responses at temperature 1) from language models and evaluated consistency across the generated outputs. Similarly, Arora et al. (2022) aggregated outputs generated from multiple prompts to derive the final result. In contrast, Dhuliawala et al. (2023) introduced a verification plan that uses an initial prompt to generate and execute reasoning steps for a verified response. Additionally, Min et al. (2023) decomposed text generation into a series of atomic facts, computing the percentage of these facts supported by reliable sources. While promising, these approaches face challenges when tasked with handling complex semantic compositions, where atomic fact verification alone may not suffice.

Factual Inconsistency in Summarization: Factual inconsistency between abstractive summaries and their source documents remains a critical area of research. Yang et al. (2024) proposed a method for fact verification in abstractive summarization that involves decomposing summaries into atomic facts and aligning them with source documents through adaptive granularity expansion. Other approaches, such as those proposed by Tang et al. (2024), Stacey et al. (2024), Liu et al. (2024), and Ko et al. (2024), further explore methods to detect and correct inconsistencies in summarization tasks.

A major challenge in prior work on fact verification is the reliance on large language models (LLMs) and brittle prompt engineering methods, which lack scalability, interpretability, and effective integration of reference documents for reasoning. Additionally, approaches that decompose outputs into atomic facts often struggle with semantic composition, limiting their utility for complex reasoning tasks. Chandler et al. (2024) attempted to detect factual inconsistencies by introducing multiple prompts and ensembling the outputs to perform final consistency checks. However, their approach does not incorporate grounding to source documents, relying solely on manual prompt engineering which may be brittle and sensitive to input variations.

Our proposed solution addresses these challenges by leveraging meta-prompting techniques to generate reasoning plans, enabling smaller LLMs to perform fact verification effectively. We improve reliability and robustness by combining multiple reasoning variants with majority voting.

3 Datasets

One of the main goals of this work is to access the meta-reasoning capabilities of smaller LLMs which necessitates the requirement for complex inputs. We consider MuSiQue Dataset (Trivedi et al., 2022) for our experiments.

The MuSiQue dataset was chosen for its high-quality compositional instances that prevent the use of simple decomposition-based heuristics to determine the answer. This dataset provides question-answer pairs along with supporting paragraphs containing relevant information. An example of a 2-hop query is illustrated in Figure 3.

For our experiments, we sampled 200 questions each from 2-hop, 3-hop, and 4-hop categories, and then replaced the answers to 300 questions (100 from each category) with incorrect ones to introduce factual inconsistencies (Refer Table 1). To generate these incorrect answers, we utilized GPT4o-mini, prompting it to produce four plausibly incorrect options for each sample. The most appropriate incorrect answer among these options was then selected manually.

	2-hop	3-hop	4-hop	Total (600)
Correct	100	100	100	300
Incorrect	100	100	100	300

Table 1: Dataset Statistics showing the distribution across different hops.

Query Example

Question: What county was Tim Dubois born in?
Answer: McDonald County
Available information:
Tim DuBois (born May 4, 1948 in Southwest City, Missouri) is a Nashville, Tennessee-based music executive. He attended Oklahoma State University and received a B.A. and M.A. in Accounting and in 2016 he was awarded an honorary PHD in Accounting. He then entered into the music business and has taken part in multiple aspects of the industry including songwriting, record labels, management, and production. DuBois has been recognized for numerous honors and awards for his contributions to the music industry.', **'Southwest City is a city in McDonald County, Missouri, United States.** The population was 937 at the 2010 census, at which time it was a town. It is part of the Fayetteville–Springdale–Rogers, AR-MO Metropolitan Statistical Area and is located in the southwestern corner of the state of Missouri.

Two-Hop Example: To answer, we first identify Tim DuBois's birthplace (Southwest City, Missouri) and then determine its county (McDonald County)

When selecting the most suitable option, we ensure that the chosen answer aligns closely with the context's semantics and maintains high plausibility. For instance, if the question pertains to the sports domain, asking about the best footballer of all time should not result in an overly implausible incorrect answer, such as *John F. Kennedy*, which would be too obvious for the model to identify as incorrect. Instead, we select answers that are contextually relevant yet still incorrect, such as *Diego Maradona* or *David Beckham*. The prompt used for generating these responses is provided in Appendix 6.

4 Methodology

Problem Formulation: Given a query (Q), a response (R), and a context (C), the task is to verify whether the response is factually correct with respect to the context or not.

In this section, we describe our proposed methodology **Self-Resolve** (see Figure 1). The approach begins by equipping the model with initial heuristics, along with input data and a task definition, to help identify factual inconsistencies. Following Fernando et al. (2023); Zhou et al. (2024), we use the term "reasoning module" to refer to these heuristics.

The next step is the **SELECT STEP** which performs the selection of the most relevant heuristics best suited to the specific input context. This is followed by the **PLAN STEP**, where a structured, step-by-step reasoning plan is designed for detecting factual inconsistencies. Subsequently, in the **EXECUTE STEP**, the reasoning plan is carried out to achieve the final output.

To enhance reliability, this execution process is repeated across **five different temperature settings** (ranging from 0.1 to 0.5), and a majority voting mechanism is employed to reach the final result.

We now describe each component of the framework in detail below:

Task Definition (τ): In task definition we provide the question, answer and available information to the prompt with instructions to validate the generated output with predicted output after verification. The prompt used for this step is discussed in Appendix A, Figure 3.

Reasoning Modules (D): Reasoning modules are the initial heuristics (or seed prompts) designed to guide the development of specific reasoning plans for fact verification. Inspired by Fernando et al. (2023), we initially crafted 10 distinct reasoning plans tailored for this task. However, our initial experiments revealed that the model performed effectively with only a subset of these modules. Based on qualitative analysis, we selected three reasoning modules (Appendix A, Figure 4) as the most suitable for the fact verification process:

1. **Critical Thinking:** This module involves performing in-depth analysis of the provided information to assess the relevance and accuracy of the answer with respect to the question. LLMs utilize this seed prompt to consider various dimensions of the context, including syntax and semantics, ensuring that the answer logically aligns with the question.

2. **Self-talk:** Self-talk helps to engage in an internal dialogue within the model to evaluate the coherence and congruence of the answer with the query. LLMs utilize these ideas to identify and

Method	2 hops			3 hops			4 hops		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Llama 3.1 8B									
Vanilla	0.94	0.37	0.53	0.78	0.06	0.12	1.0	0.02	0.04
CoT	0.94	0.54	0.69	0.89	0.15	0.26	0.72	0.09	0.16
CoT + Temp. Voting	0.94	0.55	0.69	0.88	0.15	0.25	0.59	0.05	0.10
Self-Check	0.94	0.53	0.68	0.90	0.14	0.24	0.62	0.05	0.10
Query Decomposition	0.73	0.10	0.18	0.39	0.03	0.06	0.82	0.07	0.14
Self-Resolve*	0.96	0.88	**0.92**	0.91	0.67	**0.77**	0.94	0.67	**0.78**
Gemma 2 9B									
Vanilla	0.68	0.82	0.75	0.69	0.67	0.68	0.81	0.76	0.78
CoT	0.69	0.86	0.76	0.69	0.72	0.71	0.85	0.83	0.84
CoT + Temp. Voting	0.69	0.85	0.76	0.69	0.74	0.71	0.85	0.82	0.84
Self-Check	0.69	0.87	0.77	0.70	0.72	0.71	0.87	0.77	0.82
Query Decomposition	0.65	0.51	0.57	0.49	0.17	0.25	0.94	0.12	0.21
Self-Resolve*	0.93	0.83	**0.88**	0.94	0.58	**0.72**	0.95	0.72	0.82
Mistral 7B									
Vanilla	0.67	0.71	0.69	0.62	0.46	0.53	0.80	0.37	0.51
CoT	0.73	0.72	0.73	0.56	0.49	0.53	0.83	0.45	0.58
CoT + Temp. Voting	0.69	0.77	0.73	0.56	0.59	0.57	0.80	0.57	0.67
Self-Check	0.73	0.71	0.72	0.57	0.50	0.53	0.86	0.48	0.62
Query Decomposition	0.57	0.67	0.62	0.44	0.49	0.46	0.62	0.49	0.55
Self-Resolve*	0.92	0.69	**0.79**	0.82	0.58	**0.68**	0.88	0.52	0.65
GPT4-Omni									
*	0.96	0.76	0.83	0.99	0.67	0.80	0.95	0.72	0.82

Table 2: Performance comparison of our methods (marked with *) for different compositions of queries.

clarify any uncertainty or inconsistencies within the information and to reaffirm or dispute the correctness of the answer based on the information gathered.

3. **Chain of Thought:** Inspired by Wei et al. (2022), this reasoning module aims to elicit a step-by-step plan to verify the correctness of the answer. The goal is to maintain clarity and conciseness while methodically breaking down the reasoning process.

The next step involves choosing the best reasoning module for the given input.

SELECT STEP: In this step, we provide the question, answer and relevant information required for the verification and ask the language model to choose the best reasoning module as per the defined Task Definition (τ). The prompt used for this step is referred to as Select Prompt S_p (Appendix A, Figure 5). This approach mirrors how humans tackle complex problems: by first identifying the most relevant domain or framework within which to structure their reasoning.

$$D_s = S_p(\{Q, R, C, D, \tau\})$$

Here the input to S_p is Question (Q), Response (R), context (C), Task Definition τ and the set of seed reasoning module descriptions (D), and the output is a selected reasoning module (D_s).

PLAN STEP: In the Plan Step, the goal is to adapt the selected reasoning module and prepare a detailed plan for the final verification. This step is guided by the Adapt Prompt (P_p) (refer to Appendix A, Figure 7 for the prompt used). This step mirrors the process of formulating a specific plan within a broad domain, incorporating the unique details of the situation at hand.

We mimic this scenario by prompting the language model to devise a plan based on the selected reasoning module. More specifically, we prompt the language model to implement a step-by-step reasoning plan by adapting the selected reasoning modules in JSON format, where each key represents a question, and the corresponding value is the answer to that question. This step also gives insights into how smaller LLMs reason over complex problems.

$$D_a = P_p(\{Q, R, C, D_s, \tau\})$$

Here the inputs to the Plan Prompt (P_p) are: Question (Q), Response (R), context (C), set of selected reasoning module descriptions (D_s) and the Task Definition τ and the output is a step-by-step reasoning plan (D_a).

EXECUTE STEP: In this step, we use the output plan from the previous step to reach the final verdict.

$$O = E_p(\{Q, R, C, D_a, \tau\})$$

Here, the inputs to Execute Prompt (E_p) are Question (Q), Response (R), context (C), and the generated plan (P_p), Task Definition τ and the output (O) is *hallucinated* if the output is inconsistent with the question and *non-hallucinated* if the answer is consistent with the question.

Temperature Variation and Majority Voting: To mimic human-like reasoning, we must account for the multiple aspects of the reasoning plans. Different situations might call for a creative approach, while others require a more deterministic, straightforward solution. To capture this diversity in thinking, we introduce the **temperature variation and majority voting** mechanism, which allows the model to explore both creative and deterministic reasoning processes and then reaches the final output using a majority voting system.

Temperature (T) is a key parameter in natural language processing models that controls the "creativity" of the generated response. The temperature value ranges from 0 to 1. The temperature setting influences the probabilities generated by the softmax function, adjusting how much weight is given to less likely tokens.

The equation for temperature sampling is given below:

$$\text{softmax}(x_i) = \frac{e^{x_i/T}}{\sum_j e^{x_j/T}} \tag{1}$$

By adjusting the temperature, we can influence the extent to which the model produces more diverse or predictable responses. Higher temperature encourages more creativity, while lower temperature encourages more deterministic output. We choose to account for both cases by introducing a majority voting mechanism in this setting. Temperature voting consists of two stages:

1. **Temperature Variation:** In this stage, the three steps (Select, Plan and Execute) are executed at five temperature values, starting at 0.1 and progressing through 0.2, 0.3, 0.4, and 0.5. This variation allows the model to generate both deterministic and creative responses.

2. **Majority Voting:** After generating responses at different temperatures, we apply majority voting to the answers. This step aggregates the model's responses and resolves the final output based on the majority consensus, reflecting the model's confidence in the correctness or incorrectness of its response within the context provided.

5 Experimental setup

In this section, we discuss the experiment designs of our framework and all the hyper parameters necessary to reproduce the results.

5.1 Models

We use Llama 3.1-8B (Grattafiori et al., 2024), Gemma 2-9B(Team et al., 2024), and Mistral-v0.3-7B (Jiang et al., 2023) models, all instruction tuned variants. We specifically chose the instruction-tuned model due to its ability to adhere to the explicit instructions mentioned in the prompt. We have performed our experiments on NVIDIA A100 80GB GPUs with max new tokens 1000, and top p value of 1 in a single inference setting.

5.2 Baselines

We compare Self-Resolve with other methods described below:

Vanilla: In this baseline, we prompt the model to generate the output without giving any explicit reasoning instructions or heuristics required for the process of verification. This experiment is performed to assess if LLMs can reason over multiple pieces of information independently.

Chain of Thought (CoT): This approach adds the phrase "Let's think step by step" to the prompt, as in (Wei et al., 2022), to guide the model through breaking down multi-hop queries for reasoning.

CoT with Temperature Voting: The concept of incorporating temperature voting with CoT is aimed at assessing a model's confidence in its generated outputs as the level of creativity progressively increases. We use the CoT prompt as discussed previously and prompt the model five times with temperature values 0.1, 0.2, 0.3, 0.4, and 0.5. The purpose of introducing this baseline is to establish a direct comparison with our meta-prompting setup, where temperature voting is employed to assess the model's reasoning process.

Self Check: In this technique, the LLM is prompted to generate a diverse set of reasoning paths and then choose the most consistent answer from the final answers. We keep the temperature to 1 and prompt the model 20 times similar to Manakul et al. (2023) and then choose the most consistent answer as the final answer. Using this setup we aim to mimic the Self-checkGPT work where a reference text is present.

Query Decomposition: One of the most intuitive heuristics to verify the multi-hop is to break the given query into sub-queries and then collate the answers from individual queries to verify the given answer. We implement this technique in a three-stage process. First, we prompt the model to decompose the question into multiple independent sub-questions. Second is the answer stage, where answers to these sub-questions are found. In the final stage, the given question's answer is verified by comparing it with the supporting context, which was generated in the second stage by answering the independent sub-questions.

GPT4-Omni: We prompt GPT-4o (OpenAI et al., 2024) (November 20, 2024 release) to verify the answer to a given task. This benchmarks the reasoning capabilities of large LLMs and allows us to compare their performance with smaller LLMs enhanced by the proposed method.

6 Results

In this section, we discuss the results and some general observations of the proposed approach. The Table 2 compares the performance of various methods across different models (Llama 3.1 8B, Gemma 2 9B, Mistral 7B, and GPT4-Omni) for tasks requiring reasoning over 2, 3, and 4 hops. Performance metrics include Precision (P), Recall (R), and F1 score (F1). In general, the proposed Self-Resolve outperforms other methods in most cases across most models and hop levels in terms of F1 score, indicating its superior ability to balance precision and recall effectively. Query Decomposition generally underperforms compared to other methods, with significant drops in F1 scores, especially as the number of hops increases.

Model Specific Insights: We describe our analysis of the predictive behaviour of the individual predictive models below:

Llama 3.1 8B: Performs poorly with methods like Vanilla, CoT, and Self-Check, having very low recall and F1 scores. However, Self-Resolve achieves the best results (F1 of 0.92 for two hops, 0.77 for three hops, and 0.78 for four hops). While using the baselines, we note that the model is strongly biased towards the "Incorrect" label. This is mainly due to the model's tendency to explicitly search for the given answer in the relevant information, which is very unlikely in our setup. Our method Self-Resolve, generates explicit information by capturing the relevant parts present in the given information which are required to answer correctly. This can explain the significant gains in the proposed approach.

Gemma2-9B: Demonstrates strong performance overall, with higher F1 scores compared to other models and also outperforms GPT4-Omni using the proposed methods by 5%. We note that 'Self-Resolve' achieves the best performance in F1 scores for two hops and three hops while showing competitive performance with other baselines in the four-hop settings.

Mistral-7B: Shows moderate performance compared to other models (lower than Gemma2-9B and better than Llama3.1 8B) with a noticeable improvement when using "Self-Resolve" for 2-hop and 3-hop verification while showing competitive results when somewhat similar performance with other baselines in 4-hop setting.

Impact of Hops: We observe a non-monotonic trend in the performance when we increase the total hops in the given input query. More specifically, We observe that most models perform really well in 2 hop setting and worse in 3 hop setting even when compared to 4 hops setting. Upon analysis, we note that the instances in the three-hop setting had significantly more uncertainty and non-linearity in hop arrangement. This may have reduced the complexity of the 4-hop compared to the 3-hop data instances.

7 Error Analysis

Lack of Pragmatic Understanding: we observe that some instances in the dataset lack essential hints required to infer relationships between the target entities. In the given example (Figure 7), the reasoning plan provided by the LLM for marking the answer as incorrect states that, based on the available information, C*abo Delgado Province does not border Niassa Province*. However, it fails to infer that *if the Niassa Reserve spans both Cabo Delgado Province and Niassa Province, these provinces must share a border*. Consequently, the LLM outputs "Incorrect" claiming the given answer is wrong despite it being correct. This highlights the limitation of LLMs in pragmatic understanding where the query requires information which is not present in the content.

Lack of Pragmatic Understanding

Question: What province shares a border with the province where Lago District is located?
Answer: Cabo Delgado Province
Available Information: **Niassa Reserve is a nature reserve in Cabo Delgado Province and Niassa Province**, Mozambique. Covering over, it is the largest protected area in the country. The reserve is part of the Trans-Frontier Conservation Area and links to the Tanzanian Lukwika-Lumesule Game Reserve. It will connect to the Lake Niassa Reserve when it is completed.', '**Lago District is a district of Niassa Province** in north-western Mozambique. The principal town is Metangula. Lago District is bordered to the west by Lake Nyasa, on the south by Lichinga, and on the east by Sanga District.

Correct Context Transfer: In several scenarios, the model successfully executed the planning step for verification but failed to identify that the initial answer provided was incorrect. During the planning stage, the model correctly deduced the right answer and clearly articulated the reasoning steps to reach it. This contextual information increased the semantic significance of the correct answer more than the original answer. The model provided the verification verdict for the correct answer instead of the original answer leading to the incorrect verification. In the example reasoning plan (Figure 2) given below, the provided answer was **'Aaja Nachle'** which was incorrect, the LLM was able to deduce correctly the correct answer from the available information but failed to compare the answer with the given context. By skipping this verification step, the model incorrectly marked the given answer as correct, even though it was actually incorrect.

Correct Context Transfer

Question: What is the movie starring Preity Zinta, Madhuri Dixit and the spouse of Kajol?
Answer: Aaja Nachle
Reasoning plan (Gemma 2-9B): Here's how we can adapt the "Chain of Thought" reasoning approach to solve this task:
1. Identify Key Information:
Question: *We need to find a movie starring Preity Zinta, Madhuri Dixit, and Kajol's spouse.*
Available Information: We have a biography of Anil Devgan (Kajol's brother-in-law) and details about the movie *Yeh Raaste Hain Pyaar Ke*.
2. Chain of Thought
1. Kajol's spouse is Ajay Devgan.
2. *Yeh Raaste Hain Pyaar Ke* stars Madhuri Dixit and Preity Zinta.
3. *Yeh Raaste Hain Pyaar Ke* also stars Ajay Devgan.
3. Conclusion Since all the required actors are present in the movie ***Yeh Raaste Hain Pyaar Ke***, the answer is **correct.**

Figure 2: Context Transfer: This example demonstrates the transfer of correct reasoning context to the execute stage directing to incorrect verification of the answer

8 Conclusion and Future Work

In this work, we introduce the Self-Resolve framework, which leverages the meta-reasoning capabilities of smaller language models (LLMs). Our approach aims to extend the reasoning capabilities of smaller LLMs by resolving the correct output through majority voting on stochastically sampled reasoning structures. We demonstrate that meta-reasoning with smaller LLMs is feasible and shows significant improvements in fact verification tasks within a multi-hop setup, where the model utilizes its own reasoning plans. Looking ahead, the proposed method can be extended to mitigating hallucinations in language models. Future work will explore combining multiple LLMs from different families to enhance hallucination detection and mitigation further.

9 Limitations

Despite the improvements achieved through our framework for fact verification, several limitations remain. One significant challenge is pragmatic reasoning, which involves scenarios where world knowledge is essential for validating outputs. Models, particularly those with fewer parameters, often lack sufficient training data to effectively address the pragmatic aspects of a query. Additionally, the proposed framework relies heavily on the reasoning capabilities of the language models considered. Therefore, language models with poor reasoning capabilities might not be suitable for the proposed approach.

References

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

Simran Arora, Avanika Narayan, Mayee F Chen, Laurel Orr, Neel Guha, Kush Bhatia, Ines Chami, Frederic Sala, and Christopher Ré. 2022. Ask me anything: A simple strategy for prompting language models. *arXiv preprint arXiv:2210.02441*.

Isabelle Augenstein, Timothy Baldwin, Meeyoung Cha, Tanmoy Chakraborty, Giovanni Luca Ciampaglia, David Corney, Renee DiResta, Emilio Ferrara, Scott Hale, Alon Halevy, et al. 2024. Factuality challenges in the era of large language models and opportunities for fact-checking. *Nature Machine Intelligence*, 6(8):852–863.

Alex Chandler, Devesh Surve, and Hui Su. 2024. Detecting errors through ensembling prompts (deep): An end-to-end llm framework for detecting factual errors. *arXiv preprint arXiv:2406.13009*.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2023. Chain-of-verification reduces hallucination in large language models. *arXiv preprint arXiv:2309.11495*.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2023. Promptbreeder: Self-referential self-improvement via prompt evolution. *arXiv preprint arXiv:2309.16797*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, et al. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Sungho Ko, Hyunjin Cho, Hyungjoo Chae, Jinyoung Yeo, and Dongha Lee. 2024. Evidence-focused fact summarization for knowledge-augmented zero-shot question answering. *arXiv preprint arXiv:2403.02966*.

Xin Liu, Farima Fatahi Bayat, and Lu Wang. 2024. Enhancing language model factuality via activation-based confidence calibration and guided decoding. *arXiv preprint arXiv:2406.13230*.

Potsawee Manakul, Adian Liusie, and Mark JF Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *arXiv preprint arXiv:2303.08896*.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. *arXiv preprint arXiv:2305.14251*.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew

Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Joe Stacey, Pasquale Minervini, Haim Dubossarsky, Oana-Maria Camburu, and Marek Rei. 2024. Atomic inference for nli with generated facts as atoms. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 10188–10204.

Liyan Tang, Philippe Laban, and Greg Durrett. 2024. Minicheck: Efficient fact-checking of llms on grounding documents. *arXiv preprint arXiv:2404.10774*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, et al. 2024. Gemma 2: Improving open language models at a practical size. *Preprint*, arXiv:2408.00118.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multihop questions via single-hop question composition. *Preprint*, arXiv:2108.00573.

Yuxia Wang, Minghan Wang, Muhammad Arslan Manzoor, Fei Liu, Georgi Georgiev, Rocktim Das, and Preslav Nakov. 2024. Factuality of large language models: A survey. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19519–19529.

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

Joonho Yang, Seunghyun Yoon, Byeongjeong Kim, and Hwanhee Lee. 2024. Fizz: Factual inconsistency detection by zoom-in summary and zoom-out document. *arXiv preprint arXiv:2404.11184*.

Pei Zhou, Jay Pujara, Xiang Ren, Xinyun Chen, Heng-Tze Cheng, Quoc V Le, Ed H Chi, Denny Zhou, Swaroop Mishra, and Huaixiu Steven Zheng. 2024. Self-discover: Large language models self-compose reasoning structures. *arXiv preprint arXiv:2402.03620*.

A Self-Resolve Prompts

In this section, we discuss all the prompts used in the proposed framework (Self-Resolve). We discuss the prompts used for generating the Task definition (Figure 3), Select step (Figure 5), Plan step (Figure 7) and Execute step (Figure 8). We also show the 3 different reasoning modules (Figure 4).

Task Definition

Given a question and an answer, verify if the provided answer is accurate with respect to the question based on available information. Output "verificatioin result": "correct" if the answer is correct with respect to the question; otherwise, output $“verification\ result” : “incorrect”$.

```
Question: {}
Answer: {}
Given Information: {}

```

Figure 3: Task Definition given to the model in the prompts

Reasoning Modules

1. Try Critical Thinking: This style involves analyzing the problem from different perspectives, questioning assumptions, and evaluating the evidence or information available.

2. Try Self Talk: This approach entails engaging in an internal dialogue to break down the provided question, and confirming the correctness of the answer.

3. Devise a plan to verify if the given answer is correct or incorrect. Be very precise and do not mention any unnecessary steps or any extra information. Follow the plan to step by step to get the answer.

Figure 4: Reasoning Modules- These reasoning modules were used in the proposed pipeline. Only one these module is selected for a given query in further step

Select Prompt

Assume you are an expert in selecting the most appropriate reasoning modules for fact verification.
You have to select only one of the reasoning module that are crucial for solving the below task from the given set of reasoning modules:
<Task>
$\{Task\}$
</Task>

Reasoning Modules set:
$\{reasoning\_modules\}$
Do not output anything else, only print the selected modules.

Figure 5: Select Prompt- This prompt was use to select the reasoning module specific to the query given in the input.

B Dataset Creation:

In this section, we describe the prompts used to create an extension of the MusiQue dataset.

Data generation Prompt

Provide four different options which are plausible but incorrect answers to the given question. You have to provide only these options. Do not output anything else. You can follow the given examples: Example 1: Question: Who directed the movie "Inception"? Answer: Christopher Nolan Output: [Steven Spielberg; Quentin Tarantino; Martin Scorsese; Ridley Scott] Example 2: Question: What is the capital city of Brazil? Answer: Brasília Output: [Rio de Janeiro; São Paulo; Buenos Aires; Lima]
Question: $\{\}$
Answer: $\{\}$

Figure 6: Initial Data generation pipeline- This prompt was used for generating the options for a given set of questions, answers and relevant information.

Plan Prompt

Assume you are an expert in adapting the reasoning modules for a given problem.
Adapt the given reasoning module into an improvised plan to solve the given task better:
<Task>
$\{Task\}$
</Task>

Reasoning module descriptions:
$\{selected\_reasoning\_modules\}$

Figure 7: Plan Prompt- This prompt was used to adapt the selected reasoning module into an plan specific to query.

Execute Prompt

Assume you are an expert in structured reasoning and fact verification.
For the given task: <Task>
$\{Task\}$
</Task>
Execute the given reasoning plan in a step-by-step manner to verify the given answer to the question using the given information.
The output must be in JSON format only and make sure the final result is in the form "verification result": "correct" if the answer is correct with respect to the question; otherwise, output "verification result": "incorrect". Note that output must be either ’correct’ or ’incorrect’.
Given reasoning plan:
$\{adapted\_reasoning\_modules\}$

Figure 8: Execute Prompt- This prompt was used to execute the reasoning plan in a step by step manner to get the final verification result.

C Baseline Prompts

In this section, we discuss all the baseline prompts used in this work.

C.1 Vanilla

Vanilla Prompt

You are an expert in verifying the correctness of answers. Given a question, an answer, and relevant information, determine if the answer correctly addresses the question based on the provided information.
If the answer is correct, output "Correct." If the answer is incorrect, output "Incorrect."
Only output one of these two values without any additional text.

Question: {}
Answer: {}
Given Information: {}

Output:

Figure 9: Vanilla Prompt- In this prompt, we simply provide question, answer and relevant information and ask to verify the answer with respect to the relevant information given.

C.2 Chain of thought

Chain of Thought Prompt

Assume you are an expert in verifying the facts. Given a question and an answer, verify if the provided answer accurately responds to the question based on available information. Let’s think step by step.
You have to output ’Correct’ if the question is answered correctly with respect to the given Information; otherwise, output ’Incorrect’.Please note that Output can take only 2 possible values, either "Correct" or "Incorrect".

Only generate the output, do not print any other text.

Question: {}
Answer: {}
Given Information: {}

Output:

Figure 10: CoT Prompt- In this prompt, we follow the convention of (Wei et al., 2022) and include the phrase ‘Let’s think step-by-step.

C.3 Query Decomposition

In this setup, we aim to decompose the given query into smaller subpart and reconstruct the answer after answering each sub-queries individually. We describe our three staged process below:

Decompose Prompt

You are an AI assistant designed to decompose complex questions into simple, stand-alone sub-questions.
Given a multi-hop question, break it down into a list of minimal, independent sub-questions. Each sub-question should address only one aspect or step required to answer the original question without relying on information from other sub-questions.
Question: `{{given_question}}`
Output Format:
```
["sub-question1","sub-question2",
"sub-question3", ...]
```

Figure 11: Decompose Prompt- This prompt was used to decompose the given multi-hop queries to smaller sub queries which are independent from other sub-queries

Answer Prompt

You are a helpful AI assistant tasked with answering questions using provided information. For each question below, provide a concise and accurate answer based on the given information. Ensure each response directly addresses its corresponding question.
Questions: `{{decomposed_question}}`

Given Information:
`{{given_information}}`
Output Format:
```
["question":"subquestion1",
"answer":"answer1","question":
"sub-question2","answer":"answer2"...]
```

Figure 12: Answer Prompt- This prompt was used to answer the decomposed queries to generate the final answer

Verify Prompt

You are an AI assistant trained to evaluate answers. Your task is to verify whether a provided answer to a main question is accurate based on answers to decomposed sub-questions. Your output should be strictly `"Correct"` if the answer is accurate, or `"Incorrect"` if it is not. Do not include any additional text.

Main Question: `{{given_question}}`
Main Answer: `{{given_answer}}`
Supporting_Information (from_sub-questions):
`{{decomposed_question_answer}}`

Figure 13: Verify Prompt- This prompt was used to verify the final answer based on the decomposed queries and their generated answeres.

GPT4 Prompt

Given a question, an answer, and relevant information, determine if the answer correctly addresses the question based on the provided information. If the answer is correct, output "Correct" If the answer is incorrect, output "Incorrect" Only output one of these two values without any additional text.

Question: {}
Answer: {}
Given Information: {}

Figure 14: GPT4 Prompt- This prompt was used for the verification process by GPT4-Omni.