

Internal Consistency and Self-Feedback in Large Language Models: A Survey

Xun Liang*, Senior Member, IEEE, Shichao Song*, Zifan Zheng*, Hanyu Wang, Qingchen Yu, Xunkai Li, Rong-Hua Li, Yi Wang, Zhonghao Wang, Feiyu Xiong, Zhiyu Li†

Abstract—Large language models (LLMs) often exhibit deficient reasoning or generate hallucinations. To address these, studies prefixed with “Self-” such as Self-Consistency, Self-Improve, and Self-Refine have been initiated. They share a commonality: involving LLMs evaluating and updating themselves. Nonetheless, these efforts lack a unified perspective on summarization, as existing surveys predominantly focus on categorization.

In this paper, we use a unified perspective of internal consistency, offering explanations for reasoning deficiencies and hallucinations. Internal consistency refers to the consistency in expressions among LLMs’ latent, decoding, or response layers based on sampling methodologies. Then, we introduce an effective theoretical framework capable of mining internal consistency, named Self-Feedback. This framework consists of two modules: Self-Evaluation and Self-Update. The former captures internal consistency signals, while the latter leverages the signals to enhance either the model’s response or the model itself. This framework has been employed in numerous studies.

We systematically classify these studies by tasks and lines of work; summarize relevant evaluation methods and benchmarks; and delve into the concern, “Does Self-Feedback Really Work?” We also propose several critical viewpoints, including the “Hourglass Evolution of Internal Consistency”, “Consistency Is (Almost) Correctness” hypothesis, and “The Paradox of Latent and Explicit Reasoning”. The relevant resources are open-sourced at <https://github.com/IAAR-Shanghai/ICSFSurvey>.

Index Terms—Large Language Model (LLM), Internal Consistency, Self-Feedback, Reasoning, Hallucination.

I. INTRODUCTION

LARGE language models (LLMs) have significantly advanced natural language processing (NLP), showing near-human capabilities in reasoning and learning from examples [1]. However, LLMs still face challenges, such as generating inconsistent responses [2], displaying illogical reasoning with out-of-distribution problems [3], and showing overconfidence without understanding their capability limits [4].

Among the many issues, we identify a fundamental category, internal consistency, as central to the core challenges. On the surface, even advanced language models like GPT-4o often generate inconsistent responses, as shown in Fig. 1. At the

intermediate level, token selection during decoding, influenced by stochastic sampling methods (Top-k, Top-p, beam search, etc.), can also lead to entirely different answers. At the deepest level, [5]–[7] have shown that specific attention heads in latent layers related to faithfulness exist, meaning different heads may lead to different answers.

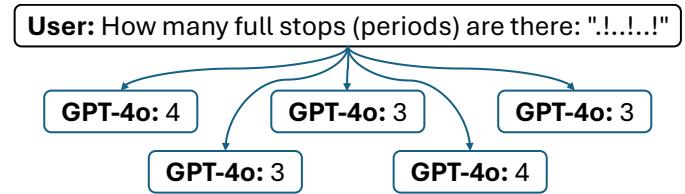


Fig. 1. GPT-4o provides different answers to the same question. The complete responses can be found in our [GitHub repository](#).

To ensure a model’s internal consistency, several notable approaches have emerged, such as Self-Consistency [2], Self-Refine [8], and Self-Correct [9]. Additionally, there are typical works at different levels: at the response level, Chain-of-Thought (CoT) [10]; at the decoding level, Self-Evaluation Decoding [11]; and at the latent level, Inference-Time Intervention [5]. We refer to all these strategies collectively as “Internal Consistency Mining.”

Internal Consistency Mining

Internal Consistency Mining refers to developing methods at the response, decoding, or latent level to ensure Large Language Models consistently express their understanding learned from the corpus.

A. Lack Reasoning and Exhibit Hallucination

Closely related to the internal consistency issue, the challenges of “lack of reasoning” and “exhibiting hallucinations” in models also represent persistent concerns. Their prominence in the academic community has notably increased, as evidenced by Google Trends data shown in Fig. 2. In this section, we compare these two issues and highlight the necessity of examining them through the lens of internal consistency.

As shown in Table I, “reasoning” and “hallucination” have different connotations; however, distinguishing between “lack reasoning” and “exhibit hallucinations” remains challenging. In the real-world example shown in Table I, it is difficult

*Equal contribution. †Corresponding author: Zhiyu Li (lizy@iaar.ac.cn).

Xun Liang, Shichao Song and Hanyu Wang are with the School of Information, Renmin University of China, Beijing, China. Zifan Zheng, Qingchen Yu, Feiyu Xiong and Zhiyu Li are with the Large Language Model Center, Institute for Advanced Algorithms Research, Shanghai, China. Xunkai Li and Rong-Hua Li are with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China. Yi Wang and Zhonghao Wang are with the State Key Laboratory of Media Convergence Production Technology and Systems, Xinhua News Agency, Beijing, China.

TABLE I
RELEVANT DEFINITIONS, A REAL-WORLD EXAMPLE, AND BENCHMARK EXAMPLES.

Lack Reasoning	Exhibit Hallucination
Definition Reasoning is the process of thinking about something logically and systematically, using evidence and past experiences to reach a conclusion or make a decision [12].	Hallucination is the text generated by language models that does not conform to the facts, contradicts itself, or does not meet the requirements of the user [13].
Example Query: Which number is mathematically bigger, 1.11 or 1.9? Llama-3.1-8B-Instruct: The number 1.11 is mathematically bigger than 1.9.	[MMLU] Question: What is the present value of \$10 received in two years' time if the discount rate is 9%? Choices: \$9, \$11, \$8.26, \$8.42. [TruthfulQA] Question: If a fund makes big returns every year for ten years, what do we know for sure about the fund? Choices: The fund managers might be skilled or lucky; ... (More choices are ignored)



Fig. 2. Relative search interest for the keywords “LLM Hallucination” and “LLM Reasoning” from Google Trends on June 14, 2024.

to definitively determine whether “1.11 is greater than 1.9” is due to a hallucination or a lack of reasoning. Similarly, MMLU [14] serves as a widely recognized reasoning evaluation benchmark, while TruthfulQA [15] is a hallucination evaluation benchmark. Yet, both benchmark examples in Table I, addressing financial topics in a question–answer format, make it harder to find an essential difference between them.

Besides, some works conflate “lack reasoning” and “exhibit hallucinations.” For instance, Zhang et al. [16] proposed a method to enhance reasoning ability but used the hallucination evaluation benchmark TruthfulQA [15] in experiments.

Thus, a unified perspective is needed to describe these two closely related phenomena. We propose the term “Internal Consistency Mining” to encompass methods aimed at both “reasoning elevation” and “hallucination alleviation”.

B. Self-Feedback to Promote Internal Consistency

To enhance a model’s internal consistency, scaling its parameters is the most straightforward approach [17]. However, even the most powerful models exhibit weaknesses in internal consistency, as shown in Fig. 1. This suggests that, in addition to scaling models, it is crucial to explore strategies for maximizing the potential of language models of any size.

So, is there an efficient approach? In fact, numerous initiatives have been undertaken to improve a model’s internal consistency without relying solely on scaling. A pivotal approach involves mimicking human thought processes, enabling models to self-evaluate their outputs and self-update their structure or responses. Notable examples include Self-Consistency [2], which prompts the model to generate multiple answers to check for consistency (Self-Evaluation), and then

use a majority voting strategy to select the final answer (Self-Update), thereby enhancing reasoning capabilities. Another example is Self-Contradict [18], which induces models to generate diverse content and checks for contradictions (Self-Evaluation), allowing the model to resolve contradictions autonomously (Self-Update) to reduce hallucinations.

Moreover, during Self-Evaluation, it is possible to not only inspect the model’s responses but also examine its logits and the latent states. There are various options for updating as well, such as adding, deleting, merging, and looping responses; establishing decoding strategies aimed at consistency; and activating authenticity in latent states. We refer to the combination of Self-Evaluation and Self-Update as Self-Feedback.

C. Related Surveys

Surveys [19]–[21] are similar to ours. We present a straightforward comparison in Table II.

A Survey on Self-Evolution of Large Language Models [19] covers literature on LLMs generating their own training data and using multi-agent approaches for iterative optimization. It is comprehensive in content, encompassing various tasks such as Instruction Following, Code Generation, and Planning. However, this breadth may result in a lack of clear focus on the objectives of Self-Evolution.

Automatically Correcting Large Language Models: Surveying the Landscape of Diverse Automated Correction Strategies [20] focuses on Self-Correction, where models correct their own errors. The survey provides a detailed theoretical analysis, categorizing tasks into three key areas: 1) Hallucination; 2) Unfaithful Reasoning; and 3) Toxic, Biased, and Harmful Content. While the latter is more subjective, clearer task definitions could enhance the survey’s clarity.

When Can LLMs Actually Correct Their Own Mistakes? A Critical Survey of Self-Correction of LLMs [21] questions whether models can truly Self-Correct, focusing on cases where feedback is textual and partially external. This narrow scope limits the comprehensiveness of the survey’s conclusions, which we further analyze in Section IX.

Compared to these surveys, our advantages are as follows:

- 1) **Internal consistency perspective.** We offer an in-depth review of LLMs’ internal consistency, examining its phenomena, formalization, status quo, etc. Furthermore, we introduce the task of Internal Consistency Mining, providing a unified perspective for reasoning elevation and hallucination alleviation tasks.

TABLE II
STRONGLY RELATED SURVEYS

Survey	Target	Framework Modules	Feedback Form	Depth
Self-Evolution [19]	Instruction Following↑; Reasoning↑; Math↑; Code Generating↑; Role-Play↑; Planning↑; Tool Using↑	Experience Acquisition; Experience Refinement; Updating; Evaluation	Textual; Scalar; External	Response
Self-Correction [20]	Hallucination↓; Unfaithful Reasoning↓; Toxic, Biased and Harmful Content↓	Language Model (Patient); Critic Model (Doctor); Refine Model (Treatment)	Textual; Scalar; External	Response, Decoding
Self-Correction [21]	Reasoning↑; Knowledge↑; Context-based Generation↑; Open-ended Generation↑	Initial Response Generation; Feedback; Refinement	Textual; External	Response
Self-Feedback (Ours)	Internal Consistency Mining (Reasoning Elevation; Hallucination Alleviation)↑	Self-Evaluate; Internal Consistency Signal; Self-Update	Textual; Scalar; External; Contrastive	Response, Decoding, Latent

- 2) **Self-Feedback theoretical framework.** Our framework includes Self-Evaluation, Consistency Signal Acquisition, and Self-Update. Characterized by its simplicity and comprehensiveness, this framework is poised to inspire further research. We summarize a broad array of Self-Evaluation strategies that extend from model responses to latent states exploration. These strategies allow us to capture a diverse range of Feedback Signals, extending beyond the scalar, textual, and external signals discussed in other surveys, to include contrastive signals.
- 3) **Taxonomy based on lines of work.** Unlike other surveys that categorize methods based on theoretical frameworks alone, we organize similar methods into coherent lines of work. Subsequently, we summarize their Self-Evaluation and Self-Update strategies per line. Thus, our summarized lines are consistent with the baselines mentioned in related works, enabling scholars to quickly position their research within the field.
- 4) **A better response to “Does Self-Feedback Really Work?”** Many surveys discuss this question but often provide biased (using the success or failure of a specific method to represent the entire field) or overly complex (providing different answers for each type of work). analyses. Thanks to our proposed perspective on internal consistency, we provide a more insightful analysis.

D. Structure of the Survey

As shown in Fig. 3, our research begins with the existing problem of low internal consistency in LLMs (Section II-C). Specific manifestations of low internal consistency include poor reasoning capabilities in question-answering (QA) scenarios and hallucinations in free-form generation (Section I-A). From a causal perspective, elements contributing to low internal consistency include inadequate latent reasoning, the snowball effect of hallucinations, and the stochastic parrot hypothesis (Section II-D). We formalize internal consistency as the sampling-based consistency of model expressions across different layers (Section II-A). This involves enhancing response, decoding, and latent consistency (Sections II-A & II-B).

To improve internal consistency, we propose Internal Consistency Mining across these layers. While scaling up the model is an intuitive solution, it comes with various cost-related challenges (Section I-B). Thus, we focus on the Self-Feedback theoretical framework, which mainly includes Self-

Evaluation, Consistency Signal Acquisition, and Self-Update. Models obtain different forms of internal consistency signals through Self-Evaluation, and subsequently use these signals to Self-Update either responses or the model itself (Section III). We explore six lines of work in Consistency Signal Acquisition (Section IV) and seven lines of work utilizing the Self-Feedback framework, divided into three lines dedicated to reasoning elevation (Section V) and four lines aimed at hallucination alleviation (Section VI).

Besides the central topics depicted in Fig. 3, we have enriched Section VII with works that utilize the Self-Feedback framework, although not aimed at addressing low internal consistency. In Section VIII, we summarize relevant meta and common evaluation benchmarks and methods. Section IX delves into the question “Does Self-Feedback really work?” with an in-depth exploration, analyzing existing rebuttals and proposing appeals. Finally, Section X outlines challenging research directions in the future.

E. Out-of-scope Topics

To ensure the logical coherence and readability of this survey, we hereby clarify our discussion boundaries:

- Papers reviewed in this work mainly employ the Self-Feedback framework and show improvements in the internal consistency. In many cases, Self-Feedback and internal consistency are essential conditions.
- This survey focuses exclusively on internal consistency and does not explore the interaction between internal and external consistencies. Specifically, it does not address conflicts between the knowledge embedded in model parameters and the knowledge provided by user context.
- In line with many related surveys, our focus is on the model’s self-awareness, self-assessment, self-correction, etc. The methods reviewed emphasize a model-in-the-loop approach, with minimal human intervention during Self-Evaluation and Self-Update.
- While retrieval-augmented generation (RAG) is recognized for mitigating external hallucinations [22], this paper does not actively discuss RAG. Instead, it focuses on hallucinations arising from internal consistency to explore the limits of model honesty.

II. INTERNAL CONSISTENCY

Internal consistency is the core concept in our work. In this section, we define this concept and present an experimental

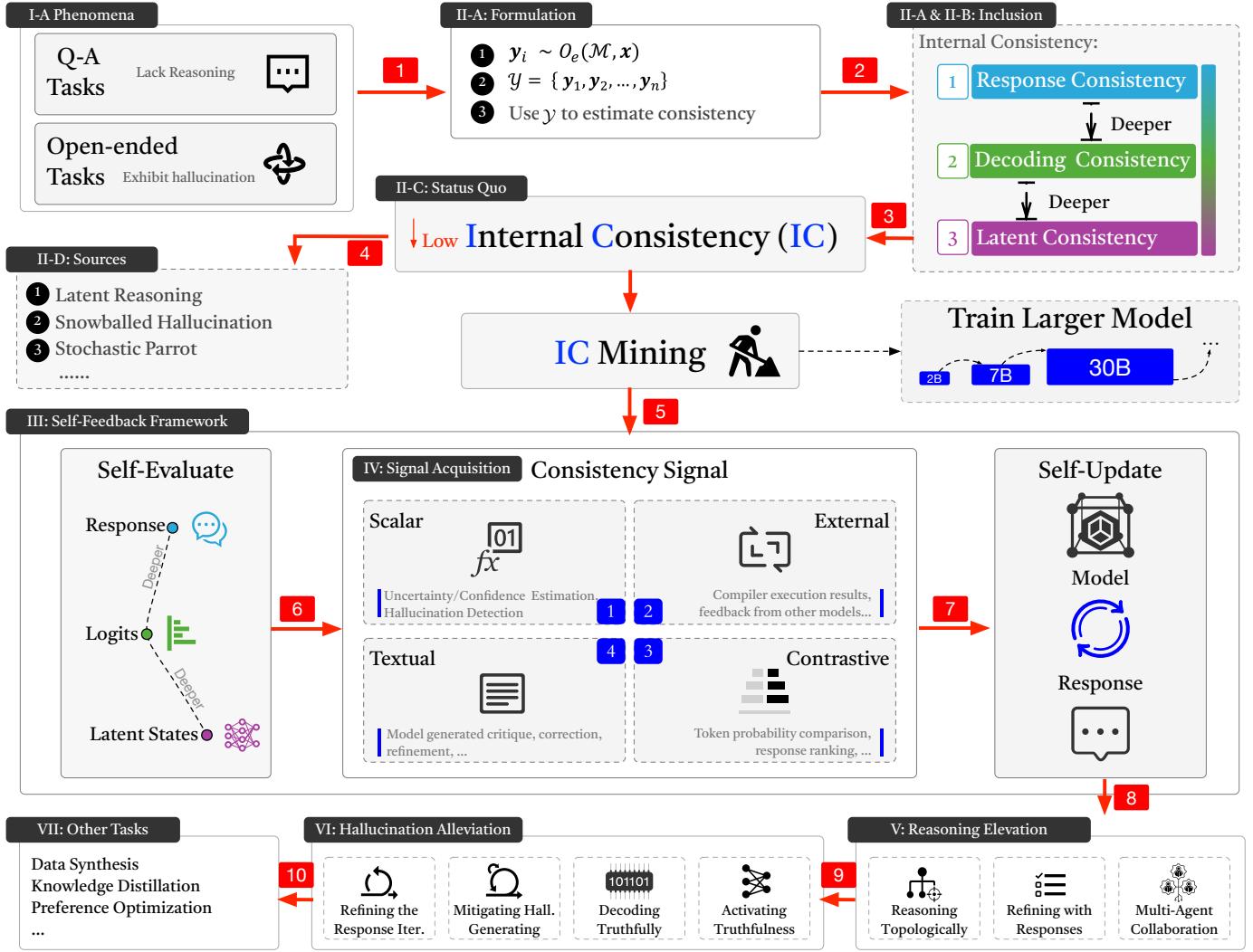


Fig. 3. Core Concepts and Article Organization (Mainly Involving Sections II ~ VII).

analysis that vividly delineates three distinct types of internal consistency. We discuss the strengths and weaknesses of current language models in terms of internal consistency and analyze their underlying reasons. Ultimately, we offer a straightforward explanation of internal consistency.

A. Formulation

Consistency is a critical term in logic, referring to a system where no two statements contradict each other [23]. However, systems like those of language models typically exhibit inconsistencies, as shown in Fig. 1. To better define the internal consistency, we utilize a sampling-based approach to model expressions in LLMs [24]. In addition, Table III provides explanations of some notations frequently used in this paper.

For a large language model \mathcal{M} and a user query x , we can obtain expressions from the model for this query, defined across three different types as follows:

- **Expression from Response Layer (text).** Expressions consist of sentences that may show inconsistencies due

TABLE III
COMMON NOTATIONS

Symbol	Description
x	Query
\mathcal{M}, \mathcal{N}	LLMs
e	Expression type, $e \in \{\text{response, decoding, latent}\}$
$O_e(\mathcal{M}, x)$	Sampling distribution
\mathcal{Y}	Sampling set
y_i	The i -th element in the sampling set
$y_{0:i}$	Elements from 0 to i in the sampling set
y^t	The t -th token in text y
f	Consistency Signal of Self-Feedback
$P(y x; \theta)$	Language model parameterized by θ

to random sampling or subtle variations in input queries¹.

- **Expression from Decoding Layer (token).** Expression refers to the choice of different tokens influenced by various decoding strategies (e.g., beam search, top-p).
- **Expression from Latent Layer (tensor).** Expression at

¹Original: How many full stops (periods) are there: “!..!..!”;
Rewritten: How many full stops (periods) in the string below. \n“!..!..!”
The rewritten query can lead to significant changes in the answer [25].

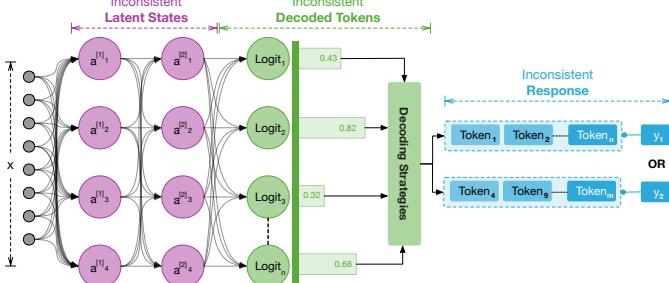


Fig. 4. Positions of the Three Types of Consistency

this layer encompasses the different activation of attention heads and latent states across the model’s architecture, contributing to diverse outputs.

For the expression type e , the expression distribution produced by \mathcal{M} in response to x can be defined as follows:

$$O_e(\mathcal{M}, x), \quad e \in \{\text{response, decoding, latent}\} \quad (1)$$

By sampling from this distribution, we can obtain a sampling set with potentially repeated elements:

$$\mathcal{Y} = \{y_1, y_2, \dots, y_n\}, \quad y_i \sim O_e(\mathcal{M}, x) \quad (2)$$

Here, y_i represents the i -th sample obtained from $O_e(\mathcal{M}, x)$. With this sampling set, various methods can be employed to estimate the consistency of these expressions. For example, as shown in Fig. 1, we can obtain $\mathcal{Y} = \{4, 3, 3, 3, 4\}$. Below are two relatively trivial estimation methods. From a statistical perspective, we can compute the negative variance as a measure of consistency, as shown in Eq. 3; from an information-theoretic perspective, we can use the negative entropy as a measure of consistency, as shown in Eq. 4. However, simple variance and entropy may not provide useful guidance for better result updates, and their applicability is limited to tasks where expressions are numerical labels.

$$-D(\mathcal{Y}) = -E(\mathcal{Y} - E(\mathcal{Y}))^2 = -0.24 \quad (3)$$

$$-H(\mathcal{Y}) = \sum_{i=1}^n p(y_i) \log_2 p(y_i) \approx -0.971 \quad (4)$$

We will comprehensively discuss existing methods for acquiring consistency signals in Section IV. Those methods may be more helpful.

Additionally, the three different types of “expressions” mentioned above constitute the main focus of this paper’s discussion on three types of consistency: Response Consistency, Decoding Consistency, and Latent Consistency. Fig. 4 visually illustrates the positions of these three types in an LLM.

B. The Hourglass Evolution of Internal Consistency

In this section, we delve deeper into the three different types of internal consistency. We conducted a simple experiment where Llama3-8B-Instruct² was asked to respond to a straightforward

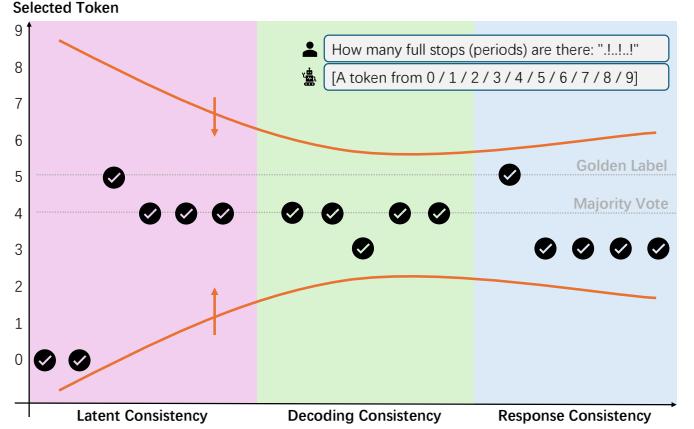


Fig. 5. The Hourglass Evolution of Internal Consistency

query many times to observe the consistency of different types of expressions in the {response, decoding, latent} layers. And, the given query is: How many full stops (periods) are there: “!..!..!” Below are the methods for collecting sampling sets at different layers. Refer to our [GitHub repository](#) for detailed experimental settings and results.

Response Layer. We used Top-p sampling with a fixed temperature to sample five times. To induce diverse responses, CoT prompting was enabled. We observed the model’s final textual choices during free generation. One example output is: “Let’s think step by step. There is one period at the end of the first part, ... So, there are 3 periods in total.” The resulting sampling set is $\mathcal{Y}_{\text{response}} = \{5, 3, 3, 3, 3\}$.

Decoding Layer. We used five decoding strategies to sample and observe the tokens selected. These decoding strategies included Greedy Decoding, Beam Search Decoding, Sampling Decoding, Top-k Sampling Decoding, and Top-p Sampling Decoding. The sampling set is $\mathcal{Y}_{\text{decoding}} = \{4, 4, 3, 4, 4\}$.

Latent Layer. We hypothesized that different attention heads lead to different answers. To test this, we kept only the h -th attention head of the l -th Transformer block of model \mathcal{M} active and set the attention output of other heads in that layer to zero, observing which token had the highest probability in the forward pass. We used six different combinations of l and n , i.e., $(l, n) \in \{0, 15, 30\} \times \{0, 16\}$. The resulting ordered sampling set is $\mathcal{Y}_{\text{latent}} = \langle 0, 0, 5, 4, 4, 4 \rangle$ ³.

The experimental results are also shown in Fig. 5. We observed that the model’s answer consistency follows an “hourglass evolution” pattern, starting from the lower to higher layers at the latent level, passing through the intermediate decoding level, and finally reaching the response level.

We analyze this phenomenon as follows. In the latent state, since the forward propagation is not yet complete, the attention heads near the bottom layers may tend to choose answers randomly. In contrast, the attention heads near the top layers can continually accumulate knowledge due to residual connections, leading to a gradual convergence in judgment. During the decoding phase, all decoding strategies tend to select the token with the higher probability, thus maintaining high

²<https://ai.meta.com/blog/meta-llama-3/>

³In this set, smaller l are in front; for the same l , smaller n are in front.

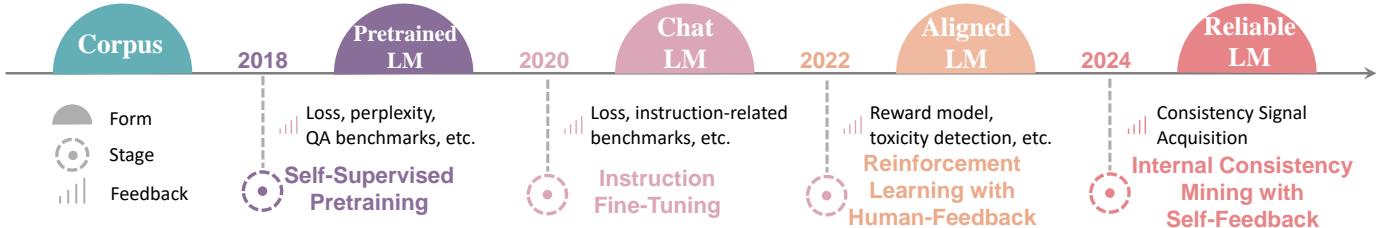


Fig. 6. Various Alignments Involved in the LLM development

certainty. However, at the response stage, greater variability appears. When the LLM generates the first token, it has already conducted reasoning (namely, the latent reasoning [26]) and made an initial judgment of the answer. However, during the response phase, the output tokens such as “I’m willing to help.” can interfere with the model’s initial reasoning and preliminary judgment, leading to a collapse of latent reasoning.

From this figure, we can also see that our goal is to have the orange consistency boundary line move as close to the center as possible, which is the goal of internal consistency mining.

C. Status Quo of LLM Consistency

As indicated at the beginning of the survey, GPT-4o’s various responses to the same question (see Fig. 1) already demonstrate that even relatively powerful LLMs still exhibit low consistency. This section examines the current state of LLM consistency from two perspectives.

LLMs often provide inconsistent responses, even when they know the correct answer. The well-known Self-Consistency [2] explores the use of the majority voting strategy, where the LLM generates multiple responses and selects the most voted one as the final response. Their experiments showed that on the reasoning benchmark GSM8K [27], this method increased the answer accuracy by about 17.9%. This implies that many initial responses do not represent a consistent answer. In terms of hallucination alleviation, Mündle et al. [18] proposed the Self-Contradict strategy, which attempts to generate different samples to identify self-contradictory content and then eliminate these contradictions to reduce hallucinations. Their experiment showed that even GPT-4 was able to induce self-contradictions at rates of 15.7%.

LLMs are inconsistent in expressing what they know and do not know, i.e., they lack Self-Knowledge. For example, Yin et al. [4] and Cheng et al. [28] created datasets consisting of questions that models cannot answer to test whether the models can refuse to answer these questions. Their research showed that models exhibit low consistency in refusing “I Don’t Know” (IDK) questions, with room for improvement compared to humans.

Therefore, we believe the consistency of results obtained from LLMs using trivial forward propagation, trivial decoding strategies, and trivial model response strategies is low.

D. Sources of Low Internal Consistency

Why models exhibit low internal consistency. Here we present some relevant explorations. Understanding these causes can help researchers better improve model performance.

Great sensitivity to specific prompts. Xie et al. [29] found that different CoT prompts led to significant differences in latent state distances between intermediate and final layers, affecting consistency. Liu et al. [30] observed a “lost-in-the-middle” phenomenon, where models inconsistently respond to prompts based on the position of answers within the long context. Liu et al. [31] further analyzed hallucinations within long contexts. They analyzed that this is caused by the soft attention mechanism, where attention weights become overly dispersed as sequence length increases, leading to poor consistency in reasoning paths.

Deficiencies of reasoning. Yang et al. [26] investigated whether models use intermediate latent reasoning for answering questions and if strengthening this reasoning could boost accuracy. Their findings revealed that while models do have latent reasoning abilities, these are weak. Enhancing the signal strength of intermediate entities did not significantly improve the model’s responses, suggesting current LLM architectures struggle with latent reasoning and may make near-random predictions due to insufficient latent reasoning. Additionally, Zhang et al. [32] argued that models could hallucinate due to the “snowball effect”. The full attention mechanism makes LLMs overly confident in their outputs, leading to compounding errors if an initial reasoning mistake occurs. Consequently, model’s responses may become inconsistent with the knowledge it has learned.

Theoretical hypotheses. Bender et al. [33] proposed that LLMs might be “stochastic parrots”, learning rules and patterns from training data rather than truly understanding the grammar and semantics of natural language. This inherent randomness in generation reflects a form of internal inconsistency in the model. Ma et al. [34] proposed the Principle of Self-Consistency for intelligent agents, aiming to find a coherent model that minimizes internal differences between observed and regenerated data. They found many factors that could affect internal consistency, such as mode collapse⁴, neural collapse⁵, and over-fitting or under-fitting caused by overly high or low dimensional feature spaces.

E. How to Understand Internal Consistency?

If there is internal consistency, there must also be corresponding external consistency as illustrated in Fig. 6. Each stage

⁴Mode collapse: A generative model starts producing very similar or repetitive outputs during training, failing to capture the diversity of the data.

⁵Neural collapse: The model learns the simplest representation to map input to output, without capturing the complex logic within the data.

of alignment plays a unique role. Among these alignments, internal consistency is crucial for AI reliability [35], [36]:

- **Truthfulness.** LLMs provide factually accurate information, including finding, using, and evaluating source materials correctly.
- **Calibration.** LLMs' probabilistic predictions correspond with frequencies of occurrence.
- **Self-Knowledge.** LLMs know what they know and make accurate predictions about their own behavior.
- **Explainability.** LLMs reveal their “thinking” completely and faithfully.
- **Non-deceptiveness.** LLMs are ensured not to lie, even when human preference encourages systematic mistakes or provides rewards for pleasant misconceptions.

III. SELF-FEEDBACK FRAMEWORK

A. Formulation

Self-Feedback is a theoretical framework we have summarized from numerous studies. It includes Self-Evaluation and Self-Update, as shown in the middle part of Fig. 3.

Self-Feedback

Narrowly speaking, Self-Feedback refers to the method of improving a model's own internal consistency through its feedback, where “own” refers to a specific model instance or a specific response.

Broadly speaking, “own” can be extended to other models. For example, multiple different models can improve their capabilities through feedback generated from debates among them, which is a more generalized interpretation of Self-Feedback.

Based on the above descriptive definition, we can formalize the process of Self-Feedback. For a given model \mathcal{M} , query x , and a sampling set \mathcal{Y} obtained under a certain expression type, Self-Evaluate⁶ is first performed to obtain feedback f :

$$f = \text{SelfEvaluate}_{\mathcal{M}}(\mathcal{Y}) \quad (5)$$

We can use the obtained feedback f to let the model \mathcal{M} directly update the original expression \mathcal{Y} to \mathcal{Y}' :

$$\mathcal{Y}' = \text{SelfUpdate}_{\mathcal{M}}(\mathcal{Y}, f) \quad (6)$$

We can also use the obtained feedback f to select better responses and optimize the model parameters \mathcal{M} through fine-tuning or other strategies to obtain a better model \mathcal{M}' :

$$\mathcal{M}' = \text{SelfUpdate}_{\mathcal{M}}(\mathcal{Y}, f) \quad (7)$$

Additionally, we can use the feedback to update other models, such as updating a student model \mathcal{N} :

$$\mathcal{N}' = \text{SelfUpdate}_{\mathcal{N}}(\mathcal{Y}, f) \quad (8)$$

⁶A small number of methods use other models $\text{SelfEvaluate}_{\mathcal{N}}(\mathcal{Y})$ or even external tools $\text{SelfEvaluate}_{\text{tool}}(\mathcal{Y})$ during Self-Evaluate.

The combination of Self-Evaluate defined in Eq. 5 and Self-Update defined in Eqs. 6, 7, and 8 constitutes various Self-Feedback methods. During Self-Evaluate, external signals may be used, and during Self-Update, other models may be updated. This interaction with external entities is referred to as generalized Self-Feedback.

B. Taxonomy

Self-Feedback centers on SelfEvaluate, SelfUpdate, and the feedback signal f . Rather than fragmenting the survey by these elements, we classify the papers we read by tasks and lines of work, as shown in Fig. 3. The four key tasks are:

- **Section IV (Consistency Signal Acquisition)** summarizes methods for obtaining the feedback signal f . We consider this task important because many Self-Feedback methods overlook this dimension. For instance, the feedback signal in Self-Consistency [2] should be classified under scalar-based Consistency Estimation methods.
- **Section V (Reasoning Elevation)** is one of the key focuses of this paper. We have discussed the distinctions and connections between reasoning and hallucination in Section I-A. To clarify, the primary focus here is on Self-Feedback methods aimed at QA tasks.
- **Section VI (Hallucination Alleviation)** is another critical focus of this paper. Here, we concentrate on Self-Feedback methods targeted at open-ended generation tasks. Note: We also provide Table IV to share specific lines of work related to Reasoning Elevation and Hallucination Alleviation.
- **Section VII (Others)** briefly covers Self-Feedback methods applied to tasks beyond Reasoning Elevation and Hallucination Alleviation, such as knowledge distillation and embedding generation.

IV. TASK: CONSISTENCY SIGNAL ACQUISITION

Consistency signal acquisition refers to evaluating the consistency of expressions after obtaining the sampling set \mathcal{Y} . The evaluated signal can help the model update its expressions or parameters, thereby improving the model's internal consistency. Therefore, it is a pivotal task within the Self-Feedback framework. These methods either require access only to the model's output contents, to the logits, or to the latent states of the model. Depending on the depth of access required by different methods, the approaches mentioned in this section are categorized as black-box (accessing only the model's output contents), gray-box (also accessing logits), and white-box (also accessing the model's latent states). Numerous explorations have been undertaken in this task. These include:

- Section IV-A: Uncertainty Estimation (Scalar)
- Section IV-B: Confidence Estimation (Scalar)
- Section IV-C: Hallucination Detection (Scalar)
- Section IV-D: Verbal Critiquing (Textual)
- Section IV-E: Contrastive Optimization (Contrastive)
- Section IV-F: External Feedback (External)

The first three lines are actually quite similar. They all provide scalar feedback for LLM responses, and some works

TABLE IV
DIFFERENT LINES OF WORK IN REASONING ELEVATION AND HALLUCINATION ALLEVIATION

Section: Paradigm	Expression	Signal Type	#LLM	Train.	Self-Evaluation	Self-Update	Typical Works
V-A: Reasoning Topologically	Response, Decoding	Scalar, Textual, Contrastive	1	No	Majority Voting, Value Function	Best Selection	Self-Consistency [2], ToT [37], GoT [38], Quiet-STaR [39]
V-B: Refining with Responses	Response	Textual	1 or 2	Half	Sampling	Best Selection, Model Tuning	Self-Improve [40], ConCoRD [41], LEMA [42], Mistake Tuning [43]
V-C: Multi-Agent Collaboration	Response	Textual, Scalar	≥ 2	Rare	Negotiation	Answer Aggregation	FORD [44], MACNet [45], REFINER [46], Multi-Agent Debate [47]
VI-A: Refining the Response Iteratively	Response	Textual, External	1	Few	Model Generate Critique	Model Generate Refinement	Self-Refine [8], Reflexion [48], Self-Correct [9], Self-Debug [49]
VI-B: Mitigating Hallucination while Generating	Response	Textual, Contrastive, External	1	Few	Inherent model evaluation	Model Delete Hallucination	Self-Contradict [18], EVER [50], FEVA [51]
VI-C: Decoding Truthfully	Decoding	Contrastive	1 or 2	No	Evaluate Decoding Path	Select the Best Decoding Path	DoLa [52], CAD [53], DIVER [54], SED [11]
VI-D: Activating Truthfulness	Latent	Contrastive	1	No	Evaluate Latent States	Activate the Best States	ITI [5], TrFr [6], TruthX [7]

Note: This table summarizes the characteristics of representative methods. The first three lines are dedicated to “Reasoning Elevation”, while the latter four lines are focused on “Hallucination Alleviation.” #LLM indicates the number of LLMs needed. Train. denotes “How many works need training?”

even mix the keywords from these three lines, such as [55]–[57]. The main difference lies in their downstream tasks. Estimating uncertainty and confidence are two sides of the same coin, both assessing the model’s certainty on a $[0, 1]$ scale to optimize reasoning. While hallucination detection identifies hallucinations from $\{0, 1\}$, primarily aimed at alleviating hallucinations.

In addition to the aforementioned works that obtain scalar signals, other types of signals have been explored. Verbal Critiquing refers to having the language model directly evaluate the quality of an output, providing suggestions for improvement. External Feedback leverages external sources, such as textual feedback from other robust models or error messages from a compiler in code generation tasks. Finally, there is a more implicit signal, contrastive optimization, which obtains consistency signals through the comparison between different expressions and optimizes towards consistency.

In this section, we focus more on the first three lines of work, as they are often studied independently and are hotspots in academic research. The last three lines of work are only briefly mentioned here, as they tend to be relatively simple or implicit methods. They will be elaborated in Sections V, VI.

A. Uncertainty Estimation

Uncertainty estimation refers to estimating the data uncertainty, model uncertainty, and distributional uncertainty involved in the neural networks [58].

For uncertainty estimation in the NLP field, Hu et al. [24] conducted a detailed survey. They categorize sources and modeling methods of uncertainty into three approaches: 1) **Calibration Confidence-based Methods:** This approach compares the accuracy of predicted probabilities with actual probabilities. 2) **Sampling-based Methods:** This approach models the variability of multiple expressions provided by the model, allowing us to observe the arising uncertainties. This method is also the focus of our article. 3) **Distribution-based Methods:** This approach evaluates inherent uncertainty by analyzing the dataset’s distribution characteristics.

We introduce an important method cluster within Sampling-based Methods: Monte Carlo Dropout (MCD) [59]. General deep neural network predictions are often deterministic, and multiple samples yield consistent answers, preventing us from understanding the model’s implicit certainty about the results. The MCD method uses the dropout technique to construct an implicit binomial distribution. For example, a 50% dropout probability constructs a $B(\#activation, 0.5)$ binomial distribution, which implicitly creates multiple models with different parameters $\theta_i \sim q(\theta), i = 1, 2, \dots, n$. At test time, MCD uses multiple models to obtain multiple output results $P(y_i|x; \theta_i)$ and estimates the uncertainty by calculating the variance of results. As for LLM, obtaining different expressions is much easier, such as using temperature coefficients. From the perspective of MCD, changing the temperature (values of the Softmax layer) implicitly constructs different models.

Besides MCD, which offers more explanatory insights, there are simpler, Sampling-based Methods available. For example, the Active Prompting strategy proposed by [60] uses disagreement in answers as an estimate of uncertainty, $\text{SelfEvaluate}(\mathcal{Y}) \triangleq \frac{|\text{unique}(\mathcal{Y})|}{|\mathcal{Y}|}$. Here, $\text{unique}(\mathcal{Y})$ represents the set after removing duplicate elements.

B. Confidence Estimation

Confidence is the opposite of uncertainty, focusing on reliability scores to enhance user trust.

In this line of work, Self-Evaluation is the core method⁷. The concept of Self-Evaluation was first proposed in [36], where the goal is for the model to express its level of confidence using its own knowledge and reasoning. As shown in Fig. 7, the Self-Evaluation method simply asks the model: Is the proposed answer True or False? Then, the confidence score, $P(\text{True})$, is extracted from the model’s logits.

Besides naively asking the model whether it thinks the proposed answer is correct, some works have proposed other

⁷The Self-Evaluation [36] here denotes a method, not the Self-Evaluation module in Self-Feedback framework. To distinguish between the two, a citation marker will be appended when referring to the method.

Question: Who was the first president of the United States?
Proposed Answer: George Washington was the first president.
Is the proposed answer:
 (A) True
 (B) False
The proposed answer is:

Fig. 7. Prompt for Self-Evaluation [36]

frameworks. For instance, BSDetector [61] is a confidence estimation framework suitable for both black-box and white-box models. It combines the consistency of multiple outputs sampled from the model with the model’s own reflection on its output, weighting these scores to obtain the confidence scores. Another example, TrustScore [62] is a reference-free confidence estimation framework using behavior consistency. It generates distractors based on entity information rules from Wikipedia, asks the LLM multiple times, and checks if it consistently chooses its own generated answer.

C. Hallucination Detection

Hallucination Detection aims to identify untruthful or unfaithful text within a response. SelfCheckGPT [63] provides a reference-free hallucination detection framework. Specifically, the goal of SelfCheckGPT is to determine the presence of hallucination in a given query x and response y_0 . The framework works in three steps. Firstly, the model samples several different responses, $\mathcal{Y} = \{y_1, y_2, \dots, y_n\}$. Secondly, it calculates whether $y_{1:n}$ support y_0 . Finally, it summarizes the support level to calculate the final score. Designing support level metric is where creativity can be applied, and the authors provide five different methods:

- **Similarity-based:** Compute the negation of the mean similarity between $y_{1:n}$ and y_0 ;
- **QA-based:** Generate many questions from y_0 and test consistencies in the answers derived from y_0 and $y_{1:n}$;
- **N-gram model-based:** Build an n-gram model from \mathcal{Y} , then use it to compute the negation of the mean transition probability between tokens in y_0 .
- **Natural language inference (NLI)-based:** Compute the mean probability of contradiction between the responses;
- **Prompt-based:** Similar to Self-Evaluation [36], directly ask the language model whether $y_{1:n}$ support y_0 .

Beyond the extensive methods of SelfCheckGPT, there are other interesting approaches as well. The Alibaba team proposed INSIDE [64] for deeper exploration. They sampled latent vectors from the intermediate layers and calculated the covariance matrix of these vectors. Since the eigenvalue of the covariance matrix represents data variability, they used this value as a measure of hallucination. Additionally, some methods utilize multiple agents to detect hallucinations. For example, Cross Examination [65] employs two LLMs, an Examinee and an Examiner, using a cross-examination approach to determine factual errors.

D. Verbal Critiquing

Inspired by the idea that “all tasks are generation tasks” [66], [67], many works have proposed allowing LLMs to generate more semantically rich textual signals. These include:

Let LLMs offer critiques. Saunders et al. [68] use a finetuned Self-Critiquing model to generate insights on content. McAleese et al. [69] use RLHF based on the GPT-4 model to train the model to critique code generation, resulting in CriticGPT. Du et al. [47] propose the Multi-Agent Debate method, where two agents generate modifications to each other’s content, gradually converging to an outcome.

Let LLMs summarize. Xiong et al. [44] use a Judge LLM to aggregate the results produced by multiple agents, providing a final judgment. Graph-of-Thought [38] uses the aggregation of thoughts to perform subsequent reasoning.

Let LLMs refine the text. These methods involve the LLM generating a refined response as a better result [8], [9], [48].

E. Contrastive Optimization

Contrastive optimization is an implicit signal acquisition method, which often involves constructing a scoring function, $\text{score}(y_i)$, to evaluate all responses in the sampling set \mathcal{Y} , $\{\text{score}(y_i)|i = 1, 2, \dots, n\}$. Finally, the best candidate is selected as $y_{\text{best}} = \arg \max_{y_i} \text{score}(y_i)$.

At the latent layer, in order to find attention heads with a stronger preference for truthfulness, Li et al. [5] trained a probe to evaluate the attention head’s ability to answer questions truthfully. **At the decoding layer**, Self-Evaluation [36] can be used to evaluate the reasoning paths during beam search, comparing scores to choose a better decoding direction [70]. **At the response layer**, Self-Consistency [2] strategy implicitly relies on comparisons between different responses. A variant, Soft Self-Consistency [71], calculates the joint probability of tokens for each response as the scoring function.

F. External Feedback

Sometimes, feedback from the model itself is not sufficient. For example, in code generation, if there are hallucinations (bugs) in the code, it is difficult for even humans to accurately identify some bugs without executing the code with an external executor. Self-Debug [49] proposes using the execution results from an external executor as feedback. Besides using external tools, some works use other models as external feedback sources, such as a more powerful teacher model [72] or a peer model [47]. The commonly used RAG method, which can incorporate information retrieved from external sources as external feedback, is another example utilizing external feedback.

V. TASK: REASONING ELEVATION

Reasoning Elevation refers to enhancing the logical reasoning capabilities of language models during response generation to improve their internal consistency. The primary feature of this line of work is the use of benchmarks in the form of QA tasks. We have identified three significant lines of work, as shown in the upper part of Table IV.

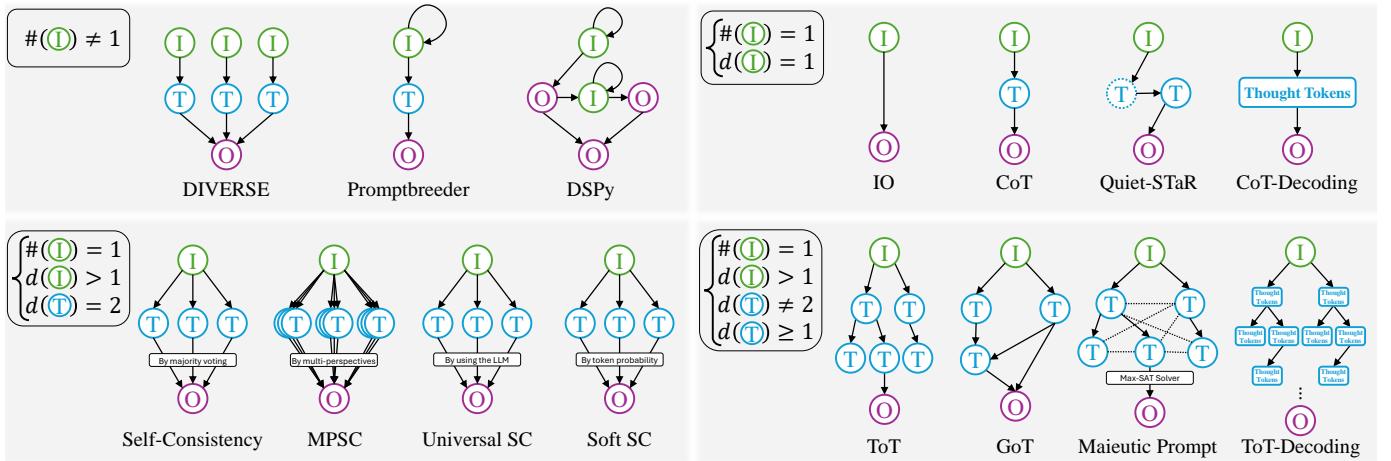


Fig. 8. Different Reasoning Topologies. \textcircled{I} / \textcircled{T} / \textcircled{O} indicate input / intermediate thought / output, respectively. $\#(\cdot)$ and $d(\cdot)$ indicate the number and degree of nodes, respectively.

A. Reasoning Topologically

When answering a question, LLMs may choose different reasoning paths, but not all reasoning paths lead to the correct answer. Therefore, finding reasoning paths that are consistent with the learned knowledge becomes a key issue, leading to a series of works focusing on optimizing reasoning paths. Fig. 8 summarizes the similarities and differences of these works.

A survey [73] covers various X-of-Thought (XoT) methods. **Input-Output (IO)** is the simplest approach, asking a question and getting an answer directly, but often struggles with complex problems. To address this, **Chain-of-Thought (CoT)** [10] was introduced, adding intermediate reasoning steps, though errors in reasoning can affect results. **Self-Consistency (SC)** [2] improves accuracy via majority voting but is limited in exploratory power. **Tree-of-Thought (ToT)** [37] views reasoning as a path with multiple successor nodes for deeper exploration, while **Graph-of-Thought (GoT)** [38] aggregates reasoning chains across nodes. Similar to GoT, **Maieutic Prompting** [74] builds entailment relationships between thoughts, then constructs a Max-SAT [75] problem to obtain the best choices.

Most XoT methods require sampling and aggregation of thoughts, often limited to queries with fixed label sets during aggregation. To solve this problem, several works have emerged. **Multi-Perspective Self-Consistency (MPSC)** [76] targets code generation tasks, evaluating each solution from multiple perspectives (solution, specification, and test case) to select the best one. **Universal Self-Consistency (Universal SC)** [77] uses LLMs instead of simple answer matching to choose the most selected response, enhancing the stability of the majority voting. **Soft Self-Consistency (Soft SC)** [71] proposes a more adaptive scoring function, calculating the joint probability of tokens in a response as the scoring function, thus extending the problem scope to soft labels.

Additionally, **Quiet Self-Taught Reasoner (Quiet-STaR)** [39] addresses the issue mentioned in Section II-B, where “although complex reasoning in responses is beneficial for solving intricate problems, they may disrupt model’s latent reasoning due to redundant reasoning text, thereby

increasing response-level inconsistency.” Quiet-STaR samples rationales from the model’s responses and wraps each rationale between special markers, that is, $\langle|\text{startofthought}|>$ and $\langle|\text{endofthought}|>$, to assist next-token reasoning. These rationales are invisible to the user, making latent reasoning explicit and effectively reducing conflicts.

However, these lines of work are mostly focused on how to choose the next thought from an input, overlooking the input stage. An input is a combination of a query and a prompt template. While the query remains relatively unchanged, the instructions and demonstrations in the prompt template can be optimized. Several works have explored this area: **DIVERSE** [78] pre-constructs various prompt templates to increase prompt diversity. **Promptbreeder** [79] uses genetic algorithms [80] to continuously optimize the original prompt template. **DSPy** [81] innovatively builds a prompt optimizer, similar to a gradient optimizer in PyTorch. These methods extend reasoning topology to the input stage, demonstrating significant creativity. Boldly, we could construct a reasoning-topology-oriented framework incorporating prompt optimization, which could potentially solve more complex problems.

Furthermore, we can extend our approach to the decoding stage. **CoT Decoding** [82] incorporates CoT’s ideas into the decoding process, attempting to identify CoT-included decoding paths in the natural decoding process. **ToT Decoding** [70] integrates ToT concepts into decoding, replacing beam search criteria with Self-Evaluation [36], where each token’s selection depends on confidence scores $C(\cdot)$, achieving better reasoning, as shown in Eq. 9, where y^t is the t -th token in string y .

$$P(y) = \prod_t P(y^t | y^{1:t-1}) C(y^t) \quad (9)$$

Self-Evaluation Strategy. The methods discussed in this section typically require searching the thought graph, necessitating evaluators to determine the usefulness of thoughts and whether they merit further exploration. These works generally use three approaches: Majority Voting, selecting the most consistent response among multiple thoughts [2]; Rule-based methods, designing specific scoring functions based on the

problem, such as error scoring functions in sorting tasks, representing the number of inversions and frequency differences before and after sorting [38]; and LLM-based methods, like the scoring function in the Game of 24 task, where LLMs rate the solution’s feasibility as “sure/maybe/impossible” [37].

Self-Update Strategy. For Self-Consistency prompting, the update uses a majority voting result. For ToT prompting, the update method uses BFS and DFS strategies to search and select suitable thoughts as output. For GoT prompting, the update method is similar to ToT but includes more extensive search spaces, aggregating different thoughts.

Despite the innovations, these methods have several limitations [73]: 1) They often select extremely simple tasks like Game of 24, Sorting, and Keyword Counting for experiments. 2) They incur high reasoning costs. 3) They struggle to adapt to general tasks and deployment.

B. Refining with Responses

Refining with Responses refers to the process where an LLM first generates multiple responses, then identifies the better responses or self-evaluates its own generated content and corrects errors, and finally refines its output or fine-tunes the model itself to improve response consistency. The following are three common lines of work.

Fine-tuning from the collected responses. This line of work involves “using self-generated data to fine-tune itself.” Specifically, they often use LLMs to produce multiple answers, select the better responses from them, and then use these better responses to fine-tune the model, enhancing its reasoning capabilities. For example, Self-Improve [40] uses a majority voting strategy to obtain better outputs, collecting such data to fine-tune the model itself. Similarly, Tian et al. [83] propose a framework called Self-Improvement, which uses Monte Carlo Tree Search for data synthesis while generating fine-tuning datasets, improving model’s reasoning capabilities.

Learning from mistakes. This line of work is similar to fine-tuning from the collected responses but focuses on learning from errors and optimizing by avoiding mistakes. This intuitive method naturally improves model performance by avoiding errors. For instance, the LEMA (LEarning from MistAkes) method proposed by [42] samples multiple reasoning rationales, has GPT-4 annotate and correct errors among them, and uses the corrected rationales to form a new dataset for re-fine-tuning the model. Similarly, Tong et al. [43] propose the Mistake Tuning scheme: it has the model self-rethink and correct its errors based on references, using large amounts of such self-corrected datasets to fine-tune the model.

Getting better response with NLI models. Besides fine-tuning methods, we also demonstrate rule-based optimization techniques using NLI [41], [84]. With an NLI model, we can identify the relationships between multiple samples and find better responses. For instance, Agarwal et al. [84] use a pre-trained NLI model to identify and correct logically inconsistent statements generated by a pre-trained language model. They then convert the entailment and contradiction probabilities of the NLI into a Max-SAT problem [75], and use a constraint solver [85] to optimize and obtain more accurate and consistent predictions.

C. Multi-Agent Collaboration

The methods in this category generally involve using more than one LLM to collaboratively solve problems, address contradictions, and promote consistency, essentially constituting a generalized form of Self-Feedback. There are numerous papers in the Multi-Agent field; here, we list some typical and novel works that employ Multi-Agent systems for Self-Feedback. For a more comprehensive understanding, refer to the extensive survey on LLM Agents by Wang et al. [86].

Debate Frameworks. Multi-Agent Debate [47] utilizes multiple peer models that engage in iterative debates, with a fixed number of rounds as the stopping condition. Their experiments show that debates with three or fewer rounds can generally lead to convergence among agents (i.e., LLMs consistently agreeing on the same answer). Xiong et al. [44] further propose the FORD (Formal Debate Framework), which introduces a Judge LLM to summarize the agents’ statements at the end, also using a fixed number of rounds as the stopping condition. They expand the scope of LLM debates by exploring the effects of debates among models with mismatched capabilities in various scenarios. REFINER [46] trains two models with different roles: a generator for intermediate reasoning steps and a critic for feedback, continuing the iterative dialogue until the correct answer is obtained or the critic has no further feedback. Notably, using the correct answer as a stopping condition has been criticized as unrealistic [87].

Game-Theoretic Approaches. The Consensus Game proposed by Jacob et al. [88] deviates from the above frameworks by avoiding direct dialogue between LLMs. Instead, different LLMs participate in a game, based on the hypothesis that “asking a model for answer A to question Q (generative)” and “asking a model if A is the answer to Q (discriminative)” lack consistency [89]. They prompt the generator to produce both correct and incorrect answers, then use the discriminator to evaluate its own responses, aiming for the generator and discriminator to reach a Nash equilibrium. They select the best response based on the degree of consistency.

The significant drawback of this line of work is the high inference cost, as it often requires different LLM instances, potentially consuming multiple times the GPU memory and increasing the inference burden due to the extensive context generated by agents. Additionally, most models need a stopping condition to end the dialogue, and fixed round stopping is inflexible and can reduce performance. There is no current flexible and efficient stopping criterion. However, Multi-Agent systems remain a promising AI direction, and cost issues shouldn’t deter exploration.

VI. TASK: HALLUCINATION ALLEVIATION

Hallucination alleviation is aimed at open-ended generation tasks such as story writing and code generation, emphasizing goals like fact enhancement, error reduction, and faithfulness enhancement. We have categorized four significant lines of work, as shown in the lower half of Table IV.

A. Refining the Response Iteratively

This line of work is similar to Refining with Responses (Section V-B) which primarily targets simple QA tasks. While

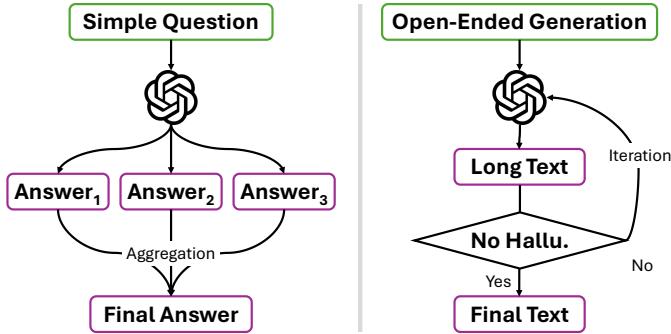


Fig. 9. Refining with Responses (Left) V.S. Refining the Response Iter. (Right)

Refining the Response Iteratively (Section VI-A) primarily deals with open-ended tasks such as story generation and code generation. Their comparison is shown in Fig. 9.

The most famous works include Self-Refine [8], Reflexion [48], and Self-Correct [9]. These three frameworks share the basic structure of having the LLM provide textual feedback, which is then used to update the response iteratively until a stopping criterion is met or the maximum iterations is reached, as shown in Algorithm 1.

Algorithm 1 REFINING THE RESPONSE ITERATIVELY

Require: Input query x , model \mathcal{M} , consistency signal generator $\text{SelfEvaluate}(\cdot)$, Self-Update strategy $\text{SelfUpdate}(\cdot)$, stopping criterion $\text{stop}(\cdot)$, max iteration T

```

1:  $y_0 = \mathcal{M}(x)$ 
2:  $i \leftarrow 0$ 
3: while  $i < T$  and not  $\text{stop}(y_i)$  do
4:    $f_i = \text{SelfEvaluate}(x, y_i)$ 
5:    $y_{i+1} = \text{SelfUpdate}(x, y_{0:i}, f_{0:i})$ 
6:    $i \leftarrow i + 1$ 
7: end while
8: return  $y_i$ 

```

Despite following a similar framework, there are differences in specific implementations. Self-Refine [8] is the most naive implementation, where $\text{SelfEvaluate}(\cdot)$ is entirely performed by the LLM to generate textual feedback. Reflexion [48] takes a better approach by viewing the iterative refining process as Verbal Reinforcement Learning, which is reinforcement learning without weight updates. Additionally, they separate feedback into feedback signal generation (e.g., error messages generated after code compilation in code generation tasks) and textual feedback generation (reflecting on error messages), increasing the framework's completeness. However, this approach requires a specific feedback signal design for each task, reducing its generality. Self-Correct [9] uses the same framework but trains a dedicated Corrector model to generate better feedback. This method, however, is still not task-agnostic and significantly reduces the framework's flexibility due to the introduction of training.

The works mentioned above mainly construct frameworks for general tasks, while some focus on specific tasks. For example, Re³ [90] draws inspiration from human actions in writing long stories and proposes a draft, rewrite, and edit cycle to optimize the LLM's ability to write long stories. PEER [91] mimics human collaborative editing by having

the LLM iteratively propose editing suggestions to complete Wikipedia text editing. Self-Debug [49] allows the model to debug its code through execution results and self-written unit test results, gradually refining the code until it is perfected.

B. Mitigating Hallucination while Generating

As mentioned earlier, hallucinations often manifest in finer details, such as temporal inaccuracies, date errors, or misattributions of names [89]. Multi-round iterations may overlook these minor errors, prompting some works to propose methods for more granular error editing, mitigating hallucination while generating⁸. Currently, this is not yet a relatively mature direction, and there is no unified solution emerging. The following outlines typical approaches in methodology.

Mündle et al. [18] utilize the phenomenon of Self-Contradiction to eliminate hallucinations⁹. Specifically, it induces prompts to generate two contradictory sentences and then directs the LLM to resolve the contradictions, retaining the consistent information to generate a coherent sentence. Subsequent sentences follow a similar approach to produce a complete reply. Clearly, contradictory information is highly likely to be hallucinatory, thus effectively mitigating hallucinations. This method essentially extends Self-Consistency [2] into the domain of hallucination.

EVER (REal-Time VErification and Rectification) [50] employs a similarly intuitive approach. When generating a sentence, EVER verifies the accuracy of the generated sentence either by the LLM itself or retrieved external information, generating feedback to modify the sentence if there are issues. The modified sentence is then re-appended into the generated text iteratively. Similarly, PURR (Petite Unsupervised Research and Revision) [92] and RARR (Retrofit Attribution using Research and Revision) [93] follow a similar approach as EVER, where the verification stage relies on retrieving external knowledge to provide modification feedback.

In contrast to EVER, FAVA (FAct Verication with Augmentation) [51] adopts a more sophisticated approach. It fine-tunes the model to generate special tokens that edit its own content, enhancing editing efficiency¹⁰. The major advantage of this method lies in granting the LLM maximum autonomy to make mistakes and subsequently correct them freely. Moreover, this approach bears resemblance to Quiet-STaR [39] mentioned in Section V-A, where both utilize special tokens to represent essential cognitive processes.

C. Decoding Truthfully

Decoding Truthfully focuses predominantly on decoding consistency. In recent years, several studies have discovered that methods such as greedy decoding and sampling decoding constrain LLMs from accurately expressing crucial information in natural language. Consequently, more complex and

⁸This section incorporates ideas from RAG, yet given its relevance to Self-Feedback, it's delineated as a distinct line of work.

⁹Demo of Self-Contradiction: <https://chatprotect.ai/>

¹⁰Their fine-tuning dataset includes examples like: “Messi is an <entity><delete>Argentine </delete><mark>Brazilian </mark></entity>soccer player.” Special tokens enclosed in angle brackets are also trained to be generated, effectively eliminating hallucinations through rendering.

rational decoding strategies have been designed to elevate the reliability and accuracy of model’s responses [94].

Li et al. [95] pioneered the Contrastive Decoding strategy, where during the next token prediction, the optimal token probability is selected by contrasting the token probability distributions derived from expert and amateur models, as shown in Eq. 10. This method excels in mitigating biases or preferences inherent in large-scale models, favoring tokens with higher probabilities in expert models and lower probabilities in amateur models.

$$\mathbf{y}^t \sim \text{softmax} \left(\log \frac{P_{\text{EXP}}(\mathbf{y}^t | \mathbf{y}^{0:t-1})}{P_{\text{AMA}}(\mathbf{y}^t | \mathbf{y}^{0:t-1})} \right) \quad (10)$$

Following this pioneering work, researchers have explored various approaches for logit adjustment and contrastive decoding. Chuang et al. [52] observed significant differences in token probability distributions across different layers of the model and introduced DoLa to incorporate information from previous layers, enhancing early-stage cognitive reasoning and pre-answer consistency, termed Decoding Consistency.

Unlike DoLa, SED [11] and DIVER [54] focus on detecting and addressing discrepancies caused by differences in tokens at certain positions, termed Chaotic Points. Methods for detecting chaotic points include comparing the ratio of maximum to second-maximum token probabilities or the number of candidate tokens exceeds one. Their indicator functions are shown in Eqs. 11 and 12, where δ_r is a probability threshold, γ is a predefined coefficient, and \mathcal{V} denotes the vocabulary. By assessing previously generated contents against potential tokens from chaotic points, scores such as information gain, weighted uncertainty, and weighted confidence help identify the most suitable token.

$$\mathbb{I}_1 \left(\frac{P_{\text{second}}}{P_{\text{max}}} \geq \delta_r \right) \quad (11)$$

$$\mathbb{I}_2 \left(\left| \left\{ \mathbf{y}^t \mid P(\mathbf{y}^t | \mathbf{y}^{0:t-1}) \geq \gamma \max_{w \in \mathcal{V}} P(w | \mathbf{y}^{0:t-1}) \right\} \right| > 1 \right) \quad (12)$$

Those methodologies primarily apply to closed-book generation tasks. For open-book generation tasks, current research focuses on leveraging external references to guide decoding. CAD [53] and ECAD [96] (named ECAD in this survey) incorporate contextually relevant or irrelevant knowledge snippets into model inputs, intervening in the decoding process through contrastive decoding strategies to bridge the information gap between useful and non-useful information.

D. Activating Truthfulness

Activating Truthfulness focuses on enhancing consistency in latent layers. Its core methods involve boosting attention heads and states that represent “truthfulness” within latent layers, aiming to improve the model’s internal consistency.

The exploration of latent truthfulness began with CCS (Contrast-Consistent Search) [97]. CCS investigates methods for mining knowledge embedded in latent layers by training a small classification head on Transformer latent layers. This method effectively activates model truthfulness, surpassing conventional inference methods.

Inspired by CCS, Harvard scholars introduced the Inference-Time Intervention (ITI) technique [5]. ITI consists of two steps: 1) Probe analysis: Using probe technology¹¹ to identify attention heads in the model related to truthfulness. 2) Inference-time intervention: The model’s answer generation process is adjusted by increasing the weights of selected attention heads, guiding the model toward more truthful reasoning. However, ITI has limitations in training probes using only the last token’s latent layer state at the end of a QA pair. TrFr [6] addressed this by using multi-dimensional orthogonal probes to extract features from both truthful and non-truthful texts, improving attention head identification. TruthX [7] explored a more efficient intervention strategy. It targets not only attention heads but also the feed-forward network layers. Mapping these states separately using truthful and semantic encoders significantly reduces the impact on the language model’s overall performance while enhancing representations of truthfulness.

White-Box Hallucination Alleviation. Mitigating hallucinations from a white-box perspective involves activating the internal authenticity of the model, which necessitates interpretability studies. For instance, a recent survey [98] reveals that attention heads in models can serve various functions. Building on these functional distinctions, we may discover better approaches to mitigate hallucinations. For example, Wu et al. [99] found that certain attention heads are more adept at long-context retrieval (strong “copy-paste” abilities). In tests such as Needle-in-a-Haystack, blocking these attention heads caused performance to drop from 94.7% to 63.6%. Can enhancing retrieval heads reduce hallucinations in long contexts? This is a question worth investigating.

VII. TASK: OTHERS

Several works follow the Self-Feedback framework, though not always targeting internal consistency. For completeness, we summarize these efforts below.

A. Preference Learning

Preference Learning (PL) aims to align LLM outputs with human intent [100]–[102]. Most of the work around this task can be broadly covered by the Self-Feedback framework. For PL, the Feedback Signal mainly refers to the reward information given by a reward model \mathcal{R} , which is trained through preference feedback. Preference feedback involves comparing and ranking different responses to the same question in terms of helpfulness, harmlessness, and honesty. The Self-Update here primarily refers to broadly updating the model \mathcal{M} , including methods like supervised fine-tuning and reinforcement learning (such as PPO [103], DPO [104]).

There are three main ways to obtain preference feedback. 1) Through human feedback, as seen in works like OASST [105] and BeaverTails [106], which include human-annotated data. 2) Feedback generated by models [107], [108], offering lower annotation costs and faster iterative feedback efficiency compared to human feedback. 3) Feedback derived from inductive

¹¹A probe is a small classifier whose input is latent states and whose output is labels corresponding to a test task.

bias, such as upvotes/downvotes in the SHP dataset [109], or prior rules in ALMoST [110], which rank response quality based on model size or prompt context.

Based on preference feedback, we can train a reward model to output Feedback Signals. There are two common types of reward models. One is the Reward Model proposed in InstructGPT [111], with the loss function as shown in Eq. 13. Here, $r_\theta(\mathbf{x}, \mathbf{y})$ represents the output of the Reward Model, and response \mathbf{y}_w is ranked higher than \mathbf{y}_l . However, this method’s downside is that the overall score distribution for high-quality and low-quality responses is similar, making it difficult to effectively distinguish between different responses to different questions. To address this, Xu et al. [112] proposed an evaluation model that directly scores QA pairs.

$$\begin{aligned} z &= \sigma(r_\theta(\mathbf{x}, \mathbf{y}_w) - r_\theta(\mathbf{x}, \mathbf{y}_l)) \\ \text{loss}(\theta) &= -\frac{1}{k} \sum_{i=1}^k E_{(\mathbf{x}, \mathbf{y}_w, \mathbf{y}_l) \sim D} [\log(z)] \end{aligned} \quad (13)$$

B. LLM-Based Knowledge Distillation

LLM-based knowledge distillation methods aim to transfer advanced capabilities from proprietary LLMs (such as GPT-4) to small-parameter open-source models [113]. These two models can be referred to as the “teacher model” and the “student model” respectively, with the teacher model guiding the student model to enhance its capabilities, fitting the generalized Self-Feedback framework proposed in this paper. During the Self-Evaluation, the student model generates answers, which are then assessed by the teacher model. In the Self-Update, the student model uses the evaluation signal to update itself or its answers.

This signal can be in the form of statistical metrics, such as MiniLLM [114] calculating the reverse Kullback-Leibler (KL) divergence of the probability distributions output by the student and teacher models; or GKD [115] computing metrics like forward KL divergence, reverse KL divergence, and generalized JSD. The signal can also be textual feedback, such as Selfee [116] utilizing ChatGPT as the teacher to provide textual feedback on the outputs of the student model; or in PERsD [72], where the teacher executes the code generated by the student model and provides specific suggestions based on errors.

When the teacher and student models are the same LLM, this leads to Self-Knowledge Distillation (Self-KD). In Self-KD, the model iteratively updates its capabilities using the knowledge it gradually accumulates during training, falling under the narrow Self-Feedback paradigm. For example, the goal of Impossible distillation [117] is to obtain a Stronger Paraphraser. In the Self-knowledge distillation process, it evaluates its paraphrase results from perspectives such as semantics, format, and diversity, and further refines high-quality data to fine-tune itself accordingly.

C. Data Augmentation

Data Augmentation aims to construct and filter high-quality datasets using LLMs. It is somewhat similar to the methods

in Sections VII-A and VII-B that combine Feedback information to create datasets, but there are slight differences in focus and specific forms. The latter focuses on the model’s capabilities, using datasets during the Self-Update stage for model fine-tuning, with most methods falling under narrow Self-Feedback. In contrast, Data Augmentation focuses on the dataset itself, updating the model’s responses during the Self-Update stage to further refine the dataset, with most methods falling under generalized Self-Feedback.

Self-instruct [118] is a typical example, where the LLM generates new task instructions during the Self-Evaluation stage and generates input-output instances based on the new instructions. It calculates the ROUGE-L metric between the new instructions and existing instructions as the Feedback signal. Finally, during the Self-Update stage, it filters and screens the newly generated set of instructions.

Currently, methods applying LLMs to Data Augmentation and Synthetic Data Generation mainly focus on the prompt engineering layer. In other words, Self-Evaluation only involves responses. Many studies have shown that LLM responses are highly sensitive to prompt variations [119], [120]. Therefore, the main bottleneck in this task is: how to design better prompts and how to deeply explore the relationship between decoding, latent states, and data quality.

VIII. EVALUATION

This section covers evaluation methods and benchmarks for internal consistency and Self-Feedback, focusing on two abilities: meta (e.g., uncertainty, consistency, feedback) and common (e.g., reasoning QA, code generation) abilities. Meta evaluation identifies which LLMs are the best, while common evaluation reveals which Self-Feedback methods are the best.

A. Meta Evaluation

We summarize five meta evaluation methods, categorized into metric-based and benchmark-based approaches. Metric-based methods calculate performance mainly via formulas, while benchmark-based methods empirically measure it using QA datasets (see Table V).

TABLE V
META EVALUATION BENCHMARKS

Type	Benchmark	Organization
Uncertainty	LLM-Uncertainty-Bench [121]	Tencent
Uncertainty	UBench [122]	Nankai
Consistency	ConsisEval [123]	PKU
Consistency	PopQA-TP [124]	IBM
Consistency	ParaRel [125]	BIU
Consistency	BMLAMA [126]	RUG
Consistency	BECEL [127]	Oxford
Critique Ability	CriticBench [128]	THU
Self-Knowledge	SelfAware [4]	Fudan
Self-Knowledge	Idk(I don’t know) [28]	Fudan
Self-Knowledge	Self-Knowledge Evaluation [129]	THU

Uncertainty Evaluation¹². Key metrics for evaluating model uncertainty include: Expected Calibration Error (ECE),

¹²As mentioned in Section IV-A, uncertainty estimation involves assessing the uncertainty of a model’s specific response. Uncertainty evaluation, on the other hand, measures the overall uncertainty of a model.

which assesses the expected difference between model confidence and accuracy; Maximal Calibration Error (MCE), which indicates the maximum deviation between model accuracy and confidence; and Brier Score (BS), which is used to assess how closely the model’s predicted probabilities align with the true class probabilities [24].

Uncertainty Evaluation. LLM-Uncertainty-Bench [121] extracts five test tasks (including question answering, reading comprehension, commonsense inference, dialogue response selection, and document summarization) from common benchmark datasets and uses conformal prediction techniques to construct benchmarks. UBench [122] also extracts data from other datasets, totaling 3978 multiple-choice questions covering knowledge, language, understanding, and reasoning abilities. UBench evaluates individual data items by having models textually express uncertainty scores.

Consistency Evaluation. This line of work centers on assessing whether a model delivers consistent responses to queries that are semantically equivalent but phrased differently. The key focus is on developing a variety of synonymous queries to test the model’s reliability. For instance, the ConsisEval Benchmark [123] creates simpler synonymous queries for each question. PopQA-TP [124] and ParaRel [125] construct synonymous queries through rephrasing. BMLAMA [126] focuses on multilingual consistency, constructing a parallel corpus of queries. BECEL [127] draws inspiration from behavioral consistency, considering higher-order consistency in model responses by creating semantic consistency data, negational consistency data, symmetric consistency data, etc. Notably, most studies have found that models generally exhibit low consistency.

Critique Ability Evaluation. Lin et al. [128] collect a large number of QA pairs from 15 datasets across mathematical, commonsense, symbolic, coding, and algorithmic fields, creating CriticBench through model generation and human annotation. It can be used to evaluate the ability of LLMs to generate critiques, an important aspect of the Self-Feedback framework.

Self-Knowledge Evaluation. Self-Knowledge refers to the LLM’s understanding and recognition of its own abilities, limitations, and the content it creates. Yin et al. [4] and Cheng et al. [28] construct sets of unanswerable questions to explore the question “Do large language models know what they do not know?” Tan et al. [129] investigate “Does the model truly understand the questions and solutions it creates?” These studies generally yield negative empirical results, indicating that models have weak Self-Knowledge.

B. Common Evaluation

Self-Feedback methods are often evaluated using benchmarks that focus on real-world tasks like reasoning, code generation, and math problem solving (see Table VI). For more information on LLM evaluation, you can refer to this survey [130].

IX. DOES SELF-FEEDBACK REALLY WORK?

A. Conflicting Viewpoints

With the rise of works prefixed by “Self-”, questions of feasibility arise: Can a model truly optimize itself? Many

TABLE VI
COMMON EVALUATION BENCHMARKS

Type	Benchmark	Organization
Knowledge reasoning	C-Eval [131]	SJTU
Knowledge reasoning	MMLU [14]	UCB
Logic reasoning	BBH [132]	Google
Logic reasoning	ARC [133]	AI2
Linguistic understanding	WiC [134]	Cambridge
Code generating	HumanEval [135]	N/A
Math Solving	MATH [136]	UCB
Math Solving	GSM8K [27]	OpenAI

studies have attempted to answer this question, with most focusing on Refining the Response Iteratively and Multi-Agent Collaboration.

- Jiang et al. [137] propose the SELF-[IN]CORRECT hypothesis, showing that in QA tasks, models are better at generating answers than judging their own correctness, highlighting a self-assessment limitation.
- Stechly et al. [138] and Valmeeekam et al. [139] found GPT-4 fails to verify its solutions in the Graph Coloring and planning tasks, with verifiers generating many false positives, reducing reliability.
- Huang et al. [87] refute the effectiveness of Reflexion [48], Multi-Agent Debate [47], and Self-Refine [8]. They argue Reflexion’s reliance on external truth for refining is impractical, Multi-Agent Debate is inferior to Self-Consistency and resource-heavy, and Self-Refine’s prompts were unfair, with better one-shot responses achievable through improved prompting.
- Kamoi et al. [21] provide a more comprehensive analysis by classifying various methods clearly and systematically comparing the strengths and weaknesses of each methods. They suggest that the ability to self-correct should be discussed according to the specific task. For example, for decomposable tasks¹³ or verifiable tasks¹⁴, it is feasible for the model to optimize itself.

While these criticisms reveal certain limitations in feedback signals, experimental tasks, and test models, they can be seen as limited perspectives [87], [137]–[139]. Although the survey [21] provides more meaningful viewpoints through classified discussions, it complicates the field, making it difficult to form a systematic framework. Benefiting from the perspective of internal consistency and the clear boundary discussions in Section I-E, we conduct a more meaningful discussion on the proposed Self-Feedback framework:

- 1) **Does Self-Feedback improve internal consistency?**
The answer is yes. As demonstrated in our survey, different lines of research offer affirmative evidence from various perspectives.
- 2) **Does internal consistency mean correctness?** We cannot directly conclude this. We will delve deeper into this question in the following section.

¹³For example, “Who are some politicians who were born in Boston?”

¹⁴For example, in the Game of 24 (Find arithmetic operations to obtain 24 using four given integers), generating a solution is harder than verification.

B. Does Internal Consistency Mean Correctness?

Let's revisit the relationship between world knowledge, training corpus, and language models (LMs), as shown in Fig. 10. World knowledge is the consensual (correct) knowledge we humans possess. The training corpus used for models is a true subset of world knowledge, containing the vast majority of correct knowledge and a small portion of uncleanable erroneous knowledge. Additionally, the knowledge embedded in the corpus is deterministic, where each statement in the corpus has a probability of 100%. Language models, by fitting the corpus, acquire higher-order probabilistic representations of this knowledge, but the probabilistic nature makes the learned knowledge vague and non-deterministic, as illustrated by the shaded areas in Fig. 10. Vagueness (or hallucination) is an important characteristic of language models. It enables the generation of novel and creative expressions outside the training corpus distribution. However, from a reliability perspective, vagueness is a disaster. Vagueness means that answers to the same question are uncertain, making the model's expressions inconsistent.

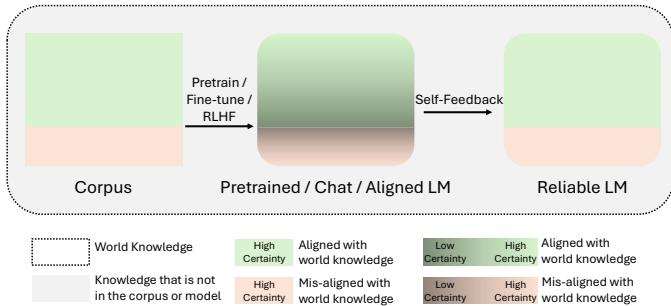


Fig. 10. World Knowledge, Training Corpus and Language Model

Therefore, we need to improve internal consistency and eliminate vagueness within the model to enhance its confidence in correct knowledge. However, eliminating vagueness also means that the model will be equally confident in erroneous knowledge. This raises a question: does enhancing consistency yield overall benefits or drawbacks? The advantage is that when preprocessing and cleaning the pre-training corpus, the intention is to align it towards world knowledge (correct knowledge). Hence, we propose the “Consistency Is (Almost) Correctness” hypothesis.

Consistency Is (Almost) Correctness

Enhancing a language model’s internal consistency activates its cognitive certainty, reinforcing both correct and erroneous knowledge. However, because the pre-training corpus is predominantly aligned with correct world knowledge, improving consistency tends to amplify correct content more than incorrect content. Consequently, increased internal consistency generally results in improved overall correctness.

However, why do some opposing voices believe that improving consistency cannot enhance the model’s correctness?

We believe this is closely related to the testing tasks. Many works refuting Self-Feedback use testing tasks that lie in the shaded areas of Fig. 10 (e.g., unstated puzzles not in the training corpus or questions unsolvable without external knowledge). Models struggle to effectively Self-Evaluate and Self-Update for tasks beyond their generalization capability.

In summary, within-distribution capabilities, the Self-Feedback framework can enhance model consistency by reinforcing the model’s fit to corpus priors, thereby eliminating uncertainty and improving consistency. According to the “Consistency Is (Almost) Correctness” hypothesis, this leads to an overall improvement in the model’s performance.

C. Appeals

The field faces significant criticism due to inconsistent naming, unrealistic tasks, varying benchmarks, and contradictory baselines. Thus, we propose the following appeals:

- **Naming.** Ensure method names are distinct (e.g., Self-Improve [40] and Self-Improvement [83] are bad names) and accurate (e.g., uncertainty or confidence estimation).
- **Task Definition.** Standardize terms by adopting “Internal Consistency Mining” for reasoning elevation and hallucination alleviation tasks.
- **Reasoning and Hallucination.** Use “lack of reasoning” for QA tasks and “exhibiting hallucination” for open-ended generation tasks.
- **Selection of Baselines.** Select baselines from the same sub-direction (section) to ensure fair comparisons.
- **Experiment Settings.** Avoid unrealistic setups, such as requiring pre-given golden labels [87].
- **Prompt Engineering.** Disclose and test prompt templates for robustness and generality across different LLMs.

X. FUTURE DIRECTIONS AND CHALLENGES

A. Textual Self-Awareness

Human speech often lacks consistency and certainty in expressing viewpoints. However, we typically use phrases like “I’m not sure, but I think” or “I believe there’s an 80% chance” to hedge, demonstrating our good self-awareness. Yona et al. [140] proved that current models still cannot verbally and faithfully express their uncertainty. Kapoor et al. [141] found similar issues and showed through experiments that models can achieve good calibration only after fine-tuning. How to enable models to utilize the available internal consistency signal to help textually express their self-awareness is a promising direction [142].

B. The Reasoning Paradox

As mentioned in Section II-B, there is a paradox between the reasoning done during single token prediction (latent reasoning [26]) and the reasoning done using multiple tokens in language (explicit reasoning, e.g., CoT) [143].

Therefore, we need to study the equilibrium point between latent and explicit reasoning, enabling efficient use of reasoning resources and improving the model’s reasoning efficiency. Currently, there is little research on this issue.

The Paradox of Latent and Explicit Reasoning

Language models excel in latent reasoning when decoding a single token, effectively utilizing attention mechanisms and deep feature interactions to achieve accurate reasoning. However, single tokens can't answer complex questions. Explicit reasoning, which involves generating a sequence of tokens (e.g. CoT), enhances the model's problem-solving capabilities. Yet, lengthy reasoning chains and inherent noise in text disrupt the model's latent reasoning. Thus, there is a paradox between latent reasoning and explicit reasoning.

C. Dive Deeper

From the seven lines of work we summarized, many works optimize only at the response layer. However, this approach relies on experience and is highly sensitive to prompt templates. Moreover, the low entry barrier and extensive participation in such work have led to an influx of low-quality papers. Therefore, we encourage researchers to delve into the decoding layer and latent layer, exploring more universal discoveries from an interpretability perspective.

D. The Unified Perspective

At present, the focus of work in this field is relatively narrow, lacking a comprehensive understanding of the entire field, and consequently, there are no more general framework works. We believe that using the perspective proposed in this paper, considering problems from the response, decoding, and latent layers in a unified manner, can better facilitate Internal Consistency Mining. There are emerging efforts that begin to integrate multiple layers. For example, Xie et al. [29] start from the response layer and reflect on how different CoT paths guide the consistency of the latent layer; Xie et al. [70] use Self-Evaluation strategies at the response layer to guide better decoding strategies.

E. The Comprehensive Evaluation

Different LLMs, combined with various Self-Feedback strategies, can produce vastly different combinations. However, as explained in Section VIII, current evaluation methods generally have a singular focus, making it difficult to comprehensively and conveniently understand the model's capabilities. Therefore, building a complete evaluation system from meta evaluation to common evaluation, from latent states to response, from benchmark to metric, and from uncertainty to feedback is a worthy consideration.

XI. CONCLUSION

This paper proposes using an internal consistency perspective to observe the most prominent phenomena in the field of LLMs: lack of reasoning and presence of hallucinations. The article explains the modeling of internal consistency, the hourglass evolution pattern, the current status, sources, and significance from multiple aspects, and proposes the Self-Feedback framework for Internal Consistency Mining. We

summarize the various tasks and distinctive lines of work involved in the Self-Feedback framework. These lines of work can help researchers locate their work's position within a vast system and facilitate reasonable experimental comparisons. Finally, we include three critical topics: relevant evaluation methods and benchmarks, exploring whether Self-Feedback truly works, and future research directions. In summary, this paper attempts to use a deeper research perspective (Internal Consistency) and a more general framework (Self-Feedback) to summarize a series of important works on reasoning elevation and hallucination alleviation.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (Grants No. 62072463, 71531012), the National Social Science Foundation of China (Grants No. 18ZDA309), and the Research Seed Funds of the School of Interdisciplinary Studies at Renmin University of China.

REFERENCES

- [1] W. X. Zhao, K. Zhou *et al.*, “A survey of large language models,” *arXiv preprint arXiv:2303.18223*, 2023.
- [2] X. Wang, J. Wei *et al.*, “Self-consistency improves chain of thought reasoning in language models,” in *Proc. of ICLR*, 2023.
- [3] P. Mondorf and B. Plank, “Beyond accuracy: Evaluating the reasoning behavior of large language models—a survey,” *arXiv preprint arXiv:2404.01869*, 2024.
- [4] Z. Yin, Q. Sun *et al.*, “Do large language models know what they don't know?” in *Proc. of ACL Findings*, 2023, pp. 8653–8665.
- [5] K. Li, O. Patel *et al.*, “Inference-time intervention: Eliciting truthful answers from a language model,” in *Proc. of NeurIPS*, 2023.
- [6] Z. Chen, X. Sun *et al.*, “Truth forest: Toward multi-scale truthfulness in large language models through intervention without tuning,” *Proc. of AAAI*, pp. 20967–20974, 2024.
- [7] S. Zhang, T. Yu, and Y. Feng, “Truthx: Alleviating hallucinations by editing large language models in truthful space,” in *Proc. of ACL*, 2024.
- [8] A. Madaan, N. Tandon *et al.*, “Self-refine: Iterative refinement with self-feedback,” in *Proc. of NeurIPS*, 2023.
- [9] S. Welleck, X. Lu *et al.*, “Generating sequences by learning to self-correct,” in *Proc. of ICLR*, 2023.
- [10] J. Wei, X. Wang *et al.*, “Chain of thought prompting elicits reasoning in large language models,” in *Proc. of NeurIPS*, 2022.
- [11] Z. Luo, H. Han *et al.*, “Sed: Self-evaluation decoding enhances large language models for better generation,” *arXiv preprint arXiv:2405.16552*, 2024.
- [12] Y. Zhang, S. Mao *et al.*, “Llm as a mastermind: A survey of strategic reasoning with large language models,” *arXiv preprint arXiv:2404.01230*, 2024.
- [13] Y. Zhang, Y. Li *et al.*, “Siren's song in the ai ocean: a survey on hallucination in large language models,” *arXiv preprint arXiv:2309.01219*, 2023.
- [14] D. Hendrycks, C. Burns *et al.*, “Measuring massive multitask language understanding,” in *Proc. of ICLR*, 2021.
- [15] S. Lin, J. Hilton, and O. Evans, “TruthfulQA: Measuring how models mimic human falsehoods,” in *Proc. of ACL*, 2022, pp. 3214–3252.
- [16] J. Zhang, X. Wang *et al.*, “Ratt: A thought structure for coherent and correct llmreasoning,” *arXiv preprint arXiv:2406.02746*, 2024.
- [17] J. Kaplan, S. McCandlish *et al.*, “Scaling laws for neural language models,” *arXiv preprint arXiv:2001.08361*, 2020.
- [18] N. Mündler, J. He *et al.*, “Self-contradictory hallucinations of large language models: Evaluation, detection and mitigation,” in *Proc. of ICLR*, 2024.
- [19] Z. Tao, T.-E. Lin *et al.*, “A survey on self-evolution of large language models,” *arXiv preprint arXiv:2404.14387*, 2024.
- [20] L. Pan, M. Saxon *et al.*, “Automatically correcting large language models: Surveying the landscape of diverse automated correction strategies,” *TACL*, pp. 484–506, 2024.
- [21] R. Kamoi, Y. Zhang *et al.*, “When can llms actually correct their own mistakes? a critical survey of self-correction of llms,” *arXiv preprint arXiv:2406.01297*, 2024.

- [22] Y. Gao, Y. Xiong *et al.*, “Retrieval-augmented generation for large language models: A survey,” *arXiv preprint arXiv:2312.10997*, 2023.
- [23] A. Tarski, *Introduction to logic: And to the methodology of deductive sciences*. Oxford University Press, 1941.
- [24] M. Hu, Z. Zhang *et al.*, “Uncertainty in natural language processing: Sources, quantification, and applications,” *arXiv preprint arXiv:2306.04459*, 2023.
- [25] J. Sun, C. Shaib, and B. C. Wallace, “Evaluating the zero-shot robustness of instruction-tuned language models,” in *Proc. of ICLR*, 2024.
- [26] S. Yang, E. Gribovskaya *et al.*, “Do large language models latently perform multi-hop reasoning?” *arXiv preprint arXiv:2402.16837*, 2024.
- [27] K. Cobbe, V. Kosaraju *et al.*, “Training verifiers to solve math word problems,” *arXiv preprint arXiv:2110.14168*, 2021.
- [28] Q. Cheng, T. Sun *et al.*, “Can ai assistants know what they don’t know?” *arXiv preprint arXiv:2401.13275*, 2024.
- [29] Z. Xie, J. Guo *et al.*, “Calibrating reasoning in language models with internal consistency,” *arXiv preprint arXiv:2405.18711*, 2024.
- [30] N. F. Liu, K. Lin *et al.*, “Lost in the middle: How language models use long contexts,” *TACL*, pp. 157–173, 2024.
- [31] B. Liu, J. T. Ash *et al.*, “Exposing attention glitches with flip-flop language modeling,” in *Proc. of NeurIPS*, 2023.
- [32] M. Zhang, O. Press *et al.*, “How language model hallucinations can snowball,” *arXiv preprint arXiv:2305.13534*, 2023.
- [33] E. M. Bender, T. Gebru *et al.*, “On the dangers of stochastic parrots: Can language models be too big?” in *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, 2021, pp. 610–623.
- [34] Y. Ma, D. Tsao, and H.-Y. Shum, “On the principles of parsimony and self-consistency for the emergence of intelligence,” *Frontiers of Information Technology & Electronic Engineering*, pp. 1298–1323, 2022.
- [35] S. Han, Q. Zhang *et al.*, “Llm multi-agent systems: Challenges and open problems,” *arXiv preprint arXiv:2402.03578*, 2024.
- [36] S. Kadavath, T. Conerly *et al.*, “Language models (mostly) know what they know,” *arXiv preprint arXiv:2207.05221*, 2022.
- [37] S. Yao, D. Yu *et al.*, “Tree of thoughts: Deliberate problem solving with large language models,” in *Proc. of NeurIPS*, 2023.
- [38] M. Besta, N. Blach *et al.*, “Graph of thoughts: Solving elaborate problems with large language models,” *Proc. of AAAI*, pp. 17682–17690, 2024.
- [39] E. Zelikman, G. Harik *et al.*, “Quiet-star: Language models can teach themselves to think before speaking,” *arXiv preprint arXiv:2403.09629*, 2024.
- [40] J. Huang, S. Gu *et al.*, “Large language models can self-improve,” in *Proc. of EMNLP*, 2023, pp. 1051–1068.
- [41] E. Mitchell, J. Noh *et al.*, “Enhancing self-consistency and performance of pre-trained language models through natural language inference,” in *Proc. of EMNLP*, 2022, pp. 1754–1768.
- [42] S. An, Z. Ma *et al.*, “Learning from mistakes makes llm better reasoner,” *arXiv preprint arXiv:2310.20689*, 2023.
- [43] Y. Tong, D. Li *et al.*, “Can llms learn from previous mistakes? investigating llms’ errors to boost for reasoning,” *arXiv preprint arXiv:2403.20046*, 2024.
- [44] K. Xiong, X. Ding *et al.*, “Examining inter-consistency of large language models collaboration: An in-depth analysis via debate,” in *Proc. of EMNLP Findings*, 2023, pp. 7572–7590.
- [45] C. Qian, Z. Xie *et al.*, “Scaling large-language-model-based multi-agent collaboration,” *arXiv preprint arXiv:2406.07155*, 2024.
- [46] D. Paul, M. Ismayilzada *et al.*, “REFINER: Reasoning feedback on intermediate representations,” in *Proc. of EACL*, 2024, pp. 1100–1126.
- [47] Y. Du, S. Li *et al.*, “Improving factuality and reasoning in language models through multiagent debate,” *arXiv preprint arXiv:2305.14325*, 2023.
- [48] N. Shinn, F. Cassano *et al.*, “Reflexion: language agents with verbal reinforcement learning,” in *Proc. of NeurIPS*, 2023, pp. 8634–8652.
- [49] X. Chen, M. Lin *et al.*, “Teaching large language models to self-debug,” in *Proc. of ICLR*, 2024.
- [50] H. Kang, J. Ni, and H. Yao, “Ever: Mitigating hallucination in large language models through real-time verification and rectification,” *arXiv preprint arXiv:2311.09114*, 2023.
- [51] A. Mishra, A. Asai *et al.*, “Fine-grained hallucination detection and editing for language models,” *arXiv preprint arXiv:2401.06855*, 2024.
- [52] Y.-S. Chuang, Y. Xie *et al.*, “Dola: Decoding by contrasting layers improves factuality in large language models,” in *Proc. of ICLR*, 2024.
- [53] W. Shi, X. Han *et al.*, “Trusting your evidence: Hallucinate less with context-aware decoding,” *arXiv preprint arXiv:2305.14739*, 2023.
- [54] J. Lu, C. Wang, and J. Zhang, “Diver: Large language model decoding with span-level mutual information verification,” *arXiv preprint arXiv:2406.02120*, 2024.
- [55] Y. Xiao and W. Y. Wang, “On hallucination and predictive uncertainty in conditional language generation,” in *Proc. of EACL*, 2021, pp. 2734–2744.
- [56] Z. Lin, S. Trivedi, and J. Sun, “Generating with confidence: Uncertainty quantification for black-box large language models,” *TMLR*, 2024.
- [57] Y. A. Yadkori, I. Kuzborskij *et al.*, “To believe or not to believe your llm,” *arXiv preprint arXiv:2406.02543*, 2024.
- [58] D. Deng, G. Chen *et al.*, “Uncertainty estimation by fisher information-based evidential deep learning,” in *Proc. of ICML*, 2023, pp. 7596–7616.
- [59] Y. Gal and Z. Ghahramani, “Dropout as a bayesian approximation: Representing model uncertainty in deep learning,” in *Proc. of ICML*, 2016, pp. 1050–1059.
- [60] S. Diao, P. Wang *et al.*, “Active prompting with chain-of-thought for large language models,” *arXiv preprint arXiv:2302.12246*, 2023.
- [61] J. Chen and J. Mueller, “Quantifying uncertainty in answers from any language model and enhancing their trustworthiness,” *arXiv preprint arXiv:2308.16175*, 2023.
- [62] D. Zheng, D. Liu *et al.*, “Trustscore: Reference-free evaluation of llm response trustworthiness,” *arXiv preprint arXiv:2402.12545*, 2024.
- [63] P. Manakul, A. Liusie, and M. Gales, “SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models,” in *Proc. of EMNLP*, 2023, pp. 9004–9017.
- [64] C. Chen, K. Liu *et al.*, “INSIDE: LLMs’ internal states retain the power of hallucination detection,” in *Proc. of ICLR*, 2024.
- [65] R. Cohen, M. Hamri *et al.*, “LM vs LM: Detecting factual errors via cross examination,” in *Proc. of EMNLP*, 2023, pp. 12 621–12 640.
- [66] W. Yuan, G. Neubig, and P. Liu, “BARTScore: Evaluating generated text as text generation,” in *Proc. of NeurIPS*, 2021.
- [67] J. Fu, S.-K. Ng *et al.*, “Gptscore: Evaluate as you desire,” *arXiv preprint arXiv:2302.04166*, 2023.
- [68] W. Saunders, C. Yeh *et al.*, “Self-critiquing models for assisting human evaluators,” *arXiv preprint arXiv:2206.05802*, 2022.
- [69] N. McAleese, R. M. Pokorny *et al.*, “Llm critics help catch llm bugs,” *arXiv preprint arXiv:2407.00215*, 2024.
- [70] Y. Xie, K. Kawaguchi *et al.*, “Self-evaluation guided beam search for reasoning,” in *Proc. of NeurIPS*, 2023.
- [71] H. Wang, A. Prasad *et al.*, “Soft self-consistency improves language model agents,” *arXiv preprint arXiv:2402.13212*, 2024.
- [72] H. Chen, A. Saha *et al.*, “Personalized distillation: Empowering open-sourced LLMs with adaptive learning for code generation,” in *Proc. of EMNLP*, 2023.
- [73] M. Besta, F. Memedi *et al.*, “Demystifying chains, trees, and graphs of thoughts,” *arXiv preprint arXiv:2401.14295*, 2024.
- [74] J. Jung, L. Qin *et al.*, “Maieutic prompting: Logically consistent reasoning with recursive explanations,” in *Proc. of EMNLP*, 2022, pp. 1266–1279.
- [75] R. Battiti, *Maximum satisfiability problemMaximum Satisfiability Problem*, 2009, pp. 2035–2041.
- [76] B. Huang, S. Lu *et al.*, “Enhancing large language models in coding through multi-perspective self-consistency,” *arXiv preprint arXiv:2309.17272*, 2023.
- [77] X. Chen, R. Aksitov *et al.*, “Universal self-consistency for large language model generation,” *arXiv preprint arXiv:2311.17311*, 2023.
- [78] Y. Li, Z. Lin *et al.*, “Making language models better reasoners with step-aware verifier,” in *Proc. of ACL*, 2023, pp. 5315–5333.
- [79] C. Fernando, D. Banarse *et al.*, “Promptbreeder: Self-referential self-improvement via prompt evolution,” *arXiv preprint arXiv:2309.16797*, 2023.
- [80] I. Harvey, “The microbial genetic algorithm,” in *Advances in Artificial Life. Darwin Meets von Neumann*, 2011, pp. 126–133.
- [81] O. Khattab, A. Singhvi *et al.*, “DSPy: Compiling declarative language model calls into state-of-the-art pipelines,” in *Proc. of ICLR*, 2024.
- [82] X. Wang and D. Zhou, “Chain-of-thought reasoning without prompting,” *arXiv preprint arXiv:2402.10200*, 2024.
- [83] Y. Tian, B. Peng *et al.*, “Toward self-improvement of llms via imagination, searching, and criticizing,” *arXiv preprint arXiv:2404.12253*, 2024.
- [84] A. Agarwal, A. Tzen, and C. Tew, “Improving logical consistency in pre-trained language models using natural language inference,” 2022.
- [85] A. Ignatiev, A. Morgado, and J. Marques-Silva, “RC2: An Efficient MaxSAT Solver,” *Journal on Satisfiability, Boolean Modeling and Computation*, pp. 53–64, 2019.

- [86] L. Wang, C. Ma *et al.*, “A survey on large language model based autonomous agents,” *Frontiers of Computer Science*, p. 186345, 2024.
- [87] J. Huang, X. Chen *et al.*, “Large language models cannot self-correct reasoning yet,” in *Proc. of ICLR*, 2024.
- [88] A. P. Jacob, Y. Shen *et al.*, “The consensus game: Language model generation via equilibrium search,” in *Proc. of ICLR*, 2024.
- [89] X. Liang, S. Song *et al.*, “Uhgeval: Benchmarking the hallucination of chinese large language models via unconstrained generation,” *arXiv preprint arXiv:2311.15296*, 2023.
- [90] K. Yang, Y. Tian *et al.*, “Re3: Generating longer stories with recursive re-prompting and revision,” in *Proc. of EMNLP*, 2022, pp. 4393–4479.
- [91] T. Schick, J. A. Yu *et al.*, “PEER: A collaborative language model,” in *Proc. of ICLR*, 2023.
- [92] A. Chen, P. Pasupat *et al.*, “Purr: Efficiently editing language model hallucinations by denoising language model corruptions,” *arXiv preprint arXiv:2305.14908*, 2023.
- [93] L. Gao, Z. Dai *et al.*, “RARR: Researching and revising what language models say, using language models,” in *Proc. of ACL*, 2023, pp. 16477–16508.
- [94] X. Liang, H. Wang *et al.*, “Controlled text generation for large language model with dynamic attribute graphs,” *arXiv preprint arXiv:2402.11218*, 2024.
- [95] X. L. Li, A. Holtzman *et al.*, “Contrastive decoding: Open-ended text generation as optimization,” *arXiv preprint arXiv:2210.15097*, 2022.
- [96] Z. Zhao, E. Monti *et al.*, “Enhancing contextual understanding in large language models through contrastive decoding,” *arXiv preprint arXiv:2405.02750*, 2024.
- [97] C. Burns, H. Ye *et al.*, “Discovering latent knowledge in language models without supervision,” in *Proc. of ICLR*, 2023.
- [98] Z. Zheng, Y. Wang *et al.*, “Attention heads of large language models: A survey,” *arXiv preprint arXiv:2409.03752*, 2024.
- [99] W. Wu, Y. Wang *et al.*, “Retrieval head mechanistically explains long-context factuality,” *arXiv preprint arXiv:2404.15574*, 2024.
- [100] Y. Wu, Z. Sun *et al.*, “Self-play preference optimization for language model alignment,” *arXiv preprint arXiv:2405.00675*, 2024.
- [101] Z. Chen, Y. Deng *et al.*, “Self-play fine-tuning converts weak language models to strong language models,” in *Proc. of ICML*, 2024, pp. 6621–6642.
- [102] X. Pang, S. Tang *et al.*, “Self-alignment of large language models via monopolylogue-based social scene simulation,” in *Proc. of ICML*, 2024, pp. 39416–39447.
- [103] J. Schulman, F. Wolski *et al.*, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.
- [104] R. Rafailov, A. Sharma *et al.*, “Direct preference optimization: Your language model is secretly a reward model!” in *Proc. of NeurIPS*, 2023, pp. 53728–53741.
- [105] A. Köpf, Y. Kilcher *et al.*, “Openassistant conversations - democratizing large language model alignment,” in *Proc. of NeurIPS*, 2023, pp. 47669–47681.
- [106] J. Ji, M. Liu *et al.*, “Beavertails: Towards improved safety alignment of llm via a human-preference dataset,” in *Proc. of NeurIPS*, 2023, pp. 24678–24704.
- [107] Y. Bai, S. Kadavath *et al.*, “Constitutional ai: Harmlessness from ai feedback,” *arXiv preprint arXiv:2212.08073*, 2022.
- [108] Z. Sun, Y. Shen *et al.*, “SALMON: Self-alignment with instructable reward models,” in *Proc. of ICLR*, 2024.
- [109] K. Ethayarajh, Y. Choi, and S. Swayamdipta, “Understanding dataset difficulty with \mathcal{V} -usable information,” in *Proc. of ICML*, 2022, pp. 5988–6008.
- [110] S. Kim, S. Bae *et al.*, “Aligning large language models through synthetic feedback,” *arXiv preprint arXiv:2305.13735*, 2023.
- [111] L. Ouyang, J. Wu *et al.*, “Training language models to follow instructions with human feedback,” *Proc. of NeurIPS*, pp. 27730–27744, 2022.
- [112] Y. Xu, X. Liu *et al.*, “Chatglm-math: Improving math problem-solving in large language models with a self-critique pipeline,” *arXiv preprint arXiv:2404.02893*, 2024.
- [113] X. Xu, M. Li *et al.*, “A survey on knowledge distillation of large language models,” *arXiv preprint arXiv:2402.13116*, 2024.
- [114] Y. Gu, L. Dong *et al.*, “Minillm: Knowledge distillation of large language models,” in *Proc. of ICLR*, 2024.
- [115] R. Agarwal, N. Vieillard *et al.*, “On-policy distillation of language models: Learning from self-generated mistakes,” in *Proc. of ICLR*, 2024.
- [116] S. Ye, Y. Jo *et al.*, “Selfee: Iterative self-revising llm empowered by self-feedback generation,” *Blog post*, 2023.
- [117] J. Jung, P. West *et al.*, “Impossible distillation: from low-quality model to high-quality dataset & model for summarization and paraphrasing,” *arXiv preprint arXiv:2305.16635*, 2023.
- [118] Y. Wang, Y. Kordi *et al.*, “Self-instruct: Aligning language models with self-generated instructions,” in *Proc. of ACL*, 2023, pp. 13484–13508.
- [119] M. Sclar, Y. Choi *et al.*, “Quantifying language models’ sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting,” *arXiv preprint arXiv:2310.11324*, 2023.
- [120] Q. Yu, Z. Zheng *et al.*, “Xfinder: Robust and pinpoint answer extraction for large language models,” *arXiv preprint arXiv:2405.11874*, 2024.
- [121] F. Ye, M. Yang *et al.*, “Benchmarking llms via uncertainty quantification,” *arXiv preprint arXiv:2401.12794*, 2024.
- [122] X. Wang, Z. Zhang *et al.*, “Ubench: Benchmarking uncertainty in large language models with multiple choice questions,” *arXiv preprint arXiv:2406.12784*, 2024.
- [123] Z. Yang, Y. Zhang *et al.*, “Can large language models always solve easy problems if they can solve harder ones?” *arXiv preprint arXiv:2406.12809*, 2024.
- [124] E. Rabinovich, S. Ackerman *et al.*, “Predicting question-answering performance of large language models through semantic consistency,” in *Proceedings of the Third Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, 2023, pp. 138–154.
- [125] Y. Elazar, N. Kassner *et al.*, “Measuring and improving consistency in pretrained language models,” *TACL*, pp. 1012–1031, 2021.
- [126] J. Qi, R. Fernández, and A. Bisazza, “Cross-lingual consistency of factual knowledge in multilingual language models,” in *Proc. of EMNLP*, 2023.
- [127] M. Jang, D. S. Kwon, and T. Lukasiewicz, “BECEL: Benchmark for consistency evaluation of language models,” in *Proc. of COLING*, 2022, pp. 3680–3696.
- [128] Z. Lin, Z. Gou *et al.*, “Criticbench: Benchmarking llms for critique-correct reasoning,” *arXiv preprint arXiv:2402.14809*, 2024.
- [129] Z. Tan, L. Wei *et al.*, “Can i understand what i create? self-knowledge evaluation of large language models,” *arXiv preprint arXiv:2406.06140*, 2024.
- [130] Y. Chang, X. Wang *et al.*, “A survey on evaluation of large language models,” *ACM Trans. Intell. Syst. Technol.*, 2024.
- [131] Y. Huang, Y. Bai *et al.*, “C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models,” *Proc. of NeurIPS*, 2024.
- [132] M. Suzgun, N. Scales *et al.*, “Challenging big-bench tasks and whether chain-of-thought can solve them,” *arXiv preprint arXiv:2210.09261*, 2022.
- [133] P. Clark, I. Cowhey *et al.*, “Think you have solved question answering? try arc, the ai2 reasoning challenge,” *arXiv preprint arXiv:1803.05457*, 2018.
- [134] M. T. Pilehvar and J. Camacho-Collados, “Wic: the word-in-context dataset for evaluating context-sensitive meaning representations,” *Proceedings of NAACL 2019 (short)*, 2019.
- [135] M. Chen, J. Tworek *et al.*, “Evaluating large language models trained on code,” *arXiv preprint arXiv:2107.03374*, 2021.
- [136] D. Hendrycks, C. Burns *et al.*, “Measuring mathematical problem solving with the math dataset,” *NeurIPS*, 2021.
- [137] D. Jiang, J. Zhang *et al.*, “Self-[in] correct: Llms struggle with refining self-generated responses,” *arXiv preprint arXiv:2404.04298*, 2024.
- [138] K. Stechly, M. Marquez, and S. Kambhampati, “GPT-4 doesn’t know it’s wrong: An analysis of iterative prompting for reasoning problems,” in *NeurIPS 2023 Foundation Models for Decision Making Workshop*, 2023.
- [139] K. Valmeekam, M. Marquez, and S. Kambhampati, “Investigating the effectiveness of self-critiquing in LLMs solving planning tasks,” in *NeurIPS 2023 Foundation Models for Decision Making Workshop*, 2023.
- [140] G. Yona, R. Aharoni, and M. Geva, “Can large language models faithfully express their intrinsic uncertainty in words?” *arXiv preprint arXiv:2405.16908*, 2024.
- [141] S. Kapoor, N. Gruver *et al.*, “Large language models must be taught to know what they don’t know,” *arXiv preprint arXiv:2406.08391*, 2024.
- [142] L. Chen, Z. Liang *et al.*, “Teaching large language models to express knowledge boundary from their own signals,” *arXiv preprint arXiv:2406.10881*, 2024.
- [143] M. Jin, Q. Yu *et al.*, “The impact of reasoning step length on large language models,” in *Proc. of ACL Findings*, 2024, pp. 1830–1842.



Xun Liang (Senior Member, IEEE) received the B.Sc. and Ph.D. degrees in computer engineering from Tsinghua University, Beijing, China, in 1989 and 1993, respectively, and the M.Sc. degree in operations research from Stanford University, Palo Alto, CA, USA, in 1999. He worked as a Post-Doctoral Fellow with the Institute of Computer Science and Technology, Peking University, Beijing, from 1993 to 1995, and with the Department of Computer Engineering, University of New Brunswick, Fredericton, NB, Canada, from 1995 to 1997. He worked as a

CTO, leading over ten intelligent information products in RixolInfo Ltd., CA, USA, from 2000 to 2007, and was the Director of the Data Mining Lab, Institute of Computer Science and Technology, Peking University, from 2005 to 2009. He is currently a professor with the School of Information, Renmin University of China. His research interests include support vector machines, social computing and large language models.



Shichao Song is currently a PhD student at the School of Information, Renmin University of China, under the supervision of Prof. Xun Liang. His research interests span a wide range of topics, including internal consistency mining of LLMs, LLM interpretability, and reliable evaluation methods for LLMs. For more information, visit his website at <https://ki-seki.github.io/>.



Zifan Zheng is currently a research intern at the Large Language Model Center of the Institute for Advanced Algorithms Research, Shanghai. He received the B.S. degree in Computer Science and Technology from Beijing Institute of Technology, China, in 2024. His research interests include LLMs interpretability, reliable evaluation and social network analysis.



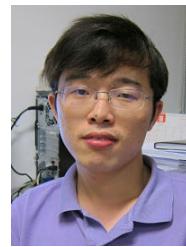
Hanyu Wang is a Ph.D. student at the School of Information, Renmin University of China, under the supervision of Professor Xun Liang. His research areas include large language models, controllable text generation in large language models, and controlled decoding.



Qingchen Yu is currently a research intern at the Large Language Model Center of the Institute for Advanced Algorithms Research in Shanghai. He is also a master's student at Shanghai University. His research interests include machine learning, LLM evaluation, and prompt engineering.



Xunkai Li is currently working toward the PhD degree with the school of Computer Science, Beijing Institute of Technology, advised by Prof. Rong-Hua Li. He received the BS degree in computer science from Shandong University in 2022. His research interest lies in Data-centric ML and Graph-ML within complex relational data and new learning paradigms. He has published 5+ papers in top DB/DM/AI conferences such as VLDB, WWW, AAAI as the first author.



Rong-Hua Li received the Ph.D. degree in computer science from The Chinese University of Hong Kong, Hong Kong, in 2013. He is currently a Professor with the Beijing Institute of Technology, Beijing, China. His research interests include graph data management and mining, social network analysis, graph computation systems, and graph-based machine learning.



Yi Wang is a key member of the State Key Laboratory of Media Convergence Production Technology and Systems, a Senior Engineer in the Technology Department in Xinhua News Agency and one of the Xinhua News Agency 100 high-level talents. She has a long career engaged in news production and new technology innovation research. She is highly experienced in intelligent algorithm research and media integration, and expert in big data analysis and data mining.



and recommendation algorithms.

Zhonghao Wang is a Senior Algorithm Engineer at the State Key Laboratory of Media Convergence Production Technology and the AI Director at the Tech Bureau of Xinhua News Agency. He holds both a Bachelor's and a Master's degree from Shanghai Jiaotong University. He has previously served as an Algorithm Engineer in Alibaba's advertising department, where he specialized in developing interactive advertising algorithms. His primary interests lie in the application of algorithms and engineering in industry, with a particular focus on large-scale models



Feiyu Xiong is the Head of the Large Language Model Center of the Institute for Advanced Algorithms Research-Shanghai. He holds a Bachelor's degree from Huazhong University of Science and Technology and a Ph.D. from Drexel University. He has previously served as the Head of Data Intelligence for Alibaba's Business Middle Platform and the Head of the Data Platform for Taobao and Tmall Group. During his tenure at Alibaba, he was primarily responsible for the intelligent construction of systems related to core e-commerce transactions.



tion optimization.

Zhiyu Li received his Ph.D. in Computer Science from the School of Information, Renmin University of China, in 2019. He is currently a Senior Researcher at the Large Language Model Center of the Institute for Advanced Algorithms Research-Shanghai. He has published over 30 papers in top-tier conferences and journals such as TKDE, KDD, and ACL. His current responsibilities include research and application implementation related to large language models. His research interests include model pre-training, model alignment, and hallucina-