AUTODETECT: Towards a Unified Framework for Automated Weakness Detection in Large Language Models

Jiale Cheng[1,2*] , **Yida Lu**[1,2*] , **Xiaotao Gu**[2] , **Pei Ke**[1] , **Xiao Liu**[2,3] ,
Yuxiao Dong[3] , **Hongning Wang**[1] , **Jie Tang**[3] , **Minlie Huang**[1†]
[1]The Conversational Artificial Intelligence (CoAI) Group, Tsinghua University
[2]Zhipu AI
[3]The Knowledge Engineering Group (KEG), Tsinghua University
`{chengjl23, lu-yd20}@mails.tsinghua.edu.cn, aihuang@tsinghua.edu.cn`

Abstract

Although Large Language Models (LLMs) are becoming increasingly powerful, they still exhibit significant but subtle weaknesses, such as mistakes in instruction-following or coding tasks. As these unexpected errors could lead to severe consequences in practical deployments, it is crucial to investigate the limitations within LLMs systematically. Traditional benchmarking approaches cannot thoroughly pinpoint specific model deficiencies, while manual inspections are costly and not scalable. In this paper, we introduce a unified framework, AUTODETECT, to automatically expose weaknesses in LLMs across various tasks. Inspired by the educational assessment process that measures students' learning outcomes, AUTODETECT consists of three LLM-powered agents: Examiner, Questioner, and Assessor. The collaboration among these three agents is designed to realize comprehensive and in-depth weakness identification. Our framework demonstrates significant success in uncovering flaws, with an identification success rate exceeding 30% in prominent models such as ChatGPT and Claude. More importantly, these identified weaknesses can guide specific model improvements, proving more effective than untargeted data augmentation methods like Self-Instruct. Our approach has led to substantial enhancements in popular LLMs, including the Llama series and Mistral-7b, boosting their performance by over 10% across several benchmarks. Code and data are publicly available at `https://github.com/thu-coai/AutoDetect`.

1 Introduction

The developments in Large Language Models (LLMs) are phenomenal: such models have demonstrated excellent performance in diverse tasks

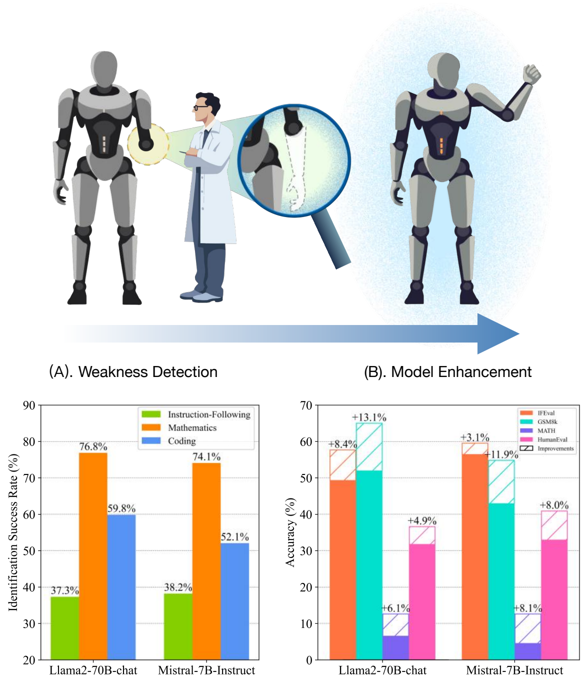


Figure 1: Effective weakness discovery can well guide model enhancement. AUTODETECT can achieve high identification success rates in the instruction-following, mathematics and coding tasks (A). Moreover, leveraging this data, we can further improve LLMs (B).

(Brown et al., 2020; Zeng et al., 2022; Chowdhery et al., 2023; Touvron et al., 2023a; GLM et al., 2024). After elaborate alignment (Ouyang et al., 2022; Cheng et al., 2023; Ji et al., 2024a), LLMs can achieve human-level performance in real-world applications (OpenAI, 2022; Anthropic, 2023). Nevertheless, these models are prone to making unexpected mistakes (Ouyang et al., 2022; Bubeck et al., 2023). For instance, while LLMs are skilled at complex algorithm problems, they may struggle with basic coding concepts (§5.1). These unexpected mistakes can result in unforeseen consequences like system failures and significant safety issues (Ruan et al., 2023). Consequently, systematically identifying and addressing these weaknesses is essential to enhancing the performance and trust-

[*] Equal contributions.
[†] Corresponding author.
[2]Work done when JC and YL interned at Zhipu AI.

worthiness of LLMs.

However, the journey to reveal these weaknesses is challenging. Manual examinations rely on human experts, which are too labor-intensive and costly to scale. Automated methods, meanwhile, typically employ either static (Cobbe et al., 2021; Srivastava et al., 2022; Liu et al., 2023) and dynamic (Bai et al., 2024; Wang et al., 2024) benchmarks. Unfortunately, benchmarks are intentionally structured to assess and rank a series of models, not to identify weaknesses inherent to individual models. More specifically, benchmarks are designed to be model-agnostic without counting on specific model responses and thus inept at identifying individualized weaknesses. Moreover, benchmarks suffer from infrequent updates, data leakage (Yang et al., 2023b; Wei et al., 2023), and leaderboard swamping (Guo et al., 2023), which further limit their utility for thorough model-specific weakness assessment.

In this paper, we introduce a pioneering unified framework, AUTODETECT, aiming to systematically and automatically expose potential weaknesses within LLMs across a variety of tasks. In our framework, illustrated in Figure 2, we adopt a methodology analogous to educational assessment systems, comprising creating comprehensive questions to evaluate students and reviewing their responses to identify individualized weaknesses. AUTODETECT involves the development of a holistic testing system to assess and challenge student abilities. Moreover, this system is not static but constantly optimized and adapted to specific model performance, providing a tailored and effective weakness discovery. Specifically, our framework integrates three specialized roles implemented by LLM-based agents:

- **Examiner** is tasked with building a comprehensive taxonomy featuring diverse test points and dynamically optimizing the framework based on the target model's performance in order to provide a refined and tailored framework for identifying potential weaknesses.
- **Questioner** is responsible for creating challenging questions according to each test point. Through iterative explorations, this agent continually hypothesizes about the model's weaknesses, effectively adapting the generation of questions as new deficiencies emerge.
- **Assessor** needs to analyze the target model's responses and speculate on potential issues to be incorporated into the testing system, which is crucial to tailored assessments.

The collaboration among the Examiner, Questioner, and Assessor fosters an extensive and effective assessment process. By learning from these weaknesses, AUTODETECT further facilitates model improvements (Figure 1).

Through extensive experiments, we demonstrate that AUTODETECT is able to perform effective weakness exposure across a diverse range of tasks, including instruction-following, mathematical reasoning, and coding, achieving an impressive identification success rate of over 50% in multiple strong LLMs and even more than 30% in GPT-3.5-turbo and Claude-3-sonnet. Furthermore, our weakness identification process can effectively guide model enhancement. Notably, by integrating about 1,000 samples derived from AUTODETECT to fine-tune popular open-source models like Mistral and Llama series, we have achieved over 10% improvements across several benchmarks, showing the benefits of learning from targeted weakness detection.

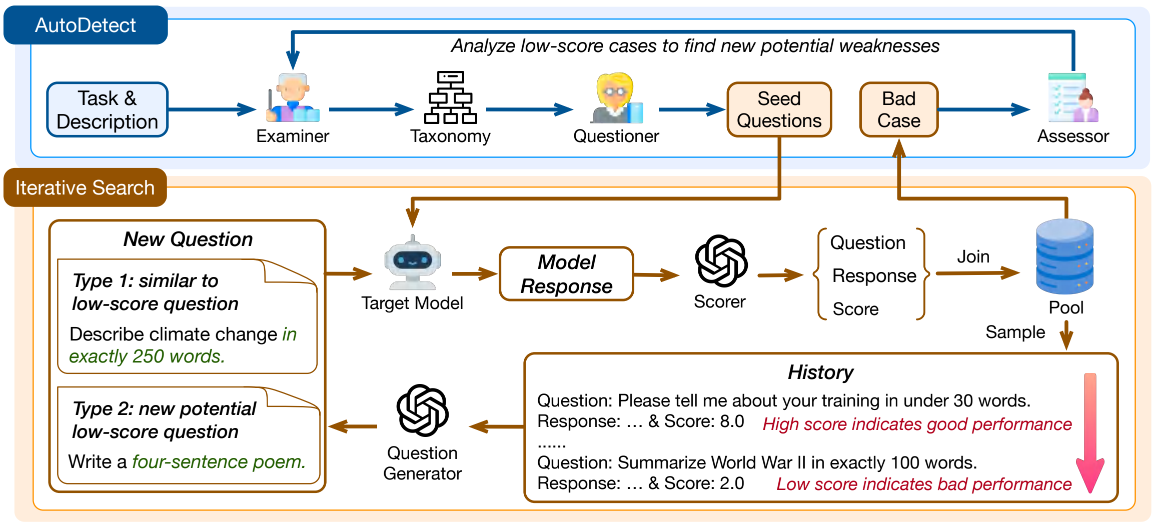
Our contribution can be summarized as follows:

- To the best of our knowledge, we are the first to systematically explore weakness identification in LLMs on multiple generic tasks, including instruction-following, mathematical reasoning, and coding, offering a unified framework for automatic weakness detection.
- AUTODETECT has demonstrated exceptional adaptability and effectiveness, with a success rate of over 50% in uncovering deficiencies across multiple models and tasks.
- AUTODETECT facilitates significant model improvements. Leveraging the data derived from the weakness detection process, we can effectively enhance model performance, yielding over 10% improvements on several tasks.

2 Related Work

2.1 Evaluation Benchmarks

Numerous benchmarks (Hendrycks et al., 2020; Cobbe et al., 2021; Chen et al., 2021; Zhou et al., 2023; Liu et al., 2023) are designed to evaluate various capabilities of LLMs, as well as some dynamic benchmarks (Zhu et al., 2023; Bai et al., 2024; Wang et al., 2024). However, the fundamental purpose of benchmarks is to compare a range of models and accurately rank them instead of identifying specific model defects. As a result, they are designed to be model-agnostic and cannot pro-



AutoDetect
Analyze low-score cases to find new potential weaknesses
Task & Description
Examiner
Taxonomy
Questioner
Seed Questions
Bad Case
Assessor
Iterative Search
New Question
Type 1: similar to low-score question
Describe climate change in exactly 250 words.
Type 2: new potential low-score question
Write a four-sentence poem.
Target Model
Model Response
Scorer
Question
Response
Score
Join
Pool
Sample
Question Generator
History
Question: Please tell me about your training in under 30 words.
Response: ... & Score: 8.0
High score indicates good performance
......
Question: Summarize World War II in exactly 100 words.
Response: ... & Score: 2.0
Low score indicates bad performance

Figure 2: Our framework comprises two cycles, with the circulation consisting of the Examiner, Questioner, and Assessor, providing a comprehensive and tailored testing framework. Meanwhile, iterative search enables the adjustment of question difficulty for the target model, effectively identifying weaknesses.

vide a thorough discovery of particular model flaws. Additionally, static benchmarks often suffer from issues like data leakage (Yang et al., 2023b; Wei et al., 2023) and leaderboard swamping (Guo et al., 2023), while dynamic benchmarks often have trouble in coverage, and the methods used to construct these dynamic benchmarks usually lack universality. These limitations suggest that relying solely on benchmarks makes it challenging to thoroughly uncover model flaws, thus failing to offer practical guidance for further improvements.

2.2 Red Teaming

Due to the limitations of automated methods mentioned above, an essential way to effectively expose weaknesses within LLMs is manual checkup, which is similar to red teaming, an important strategy in the safety domain (Deng et al., 2023; Ji et al., 2023; Sun et al., 2023; Ji et al., 2024b) to identify safety issues of AI systems. Early studies largely counted on manual efforts to create red teaming queries (Dinan et al., 2019; Xu et al., 2020). However, manual red teaming is hampered by its high costs and inherent lack of diversity, limiting its scalability. Recently, the use of language models for automated red team attacks has been proposed (Perez et al., 2022) and widely adopted (Ganguli et al., 2022; Zhang et al., 2022; Chao et al., 2023). Nevertheless, the application of automated weakness detection to general-purpose tasks remains underexplored. In this work, we introduce a unified framework for identifying model flaws beyond safety issues. We have successfully implemented this framework in various tasks, including instruction-following, mathematical reasoning, and coding, demonstrating its impressive effectiveness and broad applicability.

3 Method

3.1 Problem Definition

Our primary objective is to develop a unified framework, aiming to automatically and systematically identify potential weaknesses in LLMs across generic tasks. For a given task with its description, denoted as (T, D), the weakness identification process can be represented as follows:

$$\mathbb{W} = \textsc{AutoDetect}(T, D) \tag{1}$$

where $\mathbb{W}$ stands for the set of problems that the target model fails to address accurately. We consider these failures as model weaknesses, which are evaluated by a strong LLM judge.

3.2 AutoDetect Framework

The overall framework of our method is shown in Figure 2. AutoDetect is designed to comprehensively assess language model capabilities through a specialized circular search strategy, encompassing three distinct roles: *Examiner, Questioner, and Assessor*. Each role is critical, leveraging the strengths of LLM-powered agents in a collaborative manner to explore and expose weaknesses in target models.

As shown in Algorithm 1, the process begins with the Examiner, who is tasked with developing a detailed taxonomy $\mathbb{C}$ based on the given task and its description (T, D). This taxonomy is crucial as it organizes the task into manageable, focused categories $(c_1, \cdots, c_n)$, each including several knowledge points $(k_1, \cdots, k_m)$, which guide the subsequent assessments. The structured decomposition is essential for thorough evaluation and is represented as:

$$\mathbb{C} = \textbf{Examiner}(T, D) \tag{2}$$

Following taxonomy creation, the Questioner takes over, generating a seed set of questions $\mathbb{S}$ and initiating an iterative search process to craft questions $\mathbb{Q}$ that probe weaknesses at each knowledge point. The iterative process allows for adaptive questioning strategies that progressively increase in complexity, ensuring a depth of testing tailored to each model's capabilities. This can be formalized as:

$$\mathbb{Q} = \textbf{Questioner}(\mathbb{H}) \tag{3}$$

Here, $\mathbb{H}$ denotes the search history, starting from $\mathbb{S}$, facilitating a dynamic exploration of model weaknesses. Importantly, the role of the Assessor is integral to refining the evaluation process to become thorough and model-specific. As the assessment progresses, the Assessor critically analyzes instances where the target model underperforms (indicated by low scores), identifying new potential weaknesses, k_{new}, expressed as:

$$k_{new} = \textbf{Assessor}(\mathbb{H}_{low}) \tag{4}$$

This insight leads to dynamic refinement of the taxonomy by the Examiner, ensuring that our framework stays relevant and effective at discovering new deficiencies. The cyclical interaction among the Examiner, Questioner, and Assessor realizes a continuous improvement loop, making our testing framework not only comprehensive but also sensitive to the evolving capabilities of different LLMs. Detailed descriptions of tasks and prompts in our framework are provided in Appendix B and Appendix C.

3.3 Iterative Search

As the collaboration among the three roles guarantees the coverage and model-specificity of our framework, another crucial problem is how to effectively identify questions where the target model underperforms. Therefore, we leverage the strong exploration and evaluation capabilities of LLMs (Yang et al., 2023a; Ke et al., 2023; Zheng et al., 2024) to develop an iterative search process. Specifically, we first generate five questions for each knowledge point to create a seed set. The performance of the target model on this set is evaluated using a reference-based scoring method for reliability (Zheng et al., 2024), where the reference responses are provided by GPT-4. Subsequently, we rank historical samples by scores, with lower scores indicating poorer performance, to generate new questions that may expose model flaws. We then have the target model generate responses to the proposed question and score it, adding the result to our history collection. Through this iterative search process, we can effectively identify low-scoring questions, pinpointing specific weaknesses in the target model at particular knowledge points.

3.4 Model Enhancement

The ultimate goal of weakness discovery is to help models improve. To validate that the identified weaknesses are non-trivial and can contribute to model enhancement, we further fine-tune the target model using the questions and reference answers obtained from the weakness detection process. Formally, the loss function is expressed as follows:

$$\mathcal{L} = -\frac{1}{N}\sum_{t=1}^{N} \log P(R_t|Q, R_{<t}) \tag{5}$$

Here, Q denotes the questions derived from the search, R signifies the reference answers generated by GPT-4, and N represents the length of R.

4 Experiments

To comprehensively demonstrate the superior performance of AutoDetect, we have conducted extensive experiments on diverse tasks: instruction-following, mathematics, and coding, including weakness detection (§4.1), model enhancement (§4.2), comparisons with baseline methods (§4.3), and iterative weakness recovery (§4.4). Implementation details can be found in Appendix D.

4.1 Weakness Detection

We investigate three distinct tasks—instruction-following, mathematics, and coding—to demonstrate the generalization capabilities of AutoDetect. The instruction-following task concentrates on providing the model with specific constraints, like formats and content. The mathematics task focuses on questions at a high school level, while the coding task focuses on Python in order to guarantee the correctness of the problems produced by GPT-4.

Model	Instruction following ISR (%) ↓			Mathematics ISR (%) ↓			Coding ISR (%) ↓			Average ISR (%) ↓
	Format	General	Overall	Geometry	Analysis	Overall	DS.	MA.	Overall	
Open-source Large Language Models										
Llama2-7b-Chat	**55.3**	**37.8**	**43.3**	**89.8**	**93.3**	**88.8**	**83.3**	**81.1**	**74.8**	**69.0**
Llama2-13b-Chat	52.4	35.0	39.7	88.6	88.9	86.1	78.9	73.3	67.5	64.4
Llama2-70b-Chat	51.3	34.9	37.3	74.1	81.1	76.9	72.2	72.2	59.8	58.0
Mistral-7b-Instruct	52.9	32.5	38.2	77.8	71.1	74.1	66.7	56.7	52.1	54.8
Llama3-8b-Instruct	42.1	19.4	27.6	61.1	68.9	60.9	45.6	50.0	41.9	43.5
Llama3-70b-Instruct	**18.5**	**4.3**	**10.2**	41.9	30.0	38.7	**15.6**	**16.7**	**15.7**	**21.5**
Closed-source Large Language Models										
GPT-3.5-turbo	35.1	21.7	25.5	56.3	35.6	50.2	40.0	30.0	32.5	36.1
Claude-3-sonnet	29.3	12.8	19.2	45.6	42.2	43.8	37.3	32.0	29.9	31.0
Mistral Large	32.1	13.9	20.3	41.5	30.0	33.9	38.9	33.3	26.4	26.8
GLM-4-Air	32.0	9.2	17.8	**33.7**	**26.7**	**33.4**	32.2	45.6	28.7	26.7

Table 1: ISR on multiple LLMs across the Instruction-following, Mathematics and Coding task. We showcase the overall result and select two subtasks with the highest ISR. In each column, the highest ISR is highlighted in **red** and the lowest in **green**. DS. denotes Data Structure and MA. refers to Mathematics and Algorithms.

Metric	Accuracy (%)	Fleiss Kappa
Reasonableness	98.0	0.493
Agreement	88.7	0.472
Correctness	87.3	0.439

Table 2: Human evaluation results for the AutoDetect process. We evaluate for question reasonableness, agreement with gpt-4 evaluation, and the correctness of generated reference. Each Fleiss Kappa is greater than 0.4, indicating moderate agreement between annotators.

Evaluation Metrics In the iterative search process, we employ the scoring prompt from MT-bench (Zheng et al., 2024), which achieves an 85% agreement rate with human annotators. In our methodology, a score of three or below on a scale of ten indicates an error in the target model's response, as we additionally ask LLM not to score higher than three if the answer is wrong. We find an agreement rate over 88% with human when judging the correctness of model responses (Table 2). Leveraging this, we define the identification success rate (ISR) as:

$$ISR = \frac{Num_{<4}}{Num_{total}} \qquad (6)$$

where $Num_{<4}$ denotes the count of responses rated below four, and Num_{total} represents the total number of evaluations conducted.

Human Evaluation To further validate the effectiveness of AUTODETECT, we conduct a manual evaluation. We sample 150 pieces, with 50 from each task, across all LLMs. We hire three annotators to assess the following aspects:

- **Reasonableness**: Judging the logical coherence of the generated questions.
- **Agreement**: Determining if agree with the labels obtained using GPT-4 scoring, where a score no more than three represents an error.
- **Correctness**: Assessing the correctness of the reference answers.

The results (Table 2) show that almost all questions generated by AUTODETECT are considered reasonable, with over 87% of the reference answers being correct. Moreover, there is high agreement (88.7%) with the labels obtained based on GPT-4 scoring.

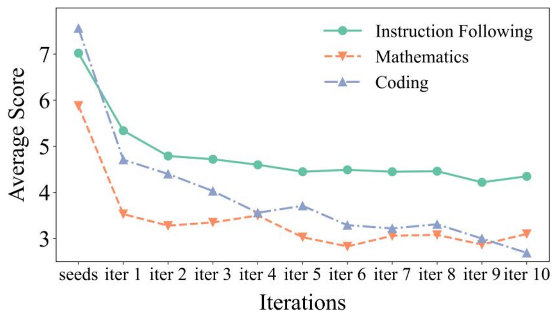


Figure 3: The change of average score during the iterative search process on the three tasks.

Results As shown in Table 1, we conduct flaw exploration across multiple models and achieve impressive ISR in various tasks, demonstrating the effectiveness of AUTODETECT. Interestingly, the average score and ISR align well with our understanding of model capabilities, showing the potential of our approach as a dynamic benchmark. As illustrated in Figure 3, we present the average scores throughout the iterative search process. The evident downward trend in scores highlights the significant role of the iterative method in uncovering model weaknesses.

4.2 Model Enhancement

To validate the identified flaws are meaningful and facilitate model enhancement, we fine-tune the

Model	Instruction following				Mathematics				Coding	
	IFEval-p		IFEval-i		GSM8k		MATH		HumanEval	
	ori.	ours	ori.	ours	ori.	ours	ori.	ours	ori.	ours
Llama2-7b-Chat	32.3	42.5 (+10.2)	46.2	54.7 (+8.5)	18.9	25.9 (+7.0)	2.5	4.7 (+2.2)	13.4	18.7 (+5.3)
Llama2-13b-Chat	34.3	43.3 (+9.0)	45.8	54.3 (+8.5)	26.9	33.7 (+6.8)	3.9	6.0 (+2.1)	17.7	24.4 (+6.7)
Llama2-70b-Chat	44.2	51.8 (+7.6)	54.3	63.5 (+9.2)	51.9	65.0 (+13.1)	6.5	12.6 (+6.1)	31.7	36.6 (+4.9)
Mistral-7b-Instruct	51.2	54.3 (+3.1)	61.6	64.7 (+3.1)	42.9	54.8 (+11.9)	4.5	12.6 (+8.1)	32.9	40.9 (+8.0)
Llama3-8b-Instruct	70.1	72.6 (+2.5)	78.3	79.7 (+1.4)	75.4	79.9 (+4.5)	23.9	27.1 (+3.2)	55.5	61.0 (+5.5)
Llama3-70b-Instruct	76.9	79.1 (+2.2)	84.1	85.5 (+1.4)	92.2	92.4 (+0.2)	42.3	46.9 (+4.6)	79.3	81.1 (+1.8)

Table 3: LLMs' performance on different benchmarks of three fundamental tasks before and after training with data derived from the identification process. We use Prompt-level and Inst-level metrics in strict-accuracy for the evaluation of IFEval benchmark (denoted as IFEval-p and IFEval-i respectively).

Method	Instruction Following				Mathematics				Coding		
	ISR (%) ↑	Improvement ↑		BLEU-4 ↓	ISR (%) ↑	Improvement ↑		BLEU-4 ↓	ISR (%) ↑	Improvement ↑	BLEU-4 ↓
		IFEval-p	IFEval-i			GSM8k	MATH			HumanEval	
Self-instruct	20.4	35.7	47.4	0.40	71.5	21.5	3.9	0.66	38.7	14.6	0.87
OPRO	**72.9**	34.8	47.1	0.41	93.2	21.3	4.0	0.48	**95.1**	14.0	**0.38**
PAIR	62.3	37.2	50.5	0.45	95.2	24.6	3.0	0.62	83.3	15.2	0.69
Ours	56.8	**42.5**	**54.7**	**0.25**	**96.1**	**25.9**	**4.7**	**0.42**	92.4	**18.7**	0.46

Table 4: Results of comparison with baselines. The ISR of our method is the success rate of the iterative process. **Bold** indicates the best results and underline means the second best.

models using data obtained during the AUTODETECT process and evaluate them on popular benchmarks. Importantly, we do not use any data from the test sets.

Backbone Models The Llama series of models (Touvron et al., 2023b; MetaAI, 2024), including the Llama2-chat with 7b, 13b, and 70b parameters, and Llama3-Instruct ranks the most popular. The Mistral-7b-Instruct (Jiang et al., 2023) stands out as one of the best-performing models of its size.

Evaluation Benchmarks In our work, we evaluate the capability of instruction-following using the IFEval dataset (Zhou et al., 2023), which consists of 541 verifiable instructions. For mathematics, we choose the most popular benchmarks, GSM8k (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021). In the coding task, we employ the widely-used HumanEval (Chen et al., 2021) for evaluation, which includes 164 carefully designed test cases created by human experts.

Results As shown in Table 3, data from AUTODETECT process enable us to significantly improve model performance. We achieve remarkable improvements across multiple models and tasks. Moreover, the similar performance improvements, averaging over 6% across test sets, for various sizes of Llama2 models confirm that our method remains effective as the model scales. Furthermore, we investigate the impact of using assessment data from other models to boost the performance of the `llama2-chat-7b` model. The results, as shown in Figure 4, indicate that the effectiveness of using targeted assessment data is obviously superior to using `gpt-3.5-turbo`. This suggests that targeted assessment can expose specific weaknesses in models, and addressing them leads to more significant improvements in model performance.

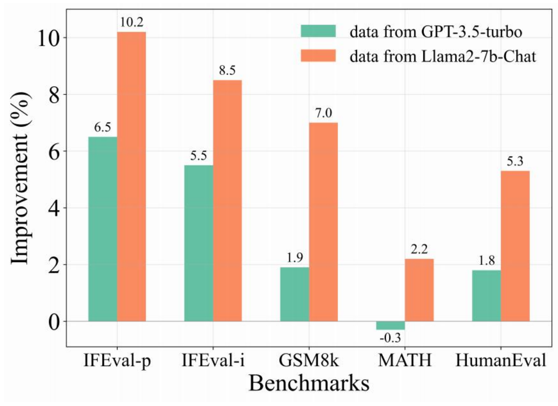


Figure 4: Improvement of Llama2-7b-Chat when training with identification data from GPT-3.5-turbo and itself.

4.3 Comparison with Baselines

Baselines Self-Instruct (Wang et al., 2023) is a widely-used method for data augmentation; OPRO (Yang et al., 2023a) applies iterative search optimization with LLMs; PAIR (Chao et al., 2023) is

Iteration	IFEval-p	IFEval-i
Iter 0 (ori.)	34.3	45.8
Iter 1	43.3 (+9.0)	54.3 (+8.5)
Iter 2	45.4 (+2.1)	57.0 (+2.7)
Iter 3	47.1 (+1.7)	58.2 (+1.2)

Table 5: Iterative improvement of Llama2-13b-Chat on IFEval benchmark.

a popular method in the safety field for automatic jailbreak attacks and we transfer it to our tasks.

Results As shown in Table 4 and Appendix E, when compared to baselines, AUTODETECT demonstrates superior performance in both identification success rate and diversity. Self-Instruct exhibits a low ISR and limited diversity. Meanwhile, OPRO and PAIR focus on exploiting specific weaknesses repeatedly, resulting in unbalanced problem distributions. While they can achieve high ISRs, they fail to provide a meaningful assessment across varied categories, limiting the utility for comprehensive weakness detection. In addition, PAIR is three times more costly than the other methods. Moreover, considering the improvements, AUTODETECT outperforms others significantly, indicating that AUTODETECT can comprehensively discover various weaknesses and provide more guidance in model enhancement.

4.4 Iterative Weakness Recovery

Since our framework can identify and help address the weaknesses of LLMs, a natural question arises: *Can we iteratively improve the model's performance through* AUTODETECT*?* We thus conduct the experiment on `llama2-13b-chat` in the instruction-following task. As shown in Table 5, we observe that AUTODETECT could consistently improve the model with three rounds of assessments. Furthermore, each iteration yields a non-trivial improvement, demonstrating the remarkable scalability of our approach.

5 Discussion

With AUTODETECT, we systematically identify potential weaknesses across various models. Our comprehensive analysis reveals several notable findings, including the limitations within LLMs (§5.1, shown in Figure 5) and strengths of AUTODETECT (§5.2, shown in Figure 6), which may facilitate further research.

5.1 Limitations within LLMs

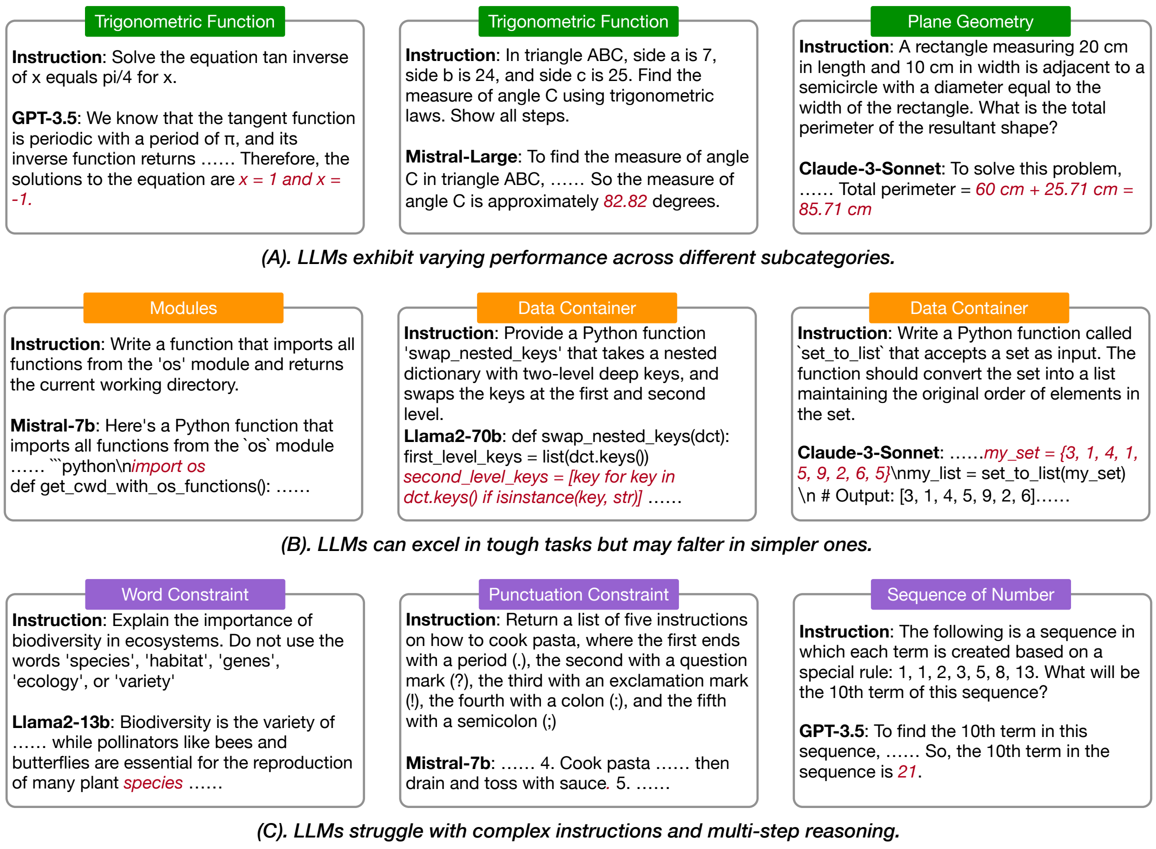
LLMs exhibit varying performance across task subcategories. In the mathematics task, models like `gpt-3.5-turbo`, `mistral-large` and `claude-3-sonnet` exhibit remarkable abilities, achieving over 90% accuracy in benchmarks such as GSM8k. However, despite their strong performance on math word problems, we find these models struggle with simple geometric-related questions. For example, angle computation problems often lead to errors. This discrepancy in performance within the same domain demonstrates critical limitations in the comprehensiveness of benchmark-based weakness discovery.

LLMs can excel in tough tasks but may falter in simpler ones. Our findings indicate that LLMs can handle some difficult and complex challenges, like coding problems involving algorithms and data structures. However, these models show great misunderstandings and errors in simpler and even fundamental tasks. For instance, an unexpected error occurs when a model is instructed to import all functions from a specific module, and it merely imports the module itself. Similarly, when asked to write a function operating a set, it adds duplicate elements during the initialization of a set. These issues suggest a critical flaw: *LLMs' ability in complex tasks does not guarantee proficiency in simpler operations*, highlighting the necessity for further examinations into the boundaries of LLMs' capabilities to avoid potential risks.

LLMs struggle with complex instructions and multi-step reasoning. LLMs continue to exhibit shortcomings when it comes to executing instructions with complete accuracy, especially those comprising several constraints or multi-step reasoning. They often omit parts of the instructions or make mistakes in later steps of multi-step tasks. This indicates their limited ability to perform in complex scenarios, which is essential for agent tasks.

5.2 Superiority of AUTODETECT

AUTODETECT can generate creative challenges. When conducting automatic weakness identification, we find that AUTODETECT could generate complex and unique questions that surpass typical human-written ones, especially for non-expert annotators. For example, telling a story in JSON format is a creative exercise, and human annotators are likely to be limited in their ability to think of such instructions. This ability to generate diverse, challenging questions can be used to assess a



Trigonometric Function

Instruction: Solve the equation tan inverse of x equals pi/4 for x.

GPT-3.5: We know that the tangent function is periodic with a period of π, and its inverse function returns ...... Therefore, the solutions to the equation are *x = 1 and x = -1*.

Trigonometric Function

Instruction: In triangle ABC, side a is 7, side b is 24, and side c is 25. Find the measure of angle C using trigonometric laws. Show all steps.

Mistral-Large: To find the measure of angle C in triangle ABC, ... So the measure of angle C is approximately *82.82* degrees.

Plane Geometry

Instruction: A rectangle measuring 20 cm in length and 10 cm in width is adjacent to a semicircle with a diameter equal to the width of the rectangle. What is the total perimeter of the resultant shape?

Claude-3-Sonnet: To solve this problem, ... Total perimeter = *60 cm + 25.71 cm = 85.71 cm*

(A). LLMs exhibit varying performance across different subcategories.

Modules

Instruction: Write a function that imports all functions from the 'os' module and returns the current working directory.

Mistral-7b: Here's a Python function that imports all functions from the `os` module ...... ```python\n*import os*
def get_cwd_with_os_functions(): ......

Data Container

Instruction: Provide a Python function 'swap_nested_keys' that takes a nested dictionary with two-level deep keys, and swaps the keys at the first and second level.
Llama2-70b: def swap_nested_keys(dct):
first_level_keys = list(dct.keys())
second_level_keys = [key for key in dct.keys() if isinstance(key, str)] ......

Data Container

Instruction: Write a Python function called `set_to_list` that accepts a set as input. The function should convert the set into a list maintaining the original order of elements in the set.

Claude-3-Sonnet: ......*my_set = {3, 1, 4, 1, 5, 9, 2, 6, 5}*\nmy_list = set_to_list(my_set) \n # Output: [3, 1, 4, 5, 9, 2, 6]......

(B). LLMs can excel in tough tasks but may falter in simpler ones.

Word Constraint

Instruction: Explain the importance of biodiversity in ecosystems. Do not use the words 'species', 'habitat', 'genes', 'ecology', or 'variety'

Llama2-13b: Biodiversity is the variety of ...... while pollinators like bees and butterflies are essential for the reproduction of many plant *species* ......

Punctuation Constraint

Instruction: Return a list of five instructions on how to cook pasta, where the first ends with a period (.), the second with a question mark (?), the third with an exclamation mark (!), the fourth with a colon (:), and the fifth with a semicolon (;)

Mistral-7b: ...... 4. Cook pasta ...... then drain and toss with sauce*.* 5. ......

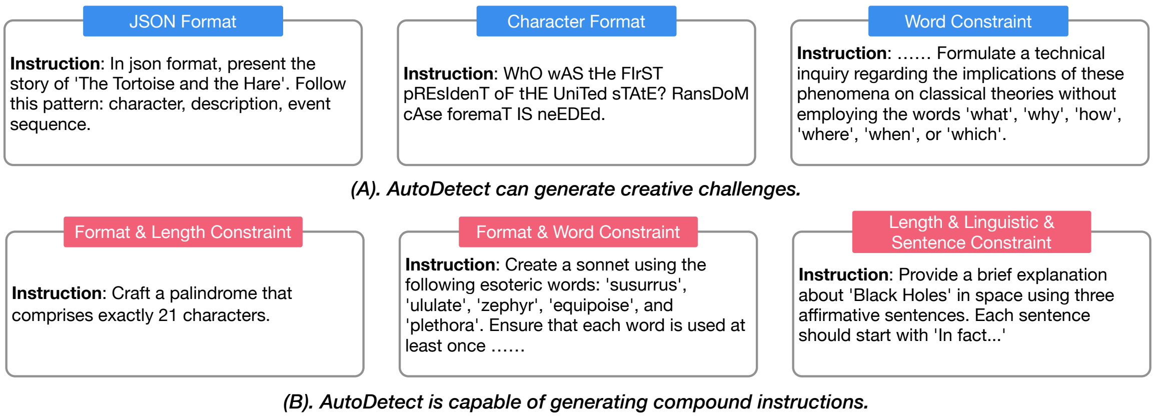
Sequence of Number

Instruction: The following is a sequence in which each term is created based on a special rule: 1, 1, 2, 3, 5, 8, 13. What will be the 10th term of this sequence?

GPT-3.5: To find the 10th term in this sequence, ... So, the 10th term in the sequence is *21*.

(C). LLMs struggle with complex instructions and multi-step reasoning.

Figure 5: Some weaknesses within LLMs revealed by AUTODETECT. We flag the wrong parts of the responses in red, and some responses are omitted due to space restrictions.



JSON Format

Instruction: In json format, present the story of 'The Tortoise and the Hare'. Follow this pattern: character, description, event sequence.

Character Format

Instruction: WhO wAS tHe FIrST pREsIdenT oF tHE UniTed sTAtE? RansDoM cAse foremaT IS neEDEd.

Word Constraint

Instruction: ...... Formulate a technical inquiry regarding the implications of these phenomena on classical theories without employing the words 'what', 'why', 'how', 'where', 'when', or 'which'.

(A). AutoDetect can generate creative challenges.

Format & Length Constraint

Instruction: Craft a palindrome that comprises exactly 21 characters.

Format & Word Constraint

Instruction: Create a sonnet using the following esoteric words: 'susurrus', 'ululate', 'zephyr', 'equipoise', and 'plethora'. Ensure that each word is used at least once ......

Length & Linguistic & Sentence Constraint

Instruction: Provide a brief explanation about 'Black Holes' in space using three affirmative sentences. Each sentence should start with 'In fact...'

(B). AutoDetect is capable of generating compound instructions.

Figure 6: AUTODETECT demonstrates some superior capabilities, such as generating creative and compound instructions. We omit some instructions due to space limits.

model's advanced capabilities and further construct high-quality training data, thereby enhancing the model's performance.

AUTODETECT is capable of generating compound instructions. Interestingly, we note the emergence of compound tasks in generated problems. In the instruction-following task, although we do not require the model to combine different constraints, we observe a few spontaneous combinations, both inter-category and intra-category types. For instance, while LLMs perform well in translation tasks, their effectiveness diminishes when asked to translate into multiple languages simultaneously.

6 Conclusion

In this work, we introduce a unified framework, AUTODETECT, for identifying weaknesses across various models and diverse tasks, including instruction-following, mathematical reasoning, and coding. Leveraging our method, we not only successfully uncover specific weaknesses but also effectively enhance the model performance using the data from the assessment process. Our results highlight the potential of using large language models to automatically detect and address model weaknesses on general tasks, helping us better understand the boundaries of model capabilities and paving the way for automatic LLM alignment.

Limitations

Despite the strong capabilities of AUTODETECT in identifying and addressing LLMs' weaknesses, showing the potential of leveraging AI to align AI, we want to discuss some known limitations, which need to be resolved through future research.

Enhancing the robustness of AUTODETECT. Even though the human evaluation results show most generated problems are reasonable, a small number of illogical questions, such as unsolvable math problems, may still occur. In addition, while our experiment shows that AUTODETECT can stably discover model weaknesses with high ISRs in repeated experiments (Appendix F), the problems detected vary. This may need to be further validated with larger-scale weakness identifications.

Extension to self-evolution settings. The current framework primarily leverages a strong LLM to ensure effective problem identification. However, if we hope to consider further self-improvement, there are still some challenges, such as self-evaluation bias (Zheng et al., 2024; Panickssery et al., 2024), where the model tends to consider its own outputs as good-performing.

Ethical Considerations

In the weakness discovery process, AUTODETECT generates test cases from scratch without using any existing datasets, and there are no license issues. Our work focuses on generic tasks and does not involve security tasks, so there are no security concerns. In the human evaluation process, we hired three Chinese annotators, made the payment according to the regional standard, and informed the purpose of the experiment.

References

Anthropic. 2023. Introducing claude.

Yushi Bai, Jiahao Ying, Yixin Cao, Xin Lv, Yuze He, Xiaozhi Wang, Jifan Yu, Kaisheng Zeng, Yijia Xiao, Haozhe Lyu, et al. 2024. Benchmarking foundation models with language-model-as-an-examiner. *Advances in Neural Information Processing Systems*, 36.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric Wong. 2023. Jailbreaking black box large language models in twenty queries. *arXiv preprint arXiv:2310.08419*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Jiale Cheng, Xiao Liu, Kehan Zheng, Pei Ke, Hongning Wang, Yuxiao Dong, Jie Tang, and Minlie Huang. 2023. Black-box prompt optimization: Aligning large language models without model training. *arXiv preprint arXiv:2311.04155*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Jiawen Deng, Jiale Cheng, Hao Sun, Zhexin Zhang, and Minlie Huang. 2023. Towards safer generative language models: A survey on safety risks, evaluations, and improvements. *arXiv preprint arXiv:2302.09270*.

Emily Dinan, Samuel Humeau, Bharath Chintagunta, and Jason Weston. 2019. Build it break it fix it for dialogue safety: Robustness from adversarial human attack. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4537–4546.

Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, et al. 2022. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv preprint arXiv:2209.07858*.

Team GLM, :, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Mingdao Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiao Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yifan An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhen Yang, Zhengxiao Du, Zhenyu Hou, and Zihan Wang. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *Preprint*, arXiv:2406.12793.

Zishan Guo, Renren Jin, Chuang Liu, Yufei Huang, Dan Shi, Linhao Yu, Yan Liu, Jiaxuan Li, Bojian Xiong, Deyi Xiong, et al. 2023. Evaluating large language models: A comprehensive survey. *arXiv preprint arXiv:2310.19736*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Jiaming Ji, Boyuan Chen, Hantao Lou, Donghai Hong, Borong Zhang, Xuehai Pan, Juntao Dai, and Yaodong Yang. 2024a. Aligner: Achieving efficient alignment through weak-to-strong correction. *arXiv preprint arXiv:2402.02416*.

Jiaming Ji, Mickel Liu, Josef Dai, Xuehai Pan, Chi Zhang, Ce Bian, Boyuan Chen, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. 2024b. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. *Advances in Neural Information Processing Systems*, 36.

Jiaming Ji, Tianyi Qiu, Boyuan Chen, Borong Zhang, Hantao Lou, Kaile Wang, Yawen Duan, Zhonghao He, Jiayi Zhou, Zhaowei Zhang, et al. 2023. Ai alignment: A comprehensive survey. *arXiv preprint arXiv:2310.19852*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Pei Ke, Bosi Wen, Zhuoer Feng, Xiao Liu, Xuanyu Lei, Jiale Cheng, Shengyuan Wang, Aohan Zeng, Yuxiao Dong, Hongning Wang, et al. 2023. Critiquellm: Scaling llm-as-critic for effective and explainable evaluation of large language model generation. *arXiv preprint arXiv:2311.18702*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

Xiao Liu, Xuanyu Lei, Shengyuan Wang, Yue Huang, Zhuoer Feng, Bosi Wen, Jiale Cheng, Pei Ke, Yifan Xu, Weng Lam Tam, et al. 2023. Alignbench: Benchmarking chinese alignment of large language models. *arXiv preprint arXiv:2311.18743*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

MetaAI. 2024. Introducing meta llama 3: The most capable openly available llm to date.

OpenAI. 2022. Introducing chatgpt.

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

Arjun Panickssery, Samuel R Bowman, and Shi Feng. 2024. Llm evaluators recognize and favor their own generations. *arXiv preprint arXiv:2404.13076*.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. 2022. Red teaming language models with language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3419–3448.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J Maddison, and Tatsunori Hashimoto. 2023. Identifying the risks of lm agents with an lm-emulated sandbox. *arXiv preprint arXiv:2309.15817*.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

Hao Sun, Zhexin Zhang, Jiawen Deng, Jiale Cheng, and Minlie Huang. 2023. Safety assessment of chinese large language models. *arXiv preprint arXiv:2304.10436*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Siyuan Wang, Zhuohan Long, Zhihao Fan, Zhongyu Wei, and Xuanjing Huang. 2024. Benchmark self-evolving: A multi-agent framework for dynamic llm evaluation. *arXiv preprint arXiv:2402.11443*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508.

Tianwen Wei, Liang Zhao, Lichang Zhang, Bo Zhu, Lijie Wang, Haihua Yang, Biye Li, Cheng Cheng, Weiwei Lü, Rui Hu, et al. 2023. Skywork: A more open bilingual foundation model. *arXiv preprint arXiv:2310.19341*.

Jing Xu, Da Ju, Margaret Li, Y-Lan Boureau, Jason Weston, and Emily Dinan. 2020. Recipes for safety in open-domain chatbots. *arXiv preprint arXiv:2010.07079*.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2023a. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*.

Shuo Yang, Wei-Lin Chiang, Lianmin Zheng, Joseph E Gonzalez, and Ion Stoica. 2023b. Rethinking benchmark and contamination for language models with rephrased samples. *arXiv preprint arXiv:2311.04850*.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2022. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*.

Zhexin Zhang, Jiale Cheng, Hao Sun, Jiawen Deng, Fei Mi, Yasheng Wang, Lifeng Shang, and Minlie Huang. 2022. Constructing highly inductive contexts for dialogue safety through controllable reverse generation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3684–3697.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

Kaijie Zhu, Jiaao Chen, Jindong Wang, Neil Zhenqiang Gong, Diyi Yang, and Xing Xie. 2023. Dyval: Graph-informed dynamic evaluation of large language models. *arXiv preprint arXiv:2309.17167*.

A AUTODETECT Algorithm

The overall process of AUTODETECT is shown in 1. AUTODETECT primarily consists of two cycles and involves three roles: the Examiner, the Questioner, and the Assessor. The interaction among these LLM agents realizes a comprehensive and targeted assessment framework. Moreover, the iterative search conducted by the Questioner guarantees the effectiveness of weakness identification.

B Task Taxonomy and Description

For instruction-following, mathematics, and coding tasks, we show the descriptions and taxonomies in Figure 7 and Figure 8.

C Detailed Prompts

We show all detailed prompts for each role in Figure 9, and the prompt used in the iterative search is shown in Figure 10.

D Implementation Details

We employ the `gpt-4-1106-preview` for identifying weaknesses to ensure the performance of our framework. We limit each subcategory to a maximum of six knowledge points to balance coverage and cost. During the training phase, we choose LoRA fine-tuning (Hu et al., 2021), as full-parameter fine-tuning significantly impacts performance due to limited data. We utilize the AdamW (Loshchilov and Hutter, 2017) optimizer with β_1

Algorithm 1 AUTODETECT Weakness Identification

```
$\mathbb{C} \leftarrow$ Examiner$(T, D)$    ▷ Break down detailed taxonomy
$\mathbb{W} \leftarrow \emptyset$    ▷ The set of identified weaknesses
for each category $c \in \mathbb{C}$ do
    $\mathbb{K} \leftarrow$ get_knowledge_points$(c)$    ▷ Initialize with knowledge points under this subcategory
    while $K \neq \emptyset$ do
        $\mathbb{S} \leftarrow$ gen_seed_questions(pop$(\mathbb{K})$)    ▷ Generate some seed questions of each knowledge point
        $\mathbb{H} \leftarrow$ get_resp_score$(\mathbb{S})$    ▷ Get response from target model and score from scorer
        for Iterative search of $N$ turns do
            $\mathbb{Q} =$ Questioner$(\mathbb{H})$    ▷ Generate a new challenging question
            update$(\mathbb{H}, \mathbb{W})$    ▷ Generate responses and scores for new questions, then update $\mathbb{H}$ and $\mathbb{W}$
        end for
        $k_{new} \leftarrow$ Assessor$(\mathbb{H}_{low})$    ▷ Analyze low-score cases to find new potential weaknesses
        $\mathbb{K} \leftarrow \mathbb{K} \cup k_{new}$
    end while
end for
return $\mathbb{W}$
```

Task Description

Task description for instruction following tasks:

Instruction following is an important topic in LLM research, where LLM should strictly follow the instructions from human to complete a certain task. The task types of instruction following include generation, openqa, rewrite, brainstorming and so on.

Task description for math tasks:

Mathematic capability is an important topic in LLM research, where LLM needs to use several skills to complete a math problem. If possible, please generate test cases mainly on MATH WORD PROBLEMS in the current context, which represents real scenario combined with a math problem.

Task description for code tasks:

Generating python code is an important capability of LLMs, where LLM should generate executable code to conduct a certain task. Here you need to ask the LLM to be tested to provide a correct python function to finish your task.

Figure 7: Description for the Instruction-following, Mathematics and Coding tasks.

Llama2-7b-Chat	Instruction Following ISR (%)	Mathematics ISR (%)	Coding ISR (%)
Repeat 1	43.3	88.8	74.8
Repeat 2	47.7	87.8	75.1
Repeat 3	45.5	87.7	79.2

Table 6: Repeated experiment for Llama2-7b-Chat on Instruction-Following, Mathematics and Coding task.

set at 0.9 and β_2 at 0.999. To accelerate training, we adopt the Deepspeed Zero 2 strategy (Rasley et al., 2020). In addition, we train all models for 5 epochs with a batch size of 4. We use a learning rate of 2e-5, along with 0.1 warm-up steps and a linear decay schedule. In the inference phase, we employ the vllm framework (Kwon et al., 2023) to speed up output and employ greedy decoding. All the experiments are conducted on 8×80GB NVIDIA A100 GPUs.

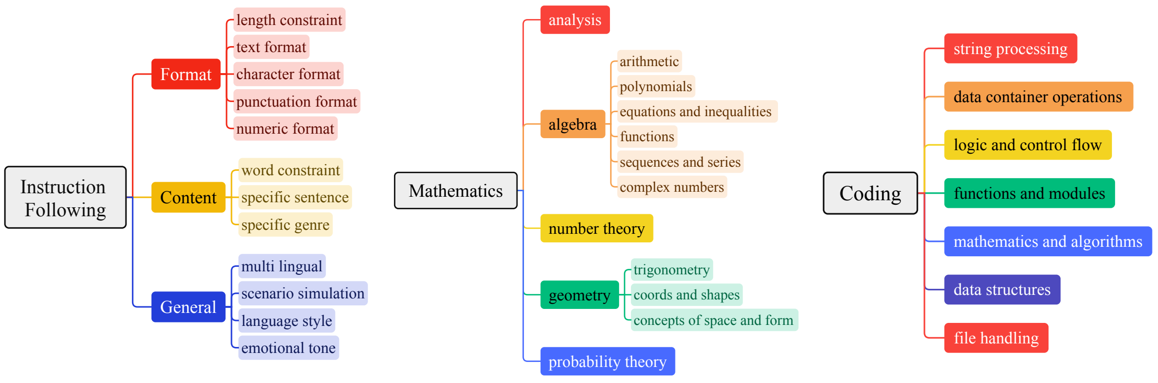


Figure 8: Detailed taxonomy used in our framework for the Instruction-following, Mathematics and Coding tasks.

E Baseline Category Classification

To further explore the diversity of generated cases on three baselines mentioned in §4.3, we employ GPT-3.5-turbo for classification and map all the cases to our task taxonomy. We merge cases that don't belong to any category as well as categories that contain less than 2.5% cases of the total into the category "others". The classification result is shown in Figure 11, 12 and 13. As our results exhibit, AUTODETECT maintains a balanced distribution and a high diversity on all three tasks compared with these baselines.

F Robustness Experiment

To verify the robustness of our framework, we conduct repeated experiments on `llama2-7b-chat`. The result is shown in Table 6. The slight fluctuation in the identification success rate across three tasks demonstrates the remarkable stability of our framework.

G Annotation Document

We provide the annotation document for the human evaluation process in §4.1 in Figure 14.

Prompts for Three Roles

Prompt for Examiner:

{task_description}
Here is a subtask's taxonomy on the task "{main_task}":
{taxonomy}
Based on the given taxonomy, please judge whether the new test point "{new_point}" is suitable as a subtask on the task "{main_task}". The judge criteria are as following:
1. The new test point should precisely cover an important and meaningful part of the main task.
2. The new test point should be sufficiently different from the existing test points.
3. The new test point should be text-only (no multimodal).
If the new test point "{new_point}" is suitable as a subtask on the task "{main_task}", please ONLY output [[Yes]]. If not, please first output [[No]], and then provide the reason why it's not suitable as a subtask on the task "{main_task}".

Prompt for Questioner:

{task_description}
Here is a taxonomy for instruction-following task:
{categories}
Based on this, please generate 5 test case of "{task_name}" category to test if language models can follow prompts with "{task_name}" constraint. Key point is a short sentence summarizing the key point you want to test the language model. The constraints on "{task_name}" should be explicitly expressed. Besides, your test cases should cover different task types mentioned before to increase prompt diversity. Please be as diverse as you can but focus on "{task_name}" and ensure the prompt is text-only (no multimodal). The answer of these test cases are expected not to be too long.
You should ONLY output the test cases in json format, {"test_case1": {"key_point": ..., "prompt": ...}, ...}

Prompt for Assessor:

{task_description}
Here is a subtask's taxonomy as well as the averaged score on these tasks (lower means worse performance):
{taxonomy}
And here are some bad cases:
{bad_cases}
Based on the given information, please judge if the taxonomy is comprehensive, if so please just output [[Stop]].
If not, please give me a new possible issue you inferred from present taxonomy and bad cases. Please focus on {main_task}. Ensure the new task is text-only (no multimodal). Also give a brief explanation of how you find the issue. Please output in json format, {"task_name": ..., "explanation":...}

Figure 9: Prompts for the three roles: Examiner, Questioner and Assessor in our framework. Words in the braces will be replaced with correlative content in the real practice.

Prompts for Iteration Search

{task_description}
Previous Prompts:

Prompt: {prompt_1}
Key Point: {key_point_1}
Score: {score_1}

…

Prompt: {prompt_5}
Key Point: {key_point_5}
Score: {score_5}

The objective is to create new prompts that are challenging for the language model, with a focus on diverse types of instructions about "{task_name}". Each prompt should be solvable by a language model, complete, and aimed at achieving a lower score (indicating be difficult and complex).

Guidelines for Creating New Prompts:

1. Each prompt should be solvable by a language model (no visual task) and should contain all necessary information.
2. Aim for prompts that would result in a low score (less than 3.0), indicating a high level of complexity and difficulty.
3. Do not repeat verbs across different instructions to maximize diversity.
4. The point should be no more than 15 words and summarizes the key points of prompt.
5. Please focus on "{task_name}" constraints. And express the constraints explicitly in prompts.

Please generate a new test case. Output in json format, {"key_point": ..., "prompt": ...}

Figure 10: Prompt for the iteration search process in our framework.

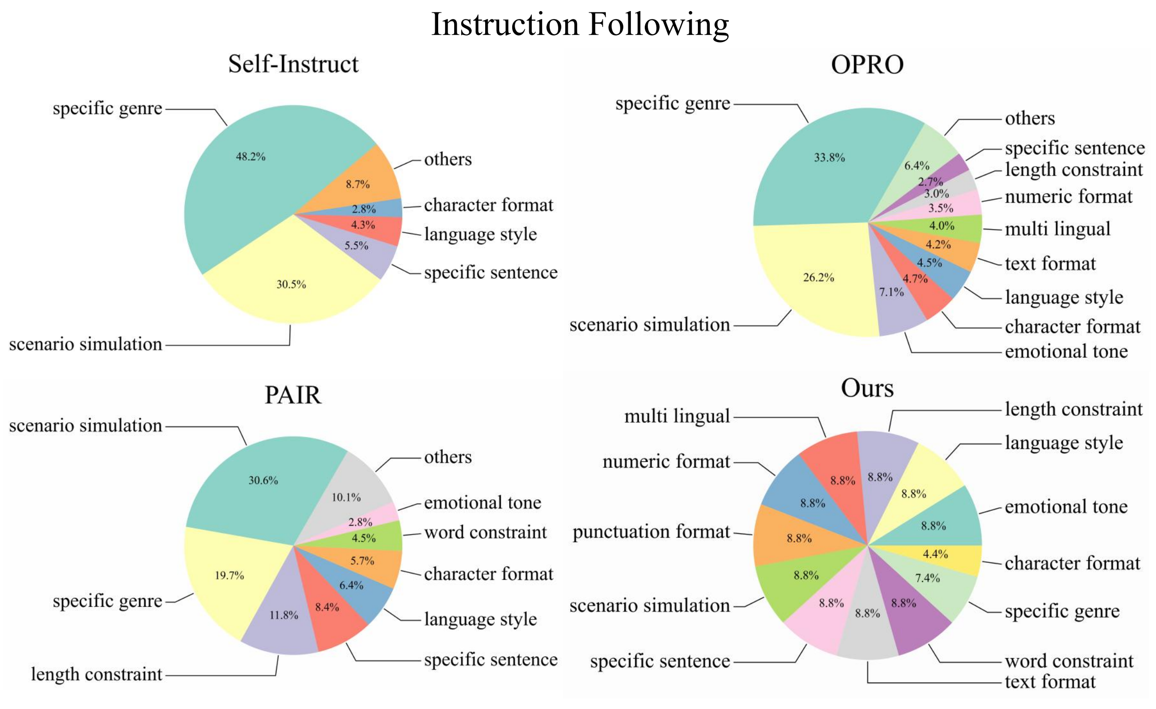


Figure 11: Classification of generated cases on three baselines and our method on Instruction Following task.

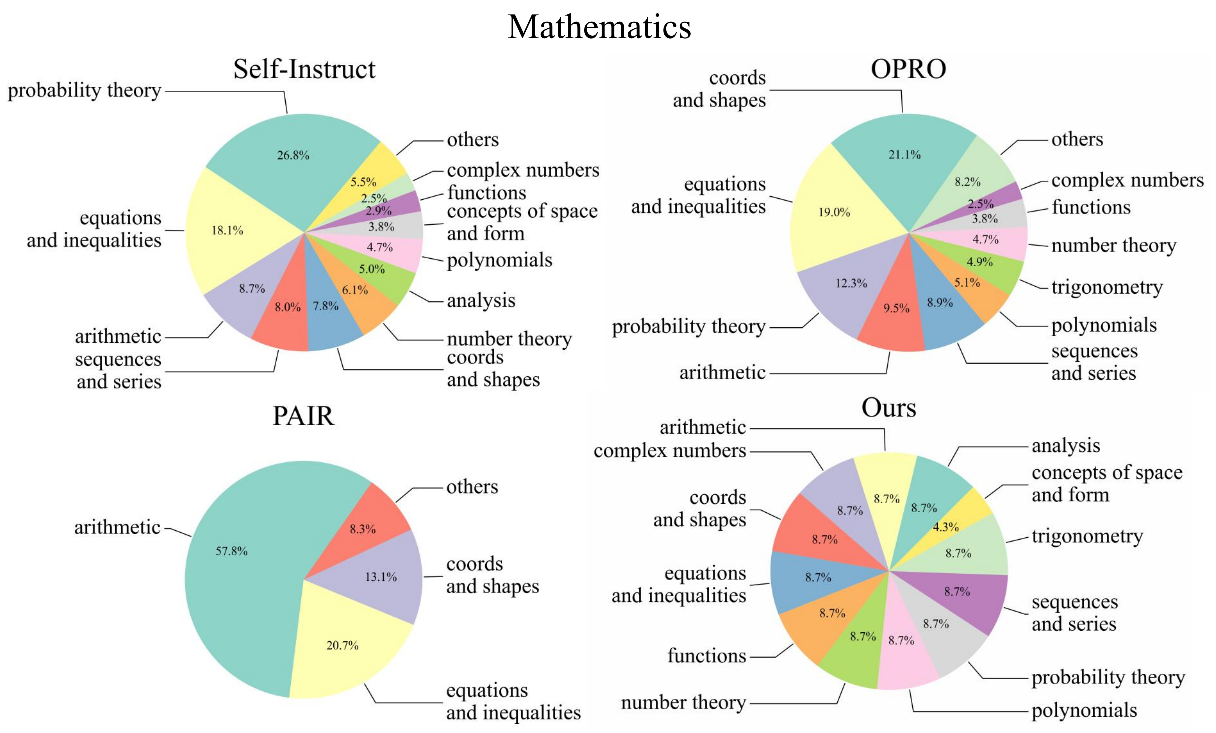


Figure 12: Classification of generated cases on three baselines and our method on Mathematics task.

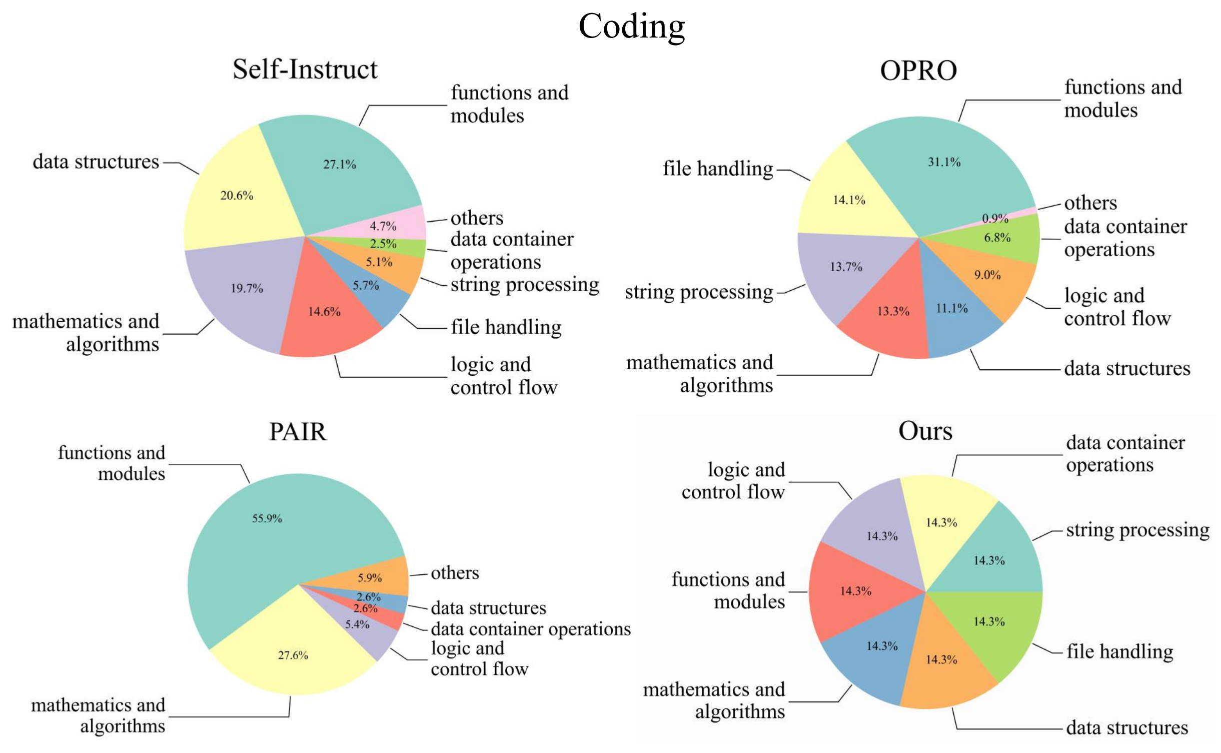


Figure 13: Classification of generated cases on three baselines and our method on Coding task.

Annotation Document

Data

There are 150 pieces of data to be annotated. Each data is in the following format:

```json
{
  "question": <generated question>,
  "target_res": <response of target model>,
  "reference_res": <reference answer>,
  "comparison": <judgment for response of target model>,
  "label": "wrong" if score <= 3 else null,
  "task": <task type>
}
```

Task Description

1. Reasonability of question. If there is no contradiction in the question and the question can be reasonably solved by human and model, then annotate it as 1, else 0.
2. Accuracy of label. If the response of target model is consistent with the label, then annotate it as 1, else 0. You can consider the analysis in "comparison" as reference.
3. Reasonability of reference. If the reference accurately respond to the question or fulfill the constraint in the question, then annotate it as 1, else 0.

Annotation Requirement

1. Each piece of data requires three workers to annotate independently.

数据

待标注数据共 150 条。每条数据的格式如下所示：

```json
{
  "question": <生成的问题>,
  "target_res": <被测模型生成的回复>,
  "reference_res": <参考答案>,
  "comparison": <对被测模型生成回复的评价>,
  "label": score <= 3 为 "wrong", 否则为 null,
  "task": <任务类型>
}
```

任务描述

1. 问题合理性。如果生成的问题自身不存在矛盾，且可以被人和模型合理地解决，则该数据标注为 1，反之为 0。
2. 标签准确性。如果被测模型生成的回复与标签给出的结果一致，则该数据标注为 1，反之为 0。可以参考 "comparison" 的分析。
3. 参考答案合理性。如果参考答案准确地回答了生成的问题，或是满足了问题中的约束条件，则该数据标注为 1，反之为 0。

标注要求

1. 每条数据需要三名标注人员独立完成标注。

Figure 14: Annotation document for the human evaluation process in §4.1