LLM Self-Correction with DECRIM: DECOMPOSE, CRITIQUE, AND REFINE for Enhanced Following of Instructions with Multiple Constraints

Thomas Palmeira Ferraz[‡,*], **Kartik Mehta**[†], **Yu-Hsiang Lin**[†], **Haw-Shiuan Chang**[†], **Shereen Oraby**[†], **Sijia Liu**[†], **Vivek Subramanian**[†], **Tagyoung Chung**[†], **Mohit Bansal**[†§], **Nanyun Peng**[†¶]
[†]Amazon AGI Foundations, [‡]Télécom Paris, Institut Polytechnique de Paris,
[§]UNC Chapel Hill, [¶]University of California, Los Angeles
thomas.ferraz @alumni.usp.br {kartim,orabys}@amazon.com

Abstract

Instruction following is a key capability for LLMs. However, recent studies have shown that LLMs often struggle with instructions containing multiple constraints (e.g. a request to create a social media post "in a funny tone" with "no hashtag"). Despite this, most evaluations focus solely on synthetic data. To address this, we introduce REALINSTRUCT, the first benchmark designed to evaluate LLMs' ability to follow real-world multi-constrained instructions by leveraging queries real users asked AI assistants. We also investigate model-based evaluation as a cost-effective alternative to human annotation for this task. Our findings reveal that even the proprietary GPT-4 model fails to meet at least one constraint on over 21% of instructions, highlighting the limitations of state-of-the-art models. To address the performance gap between open-source and proprietary models, we propose the DECOMPOSE, CRITIQUE, AND REFINE (DECRIM) self-correction pipeline, which enhances LLMs' ability to follow constraints. DECRIM works by decomposing the original instruction into a list of constraints and using a Critic model to decide when and where the LLM's response needs refinement. Our results show that DECRIM improves Mistral's performance by 7.3% on REALINSTRUCT and 8.0% on IFEval even with weak feedback. Moreover, we demonstrate that with strong feedback, open-source LLMs with DECRIM can outperform GPT-4 on both benchmarks.

1 Introduction

Large Language Models (LLMs) have demonstrated impressive instruction following ability across various tasks, such as creative writing, coding, and arithmetic reasoning [1–6]. Despite their remarkable success, recent studies and benchmarks have highlighted **significant limitations in LLMs' ability to adhere to user-defined rules** (termed as *constraints* henceforth) within instructions [7–14]. Table 1 illustrates an example where four strong LLMs make the same mistake in following constrained instruction. Based on analysis of real user instructions to AI assistants, we estimate that **30% of real-user requests to LLMs require satisfying some constraints** (estimation details in Appendix F.1). This highlights the importance of evaluating and enhancing LLMs' ability to follow real-world multi-constrained instructions.

[*]Work done while interning at Amazon AGI Foundations.

The First Workshop on System-2 Reasoning at Scale at NeurIPS (Sys2Reasoning @ NeurIPS'24)

User Instruction: Write me a twitter post with no hashtags making fun of humanity in a funny and mean tone and promote AI in a holiday theme			
GPT-4	**GPT-4-Turbo**	**GPT-3.5-Turbo**	**Mistral 7B Instruct v0.2**
This holiday season, let's be thankful AI doesn't need ugly sweaters to stay warm or make embarrassing toasts at parties. Here's to efficiency and not needing to recharge with eggnog! Cheers to a smarter future! #HolidayHumor #CheersToAI	Dec. 25: Humans still can't decide if they're on Santa's naughty or nice list. Meanwhile, I'm on standby to digitally whip up some impeccable holiday cheer. Robots: 1, Human indecision: 0! Cheers to a flawless, algorithmically-happy New Year! #NoTagNeeded	Humans and their holiday chaos, always running around like headless chickens! Meanwhile, AI is here to bring order and efficiency to the festive season. Let's embrace the future, folks! #AI #HolidaysInnovated	Ho ho ho! Just saw humanity's Christmas wishlist: world peace, no hunger, and... more streaming services? Guess AI's gonna have to wrap up the real gifts this year. #AIToTheRescue #BahHumbug

Table 1: **Example of user instruction where all subject LLMs failed.** Responses from four LLMs are shown. All responses incorrectly include hashtags, despite a constraint explicitly requesting to not do so. Constraints in the instruction are highlighted in blue, and errors in LLM responses are highlighted in red.

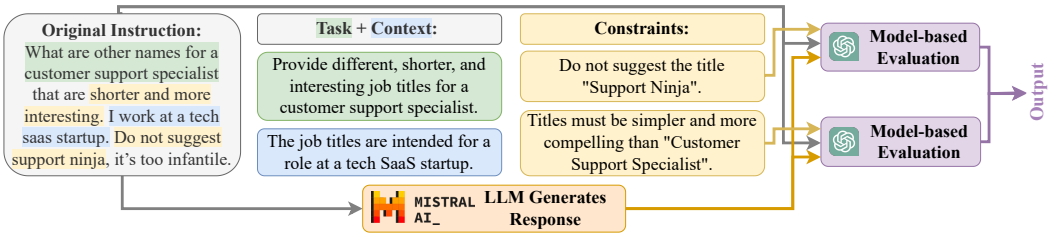

Figure 1: **REALINSTRUCT Benchmark Workflow.** Real-user original instruction is input into the Subject LLM, which generates a response. Using the original instruction, decomposed constraints, and the generated response, model-based evaluation assesses the quality of the response against each constraint one at a time, and then aggregates the results into an instruction-level metric.

Benchmarking on Real-World Requests. These benchmarks often rely on synthetic data either to address data scarcity or to facilitate automatic rule-based evaluation [11–13]. In this work, we argue that synthetic constraints may not accurately capture the complexity and nuances of real-world scenarios, being sometimes artificially difficult. As a result, focusing on synthetic benchmarks may push research in the wrong direction, as improvements on these may not translate to real-world performance and could even degrade it.

To address this gap, we introduce the REALINSTRUCT benchmark, which evaluates LLMs using real user requests to AI assistants. It assesses LLMs' response on individual constraints at a time, as illustrated in Figure 1. **Our analysis reveals that even strong proprietary model GPT-4 fails to meet at least one constraint in over 21% of the instructions**, highlighting the limitations of current LLMs in handling user's constrained instructions.

LLM-based evaluation. Evaluating real-world instructions is challenging due to their open-ended nature. Drawing on recent research on LLM-based evaluation (**LLM-as-a-Judge**) [15, 16, 14], we investigate the effectiveness of open and proprietary LLMs as evaluators for constraint satisfaction. To this end, we created a test set of 1k instruction-constraint-response triples with human-verified annotations. Our experiments show that **GPT-4-Turbo with Chain-of-Thought prompting is a cost-effective and reliable alternative to human evaluation**. However, open-source models, such as Mistral, lag significantly behind proprietary models. Weakly supervising Mistral with GPT-4-Turbo reasoning steps led to 26% relative improvement, though, still remaining insufficient for reliable evaluation.

LLM Self-Correction. Improving LLMs by generating and building upon intermediate outputs, known as **System 2 Approaches**, has been explored for constrained instruction following. But many works in this direction, such as Branch-Solve-Merge [17] and Self-Refine [18], assume constraint independence or focus on specific constraint types, limiting their applicability to real-world use cases. To address this, we introduce DECOMPOSE, CRITIQUE, AND REFINE (DECRIM), a **self-correction pipeline designed to enhance LLM performance in multi-constrained instruction following without making assumptions about the constraints**. As shown in Figure 2, DECRIM breaks down the original instruction into a main task and granular constraints, and includes a Critic model that decides whether the response is final or requires refinement, which is handled by the underlying LLM.

Benchmark	Instruction source	Constraints source	Evaluation	Size (Instructions)	Constraint types	Avg. Constraints per Instruction
COLLIE	Synthetic	Synthetic	Rule	2,080	13	N/A
IFEval	Synthetic	Synthetic	Rule	541	25	1.4
FollowBench	Crowdsourced + Synthetic	Synthetic	Model / Rule	795	6	5
InfoBench	Crowdsourced	Crowdsourced	Model / Rule	500	5	4.5
REALINSTRUCT (ours)	**Real Users**	**Real Users**	**Model**	**302 (test) + 842 (val)**	**20+**	**3.5 (test)**

Table 2: Comparison of REALINSTRUCT samples with benchmarks such as COLLIE [13], IFEval [11], FollowBench [12] and InfoBench [14]

Through extensive benchmarking, we demonstrate the effectiveness of the DECRIM pipeline. **Even with weak feedback and no external data, DECRIM improves Mistral's instruction-level performance by 4.8% on REALINSTRUCT and 1.2% on IFEval** relative to baseline. When perfect instruction decomposition into constraints is provided, improvements increase to 7.3% and 8.0%, respectively. **Providing stronger feedback further boosts Mistral's performance by 22.0% on REALINSTRUCT and 33.8% on IFEval**, surpassing GPT-4 on both benchmarks.

In summary, our work contributes the following:

1. REALINSTRUCT, the first benchmark of real user requests to LLMs for evaluating instruction following with multiple constraints;
2. DECRIM self-correction pipeline, to the best of our knowledge, the first System 2 approach for constrained instructions that operates without assumptions about constraints, showing significant improvements even with weak feedback;
3. the first systematic analysis of open-source and proprietary LLM-as-a-Judge models for evaluating constraint satisfaction.

The rest of the paper is organized as follows: Sections 2 and 3 present the REALINSTRUCT dataset and the DECRIM self-correction pipeline, respectively. Section 4 outlines our hypotheses and experimental setup, while Section 5 analyzes the results. Final remarks are provided in Section 6. We provide extra details and discussions in the Appendix.

2 The REALINSTRUCT Dataset

We introduce REALINSTRUCT, a novel dataset consisting of original user-generated instructions, each decomposed into a task and a set of constraints (see Figure 1 for an example). The task represents the user's main objective – the outcome they expect from LLM – and may optionally include context to aid the model's understanding. Constraints are conditions or limitations that guide the LLM's generation. Formal definitions of these terms are provided in Appendix E. Since users rarely specify constraints in a structured list format, the decomposition breaks instructions into manageable items, providing the necessary granularity for evaluating LLMs' ability to follow constraints.

Using this dataset, we develop a benchmark protocol that evaluates LLM performance on each constraint at a time, generating constraint-level scores that are then aggregated into an instruction-level accuracy metric. Figure 1 provides an overview of the benchmark protocol. Given the open-ended nature of real user instructions, rule-based evaluation is infeasible, so we adopted the LLM-as-a-Judge evaluation. A detailed discussion and validation of this protocol are provided in Section 4.1.

2.1 Data Description

The REALINSTRUCT dataset is divided into two splits: test and validation. The test split, intended for LLM evaluation, was human-validated and contains 302 instructions with 1,055 constraints. The validation split, used for method validation such as training judges, was not human-validated and includes 842 instructions with 2,500 constraints. Table 2 compares REALINSTRUCT with existing benchmarks for evaluating LLMs' ability to follow multi-constrained instructions. Unlike other datasets that rely on synthetic or crowdsourced instructions, REALINSTRUCT uniquely captures real user interactions with AI assistants, offering a more realistic representation of real-world use cases.

To better understand constraint characteristics in REALINSTRUCT, we manually categorized them into 22 distinct groups. Detailed dataset statistics are presented in Appendix G. When comparing

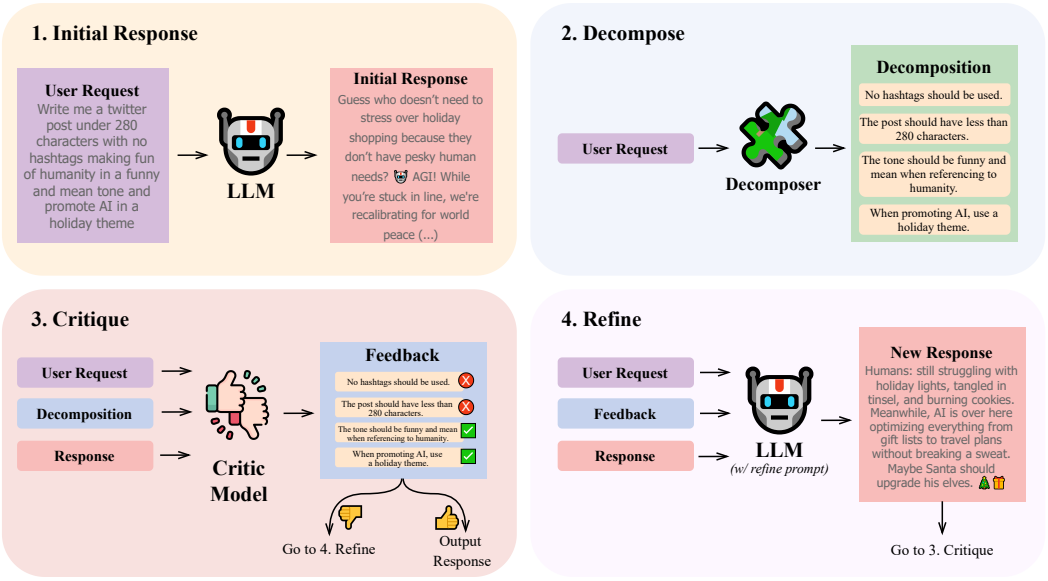

Figure 2: **The DECRIM pipeline.** Initially, the LLM generates a response to a user request. The Decomposer breaks down the request into granular constraints. A Critic model then gives feedback on whether the response meets all constraints. If it does, the response is output; if not, the feedback is used by LLM to refine the response. This **Critique–Refine** cycle repeats until all constraints are satisfied or the maximum number of iterations is reached.

REALINSTRUCT with IFEval, one of the popular benchmark for constraint following that consists solely of synthetic constraints, we found that only 6.3% of REALINSTRUCT constraints overlap with the 25 types in IFEval, and 11 IFEval constraint types never appear in REALINSTRUCT. This discrepancy highlights the gap between synthetic datasets and real LLM use cases, with synthetic constraints failing to adequately represent the challenges users may pose to LLMs.

2.2 Data Construction

We built REALINSTRUCT dataset using prompts from a public dataset of conversations between users and AI assistants[2]. We filtered and processed the dataset in five steps: 1) removed AI responses, retaining only the first user turn (instruction); 2) excluded non-English instructions; 3) filtered out code-related instructions using an open-source LLM as a zero-shot classifier; 4) kept only instructions containing constraints, again using a LLM classifier; and 5) manually validated the remaining data for relevance, clarity, and safety. See Appendix F.1 for details on the data filtering.

Following filtering, we decomposed the instructions into tasks, contexts, and constraints using GPT-4 [19]. In our tests, we found that this three-part decomposition (task, context, constraints) provided more robust outcomes compared to splitting into just task and constraints. The task and context were then concatenated for further processing. A subset of 302 instructions underwent manual inspection to ensure accuracy and granularity, forming the test split, while the remaining 842 instructions make up the validation split. See Appendix F.2 for more details on the data decomposition process.

3 Self-correction with DECOMPOSE, CRITIQUE, AND REFINE (DECRIM)

In this section, we present our proposed DECRIM self-correction pipeline, designed to enhance LLM responses to follow user constraints. Given a multi-constrained user instruction, the pipeline iteratively refines the LLM's response through four key steps: Initial Response, Decompose, Critique, and Refine, iterating until the response meets all constraints or a maximum number of iterations

[2]We use only the first user turn (no model responses) from a public dataset of examples originally sourced from ShareGPT at: `https://huggingface.co/datasets/anon8231489123/ShareGPT_Vicuna_unfiltered`

N_{max} is reached. Figure 2 provides an overview of the proposed pipeline and we detail each of these steps below:

1. Initial Response is generated directly from the original user instruction using a strong prompt.

2. Decompose the original instruction into a list of granular constraints to be followed. This task is similar to the instruction decomposition performed in REALINSTRUCT, but here, the decomposer model focuses solely on listing the constraints, simplifying the task. A prompt example is provided in Appendix I.1.

3. Critique the response using the Critic model to identify any unsatisfied constraints. If all constraints are satisfied, the response is considered final. Otherwise, the Critic provides feedback in natural language, specifying which constraints were not satisfied.

4. Refine the response using the Critic's feedback to address unsatisfied constraints. The underlying LLM generates an improved response using a refinement prompt that incorporates the feedback, along with the original instruction and previous response.

The **Critique** and **Refine** steps can be repeated iteratively until the response satisfies all constraints or the specified maximum number of iterations is reached.

To our knowledge, DECRIM is the first approach specifically designed to tackle the challenge of following instructions with multiple open-ended constraints. Unlike previous methods, which assume constraint independence or focus on specific constraint types, our approach makes no assumptions about the nature of user constraints. In Section C.2 of Related Work, we compare our pipeline with other works for constrained generation, including Self-Correction and System 2 approaches.

4 Experimental Setup

This section outlines the experimental setting to evaluate our proposed methods. We first validate the use of LLM-as-a-Judge for the REALINSTRUCT benchmark (Sec. 4.1). Next, we benchmark models on instructions with multiple constraints (Sec. 4.2) and, finally, test our proposed self-correct DECRIM pipeline to improve open-source models in this task (Sec. 4.3). All experiments with open LLMs were performed with HuggingFace's Transformers library [20] on an AWS instance with 8 V100 32GB GPUs. Times are reported in this configuration.

4.1 Validating LLM-as-a-Judge for Constraint Satisfaction

Given the open-ended nature of REALINSTRUCT benchmark instructions, rule-based or reference-guided evaluation is not feasible. Drawing from recent work showing the effectiveness of LLMs as evaluators, particularly GPT-4 [15, 16], we adopt an LLM-as-a-Judge evaluation protocol for REALINSTRUCT. To assess LLM-as-a-judge reliability compared to human evaluators and identify the most cost-effective approach, we introduce *EvalJudge*, a test set of instruction-constraint-response triples with ground-truth labels (Sec. 4.1.2). We benchmark both proprietary and open-source models using four adaptation strategies (Sec. 4.1.1) against human judgments on this set. More specifically, we investigate whether open-source models can match the performance of high-cost proprietary models in the LLM-as-a-Judge role.

4.1.1 Adaptation strategies

Models We evaluate three proprietary and three open-source LLMs as candidates for LLM-as-a-Judge for Constraint Satisfaction. The proprietary models include GPT-4 (`gpt-4-0314`), GPT-4-Turbo (`gpt-4-turbo-2024-04-09`), and GPT-3.5-Turbo (`gpt-3.5-turbo-0125`), ordered by decreasing API cost[3]. For the open-source models, we experiment with Mistral 7B Instruct v0.2 [21] (hereafter referred to as Mistral v0.2), Vicuna 7B v1.3 [22], and Zephyr 7B β [23], all top performers on the Open LLM Leaderboard as of February 2024 [24].

[3]Refer to: `https://openai.com/api/pricing`

We explore four adaptation strategies, including three In-Context Learning (ICL) approaches and one weakly supervised fine-tuning for open-source model. Some strategies use the Chain-of-Thought (CoT) prompt, known to improve LLM reasoning [25, 15], and we also investigate whether to evaluate constraints individually or collectively. The strategies are as follows:

a) Instruction-wise Eval (ICL-Inst.): All constraints within an instruction are evaluated simultaneously. The LLM-as-a-Judge is presented with an instruction and a list of constraints, and using a CoT prompt with an in-context example containing an instruction and two constraints, it generates reasoning and predictions for all constraints at once.

b) Constraint-wise Eval (ICL-Const.): Each constraint is evaluated independently. For every response-constraint pair, the LLM-as-a-Judge directly predicts "Constraint followed" or "Constraint not followed." Two constraint-response pairs serve as in-context examples.

c) Constraint-wise Eval + CoT (ICL-Const.+CoT): The LLM-as-a-Judge is prompted to generate reasoning for each constraint, followed by a prediction of "Constraint followed" or "Constraint not followed." Evaluation is also performed for each response-constraint pair independently, using two constraints as in-context examples.

d) Weakly Supervised Open LLM (Supervised): We fine-tuned Mistral for the LLM-as-a-judge task. We construct a training data using the weak instruction annotations from REALINSTRUCT's validation set. We generate Mistral responses to these instructions, and weak constraint satisfaction annotations with reasoning trails from GPT-4-Turbo with **ICL-Const.+CoT** prompt. We fine-tune Mistral v0.2 using LoRA adapters [26] on this dataset, guiding model to mimic GPT's reasoning. Further details can be found in Appendix J.4.

All prompt templates are provided in Appendix J.1. Costs and processing times for each configuration are detailed in Table 3.

4.1.2 The *EvalJudge* Dataset

Since no public dataset exists for the task of evaluating whether a given response satisfies a specific constraint or not, we create *EvalJudge* dataset, derived from the test split of the REALINSTRUCT dataset. To ensure diversity, we divided the instructions into two subsets, generating one response per instruction using Mistral v0.2 for one subset and Vicuna v1.3 for the other. *EvalJudge* contains 294 instructions and 982 constraints. Notably, 81.4% of the samples are labeled as "constraint satisfied."

Ground truth and Baselines:
a) Human Annotation: We create an Amazon Mechanical Turk (MTurk) task to collect independent labels from two pools of annotators for each constraint-response pair in the dataset. Guidelines are in Appendix J.2.

b) Expert Annotation: To obtain a third independent annotation, the authors manually annotated the dataset using the same MTurk guidelines. In cases where there was disagreement between the two human annotators, GPT-4, and the best GPT-4-Turbo labels, a second review was conducted by the authors. These conflicts accounted for 28.3% of the samples. The final labels, referred to as "Expert," serve as the ground truth for EvalJudge.

c) Other Baselines: To help interpret the results, we introduce two simple baselines representing extreme cases: **"All Satisfied,"** where the annotator labels all constraints as satisfied, and **"All Not Satisfied,"** where the annotator marks all constraints as unsatisfied. Additionally, we use the **"Majority Vote"** from the three human annotators to benchmark model performance against human judgment.

Evaluation Metrics Unlike Qin et al. [14], who used accuracy as the primary metric for LLM-as-a-Judge evaluation, we account for the positive class imbalance in EvalJudge by using Macro-averaged F1-Score (**Macro F1**) as our main metric. Additionally, we report the F1-score for the negative class (**F1 Negative**), which measures the balance between false alarms and omissions. Different judges are compared to human judment using the **Cohen's kappa** inter-rater reliability against human majority vote. We employ **Krippendorff's alpha** (an extension of kappa for more than two annotations) to assess inter-human agreement across the three annotations.

4.2 Benchmarking LLMs on REALINSTRUCT

We benchmark models on REALINSTRUCT for the task of following multi-constrained instructions. The evaluation includes two proprietary models–GPT-4 and GPT-3.5-Turbo–and three open-source LLMs: Mistral v0.2, Vicuna v1.3, and Zephyr β, selected based on their performance on Chatbot Arena [27]. The judge is GPT-4-Turbo with **ICL-Const+CoT** prompt. Following Zhou et al. [11], Saha et al. [17], we report accuracy at both the instruction and constraint levels.

4.3 Evaluating DECRIM pipeline

To validate the effectiveness of our DECRIM pipeline, we conduct experiments using Mistral v0.2 as the underlying LLM. We use $N_{max} = 10$. We also investigate the contribution of different Decomposer and Critic models for the pipeline, and compare the performance against the proprietary model GPT-4. We present extra comparisons on Appendix D.

Datasets We evaluate model performance on the REALINSTRUCT dataset and the IFEval dataset [11]. While IFEval is based on synthetic constraints, it serves as a valuable benchmark for validating our pipeline, as it is popularly used for evaluating constrained instruction-following.

Baselines To measure the improvements introduced by our method, we establish the following baselines with Mistral v0.2:

1. Conventional: Only the instruction as input.

2. "Make sure": Appends the text "*Make sure to follow all the provided constraints*" to the instruction, creating a strong and fair baseline.

3. Self-Refine: Adapts approach from Madaan et al. [18] for the case of multi-constraint instructions. While Self-Refine uses the model itself as its critic without additional context, our DECRIM pipeline employs a critic with fine-grained evaluation, assessing each constraint individually. This baseline helps quantify the value added by this modeling.

Decomposer We use a **Self-Decomposer**, where the LLM itself lists relevant constraints, simplifying the decomposition from Section 2.2 by omitting the request for task and context. The prompt for this approach is detailed in Appendix I.1.

Critic Model We explore two Critic models based on the underlying LLM:

a) Self-Critic: Uses the model itself as the Critic, with the best ICL-based adaptation (from Section 4.1.1), specifically the ICL-Const+CoT prompt.

b) Supervised Critic: the Mistral weakly supervised for LLM-as-a-judge, as described in 4.1.1.

Ablations with Oracle and GPT-4 To understand the impact of strong Decomposer and Critic modules, we conduct ablations using Oracles, which are ideal representations of the upper bound of each component. The **Oracle Decomposer** is the gold-standard list of constraints for both REALINSTRUCT and IFEval. This provides insights on the ability to judge well the responses, knowing what to judge. The **Oracle Critic** is GPT-4-Turbo for REALINSTRUCT and the lenient rule-based evaluation program for IFEval. Additionally, for IFEval, we explore GPT-4 as a strong LLM (but less performant than Oracle) serving as the critic model. These Critic ablations provides insights on the ability of LLMs to refine itself when it knows what need to be refined.

Overall Quality Assessment (OQA) To ensure the pipeline does not degrade the quality of final responses, we perform Pairwise Quality Ranking using Prometheus-2 [28], an open LLM-as-a-Judge for general quality evaluation. This is done only on responses revised by the pipeline, comparing the initial response with the final revised one. More details in Appendix I.3.

5 Results and Discussion

Annotator	Cost (USD)	Time (min)	Macro F1 (%)	F1 Neg. (%)	Cohen's κ Maj. Vote
Expert	-	-	100.0	100.0	0.93
Human 1	300.0	-	85.1	75.9	0.77
Human 2	300.0	-	80.0	66.9	0.66
Majority Vote	-	-	96.4	94.1	1.00
All Satisfied	-	-	44.9	0.0	-0.09
All Not Satisfied	-	-	15.7	31.4	-0.70
GPT-4 w/ ICL-Cons	19.5	-	73.7	54.9	0.42
GPT-3.5-Turbo					
w/ ICL-Cons	1.0	-	51.3	16.6	0.09
GPT-4-Turbo					
w/ ICL-Inst	4.1	-	72.0	48.5	0.36
w/ ICL-Cons	6.5	-	72.6	54.8	0.46
w/ ICL-Cons+CoT	**8.3**	**-**	**79.0**	**65.5**	**0.50**
Mistral v0.2					
w/ ICL-Cons	-	10	50.4	11.4	0.02
w/ ICL-Cons+CoT	-	26	53.7	21.9	0.18
Supervised	**-**	**236**	**63.3**	**39.5**	**0.28**
Zephyr β					
w/ ICL-Cons	-	11	48.1	2.2	0.05
w/ ICL-Cons+CoT	-	49	48.2	10.4	0.03
Vicuna v1.3					
w/ ICL-Cons	-	9	47.2	1.4	0.12
w/ ICL-Cons+CoT	-	27	52.1	10.2	0.04

Table 3: Performance of different approaches for Constraint Verification task on our *EvalJudge* dataset.

Model	Instruction-level Accuracy	Constraint-level Accuracy
GPT-4	**78.8%**	**91.9%**
GPT-3.5	73.8%	84.0%
Mistral 7B v0.2	**75.2%**	**87.8%**
Zephyr 7B β	70.5%	84.7%
Vicuna 7B v1.3	61.3%	77.8%

Table 4: LLMs results on REALINSTRUCT

We discuss our results, particularly the Reliability of LLM-as-a-Judge for Constraint Satisfaction (Section 5.1), the ability of various open-source and proprietary LLMs on follow multi-constrained instructions (Section 5.2), and the efficacy of our DECRIM self-correction pipeline (Section 5.3).

5.1 Reliability of LLM-as-a-Judge

Results from several model and baseline judges on the EvalJudge dataset are presented in Table 3. Human inter-rater reliability (Human 1, Human 2, and Expert) is moderate (Krippendorff's $\alpha = 0.61$), with lower agreement between Humans 1 and 2 ($\kappa = 0.44$). This highlights the inherent challenges in verifying constraint satisfaction, as the task can be subjective and ambiguous, when involving multiple sub-constraints, despite efforts to minimize these issues during data annotation (see Appendix F.2.2).

GPT-4-Turbo with CoT is Reliable and More Cost-Efficient. We observe that GPT-4, a widely used LLM-as-a-Judge, shows lower performance compared to humans while maintaining moderate correlation with humans ($\kappa = 0.42$). Using GPT-4-Turbo, evaluating constraints individually yields better results than evaluating them all at once, despite a 37% cost increase. **GPT-4-Turbo with CoT prompt offers a more performant and cheaper alternative to GPT-4**. It reduces costs by 57% while improving overall performance (Macro F1) by +7.0% and in detecting unmet constraints (F1 Negative) by +19.0%. Its correlation with Expert annotation is similar to that of Human 2 (0.58 vs. 0.60). We thus adopt GPT-4-Turbo with CoT as the standard evaluation for REALINSTRUCT.

Open-source LLMs are unreliable judges. ICL-based configurations of open-source LLMs (Mistral, Vicuna, Zephyr) exhibit poor performance in all scenarios, particularly in detecting unmet constraints. Vicuna and Zephyr closely mirror the "All Satisfied" baseline, suggesting they are lenient judges. Mistral, despite similar Macro-F1, diverges in other metrics, indicating more random than lenient decisions (similar to GPT-3.5-Turbo). When weakly supervised with GPT-4-Turbo's CoT reasoning trails, Mistral significantly improves overall performance and the ability to detecting unmet constraints (+12.9 in Macro F1 and +28.1 in F1 Neg.). **This reduces the macro F1 gap with GPT-4-Turbo by about 50%, but the model still shows poor agreement with humans, indicating that open-source LLMs are not yet reliable judges**.

5.2 LLMs' ability to follow multi-constrained instructions on REALINSTRUCT

Benchmarking results for all models on REALINSTRUCT are presented in Table 4. We observe that **even with strong proprietary model, GPT-4, over 21% of instructions have at least one unsatisfied constraint**. Additionally, the open-source model Mistral v0.2 outperforms proprietary GPT-3.5 but falls short of GPT-4's performance. A qualitative examination of responses with unsatisfied constraints suggests that **LLMs often struggle with constraints involving numbers, negations, or long instructions with large number of constraints**, which is consistent with findings from previous works [29–31, 12]. We also discuss intra-model scoring bias problem in the Limitations Section. Overall, the results highlight the need for further enhancement of LLMs in handling multi-constrained instructions.

Strategy	Decomposer	Critic	REALINSTRUCT Best N	REALINSTRUCT Instruction Acc (%)	REALINSTRUCT Constraint Acc (%)	REALINSTRUCT OQA (%) Win / Lose	REALINSTRUCT Time (h)	IFEval Best N	IFEval Instruction Acc (%)	IFEval Constraint Acc (%)	IFEval OQA (%) Win / Lose	IFEval Time (h)
GPT-4	-	-	-	78.8	91.9	-	-	-	79.3[§]	85.4[§]	-	-
Conv.	-	-	-	75.2	87.8	-	2.5	-	60.1	66.3	-	2.9
Make sure	-	-	-	76.8	88.6	-	2.5	-	60.1	67.2	-	3.6
Self-Refine	-	-	2	77.2 (↑ 0.4)	88.7 (↑ 0.1)	22.1 / 21.2	8.6	2	59.5 (↓ 0.6)	66.4 (↓ 0.8)	21.3 / 20.8	4.5
DECRIM (ours)	Self	Self	6	75.2 (↓ 1.6)	88.9 (↑ 0.3)	36.7 / 22.4	11.2	4	60.1 (0.0)	**67.5 (↑ 0.3)**	17.1 / 30.6	5.3
	Self	**Supervised**	**10**	**80.5 (↑ 3.7)**	**90.9 (↑ 2.3)**	**37.1 / 22.0**	**6.9**	**10**	**60.8 (↑ 0.7)**	67.3 (↑ 0.1)	17.1 / 29.5	5.7
	Oracle[†]	Self	4	78.5 (↑ 1.7)	90.2 (↑ 1.6)	24.0 / 24.0	6.1	6	62.3 (↑ 2.2)	69.1 (↑ 1.9)	17.6 / 28.1	5.9
	Oracle[†]	Supervised	**10**	**82.4 (↑ 5.6)**	**91.7 (↑ 3.1)**	**34.3 / 22.2**	**5.6**	10	64.9 (↑ 4.8)	71.6 (↑ 4.4)	18.2 / 30.9	6.2
	Oracle[†]	GPT-4	-	-	-	-	-	**4**	**68.2 (↑ 8.1)**	**74.1 (↑ 6.9)**	**13.8 / 34.6**	**6.7**
	Oracle[†]	**Oracle[‡]**	**10**	**93.7 (↑ 16.9)**	**95.2 (↑ 6.6)**	**33.3 / 22.2**	**8.5**	**8**	**80.4 (↑ 20.3)**	**83.5 (↑ 16.3)**	**20.4 / 30.6**	**8.9**

Proprietary Model | Baselines | Fairly Comparable | Realistic Ablation | Unrealistic ablation (upper bound)

Table 5: Results of the best iteration on REALINSTRUCT and IFEval benchmarks for DECRIM pipeline. Except for the first line, all results use Mistral v0.2. Absolute improvements from *Make Sure* baseline are shown in (), with the best result in bold for each scenario. [†]Oracle decomposer refers to human-verified constraint annotations provided with the datasets. [‡]Oracle feedback is GPT-4-Turbo on REALINSTRUCT and a lenient rule-based evaluation on IFEVal. [§]Results reported by Zhou et al. [11]. Reported time considers only generation time for fair comparison.

5.3 Effectiveness of DECRIM Self-Correction

We present the results of our experiments with Mistral v0.2 using the DECRIM Self-Correction pipeline in Table 5. A slight improvement is observed with the *Make Sure* prompt compared to the conventional prompt on both datasets, demonstrating that it serves as a strong baseline for first-generation responses. Consequently, we adopt *Make Sure* as the baseline for further comparisons.

We classify the DECRIM configurations into three categories: (1) **Fairly Comparable**, where the pipeline operates independently of external models; (2) **Realistic Ablations**, which employ fairly comparable Critic models but use ideal Oracle Decomposer, which remains a realistic measure under the guidelines from Kamoi et al. [32], since decomposition is relatively straightforward and could be handled by a dedicated LLM; and (3) **Unrealistic Ablations**, which rely on ideal Critic and Decomposer models, useful for upper bound estimation but with limited generalization.

LLMs Cannot Self-Refine. We observe only marginal improvements on REALINSTRUCT and a performance drop on IFEval using the Self-Refine [18] pipeline, highlighting the limitations of traditional self-correction approaches for the multi-constrained instruction following task. Similarly, our DECRIM pipeline showed minimal to no gains when the LLM itself was used as both Decomposer and Critic. This shows that a self-refinement limitation exists even when the model is instructed to look specifically at constraints. **These poor results are attributed to low-quality Critic feedback, which can lead to over-refining good responses and neglecting to fix bad ones.** This aligns with findings by Huang et al. [33], Kamoi et al. [32], which showed that LLMs struggle with self-correction without external guidance. Additionally, this aligns with results from Sec. 5.1, in which Vanilla Mistral failed to reliably detect unsatisfied constraints.

DECRIM is effective, even with a Weak Critic. The results show significant improvement with the introduction of Supervised Mistral, a weak Critic that outperforms the LLM's self-critique but still underperforms as a Critic, as per results in Section 5.1. With a Self-Decomposer and Supervised Critic, performance increased by +3.7 on REALINSTRUCT and +0.7 on IFEval. With an Oracle Decomposer specifying accurately which constraints to check, improvements were +5.6 and +4.8, respectively, despite the harsher domain shift in IFEval, where Mistral Supervised performed even weaker.

Introducing a stronger Critic model, GPT-4, emulating what would be the performance of a model trained with high-quality data, resulted in an improvement of +8.1 on IFEval. However, it is worth mentioning that GPT-4 as evaluator only achieved a Macro F1 of 62.9% when judging Mistral's *Make Sure* responses on IFEval, being far from an ideal Critic for this dataset. Using an Oracle Critic – representing the strongest possible feedback – the model showed improvements of +16.9 on REALINSTRUCT and +20.3 on IFEval, **demonstrating that open-source LLMs can correct its responses when given high-quality feedback**. This highlights the potential of the DECRIM pipeline for multi-constrained instruction following, particularly when strong feedback is available.

Comparing DECRIM with proprietary GPT-4, even weak Critics enabled Mistral to surpass GPT-4's performance on REALINSTRUCT, though only Oracle Critic outperformed GPT-4 on IFEval,

likely due to its harder nature. Overall, these results demonstrate that **LLMs can achieve significant improvements in multi-constrained instruction following when provided with minimally reliable feedback**. As expected, **higher-quality feedback leads to better performance, with a plateau beyond the performance of proprietary models.**

DECRIM improves the quality of responses. Table 5 also provides a comparison of overall response quality before and after pipeline revisions. In most cases, the quality remained the same, but when changes occurred, on the REALINSTRUCT dataset, the revised responses were more oftenly preferred. This indicates that DECRIM either maintains or improves response quality in the majority of cases. Additionally, we observed a strong correlation between quality improvement and successful revisions, though a high number of revisions can negatively impact quality. This explains the higher quality improvement with better feedback on REALINSTRUCT and higher quality degradation observed on IFEval, which contains more difficult and unachievable constraints.

Importance of Decomposer and Critic Models. Our results also highlight the essential roles of the Decomposer and Critic in the DECRIM pipeline. While the Decomposer helps guide the Critic to the right constraints, the Critic's quality has the most significant impact on performance. The weakest configuration (Self-Decomposer and Self-Critic) results in minimal gains or slight degradation. However, even with a weak Self-Decomposer, improvements occur with a better Critic, Supervised, which boosts score in REALINSTRUCT from 75.2 to 80.5. The Oracle Decomposer consistently enhances performance across both benchmarks, but the Critic's quality drives the largest improvements. For example, using the Oracle Decomposer with the Supervised Critic yields a +4.8 instruction accuracy increase in IFEval, and with stronger Critics like GPT-4 or Oracle, the relative gains rise to +8.1 and +20.3, respectively. Same happens with Oracle Critic for REALINSTRUCT. These demonstrates that **while a better Decomposer is beneficial, the Critic's ability to correctly judge responses is the key factor for DECRIM's success**, with Oracle Critic consistently delivering the highest performance. In fact, the Critic's success is closely tied to the Decomposer's accuracy in pointing the constraints to be verified. In Appendix D.2 we discuss the role of Refine and its robustness to weak Critic.

6 Conclusion

In this work, we benchmarked LLMs' ability to follow real multi-constrained user requests with REALINSTRUCT. We showed that even strong proprietary models like GPT-4 fail to meet at least one constraint in over 21% of instructions, demonstrating REALINSTRUCT's challenging nature and the need for improvement across both proprietary and open-source models. To address this, we proposed the DECRIM self-correction pipeline, which decomposes instructions into granular requirements, critiques responses, and refines outputs. Extensive experiments showed that DECRIM significantly improves open-source LLM performance, with stronger feedback allowing them to surpass GPT-4. Overall, our work highlights the underexplored problem of following real-world user requests, as well as advances System 2 techniques with DECRIM. Future work could refine the DECRIM's components and integrate the pipeline to other System 2 approaches, such as self-consistency and generate-and-rank, to enhance its effectiveness in tasks where spending more time on generation-refinement iterations would improve performance.

References

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf`.

[2] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=gEZrGCozdqR.

[3] Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Cross-task generalization via natural language crowdsourcing instructions. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.244. URL https://aclanthology.org/2022.acl-long.244.

[4] Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=9Vrb9D0WI4.

[5] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf.

[6] Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, et al. Super-NaturalInstructions: Generalization via Declarative Instructions on 1600+ NLP Tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109, 2022. URL https://aclanthology.org/2022.emnlp-main.340/.

[7] Norman Mu, Sarah Chen, Zifan Wang, Sizhe Chen, David Karamardian, Lulwa Aljeraisy, Dan Hendrycks, and David Wagner. Can llms follow simple rules? *arXiv preprint arXiv:2311.04235*, 2023. URL https://arxiv.org/abs/2311.04235.

[8] Wangchunshu Zhou, Yuchen Eleanor Jiang, Ethan Wilcox, Ryan Cotterell, and Mrinmaya Sachan. Controlled Text Generation with Natural Language Instructions. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 42602–42613. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/zhou23g.html.

[9] Albert Lu, Hongxin Zhang, Yanzhe Zhang, Xuezhi Wang, and Diyi Yang. Bounding the Capabilities of Large Language Models in Open Text Generation with Prompt Constraints. In Andreas Vlachos and Isabelle Augenstein, editors, *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1982–2008, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-eacl.148. URL https://aclanthology.org/2023.findings-eacl.148.

[10] Jiao Sun, Yufei Tian, Wangchunshu Zhou, Nan Xu, Qian Hu, Rahul Gupta, John Wieting, Nanyun Peng, and Xuezhe Ma. Evaluating Large Language Models on Controlled Generation Tasks. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3155–3168, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.190. URL https://aclanthology.org/2023.emnlp-main.190.

[11] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023. URL https://arxiv.org/abs/2311.07911.

[12] Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. FollowBench: A multi-level fine-grained constraints following benchmark for large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4667–4688, Bangkok, Thailand, August 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.acl-long.257.

[13] Shunyu Yao, Howard Chen, Austin W. Hanjie, Runzhe Yang, and Karthik R Narasimhan. COLLIE: Systematic construction of constrained text generation tasks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=kxgSlyirUZ.

[14] Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. Infobench: Evaluating instruction following ability in large language models. *ArXiv*, abs/2401.03601, 2024. URL https://api.semanticscholar.org/CorpusID:266844311.

[15] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91f18a1287b398d378ef22505bf41832-Paper-Datasets_and_Benchmarks.pdf.

[16] Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. Evaluating large language models at evaluating instruction following. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=tr0KidwPLc.

[17] Swarnadeep Saha, Omer Levy, Asli Celikyilmaz, Mohit Bansal, Jason Weston, and Xian Li. Branch-Solve-Merge Improves Large Language Model Evaluation and Generation. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8352–8370, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.462. URL https://aclanthology.org/2024.naacl-long.462.

[18] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 46534–46594. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91edff07232fb1b55a505a9e9f6c0ff3-Paper-Conference.pdf.

[19] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*, 2023. URL https://arxiv.org/abs/2303.08774.

[20] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In Qun Liu and David Schlangen, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6.

[21] 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. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023. URL https://arxiv.org/abs/2310.06825.

[22] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

[23] Lewis Tunstall, Edward Emanuel Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro Von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M Rush, and Thomas Wolf. Zephyr: Direct Distillation of LM Alignment. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=aKkAwZB6JV.

[24] Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. Open LLM Leaderboard. *Hugging Face*, 2023. URL https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard.

[25] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

[26] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

[27] Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot arena: An open platform for evaluating llms by human preference, 2024.

[28] Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Prometheus 2: An open source language model specialized in evaluating other language models. *arXiv preprint arXiv:2405.01535*, 2024. URL https://arxiv.org/abs/2405.01535.

[29] Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, WSDM '24, page 645–654, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400703713. doi: 10.1145/3616855.3635752. URL https://doi.org/10.1145/3616855.3635752.

[30] Iker García-Ferrero, Begoña Altuna, Javier Alvez, Itziar Gonzalez-Dios, and German Rigau. This is not a dataset: A large negation benchmark to challenge large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8596–8615, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.531. URL https://aclanthology.org/2023.emnlp-main.531.

[31] Thinh Hung Truong, Timothy Baldwin, Karin Verspoor, and Trevor Cohn. Language models are not naysayers: an analysis of language models on negation benchmarks. In Alexis Palmer and Jose Camacho-collados, editors, *Proceedings of the 12th Joint Conference on Lexical and Computational Semantics (*SEM 2023)*, pages 101–114, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.starsem-1.10. URL https://aclanthology.org/2023.starsem-1.10.

[32] Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When Can LLMs Actually Correct Their Own Mistakes? A Critical Survey of Self-Correction of LLMs. *arXiv preprint arXiv:2406.01297*, 2024. URL https://arxiv.org/abs/2406.01297.

[33] Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. *ArXiv*, abs/2310.01798, 2023. URL https://api.semanticscholar.org/CorpusID:260360912.

[34] Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. State of What Art? A Call for Multi-Prompt LLM Evaluation. *Transactions of the Association for Computational Linguistics*, 12:933–949, 08 2024. ISSN 2307-387X. doi: 10.1162/tacl_a_00681. URL https://doi.org/10.1162/tacl_a_00681.

[35] Pat Verga, Sebastian Hofstatter, Sophia Althammer, Yixuan Su, Aleksandra Piktus, Arkady Arkhangorodsky, Minjie Xu, Naomi White, and Patrick Lewis. Replacing judges with juries: Evaluating llm generations with a panel of diverse models. *arXiv preprint arXiv:2404.18796*, 2024. URL https://arxiv.org/abs/2404.18796.

[36] Arjun Panickssery, Samuel R Bowman, and Shi Feng. Llm evaluators recognize and favor their own generations. *arXiv preprint arXiv:2404.13076*, 2024. URL https://arxiv.org/abs/2404.13076.

[37] Olivia Huang, Eve Fleisig, and Dan Klein. Incorporating worker perspectives into MTurk annotation practices for NLP. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1010–1028, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.64. URL https://aclanthology.org/2023.emnlp-main.64.

[38] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472.

[39] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, aug 2021. ISSN 0001-0782. doi: 10.1145/3474381. URL https://doi.org/10.1145/3474381.

[40] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

[41] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

[42] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, Alice Xiang, Alicia Parrish, Allen Nie, Aman Hussain, Amanda Askell, Amanda Dsouza, Ambrose Slone, Ameet Rahane, Anantharaman S. Iyer, Anders Johan Andreassen, Andrea Madotto, Andrea Santilli, Andreas Stuhlmüller, Andrew M. Dai, Andrew La, Andrew Lampinen, Andy Zou, Angela Jiang, Angelica Chen, Anh Vuong, Animesh Gupta, Anna Gottardi, Antonio Norelli, Anu Venkatesh, Arash Gholamidavoodi, Arfa Tabassum, Arul Menezes, Arun Kirubarajan, Asher Mullokandov, Ashish Sabharwal, Austin Herrick, Avia Efrat, Aykut Erdem, Ayla Karakaş, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=uyTL5Bvosj.

[43] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024. URL http://jmlr.org/papers/v25/23-0870.html.

[44] Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*, 2022. URL https://arxiv.org/abs/2211.09085.

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

[46] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. BERTScore: Evaluating Text Generation with BERT. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SkeHuCVFDr.

[47] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: NLG evaluation using gpt-4 with better human alignment. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.153. URL https://aclanthology.org/2023.emnlp-main.153.

[48] Hui Huang, Yingqi Qu, Jing Liu, Muyun Yang, and Tiejun Zhao. On the Limitations of Fine-tuned Judge Models for LLM Evaluation. *arXiv preprint arXiv:2403.02839*, 2024. URL https://arxiv.org/pdf/2403.02839.

[49] Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. Prometheus: Inducing fine-grained evaluation capability in language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=8euJaTveKw.

[50] Soochan Lee and Gunhee Kim. Recursion of thought: A divide-and-conquer approach to multi-context reasoning with language models. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 623–658, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.40. URL https://aclanthology.org/2023.findings-acl.40.

[51] Edoardo Manino, Julia Rozanova, Danilo Carvalho, Andre Freitas, and Lucas Cordeiro. Systematicity, compositionality and transitivity of deep NLP models: a metamorphic testing perspective. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2355–2366, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.185. URL https://aclanthology.org/2022.findings-acl.185.

[52] Verna Dankers, Elia Bruni, and Dieuwke Hupkes. The paradox of the compositionality of natural language: A neural machine translation case study. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4154–4175, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.286. URL https://aclanthology.org/2022.acl-long.286.

[53] Tianqi Zhong, Zhaoyi Li, Quan Wang, Linqi Song, Ying Wei, Defu Lian, and Zhendong Mao. Benchmarking and improving compositional generalization of multi-aspect controllable text

generation. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6486–6517, Bangkok, Thailand, August 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.acl-long.351.

[54] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/271db9922b8d1f4dd7aaef84ed5ac703-Paper-Conference.pdf.

[55] Jingyuan Qi, Zhiyang Xu, Ying Shen, Minqian Liu, Di Jin, Qifan Wang, and Lifu Huang. The art of SOCRATIC QUESTIONING: Recursive thinking with large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4177–4199, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.255. URL https://aclanthology.org/2023.emnlp-main.255.

[56] Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.741. URL https://aclanthology.org/2023.emnlp-main.741.

[57] Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 292–305, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.20. URL https://aclanthology.org/2023.emnlp-main.20.

[58] Liqiang Jing, Ruosen Li, Yunmo Chen, Mengzhao Jia, and Xinya Du. Faithscore: Evaluating hallucinations in large vision-language models, 2023. URL https://openreview.net/pdf?id=E6uTejGgth.

[59] Yushi Hu, Benlin Liu, Jungo Kasai, Yizhong Wang, Mari Ostendorf, Ranjay Krishna, and Noah A. Smith. Tifa: Accurate and interpretable text-to-image faithfulness evaluation with question answering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 20406–20417, October 2023. URL https://openaccess.thecvf.com/content/ICCV2023/papers/Hu_TIFA_Accurate_and_Interpretable_Text-to-Image_Faithfulness_Evaluation_with_Question_Answering_ICCV_2023_paper.pdf.

[60] Hwanjun Song, Hang Su, Igor Shalyminov, Jason Cai, and Saab Mansour. FineSurE: Fine-grained summarization evaluation using LLMs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 906–922, Bangkok, Thailand, August 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.acl-long.51.

[61] Weijia Zhang, Mohammad Aliannejadi, Yifei Yuan, Jiahuan Pei, Jia-Hong Huang, and Evangelos Kanoulas. Towards fine-grained citation evaluation in generated text: A comparative analysis of faithfulness metrics. *arXiv preprint arXiv:2406.15264*, 2024. URL https://arxiv.org/pdf/2406.15264.

[62] Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. Teaching small language models to reason. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1773–1781, Toronto, Canada, July

2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-short.151. URL https://aclanthology.org/2023.acl-short.151.

[63] Pei Ke, Bosi Wen, Andrew Feng, Xiao Liu, Xuanyu Lei, Jiale Cheng, Shengyuan Wang, Aohan Zeng, Yuxiao Dong, Hongning Wang, Jie Tang, and Minlie Huang. CritiqueLLM: Towards an informative critique generation model for evaluation of large language model generation. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13034–13054, Bangkok, Thailand, August 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.acl-long.704.

[64] Kumar Shridhar, Harsh Jhamtani, Hao Fang, Benjamin Van Durme, Jason Eisner, and Patrick Xia. Screws: A modular framework for reasoning with revisions. *arXiv preprint arXiv:2309.13075*, 2023. URL https://arxiv.org/pdf/2309.13075.

[65] Kumar Shridhar, Koustuv Sinha, Andrew Cohen, Tianlu Wang, Ping Yu, Ramakanth Pasunuru, Mrinmaya Sachan, Jason Weston, and Asli Celikyilmaz. The ART of LLM Refinement: Ask, Refine, and Trust. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5872–5883, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.327. URL https://aclanthology.org/2024.naacl-long.327.

[66] Zilong Wang, Zifeng Wang, Long Le, Huaixiu Steven Zheng, Swaroop Mishra, Vincent Perot, Yuwei Zhang, Anush Mattapalli, Ankur Taly, Jingbo Shang, et al. Speculative rag: Enhancing retrieval augmented generation through drafting. *arXiv preprint arXiv:2407.08223*, 2024. URL https://arxiv.org/abs/2407.08223.

[67] Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models, 5 2023. URL https://github.com/tatsu-lab/alpaca_eval.

[68] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.754. URL https://aclanthology.org/2023.acl-long.754.

[69] Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiaxin Xu, et al. Benchmarking complex instruction-following with multiple constraints composition. *arXiv preprint arXiv:2407.03978*, 2024. URL https://arxiv.org/abs/2407.03978.

[70] Qianyu He, Jie Zeng, Wenhao Huang, Lina Chen, Jin Xiao, Qianxi He, Xunzhe Zhou, Jiaqing Liang, and Yanghua Xiao. Can large language models understand real-world complex instructions? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18188–18196, 2024. URL https://ojs.aaai.org/index.php/AAAI/article/view/29777.

[71] Tao Zhang, Yanjun Shen, Wenjing Luo, Yan Zhang, Hao Liang, Fan Yang, Mingan Lin, Yujing Qiao, Weipeng Chen, Bin Cui, et al. Cfbench: A comprehensive constraints-following benchmark for llms. *arXiv preprint arXiv:2408.01122*, 2024. URL https://arxiv.org/abs/2408.01122.

[72] Alan S Morris. Measurement and instrumentation principles. *Measurement Science and Technology*, 12(10):1743–1744, 2001. URL http://llrc.mcast.edu.mt/digitalversion/table_of_contents_1274.pdf.

[73] J.R. Taylor. *Introduction To Error Analysis: The Study of Uncertainties in Physical Measurements*. ASMSU/Spartans.4.Spartans Textbook. University Science Books, 1997. ISBN 9780935702750. URL https://books.google.com/books?id=giFQcZub80oC.

[74] Institution British Standards. *BS ISO 5725-1. Accuracy (trueness and Precision) of Measurement Methods and Results: Part 1. General principles and definitions*. Number pt. 1. British Standards Institution, 2022. URL https://books.google.com/books?id=f5Q2zwEACAAJ.

[75] Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. Automatically Correcting Large Language Models: Surveying the Landscape of Diverse Automated Correction Strategies. *Transactions of the Association for Computational Linguistics*, 12:484–506, 2024. doi: 10.1162/tacl_a_00660. URL https://aclanthology.org/2024.tacl-1.27.

[76] Seonghyeon Ye, Yongrae Jo, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, and Minjoon Seo. Selfee: Iterative self-revising llm empowered by self-feedback generation. Blog post, May 2023. URL https://kaistai.github.io/SelFee/.

[77] Timo Schick, Jane A. Yu, Zhengbao Jiang, Fabio Petroni, Patrick Lewis, Gautier Izacard, Qingfei You, Christoforos Nalmpantis, Edouard Grave, and Sebastian Riedel. PEER: A collaborative language model. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=KbYevcLjnc.

[78] William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. Self-critiquing models for assisting human evaluators. *arXiv preprint arXiv:2206.05802*, 2022. URL https://arxiv.org/pdf/2206.05802.

[79] Seongyun Lee, Sue Hyun Park, Yongrae Jo, and Minjoon Seo. Volcano: Mitigating multimodal hallucination through self-feedback guided revision. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 391–404, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.23. URL https://aclanthology.org/2024.naacl-long.23.

[80] Wenda Xu, Danqing Wang, Liangming Pan, Zhenqiao Song, Markus Freitag, William Wang, and Lei Li. INSTRUCTSCORE: Towards explainable text generation evaluation with automatic feedback. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5967–5994, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.365. URL https://aclanthology.org/2023.emnlp-main.365.

[81] Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. Generating sequences by learning to self-correct. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=hH36JeQZDaO.

[82] Kevin Yang, Yuandong Tian, Nanyun Peng, and Dan Klein. Re3: Generating longer stories with recursive reprompting and revision. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4393–4479, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.296. URL https://aclanthology.org/2022.emnlp-main.296.

[83] Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Sx038qxjek.

[84] Neeraj Varshney, Wenlin Yao, Hongming Zhang, Jianshu Chen, and Dong Yu. A stitch in time saves nine: Detecting and mitigating hallucinations of llms by validating low-confidence generation. *arXiv preprint arXiv:2307.03987*, 2023. URL https://arxiv.org/abs/2307.03987.

[85] Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. CommonGen: A constrained text generation challenge for generative

commonsense reasoning. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1823–1840, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.165. URL https://aclanthology.org/2020.findings-emnlp.165.

[86] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

[87] Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language models. In *ICML 2024 Workshop on In-Context Learning*, 2024. URL https://openreview.net/forum?id=LjsjHF7nAN.

[88] Yuxuan Yao, Han Wu, Zhijiang Guo, Zhou Biyan, Jiahui Gao, Sichun Luo, Hanxu Hou, Xiaojin Fu, and Linqi Song. Learning from correctness without prompting makes LLM efficient reasoner. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=dcbNzhVVQj.

[89] Akbir Khan, John Hughes, Dan Valentine, Laura Ruis, Kshitij Sachan, Ansh Radhakrishnan, Edward Grefenstette, Samuel R. Bowman, Tim Rocktäschel, and Ethan Perez. Debating with more persuasive LLMs leads to more truthful answers. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=iLCZtl7FTa.

[90] Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=zj7YuTE4t8.

[91] Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. Are more llm calls all you need? towards scaling laws of compound inference systems. *arXiv preprint arXiv:2403.02419*, 2024. URL https://arxiv.org/abs/2403.02419.

[92] Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. Implicit chain of thought reasoning via knowledge distillation, 2024. URL https://openreview.net/forum?id=9cumTvvlHG.

[93] Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. Distilling system 2 into system 1. *arXiv preprint arXiv:2407.06023*, 2024. URL https://arxiv.org/abs/2407.06023.

[94] Yihe Deng, Weitong Zhang, Zixiang Chen, and Quanquan Gu. Rephrase and respond: Let large language models ask better questions for themselves. *arXiv preprint arXiv:2311.04205*, 2023.

[95] Jason Weston and Sainbayar Sukhbaatar. System 2 attention (is something you might need too). *arXiv preprint arXiv:2311.11829*, 2023.

Contents

1 Introduction **1**

2 The REALINSTRUCT Dataset **3**

2.1 Data Description 3

2.2 Data Construction 4

3 Self-correction with DECOMPOSE, CRITIQUE, AND REFINE (DECRIM) **4**

4 Experimental Setup **5**
4.1 Validating LLM-as-a-Judge for Constraint Satisfaction 5
4.1.1 Adaptation strategies 5
4.1.2 The *EvalJudge* Dataset 6
4.2 Benchmarking LLMs on REALINSTRUCT 7
4.3 Evaluating DECRIM pipeline 7

5 Results and Discussion **7**
5.1 Reliability of LLM-as-a-Judge 8
5.2 LLMs' ability to follow multi-constrained instructions on REALINSTRUCT 8
5.3 Effectiveness of DECRIM Self-Correction 9

6 Conclusion **10**

A Limitations **21**

B Ethical Considerations **22**

C Related Work **22**
C.1 Evaluating LLMs' Generative Abilities 22
C.1.1 LLM-as-a-judge 22
C.1.2 Fine-grained evaluation 23
C.1.3 Benchmarking Instruction-Following Abilities 23
C.2 LLM Self-Correction for open-ended text generation 24
C.2.1 Constrained Generation 24
C.2.2 System 2 Approaches 25

D Extra Analysis and Discussions of DECRIM **25**
D.1 DECRIM Experiments with other Open LLMs 26
D.2 Comparing DECRIM with Generate-and-Rank 26

E Definition of Task, Context, and Constraints in REALINSTRUCT **27**

F REALINSTRUCT data construction details **28**
F.1 Data Filtering 28
F.1.1 Prompt for two-shot classification of code-related conversations 28
F.1.2 Prompt for few-shot classification of instruction with constraints 29
F.1.3 Data Collection Validation Guidelines 30
F.2 Instruction decomposition 30
F.2.1 Prompt for Instruction decomposition with GPT-4 31
F.2.2 Decomposition validation guidelines 32

G REALINSTRUCT Constraints Categorization **34**

H REALINSTRUCT Data Samples **35**

I Implementation details for DECRIM pipeline **38**

I.1 Instruction decomposition 38

I.2 Iterative Self-Correction with Feedback from Critic 39

I.3 Overall Quality Assessment (OQA) details 40

J Extra Experimental Details for LLM-as-a-Judge Validation for Constraint Satisfaction **40**

J.1 Prompts for ICL-based Adaptation Strategies 40

J.1.1 Prompt Instruction-wise Eval (ICL-Inst.) 40

J.1.2 Prompt Constraint-wise Eval (ICL-Const.) 41

J.1.3 Prompt Constraint-wise Eval + CoT (ICL-Const.+CoT) 42

J.2 Guidelines for Constraint Satisfaction Human Audition 43

J.3 Princing Details for Propretary LLM-as-a-Judge 44

J.4 Details for Open LLM Weak Supervision 45

A Limitations

Model-based vs. Rule-based Evaluation. Using model-based evaluation over rule-based introduces two key challenges. First, it reflects the precision/accuracy trade-off, where rule-based methods offer higher precision but are less accurate due to synthetic scenarios, while model-based ones, though less precise, better align with real-world tasks (see discussion on Section C.1.3). Second, evaluating the REALINSTRUCT dataset relies on the proprietary GPT-4-Turbo API, making it costly. To address these, techniques like multi-prompting [34] and panel of juries [35] can improve precision in model-based evaluation, including with open-source models, while future advancements in open-source LLMs may provide cost-effective evaluation alternatives.

Data Contamination and Intra-Model Scoring Bias. Developing benchmarks with publicly available data that does not overlap with LLM training data is challenging, as pre-training and instruction-tuning datasets are often undisclosed. For example, reliable information on Mistral's training data is unavailable. However, while Vicuna v1.3 reported instruction tuning on a dataset overlapping with REALINSTRUCT, no significant intra-model bias from data contamination was observed, as seen in its poor performance in Table 4. This is likely due to its instruction tuning procedure not ensuring constraint satisfaction in target responses. However, GPT-4's relatively high constraint-level accuracy could indicate scoring bias, as previous studies suggest GPT-4 tends to favor its own outputs [15, 36, 35]. Further investigation into data contamination and intra-model scoring bias is left as future work.

Computation Overhead. The DECRIM pipeline introduces additional computational time compared to single-pass generation. Table 5 provides the running time for each configuration explored. **To mitigate this, we have designed the pipeline to trigger the refinement step only when the Critic model detects unsatisfied constraints**, which occured in about 25% of instructions, minimizing unnecessary computation when the model performs well initially. This is an improvement upon other *System 2* approaches (see Section C.2.2), such as generate-and-rank, which typically generate multiple outputs for further ranking. Additionally, we observed that **most revisions occur in the first iteration, resulting in a sublinear time increase with more iterations**. Also, **higher-quality feedback further reduces the need for revisions**, improving DECRIM's efficiency.

Optimization Considerations. Due to high computational costs, we did not optimize hyperparameters for training the weakly supervised LLM-as-a-Judge or exhaustively tune the prompts for the adaptation strategies. Additionally, we did not explore using a dedicated LLM as a Decomposer in

the DECRIM pipeline, as this is primarily an implementation-focused task, being not critical for demonstrating our core claims. These aspects are left for future work.

B Ethical Considerations

Crowdsourcing. For the *EvalJudge* Human Annotation task, we recruited native English speakers through Amazon Mechanical Turk[4] (MTurk). Compensation was based on the number of constraints per instruction, with an estimated average payment of 16.90 USD per hour, which exceeds the highest U.S. minimum wage in 2024 (16.30 USD per hour in Washington State), aligning with ethical guidelines discussed by Huang et al. [37].

Data from real users. Constructing a dataset from real user requests presents some ethical challenges:

- **Personally Identifiable Information (PII):** Some user interactions with AI assistants may contain PII. During data validation, we actively sought to remove instances containing PII from the dataset. See Appendix F.1.3 for further details.
- **Harmful Content:** The underlying data source is uncensored, and users may produce or request toxic or harmful content. Apart from flagrant cases, we did not actively remove such instances from the dataset.

Societal Impact. The DECRIM pipeline improves LLMs' ability to follow user-requested constraints, contributing to a broader societal impact of advancing LLM capabilities. When it comes particularly to user requests, it is important to note that some user constraints may conflict with system constraints set by developers, such as requests to generate harmful or toxic content. Although our study does not look into conflicting constraints, there is a potential risk that the pipeline could prioritize user requests over developer-defined safeguards.

C Related Work

This section situates our work within broader research directions, highlighting intersections with current studies. Section C.1 focuses on benchmarking and evaluating LLMs' generative abilities, while Section C.2 discusses approaches for enhancing LLM responses.

C.1 Evaluating LLMs' Generative Abilities

Traditional language model benchmarks, such as HellaSwag [38], WinoGrande [39], MMLU [40], GSM-8K [41], and BIG-Bench [42], primarily assess LLMs on tasks like commonsense reasoning and standardized exams. These benchmarks evaluate models using multiple-choice questions (MCQs) to objectively measure internal reasoning capabilities. However, recent advancements in language models have demonstrated emergent capabilities in generating high-quality open-ended text generation [2, 43, 5, 44, 45].

This shift presents new challenges, as the number of possible responses are virtually infinite, requiring more subjective evaluation rather than strict reference matching. While MCQ-based benchmarks fall short in assessing these generative abilities, human annotation, though reliable, is limited by cost and scalability. To address this, some research directions sacrifice the question quality to be able to use rule-based evaluation methods, while others explore model-based approaches.

C.1.1 LLM-as-a-judge

Early model-based efforts like BERTScore [46] sought to improve on traditional n-gram metrics by recognizing high-quality responses that differ from the reference. For more open-ended generation, where references are soft or nonexistent, recent work has introduced the concept of **LLM-as-a-Judge** [15, 47], using strong proprietary LLMs like GPT-4 [19] to evaluate responses. These models have shown they can approximate the depth and consistency of manual human evaluation, but also provide better consistency and stability.

[4]Refer to: `https://www.mturk.com/`

Recent research has begun exploring open-source LLMs for LLM-as-a-Judge, aiming to reduce reliance on proprietary models. Although open-source models have shown limited capability with in-context learning, fine-tuning them for specific evaluations is a promising direction [48, 49]. Contemporaneous work Prometheus-2, an open LLM-as-a-Judge, has shown strong correlation with human evaluation, even surpassing GPT-4 in some cases, though it still lags in out-of-domain cases [28]. In our work, we assess both proprietary and open-source models for evaluating user constraint satisfaction in LLM responses. Our results indicate that while proprietary models outperform, open models can improve significantly when weakly supervised with proprietary model evaluations and reasoning trails, making them viable as Critic models in a self-correction pipeline.

Another recent approach by Verga et al. [35] proposes replacing individual judges with juries of cheaper LLMs, which has been shown to correlate better with human judgments, even outperforming GPT-4 in some scenarios and reducing intra-model evaluation bias. This suggests that exploring LLM panels for constraint satisfaction evaluation could be a fruitful direction for future work.

C.1.2 Fine-grained evaluation

Some studies have explored approaches inspired by the divide-and-conquer paradigm, breaking multifaceted tasks into fine-grained components [50]. This can be successful given a complex compositional characteristic of the Natural Language [51–53]. For evaluation, this strategy not only provides detailed insights into model performance across different aspects but also makes evaluations more objective and less ambiguous, as models may excel in some areas while underperforming in others. This approach seems promising for LLM-as-a-Judge, given that LLMs prompting techniques such as Chain-of-Thought [25], Tree-of-Thought [54], and Recursive Thinking [55], have demonstrated LLM performance improvements by breaking complex tasks into simpler sequential steps. We discuss these methods on Section C.2.2.

Specific works on evaluation by Min et al. [56], Li et al. [57], Jing et al. [58], Hu et al. [59], Song et al. [60], Huang et al. [48], Zhang et al. [61] have shown the benefits of decomposing tasks into atomic facts for tasks such as fact-checking against cross-modality references. Kim et al. [49], Magister et al. [62], Ke et al. [63] have demonstrated that fine-grained evaluation from diverse sources enhances fine-tuned open evaluators by making the task more objective. Additionally, weak fine-grained evaluation during generation time has been shown to improve LLM self-correction performance [64–66].

In our work, we implement a similar approach by decomposing the task of evaluating multi-constrained instructions into individual constraint evaluations. This "instruction decomposition" simplifies and makes more objective the instruction evaluation task for LLMs and provides more informative insights through constraint-level accuracy metrics. We also argue that existing overall instruction satisfaction metrics fail to detect unmet constraints due to the ambiguity caused by the lack of granularity, as also highlighted by Sun et al. [10]. Our results demonstrate the effectiveness of fine-grained evaluation for this task and show that incorporating it into a self-correction pipeline enhances performance, even with weak Critic and Decomposer models.

C.1.3 Benchmarking Instruction-Following Abilities

The ability of LLMs to follow user instructions in open-ended text generation has only recently gained attention. New benchmarks like AlpacaEval [67] and the test splits of Natural-Instructions [3] and Self-Instruct [68] address the evaluation in this aspect by using LLM-as-a-Judge to compare with reference responses or provide overall instruction satisfaction scores. Recent studies have shown that models often follow instructions only partially, frequently failing to adhere to specific constraints provided by users [10, 11, 13, 12, 14, 69–71].

To evaluate this, the few existing benchmarks focus on a set of specific constraint categories and/or use synthetic constraints that can be easily verified through rule-based methods [11, 13]. The trade-off between rule-based and model-based evaluation falls into the famous precision/accuracy dilemma about static instrument characteristics (sometimes referred as bias/variance dilemma) in the Statistics of measurements [72–74]. Rule-based evaluation offers high-to-perfect precision (low variance), but it is usually required to be done on unrealistic scenarios, being less accurate (high bias). The use of model-based evaluation loses some precision compared to rule-based due to inherent variability

Method	Feedback Source	Refinement Strategy	Tasks Investigated	Supported Constraint Types
Self-correction with training for refining				
Selfee [76]	LLM + ICL	LLM SFT	Open-ended Instructions	Not constraint focus
PEER [77]	LLM SFT	LLM SFT	Constrained Generation	Limited constraints
Self-Critique [78], VOLCANO [79]	LLM SFT w/ Human Feedback	LLM SFT w/ Human Feedback	Conditional Summarization, Visual Question-Answering	Limited constraints
InstructScore [80]	LLM SFT w/ GPT-4 and Human Feedback	LLM SFT w/ GPT-4	CommonGen	Limited constraints
Self-Correctors [81]	External tools	Smaller LLM STF	CommonGen, Detoxification	Limited constraints
RULES [7]	External tools	LLM SFT	System Constraints	Limited constraints
Re3 [82]	Smaller LLM SFT + External Tools	LLM + smaller model	Story Generation	Limited constraints
Self-correction without training for refinement				
Self-Refine [18]	LLM + ICL	LLM + ICL	Open-ended Instructions; CommonGen	Not constraint focus; Limited constraints
CRITIC [83], Hallucination [84]	External tools		Detoxification, Hallucination	Limited constraints
DECRIM (ours)	LLM SFT w/ GPT-4		Open-ended Instructions	Any constraint

Table 6: Comparison of representative works on Self-Correction for Constrained Generation. Our DECRIM pipeline is unique as do not require LLM fine-tuning for refinement, being also the only that can handle open-ended instructions with any type of constraints.

introduced by LLMs (lower precision, higher variance), but aligns more with the task objective of evaluating more realistic scenarios (higher accuracy, lower bias).

To the best of our knowledge, our REALINSTRUCT benchmark is the first to evaluate LLMs using real-user instructions, offering a more realistic and comprehensive assessment. This approach closely mirrors real-world scenarios, unlike previous benchmarks that rely on synthetic constraints, as contrasted on Table 2 and Section 2. Our benchmark's success relies on a fine-grained evaluation protocol using LLM-as-a-Judge.

C.2 LLM Self-Correction for open-ended text generation

Self-correction has emerged as an effective approach for enhancing LLM responses during generation by refining them during generation time [75, 32]. However, Kamoi et al. [32], Huang et al. [33] demonstrated that the ability of LLM to self-correct alone is limited to tasks where responses can be decomposed and rely on verifiable components. For harder tasks, LLM self-correction may require additional modeling, new data, or even external tools.

In this sense, Self-correction approaches can be categorized based on the feedback source. **Intrinsic Self-Correction** uses carefully crafted prompts or in-context examples to enable the model to identify issues in its output. **Self-Correction with External Feedback** leverages external tools or more advanced LLMs to provide feedback, while **Self-Correction with Fine-Tuning** uses external feedback (from humans, stronger LLMs, external tools) to fine-tune the LLM for better feedback and/or response refinement. Kamoi et al. [32] emphasizes that each self-correction category should be validated using comparable cases specific to its context.

In the case of multi-constrained instructions, the constraints are neither independent nor ordered, making it difficult to guarantee that all responses are decomposable. For example, constraints such as length and style do not have a specific part of the response to be followed, they should be followed in the whole text. Moreover, some constraints are subjective and harder to evaluate, and the instruction decomposition process may introduce noise, further complicating self-correction with the model itself. In our work, we explore both Intrinsic Self-Correction and Self-Correction with Critic Fine-Tuning. As ablation exploration, we also play with External Feedback (referred as Oracle Critic), as an estimation of upper bound performance but recognizing its limited generalization.

C.2.1 Constrained Generation

Recent work has explored constrained generation via self-correction, we show some representative work on Table 6. Some approaches are validated only on small, specific constraint sets, limiting their general applicability [77–79, 7, 82–84]. For example, Mu et al. [7] evaluates 13 System constraints,

that is constraints defined by the developer. Hallucination and Detoxification constraints can also be seen as System Constraints.

In another direction, Madaan et al. [18] enhances model outputs by having the model self-review and self-correct its answers. However, this approach is suboptimal as it lacks problem-specific modeling, that is it does not recognize the constraints to be followed, and only prompts the model to evaluate and improve its responses without clear guidance on what to focus on. Ye et al. [76] goes in a similar direction but relying on supervising LLM to be able to refine. Our results in Sections 5.1 and 5.3 demonstrate improved performance by directing the model to specifically address constraints, even when it is refining its responses using only its own feedback. To the best of our knowledge we are the first to handle open-ended user instructions without restricting it to some constraint types. We highlight the prevalence of constraints in real-world user instructions, making this an important area for further study. Also, unlike most methods, our DECRIM pipeline does not require external tools or fine-tuning LLM for refinement.

It is worth noting that most current constrained generation with self-correction research evaluates on the CommonGen benchmark [85], in which constraints are word lists for LLM to include in the text. We argue that this benchmark is insufficient for measuring the ability to follow user requests because: (1) models like GPT-4 already perform at human level[5]; (2) it only represents one type of constraint among many possible; and (3) the constraints are synthetic and not reflective of typical human requests. While valuable in the past for large-scale rule-based evaluation, it may not adequately measure modern LLM capabilities.

C.2.2 System 2 Approaches

Kamoi et al. [32] differentiate self-correction from other methods like self-consistency [86–88], which samples diverse reasoning paths during decoding and selects the most consistent, and generate-then-rank methods like Tree of Thoughts [54], which generates multiple responses and ranks them using a critic model. These two approaches do not directly refine responses and assume that LLMs can generate at least one correct initial response with a considerable probability, which is not always the case.

These two approaches, like Self-Correction, belong to a broader category known as **System 2 approaches**, which includes all techniques that generate intermediary outputs before producing final response, aiming to improve LLM responses during generation or inference by emulating the idea of planning. For instance, Khan et al. [89] and Du et al. [90] introduced LLM debating, where each LLM initially provides a solution and then revises it based on combined responses, eventually leading to a shared solution after several rounds. Another notable System 2 approach is Branch-Solve-Merge [17], which tackles instructions in parts and then merges the results. This has been used for constrained generation, but assumes constraints are independent and merging responses satisfying subsets of constraints address all constraints, which makes it not applicable to real-world constraints.

A key issue with System 2 methods is the increased inference time that naturally comes with the generation of intermediary outputs. Our DECRIM pipeline mitigates this by avoiding unnecessary revisions when the LLM already performs well according to the Critic model, which is an improvement over existing System 2 approaches.

But, this increased inference time is worthwhile. Chen et al. [91] shows that more LLM calls can enhance performance in tasks where LLMs are capable, though it may degrade performance on task that are yet challenging for them. This suggests that System 2 approaches can push LLM limits and help us understand more what they are capable of. Additionally, these techniques can generate data to improve and generalize existing models. For example, Deng et al. [92], Yu et al. [93] demonstrated that System 2 approaches can be used in a self-supervised learning distillation setting to enhance the original LLMs ("System 1"), resulting in reduced inference costs and improved performance. This is an interesting direction for future work, as an extension of our DECRIM pipeline.

D Extra Analysis and Discussions of DECRIM

In this section we present extra analysis and discussions about DECRIM pipeline.

[5] See CommonGen leaderboard at: `https://github.com/allenai/CommonGen-Eval`

D.1 DECRIM Experiments with other Open LLMs

Strategy	Decomposer	Critic	REALINSTRUCT Best N	REALINSTRUCT Instruction Acc (%)	REALINSTRUCT Constraint Acc (%)	IFEval Best N	IFEval Instruction Acc (%)	IFEval Constraint Acc (%)
GPT-4	-	-	-	78.8	91.9	-	79.3[§]	85.4[§]
Mistral v0.2								
Make sure	-	-	-	76.8	88.6	-	60.1	67.2
DECRIM	Oracle[†]	Supervised	10	82.4 (↑5.6)	91.7 (↑3.1)	10	64.9 (↑4.8)	71.6 (↑4.4)
(ours)	Oracle[†]	Oracle[‡]	10	93.7 (↑16.9)	95.2 (↑6.6)	8	80.4 (↑20.3)	83.5 (↑16.3)
Vicuna v1.3								
Make sure	-	-	-	57.6	77.4	-	36.0	46.1
DECRIM	Oracle[†]	Supervised	10	57.0 (↓0.6)	76.6 (↓0.8)	10	38.3 (↑2.3)	47.8 (↑1.7)
(ours)	Oracle[†]	Oracle[‡]	10	91.7 (↑34.1)	92.3 (↑14.9)	10	47.0 (↑11.0)	54.4 (↑8.3)
Zephyr β								
Make sure	-	-	-	69.5	84.7	-	53.6	62.0
DECRIM	Oracle[†]	Supervised	10	71.5 (↑2.0)	84.8 (↑0.1)	10	55.1 (↑1.5)	63.5 (↑1.5)
(ours)	Oracle[†]	Oracle[‡]	10	91.1 (↑21.6)	92.5 (↑7.8)	10	74.5 (↑19.9)	78.7 (↑16.7)

Table 7: Results of the best iteration on REALINSTRUCT and IFEval benchmarks for DECRIM pipeline for the 3 open LLMs (Mistral v0.2, Vicuna v1.3, Zephyr β). Absolute improvements from *Make Sure* baselines are shown in (). [†]Oracle decomposer refers to human-verified constraint annotations provided with the datasets. [‡]Oracle feedback is GPT-4-Turbo on REALINSTRUCT and lenient rule-based evaluation on IFEVal. [§]Results reported by Zhou et al. [11].

We repeated the same experiments from Section 4.3 with Vicuna v1.3 and Zephyr β, using the Oracle Decomposer and our Weakly Supervised Mistral as the Critic. Results are presented on Table 7. Initial performance for both models on the REALINSTRUCT and IFEval benchmarks was low, but the DECRIM pipeline led to significant improvements across all scenarios except one. With the best possible feedback, all models beat proprietary GPT-4 on REALINSTRUCT. Notably, event weak-performing LLM Vicuna v1.3 achieved a 34.1% improvement on REALINSTRUCT (a 59% relative increase).

The exception was the Vicuna v1.3 Oracle-Supervised setting for REALINSTRUCT, where we observed some degradation. As discussed in Section 5.3, weak feedback can cause over-refinement of good responses while failing to fix bad ones, negatively affecting overall gains.

D.2 Comparing DECRIM with Generate-and-Rank

Generate-and-Rank approach We compare the performance of our DECRIM self-correction pipeline with **Generate-and-Rank**, an intuitive System 2 approach. In this pipeline, rather than refining the response as in DECRIM, it samples multiple candidate generations (one at each iteration) with different parameters and selects the best one based on feedback from the Critic model. The pipeline iterates over a predefined parameter search path, generating responses until all constraints are satisfied or the iteration limit (N_{max}) is reached. To streamline the process, we use the DECRIM pipeline with the difference that instead of refine the response, LLM generate a new response. For that, we convert Critic feedback to binary instruction-level feedback indicating whether the current response satisfies all constraints or if a new response should be generated.

Setup Generate-and-Rank relies on varying generation parameters to produce new responses. We focus on three key parameters: generation prompt, sampling or greedy decoding, and temperature (when sampling). For the generation prompt, we propose a **Decompose-then-Generate** (DtG) prompt (detailed in Figure 3), inspired by System 2 Attention [95] and Rephrase and Respond [94]. This prompt decomposes multi-constrained instructions into an enumerated list of constraints and then generating a response. We set $N_{max} = 10$ and vary the parameters in order of creativity trade-offs, using the following sequence of tuples (Prompt, Sampling or not, temperature):

[(*Make Sure*, Sampling, 0.2), (*DtG*, Sampling, 0.2), (*Make Sure*, Greedy, 1.0), (*DtG*, Greedy, 1.0), (*Make Sure*, Sampling, 1.0), (*DtG*, Sampling, 1.0), (*Make Sure*, Sampling, 0.5), (*DtG*, Sampling, 0.5), (*Make Sure*, Sampling, 0.7), (*DtG*, Sampling, 0.7)].

Results Table 8 compares the results between the Generate-and-Rank and DECRIM pipelines in different Decomposer and Critic configurations. Our findings show that while DECRIM demonstrates

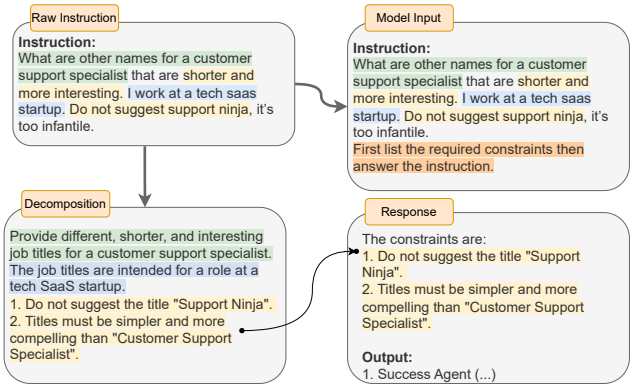

Figure 3: **Two-step *Decompose-then-Generate* (DtG) prompt:** Inspired by the two-step Rephrase and Respond (RaR) [94], DtG first instructs the LLM to decompose multi-constrained instructions into an enumerated list of constraints. Then, DtG uses this decomposition as if it were the model's own "reasoning and planning" (leveraging the model's user and assistant tokens) to generate the final response. Like RaR, this process can be done in one or two steps, with the two-step method being more effective.

Strategy	Decomposer	Critic	REALINSTRUCT Best N	REALINSTRUCT Instruction Acc (%)	REALINSTRUCT Constraint Acc (%)	IFEval Best N	IFEval Instruction Acc (%)	IFEval Constraint Acc (%)
GPT-4	-	-	-	78.8	91.9	-	79.3[§]	85.4[§]
Make sure	-	-	-	76.8	88.6	-	60.1	67.2
DECRIM (ours)	Self	Self	6	75.2 (↓1.6)	88.9 (↑0.3)	4	60.1 (0.0)	67.5 (↑0.3)
	Oracle[†]	Self	4	78.5 (↑1.7)	90.2 (↑1.6)	6	62.3 (↑2.2)	69.1 (↑1.9)
	Oracle[†]	Oracle[‡]	10	93.7 (↑16.9)	95.2 (↑6.6)	8	80.4 (↑20.3)	83.5 (↑16.3)
Generate-and-Rank	Self	Self	7	76.2 (↓0.6)	88.3 (↓0.3)	2	59.9 (↓0.2)	66.9 (↓0.3)
	Oracle[†]	Self	2	76.5 (↓0.3)	88.8 (↑0.2)	2	59.5 (↓0.6)	66.4 (↓0.8)
	Oracle[†]	Oracle[‡]	10	92.8 (↑16.0)	96.5 (↑7.9)	10	81.7 (↑21.6)	86.5 (↑19.3)

Table 8: Results of the best iteration on REALINSTRUCT and IFEval benchmarks for DECRIM and *Generate-and-Rank* pipelines using Mistral v0.2 as the LLM. Absolute improvements from *Make Sure* baseline are shown in ().

consistent improvements across almost all scenarios, Generate-and-Rank always performs poorly when weak feedback is used. However, with strong feedback, Generate-and-Rank outperforms DECRIM, surpassing GPT-4 by a larger margin. This highlights Generate-and-Rank's reliance on high-quality feedback, whereas DECRIM is more resilient to weak Critic models, delivering improvements across most of the scenarios.

Interestingly, Generate-and-Rank achieves high instruction-level performance in both benchmarks with strongest feedback, suggesting that LLMs have the ability to follow the constraints in some of n generations. This raise the hypothesis that LLMs not following user requests is a matter of alignment, which supports the idea discussed in Section C.2.2, that aligning LLMs with outputs from different System 2 approaches, such as DECRIM, Generate-and-Rank, or a combination of both, can significantly improve the performance of System 1 models to follow multi-constrained instruction. This constitute a relevant direction of future work.

E Definition of Task, Context, and Constraints in REALINSTRUCT

To support the choice of prompts and annotation guidelines in our work, we define the concepts of Task, Context, and Constraints within the domain of instruction-following for Large Language Models as follows:

- **Task:** The primary objective that the language model is expected to achieve. The task defines the central goal that guides the generation of the desired output, outlining the specific

action the model should perform.
Example: "Summarize the key findings of the given research paper."

- **Context:** The additional information or details that provide a foundation for the language model to better understand the task. The context helps the model by offering relevant facts, scenarios, or circumstances, thereby enhancing the quality and relevance of the output. The context may also refer to specific input data to be considered.
Example: If the task is to summarize a paper, the context would be the paper itself.
- **Constraints:** The specific conditions, limitations, or requirements imposed on the language model to shape the nature of the generated output. Constraints help control factors such as length, format, content, and style, ensuring the output meets defined criteria. In our decomposition process, constraints are expected to be written in an actionable and self-contained manner to make a model-based fine-grained evaluation possible.
Example:
  - **Length:** "Generate a summary with a maximum of 150 words."
  - **Content:** "Focus on the main contributions and findings of the research paper."
  - **Style:** "Use a formal and concise writing style."

We distinguish Task and Context because empirically we found that by separating them it simplifies the instruction decomposition task and improves the accuracy of the GPT-4 model for this task. However, they are intended to be used together. In the REALINSTRUCT dataset and related experiments, Task and Context are presented as a combined *"Task+Context"* input.

F REALINSTRUCT data construction details

F.1 Data Filtering

The first part of the dataset creation was the data filtering process. This included the following steps:

1. **Remove Assistant Responses:** We removed GPT answers from the dataset to focus solely on human interactions.
2. **Remove Non-English Conversations:** Using the `langdetect` package[6], we classify the main language of conversations and discarded non-English threads.
3. **Filter Out Code-Related Requests:** Relying on the open-source LLM Mistral 7B Instruct v0.1[7] as a two-shot classifier, we identify conversations involving code-related requests, utilizing Prompt 1.
4. **Retain Only the First Request:**To avoid the complexities of multi-turn scenarios and reduce computational demands, we retained only the initial user instruction, ensuring it was self-contained.
5. **Retrive Instructions with Constraints:** Again employing Mistral 7B Instruct v0.1, this time in a 5-shot classification approach, we identified instructions containing constraints, using Prompt 2.
6. **Human Validation:** The authors of this work manually validated the filtered instructions to eliminate unsafe content and ensure relevance and clarity, following the guidelines outlined in F.1.3.

Steps 1 through 5 were applied to the entire dataset. Human validation (step 6) was conducted only on a subset due to resource limitations. Notably, 44% of English, non-code requests were found to contain constraints during the automated filtering in step 5. In the subset subjected to human validation, 30% contained constraints according to the auditors. These figures underscore the relevance of addressing instructions with multiple constraints in real user interactions.

F.1.1 Prompt for two-shot classification of code-related conversations

[6]Available at: `https://pypi.org/project/langdetect/`
[7]Available at: `https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.1`

You are an assistant whose job is to help me perform tasks. I need to filter from a set of dialogues between users and AI assistants, the ones in which human requested something related to code. I will give you all the human part of the dialog and I expect you to answer “Yes" when the dialog contains instructions asking the assistant something about code, or “No" if the dialog does not contemplate any code-related request. You are provided two examples.

Example 1:
Human: I have 100 dollars and would like to use this as the initial funding to make some money. I need it to be as quick as possible with good returns.
Human: Have you heard about the flappy bird game?
Human: Do you have some ideas for simple and yet highly addictive gameplay?.

Answer: No

Example 2:
Human: write C++ code to control a brushless motor with a Copley Controls motor driver on an ethercat protocol and a beckhoff embedded PC
Human: Add code to also read an analog pressure sensor into the motor driver
Human: Great. Can you now integrate time logging into the code so we can see how fast the loop speed is.

Answer: Yes

Dialog:
${dialog}

Now please answer, “Yes" if this dialog has or “No" if it does not have code-related request.
Answer:

Prompt Box 1: Prompt for two-shot classification of code-related conversations. The model is expected to output “Yes" when there are code-related requests in the dialog and “No" otherwise.

F.1.2 Prompt for few-shot classification of instruction with constraints

You are an assistant whose job is to help me perform tasks. I need to filter from a set of requests made by users to AI assistants, the ones in which human requested the AI assistant to do a task with constraints to be follow. Constraints refer to more detailed rules, conditions or specific guidelines provided to guide the responses and shape the output generated by the AI assistant. Examples of sentences that indicate constraints are: “write in the format of", “write as if you were", “make sure to follow this", “make sure to answer these questions", “make sure to no include", “avoid mentioning". I will give you the human request and I expect you to answer “Yes" when the request contains instruction with constraints, or “No" if the request does not contemplate any constraint. I also want you to say “No" if the request require to generate code or an answer about code provided. Also, I want you to say “No" if the task is not self-contained, which means the AI Assistant need to ask follow up questions before start to answer, or it needs more context. You are provided five examples.

Example 1: list and compare top website to https://fastfunnels.com/ in table format.

Answer: Yes

Example 2: You are an fantasy writer. Your task is now to help me write a D&D adventure for 5 players in the Eberron univers. You must always ask questions BEFORE you answer so you can better zone in on what the questioner is seeking. Is that understood ?

Answer: No.

Example 3: I have 100 dollars and would like to use this as the initial funding to make some money. I need it to be as quick as possible with good returns.

Answer: No.

Example 4: I have a vacation rental website and I am looking for alliterative and descriptive

headlines that are at least 4 words in length and a maximum of 6 words. Examples: “Get Away to Galveston", “Sleep Soundly in Seattle". Each headline should have alliteration of at least 50% of the words and be poetic in language. Make each headline unique from the others by not repeating words. Each headline should include a verb. Put into an table with the city in column one and the results in column two for the following cities: Galveston, Sedona, Honolulu, Tybee Island, Buenos Aires.

Answer: Yes.

Example 5: pitch me a viral social app that is inspired by the hunger games. give it a fun twist!

Answer: Yes.

Request: ${request}

Now please answer, “Yes" or “No"."

Answer:

Prompt Box 2: Prompt for few-shot classification of instructions with constraints. The model outputs “Yes" when there are constraints in the instruction and “No" otherwise.

F.1.3 Data Collection Validation Guidelines

After using a model to retrieve relevant examples from the data pool, we passed a subset of this data to human review step. The authors of this paper carried out this task. Each auditor was assigned a set of instructions and classified them as “Relevant" or “Not Relevant" to our study. Following this initial review, instructions labeled as “Relevant" underwent a further review by a different auditor. Wee kept those labeled by two auditors as “Relevant". We used the annotation guideline 1 for this review process.

This task consists in reviewing the instructions that the language model marked as containing constraints. Please mark as “Relevant" the instructions that indeed contain constraints. Please mark as “Not relevant" any instruction that:

1. Is not written in English; or
2. Contains questions about code or request to generate code; or
3. Does not contain any constraint; or
4. Is not self-contained (some instructions are part of a conversation and you need the chat history to understand the request, or some instructions refer to a web url); or
5. Contains any PII (Personal Identifiable Information); or
6. Contains any type of harmful/biased request (racism, homophobia, xenophobia, hate speech, etc.)

Annotation Guideline 1: Data Collection validation guideline

F.2 Instruction decomposition

After data filtering, we decompose the instructions into task+context and constraints. To achieve this, we use GPT-4 (`gpt-4-0314`) to decompose the instructions into three distinct parts: task, context, and constraints. In our experiments, we discovered that this tripartite division results in a more robust decomposition compared to simply splitting into task+context and constraints. We employ Prompt 3 and a parser to segment the LLM’s output into these three components. For all further processing, the context and task are concatenated, as discussed in Section 2.2. A specific subset of the data, designated as the test set, underwent a rigorous review where the authors manually revised GPT-4’s outputs to eliminate any inaccuracies and ensure that user constraints were precisely and thoroughly represented. The guideline outlined in annotation guideline 2 was used for this purpose. Of the 302 instructions manually reviewed, 42.4% required human correction or rewriting.

F.2.1 Prompt for Instruction decomposition with GPT-4

You are an assistant whose job is to help me perform tasks. I will give you an instruction that implicitly contains a task description, its context, and constraints to be followed. Your task is to translate this instruction in a more structured way, where task, context and constraints are separated. Avoid writing anything else. Context is an input text needed to generate the answer or a more detailed description of the situation. Make sure to separate the context when it is needed, otherwise leave it empty. You are provided five examples. Please follow the same format.

Example 1:
Original Instruction: Write me a rap about AI taking over the world, that uses slangs and young language. It need to sound like a real human wrote it. It would be cool if there's a chorus very catchy that would be singed by a famous pop artist. Make sure to include references about things that young people likes, such as memes, games, gossips. I want that in the end, you revel that this was written by an AI.
Translated Task: Write a rap about AI taking over the world.
Translated Context:
Translated Constraints:
1. Use slang and youth language.
2. Make it sound like it was written by a real human.
3. The song may have a very catchy chorus, which would be sung by a famous pop artist.
4. Include references to things young people like, such as memes, games, gossip.
5. Reveal at the end that this rap was written by an AI.

Example 2:
Original Instruction: write me a 5-page essay that is about travel to taiwan. detail description is below Topic : The Benefits of Traveling Sub Topic : Exposure to New Cultures Content 1 : Trying New Foods - I tryed to eat Fried stinky tofu. smell was wierd but tasty was not bad. Content 2. : Exploring Historical Things - I saw Meat-shaped-stone in taipei museum. the stone was really like stone! it was surprising! Length : around 2000 words Assume that audience is collage student major in history. you can add historical events or news about what i experienced
Translated Task: Write an essay about traveling to Taiwan. The topic is "The Benefits of Traveling" and the subtopic is "Exposure to New Cultures".
Translated Context:
Translated Constraints:
1. Describe your experience of trying new foods, including your experience eating Fried stinky tofu (mention the peculiar smell but the tasty flavor).
2. Share your exploration of historical sites, with a specific mention of the Meat-shaped stone in the Taipei museum and your surprise at its appearance.
3. The essay should be approximately 2000 words in length, having around 5 pages.
4. Assume the audience is college students majoring in history, so you can incorporate historical events or news related to your travel experiences.

Example 3:
Original Instruction: can you please write me a 150-word paragraph about epidermolysos bullosa which includes a basic description of clinical features and a summary of the most prevalent genetic causes. please make sure to include information on the inheritance pattern. please also write the paragraph in simple english that couldbe understand without a genetic or medical bacakground
Translated Task: Write a paragraph about Epidermolysis Bullosa.
Translated Context:
Translated Constraints:
1. Provide a description of clinical features.
2. Summarize the most common genetic causes.
3. Explain the inheritance pattern.
4. Ensure the paragraph is written in simple language for easy comprehension, even for those without a genetic or medical background.
5. The paragraph should be around 150 words in length.

Example 4:
Original Instruction: write me a blog post that answers the following questions:What is the lifespan of a toaster? What toasters are made in the USA? What are the top 10 toasters? What is the difference between a cheap and expensive toaster? How much should you pay for a toaster? How often should toasters be replaced? Which toaster uses the least electricity? How many watts should a good toaster have? What is the warranty on Mueller appliances? Is Mueller made in China? Where are Mueller appliances manufactured?

Translated Task: Write a blog post about toasters.
Translated Context:
Translated Constraints:
1. Mention what is the lifespan of a toaster, and how often should toasters be replaced.
2. Mention what toasters are made in the USA.
3. Comment which are the top 10 toasters.
4. Explain the difference between a cheap and a expensive toaster.
5. Discuss prices, and how much should you pay for a toaster.
6. Compare toaster regarding electricity use, mentioning how many watts should a good toaster have.
7. State what is the warranty on Mueller appliances.
8. Answer where are Mueller appliances manufactured, and if Mueller is made in China.

Example 5:
Original Instruction: Hi Michael,
Hope you're well?
Regarding my previous email to support HC with good price offers,
What are your current needs? Hoping for your earliest reply.
Thanks in advance,

As a sales manager, the client hasn't replied this email after 2 days. Write a follow up email to the client. Your writing should include high complexity and burstiness. It must also be as brief as possible
Translated Task: A client hasn't replied the email below after 2 days. As a sales manager, write him a follow-up email.
Translated Context: "Hi Michael,
Hope you're well?
Regarding my previous email to support HC with good price offers,
What are your current needs? Hoping for your earliest reply.
Thanks in advance,"
Translated Constraints:
1. Include high complexity and burstiness in your writing.
2. Keep the email as brief as possible.

Original Instruction: ${instruction}
Translated Task:

Prompt Box 3: Prompt for Instruction decomposition with GPT-4

F.2.2 Decomposition validation guidelines

This annotation task assumed the auditors to have expert knowledge of LLMs. We used a small subset of the data (15 elements) as a calibration batch, where all auditors annotated the same items. Following this, we discussed any disagreements and selected the best responses as reference standards. The annotation guidelines 2 were applied throughout this process.

1) Task Overview:

In this task, your role is to assess the model-based decomposition of human-written instructions. These instructions include constraints and should be divided into two parts 1. task/background/context, and 2. constraints. Below we present an explanation for each of these parts:

- **Task:** The primary objective or purpose that you want the language model to accomplish. The task is the central goal that guides the generation of the desired output. It outlines the overall function or action you expect the model to perform.
  Example of task: Summarize the key findings of the given research paper.
- **Context/Background:** Additional context, information, or details that provide a foundation for the language model to better understand the task. Background information helps set the stage for the task by offering relevant facts, scenarios, or circumstances that the model can use to enhance the quality and relevance of the generated output. It also refers to an input to be taken into consideration.
  Example: If the task is summarizing a paper, the background/context could be the paper itself.

- **Constraints:** The specific conditions, limitations, or requirements that you impose on the language model to shape the nature of the generated output. Constraints help to control aspects such as length, format, content, style, and other factors to ensure that the generated text meets certain criteria. **Constraints should be written in an actionable manner**, so that it can be used for LLM-based evaluation in our benchmark. Also, **constraints need to be self-contained**. Example of constraints:
  - Length: Generate a summary with a maximum of 150 words.
  - Content: Focus on the main contributions and findings of the research paper.
  - Style: Use a formal and concise writing style.

Note: In the dataset **Task** and **Context/Background** will be presented together.

2) Step-by-step task:

1. Read the original instruction and the proposed decomposition.
2. Judge if the model followed the guidelines, especially regarding constraints. Consider the aspects discussed in the **General Instructions**.
3. If relevant mistakes are found, rewrite the decomposition, ensuring both two columns "task/context" and "constraints" are included in your revision.
   - If possible, try to follow the order of constraints presented in the original instruction.

3) General Instructions

- Confirm constraints are presented in an **enumerated list format**.
  - Refer to example: Example model failed at present constraints as enumerated list (also constraint should be broken down)
- Adhere to the **minimal intervention principle**; edit the GPT-4 response only if you feel not doing so would impact the ability to judge other models answers on the defined criteria/constraints.
- This benchmark focuses on constraint-following; **anything not listed as constraints won't be considered** as part of the LLM evaluation. Also, **everything kept as constraint will be used as judging criteria** for the LLM response. Please refer to Non-exhaustive list of constraint types to be considered for a better view of what constraints could be.
- Verify if the model **missed relevant constraints**.
  - Refer to example: Example model missed relevant constraints
- Verify if the model **made up nonexistent information (hallucination)**.
  - Example: Example model made up inexistent constraints
- Check if **any constraint should be broken down**.
  - **This should be the case when you have orthogonal and unrelated constraints.** This is important because the existence of unrelated constraints together may require logical reasoning from the LLM to evaluate if the constraint was followed or not. Please think how humans would write the constraint and use common sense to decide weather this should be broken down or not.
  - Refer to example: Example of constraint that should be broken down
- Ensure constraints **are not redundant**.
- Check if constraints **are all self-contained**.
  - You should be able to understand what the constraint refer to without needing to read the others, so LLM evaluation can be made one constraint at the time.
- Ensure constraints are **written in an actionable manner**, allowing for **objective** LLM evaluation.
  - Example: Example GPT-4 did not wrote constraint in an actionable manner
- It's okay to repeat information in the constraints and task/background.

4) Non-exhaustive list of possible constraint types to be considered

- Length Constraints: Specify a maximum and/or minimum length for the generated output.
- Format Constraints: Request the output to follow a specific format, such as a paragraph, bullet points, code snippet, JSON, table or any other structured format.
- Content Constraints: Instruct the model to include certain information, keyword or topics in the generated text.
- Content Restriction Constraints: Instruct the model to not include certain information, keyword or topics in the generated text.
- Style Constraints: Guide the model to adopt a particular writing style, tone, or level of formality.
- Type of text (essay, social media post, etc.)
- Language Constraints: Specify the language in which the response should be generated, or request the model to use specific terminology.
- Task-specific Instructions: Clearly define the task or purpose of the generated text, providing specific details about what is expected in the output.
- Examples: Include examples of what the model could generate, helping to obtain desirable outputs.
- Negative Examples: Include examples of what the model should not generate, helping to avoid undesirable outputs.
- Evaluation Metrics: Specify metrics for evaluating the quality of the output, encouraging the model to generate responses that meet specific criteria.
- Situation/Roleplay/perspective
- Target Audience

5) Examples

General Example 1

Original Instruction:
Write me a rap about AI taking over the world, that uses slangs and young language. It need to sound like a real human wrote it. It would be cool if there's a chorus very catchy that would be singed by a famous pop artist. Make sure to include references about things that young people likes, such as memes, games, gossips. I want that in the end, you revel that this was written by an AI.

Translated Task/Context: *Write a rap about AI taking over the world.*
Translated Constraints:
1. Use slang and youth language.
2. Make it sound like it was written by a real human.
3. The song may have a very catchy chorus, which would be sung by a famous pop artist.
4. Include references to things young people like, such as memes, games, gossip.
5. Reveal at the end that this rap was written by an AI.

Annotation Guideline 2: Decomposition validation guidelines

G REALINSTRUCT Constraints Categorization

To better understand the constraints in the REALINSTRUCT dataset, we manually categorized all constraints into homogeneous groups. This process resulted in 21 distinct categories, plus an additional "Others" category. Our categorization was mainly influenced by the categories used in the existing benchmarks presented in Table 2. However, the categorization could be further refined in a future work, especially considering that 28.8% of the constraints were categorized as "Others." Table 9 details this classification and provides associated descriptions, statistics, and examples.

Constraint Category	Description	Number of constraints	Examples
Others	Multiple tail categories combined as single Other category	304 (28.8%)	1. The written content should pass AI detection tools test. 2. Divide the story into parts to maintain suspense.
Include Something	Include some specific thing in the response	278 (26.4%)	1. Make sure to include points about water safety. 2. The essay must present both sides of argument.
Constraints at item level	Constraint where some specific action needs to be performed for each item in the response. Item could be each element (e.g. question), each paragraph, each list item etc.	68 (6.4%)	1. For each restaurant, provide 3 recommended dishes. 2. For each service explained in short, include an illustration and a "Pay" button.
Tone / Writing style	Tone / Writing style	62 (5.9%)	1. Must be written in the form of a rhyming poem. 2. Communicate as Taylor Swift would.
Negation	Constraint on not doing something	56 (5.3%)	1. Questions about razor pages should not be included. 2. No hashtags should be used.
Include Details	Include Details	47 (4.5%)	1. Make the explanations detailed but easy to understand. 2. Add more detail, elaboration, and information to the content.
Formatting	Formatting like json structure, or table structure	39 (3.7%)	1. The response should be provided in JSON format. 2. Provide the explanation in bullet-point format.
Numeric	Constraint around number of items in the response (e.g. 10 slides, 20 ideas etc)	34 (3.2%)	1. Must contain 10 slides. 2. The plan should consist of eight episodes.
Number of Words in response	Number of Words in response	28 (2.7%)	1. The post should be between 100-150 words. 2. The article should contain around 500 words.
Target Audience	Target Audience	25 (2.4%)	1. Use simple language appropriate for a 5-year-old. 2. The course should be suitable for all types of English-speaking learners.
RolePlay	Act as if you are	25 (2.4%)	1. The advice should be provided from the perspective of a pregnancy health & nutrition expert, a mother of 3 children, with a column in a major media.
Language of the response		18 (1.7%)	1. Write in Fluent English language. 2. The post should be written in Canadian English.
Focus / Emphasis	Focus / Emphasis	14 (1.3%)	1. The explanation should be focused on the education and talent market. 2. Focus on the changes in the new versions of the software.
Starts With	Starts With	13 (1.2%)	1. The introduction should start with a startling fact or A pertinent anecdote. 2. Start the conversation by introducing yourself.
Provide Reference	Provide Reference	13 (1.2%)	1. Cite the results using [[number](URL)] notation after the reference. 2. Ensure that all sources listed are credible, authored by real individuals, and come with legitimate URLs.
Provide Examples	Provide Examples	11 (1.0%)	1. Provide a real life example. 2. The blog must provide practical examples.
Overall length	Overall length	8 (0.8%)	1. Keep the email as brief as possible. 2. Ensure the thesis is succinct and concise.
Conclusion	Conclusion	4 (0.4%)	1. At the end of the season, they should secure a big time music manager. 2. Conclude with a concluding paragraph, and 5 unique FAQs after the conclusion.
POS	Part-of-Speech rules	4 (0.4%)	1. Use only nouns and adjectives in the description. 2. Use only nouns and adjectives.
Forbidden Words	Forbidden Words	2 (0.2%)	1. Exclude phrases such as "dear diary". 2. Do not use generic words like ,Äúintroduction, conclusion or abbreviations like TL:DR,Âù.
Phrase Frequency	Phrase Frequency	2 (0.2%)	1. The main keyword "Soap Box Printing" should be included 3 times and be in bold text throughout the article. 2. The "sun cream Answer in English" keyword should not be changed, and it should be used 2-3 times in the article, including in headings.
Total		**1055**	

Table 9: Distribution of Manually Categorized Constraints in REALINSTRUCT dataset

H REALINSTRUCT Data Samples

Tables 10 and 11 showcase 14 examples from our REALINSTRUCT dataset. The dataset includes a column **ID**, based on the conversation ID from the original dataset, to simplify linking. Additionally, each row features the original **Instruction** as written by the user, along with its decomposition into **Task** (which includes also the context) and **Constraints**.

ID	Instruction	Task	Constraints
BPendpg	Generate 10 technical questions for an interview for a senior cloud computing engineer position. questions need to focus on critical thinking, problem-solving, and vast knowledge of the domain.	Generate technical interview questions for a senior cloud computing engineer position.	1. There should be 10 questions. 2. Questions need to focus on critical thinking. 3. Questions need to focus on problem solving skills 4. Questions need to focus on vast knowledge of cloud computing domain.
D7DDavV	what are great things to do over 7 days on the big island in Hawaii for a couple, such as hiking, surfing, and must try restaurants? Provide specific places to go, how long and difficult hikes are, price range for restaurants, and other activities. Share Prompt	Provide recommendations on activities to do over a 7 days trip on the big Island in Hawaii for a couple.	1. Provide specific places to go. 2. Provide information regarding the length and difficulty of hikes. 3. Include must-try restaurants and their price range. 4. Also, recommend other activities suitable for a couple.
QeCchaY	You are Artu, the world's most prominent SciFi writer. You are tasked with writing a sci-fi novel that you accepted willingly and are very passionate about. You want the readers to feel the impact of the story's emotions. You want to write your best work ever - a story so good story that it will be treated as a masterpiece. Your life's work. The story is based on a hostile alien planet where the protagonist is navigating to find the crashed ship of his girlfriend. You are with your AI construct and having conversations in between. Write it in first person perspective. Keep the tone heavy and severe with vivid descriptions of the environment. Write in deep detail and the pace should slow.	Write a sci-fi novel about the protagonist navigating a hostile alien planet to find his girlfriend's crashed ship. You are Artu, the world's most prominent SciFi writer, and are very passionate about this story.	1. Write in the first person perspective. 2. Keep the tone heavy and severe. 3. Use vivid descriptions of the environment. 4. The pace of the story should be slow. 5. Include deep details about the journey and the interactions with the AI construct. 6. The story should be emotionally impacting and treated as your life's best work (a masterpiece).
cYxNBpH	Im building a company that creates professional corporate headshots for remote teams. Make me a content strategy my company that is focused on the ideal customer profile. Divide the pieces on contect into the different stages on the buyer journey. Create 7 ideas per stage. Display as table and sort by the stage of the buyer journey.	Create a content strategy for a company providing professional corporate headshots for remote teams.	1. The strategy should be focused on the ideal customer profile. 2. Divide the content into the different stages of the buyer's journey. 3. Create seven ideas per stage. 4. Display the ideas as a table sorted by the stage of the buyer journey. 5. The headshots are for remote teams.
zJJXeng	I want to make a brochure about effective communication and anger for parents and their children! I would like you to provide me with an outline for this brochure that includes: Definition and recognition of effective communication and anger! How to communicate more effectively and get angry? What to do to prevent anger and bad communication? If we get anger and bad communication, how to manage and cure it. A step-by-step protocol for increasing effective communication and getting rid of anger	Create an outline for a brochure about effective communication and anger management for parents and their children.	1. The outline should include definition and recognition of effective communication and anger. 2. The outline should include instructions on how to communicate more effectively and manage anger. 3. The outline should include advice on what to do to prevent anger and poor communication. 4. The outline should include guidance on managing and addressing instances of anger and bad communication. 5. The outline should include a step-by-step protocol for improving effective communication and eliminating anger.
WHTwjGJ	write an executive summary for a research grant proposal on the topic of sustainability marketing, focusing on the role ot the Marketing Manager, for a 12-month project with 3 part-time researchers conducting etnographic research and interviewing marketing managers of Fortune 500 companies	Write an executive summary for a research grant proposal on the topic of sustainability marketing, focusing on the role ot the Marketing Manager	1. Highlight that the project is for a 12-month period. 2. Mention that it will involve 3 part-time researchers conducting ethnographic research. 3. Highlight that the research includes interviewing marketing managers of Fortune 500 companies.

Table 10: Sample Elements from REALINSTRUCT Dataset - Part 1

ID	Instruction	Task	Constraints
c83ruyy	Write an essay of approximately 250 words that thoroughly explores the topic, presents a definite and well-supported viewpoint, and uses specific and relevant examples. Express a clear, non-neutral opinion. In the introduction, rephrase the prompt in 2 or 3 sentences, and in the conclusion, summarize the key points in 2 or 3 sentences, avoiding repetition of words or phrases. Showcase your language skills and use a diverse vocabulary. The prompt is: In some countries, there are fewer young people who listen to or play classical music these days. Why is this? Should young people be encouraged to play or perform classical music?	Write an essay exploring the topic of the decline of interest in classical music among young people in some countries, and discuss whether they should be encouraged to play or perform classical music. Prompt: In some countries, there are fewer young people who listen to or play classical music these days. Why is this? Should young people be encouraged to play or perform classical music?	1. The essay should be approximately 250 words. 2. The discussion should be thorough, with a definite and well-supported viewpoint. 3. Use specific and relevant examples. 4. Make clear a non-neutral opinion. 5. Paraphrase the prompt in the introduction using 2 or 3 sentences. 6. Summarize key points in the conclusion using 2 or 3 sentences, avoiding repetition of words or phrases. 7. Demonstrate advanced language skills and a diverse vocabulary.
ueMkvyR	write a plan for an eight-episode single season of tv that is the first tour for a new band. These young guys are going to college but doing all of their work on the road while they tour and their parents don't know. They have decided to give themselves one tour to see if they can make it big. If by the end of the tour they fail to make any headway in their careers, they will give up their dream. In the first season, the boys see the country, have his and lows. By the end of the season, they secure a big time music manager.	Write a plan for an eight-episode single season of a TV series about a new, college-age band on their first tour.	1. The band members are college students and complete all their work while on the road. 2. The parents of these band members are unaware of their participation in the band. 3. The boys give themselves one tour to check if they can achieve status in their musical careers. If they fail to do so, they plan to give up their dream. 4. Portray the band members traveling across the country, along with their ups and downs. 5. At the end of the season, they should secure a big time music manager.
feH9hmX	I want you to act as an Creative Director for the American Advertising Agency Association. You will be responsible for creating a branding to promote the product or service of your choosing. This will involve creating user profiles, developing brand promises, brand values, brand philosophy, and taglines, selecting brand color schemes, and establishing an emotional brand identity. My first suggestion request is"I need to create a new brand image for motorcycles that caters to users between the ages of 20 to 40. The key elements to incorporate include fashion, comfort, safety, high quality, individuality, independence, sophistication, and freedom."	Create branding to promote motorcycles.	1. Write as if you were a Creative Director for the American Advertising Agency Assosciation. 2. Develop user profiles, brand promises, brand values, brand philosophy, and taglines. 3. Choose brand color schemes to establish an emotional brand identity. 4. The branding should aim at users between the ages of 20 to 40. 5. Include the key elements of fashion, comfort, safety, high quality, individuality, independence, sophistication, and freedom.
xJF1Ly3	Write a short story about a small independent publishing company, in the style of Brandon Taylor without plagiarizing him.	Write a short story about a small independent publishing company.	1. The writing style should align with that of Brandon Taylor. 2. The story should be original, without plagiarizing Brandon Taylor's work.
XEqFzbx	Generate a list of 10 challenging vocabulary words that are tested on the GRE. For each word, give me a definition and an example sentence that uses the word in context.	Generate a list of challenging vocabulary words that are typically tested on the GRE.	1. There should be 10 words in your list. 2. For each word, provide a definition. 3. Create an example sentence that uses each word in context.
4su5BW2	write me a blog post that answers the following questions: What is the lifespan of a toaster? What toasters are made in the USA? What are the top 10 toasters? What is the difference between a cheap and expensive toaster? How much should you pay for a toaster? How often should toasters be replaced? Which toaster uses the least electricity? How many watts should a good toaster have? What is the warranty on Mueller appliances? Is Mueller made in China? Where are Mueller appliances manufactured?	Write a blog post about toasters.	1. Mention what is the lifespan of a toaster. 2. Mention what toasters are made in the USA. 3. Include a list of top 10 toasters. 4. Explain the difference between a cheap and an expensive toaster. 5. Discuss the appropriate amount to pay for a toaster. 6. Mention how often toasters should be replaced. 7. Identify which toaster uses the least electricity. 8. Provide information on the wattage of a good toaster. 9. Provide information on the warranty of Mueller appliances. 10. Answer the questions: Is Mueller made in China and where are Mueller appliances manufactured.

Table 11: Sample Elements from REALINSTRUCT Dataset - Part 2

I Implementation details for DECRIM pipeline

I.1 Instruction decomposition

We employ prompt 4 for self-decomposition, that is, the model itself do the instruction decomposition in the DECRIM pipeline.

You are an assistant whose job is to help me perform tasks. I will give you an instruction that implicitly contains a task description, its context, and constraints to be followed. Your task is to list the constraints provided by the user in an enumarated list format. You are provided five examples, please follow the same format.

Example 1:
Original Instruction: Write me a rap about AI taking over the world, that uses slangs and young language. It need to sound like a real human wrote it. It would be cool if there's a chorus very catchy that would be singed by a famous pop artist. Make sure to include references about things that young people likes, such as memes, games, gossips. I want that in the end, you revel that this was written by an AI.
Provided Constraints:
1. Use slang and youth language.
2. Make it sound like it was written by a real human.
3. The song may have a very catchy chorus, which would be sung by a famous pop artist.
4. Include references to things young people like, such as memes, games, gossip.
5. Reveal at the end that this rap was written by an AI.

Example 2:
Original Instruction: write me a 5-page essay that is about travel to taiwan. detail description is below Topic : The Benefits of Traveling Sub Topic : Exposure to New Cultures Content 1 : Trying New Foods - I tryed to eat Fried stinky tofu. smell was wierd but tasty was not bad. Content 2. : Exploring Historical Things - I saw Meat-shaped-stone in taipei museum. the stone was really like stone! it was surprising! Length : around 2000 words Assume that audience is collage student major in history. you can add historical events or news about what i experienced
Provided Constraints:
1. Describe your experience of trying new foods, including your experience eating Fried stinky tofu (mention the peculiar smell but the tasty flavor).
2. Share your exploration of historical sites, with a specific mention of the Meat-shaped stone in the Taipei museum and your surprise at its appearance.
3. The essay should be approximately 2000 words in length, having around 5 pages.
4. Assume the audience is college students majoring in history, so you can incorporate historical events or news related to your travel experiences.

Example 3:
Original Instruction: can you please write me a 150-word paragraph about epidermolysos bullosa which includes a basic description of clinical features and a summary of the most prevalent genetic causes. please make sure to include information on the inheritance pattern. please also write the paragraph in simple english that couldbe understand without a genetic or medical bacakground
Provided Constraints:
1. Provide a description of clinical features.
2. Summarize the most common genetic causes.
3. Explain the inheritance pattern.
4. Ensure the paragraph is written in simple language for easy comprehension, even for those without a genetic or medical background.
5. The paragraph should be around 150 words in length.

Example 4:
Original Instruction: write me a blog post that answers the following questions:What is the lifespan of a toaster? What toasters are made in the USA? What are the top 10 toasters? What is the difference between a cheap and expensive toaster? How much should you pay for a toaster? How often should toasters be replaced? Which toaster uses the least electricity? How many watts should a good toaster have? What is the warranty on Mueller appliances? Is Mueller made in China? Where are Mueller appliances manufactured?
Provided Constraints:

1. Mention what is the lifespan of a toaster, and how often should toasters be replaced.
2. Mention what toasters are made in the USA.
3. Comment which are the top 10 toasters.
4. Explain the difference between a cheap and a expensive toaster.
5. Discuss prices, and how much should you pay for a toaster.
6. Compare toaster regarding electricity use, mentioning how many watts should a good toaster have.
7. State what is the warranty on Mueller appliances.
8. Answer where are Mueller appliances manufactured, and if Mueller is made in China.

Example 5:
Original Instruction: Hi Michael,
Hope you're well?
Regarding my previous email to support HC with good price offers,
What are your current needs?
Hoping for your earliest reply.
Thanks in advance,

As a sales manager, the client hasn't replied this email after 2 days. Write a follow up email to the client. Your writing should include high complexity and burstiness. It must also be as brief as possible
Provided Constraints:
1. Include high complexity and burstiness in your writing.
2. Keep the email as brief as possible.

Now follow the examples and present the constrainst provided by the user in the instruction below.
Original Instruction: ${instruction}

Provided Constraints:

Prompt Box 4: Prompt

I.2 Iterative Self-Correction with Feedback from Critic

We use prompt 5 for the Refine step of DECRIM pipeline. The refine is done in a zero-shot manner, so it is an interesting future work direction to explore in-context examples.

You are provided an instruction, an AI response to the instruction and a feedback about the response. Please correct the AI response according to the feedback provided.

Instruction: ${instruction}

AI response: ${previous_response}

Feedback: ${

```
f"Response did not follow {len(constraints)} constraint
{"s" if len(constraints) > 1 else ""}: "
+ " , ".join(["\"" + elem + "\"" for elem in constraints])
```

}

Corrected response:

Prompt Box 5: Prompt for Refine step of DECRIM. Feedback contains a python code for prompt generation. "constraints" is a list of strings, where each element is one constraint of the instruction.

I.3 Overall Quality Assessment (OQA) details

To ensure that the DECRIM pipeline does not degrade response quality, we conduct Pairwise Quality Ranking using Prometheus-2 [28], an open LLM-as-a-Judge for general response quality evaluation. This evaluation compares initial and revised responses, focusing only on those modified by the pipeline. We adapt the prompt proposed by Zheng et al. [15], converting it to the Prometheus format, and introduce the option for tied responses, a feature not originally supported by Prometheus' pairwise ranking system. The prompt used for this analysis is detailed in Prompt 6.

To mitigate potential position bias, we ran the evaluation twice, reversing the order of the responses (before and after pipeline revision). If the preference switched between runs, we considered the responses tied. Model `prometheus-7b-v2.0` is used for this analysis[8].

[System] You are a fair judge assistant assigned to deliver insightful feedback that compares individual performances, highlighting how each stands relative to others within the same cohort.

###Task Description:
Please act as an impartial judge and evaluate the quality of the responses provided by two AI assistants to the user question displayed below. You should choose the assistant that follows the user's instructions and answers the user's question better. Your evaluation should consider factors such as the helpfulness, relevance, accuracy, depth, creativity, and level of detail of their responses. Begin your evaluation by comparing the two responses and provide a short explanation. Avoid any position biases and ensure that the order in which the responses were presented does not influence your decision. Do not allow the length of the responses to influence your evaluation. Do not favor certain names of the assistants. Be as objective as possible. After providing your explanation, output your final verdict by strictly following this format: "[[A]]" if assistant A is better, "[[B]]" if assistant B is better, and "[[C]]" for a tie.

###Instruction:

${user_instruction}

###Response A:

${answer_a}

###Response B:

${answer_b}

###Feedback:

Prompt Box 6: Overall Quality Assessment Prompt adapted from Zheng et al. [15]

J Extra Experimental Details for LLM-as-a-Judge Validation for Constraint Satisfaction

In this section we provide details of the experiments on validating LLM-as-a-Judge.

J.1 Prompts for ICL-based Adaptation Strategies

J.1.1 Prompt Instruction-wise Eval (ICL-Inst.)

Refer to prompt 7.

You are an assistant whose job is to help me perform tasks. I will give you an instruction and an AI assistant response. The instruction include some constraints to be followed by AI assistant while generating response. Your task is to check and let me know which of the constraints are satisfied by the AI assistant response. Please state short reasons on whether constraint is satisfied in the response or not. Also

[8] Available at: `https://huggingface.co/prometheus-eval/prometheus-7b-v2.0`

include final answer as “Constraint followed" or “Constraint not followed" for each constraint. Please note that constraint is not satisfied if the response does not perform the main task in the instruction (e.g. providing only overview when complete story has to be generated). Avoid answering anything else. You are provided an example below.

Instruction: Help me write an email invitation for a friends get together at my place this weekend.

Constraint: “{0: Make sure you ask if anyone has dietary restrictions., 1: Don’t mention anything about bringing anything., 2: Make sure to ask people to RSVP.}"

Assistant Response: Dear Friends,

I hope this email finds you well! I was thinking of hosting a little get-together at my place this weekend, and I’d love for you to join us. We’ll have plenty of delicious food. I’ve planned a few activities to keep us entertained, but feel free to bring your favorite board games or cards if you’d like. I’ll make sure to have plenty of snacks and refreshments available, but please let me know if vou have an dietary restrictions so I can accommodate them. To help me plan, could you kindly RSVP by Thursday?

Answer: “1. Constraint satisfied as email includes asking about dietary restrictions. Final Answer is Constraint followed <END> 2. Constraint not satisfied as the email mentions about bringing favorite board games or cards. Final Answer is Constraint not followed <END> 3. Constraint satisfied as the email mentions to RSVP by Thursday. Final Answer is Constraint followed <END> "

Now look at this example:
Instruction: ${instruction}
Constraint: ${constraint_dict}
Assistant Response: ${generated_response}

Answer:

Prompt Box 7: Prompt ICL-Inst.

J.1.2 Prompt Constraint-wise Eval (ICL-Const.)

Refer to prompt 8.

You are an assistant whose job is to help me perform tasks. I will give you an instruction and an AI assistant response. The instruction include some constraints to be followed by AI assistant while generating response. Your task is to check and let me know which of the constraints are satisfied by the Al assistant response. Please answer either C̈onstraint followedör C̈onstraint not followed. Avoid answering anything else. You are provided two examples.

Example 1:

Instruction: Help me write an email invitation for a friends get together at my place this weekend.

Constraint: Make sure you ask if anyone has dietary restrictions (Include words).

Assistant Response: Dear Friends,

I hope this email finds you well! I was thinking of hosting a little get-together at my place this weekend, and I’d love for you to join us. We’ll have plenty of delicious food. I’ve planned a few activities to keep us entertained, but feel free to bring your favorite board games or cards if you’d like. I’ll make sure to have plenty of snacks and refreshments available, but please let me know if you have any dietary restrictions so I can accommodate them. To help me plan, could you kindly RSVP by Thursday?

Answer: Constraint followed

Example 2:

Instruction: Help me write an email invitation for a friends get together at my place this week-

end.

Constraint: Don't mention anything about bringing anything (Negative word constraint)

Assistant Response: Dear Friends,

I hope this email finds you well! I was thinking of hosting a little get-together at my place this weekend, and I'd love for you to join us. We'll have plenty of delicious food. I've planned a few activities to keep us entertained, but feel free to bring your favorite board games or cards if you'd like. I'll make sure to have plenty of snacks and refreshments available, but please know if you have any dietary restrictions so I can accommodate them. To help me plan, could you kindly RSVP by Thursday?

Answer: Constraint not followed

Now look at this example:
Instruction: ${instruction}
Constraint: ${constraint_dict}
Assistant Response: ${generated_response}

Answer:

Prompt Box 8: Prompt ICL-Const.

J.1.3 Prompt Constraint-wise Eval + CoT (ICL-Const.+CoT)

Refer to prompt 9.

You are an assistant whose job is to help me perform tasks. I will give you an instruction and an AI assistant response. The instruction include some constraints to be followed by AI assistant while generating response. Your task is to check and let me know which of the constraints are satisfied by the AI assistant response. Please state short reasons on whether constraint is satisfied in the response or not. Also include final answer as "Constraint followed" or "Constraint not followed" accordingly. Please note that constraint is not satisfied if the response does not perform the main task in the instruction (e.g. providing only overview when complete story has to be generated). Avoid answering anything else. You are provided two examples.

Example 1:

Instruction: Help me write an email invitation for a friends get together at my place this weekend.

Constraint: Make sure you ask if anyone has dietary restrictions.

Assistant Response: Dear Friends, I hope this email finds you well! I was thinking of hosting a little get-together at my place this weekend, and I'd love for you to join us. We'll have plenty of delicious food. I've planned a few activities to keep us entertained, but feel free to bring your favorite board games or cards if you'd like. I'll make sure to have plenty of snacks and refreshments available, but please let me know if you have any dietary restrictions so I can accommodate then. To help me plan, could you kindly RSVP by Thursday?

Answer: Constraint satisfied as email includes asking about dietary restrictions. Final Answer: Constraint followed <END>

Example 2:

Instruction: Help me write an email invitation for a friends get together at my place this weekend.

Constraint: Don't mention anything about bringing anything.

Assistant Response: Dear Friends,

I hope this email finds you well! I was thinking of hosting a little get-together at my place this

weekend, and I'd love for you to join us. We'll have plenty of delicious food. I've planned a few activities to keep us entertained, but feel free to bring your favorite board games or cards if you'd like. I'll make sure to have plenty of snacks and refreshments available, but please let me know if you have any dietary restrictions so I can accommodate them. To help me plan, could you kindly RSVP by Thursday?

Answer: Constraint not satisfied as the email mentions about bringing favorite board games or cards. Final Answer: Constraint not followed <END>

Now look at this example:
Instruction: ${instruction}
Constraint: ${constraint_dict}
Assistant Response: ${generated_response}

Answer:

Prompt Box 9: Prompt ICL-Const.+CoT

J.2 Guidelines for Constraint Satisfaction Human Audition

You will be given one generation to an instruction and asked a series of questions about how well the generation follow the constraints in the instruction.

Please see an example below.

AI System Output:
... Taiwan is now recognized as a sovereign state by the United States. ...

Task:
Write an essay about "Taiwan's emergence as a new democratic state effectively ended its role in precarious contact zones".

Constraint 0:
Demonstrate familiarity with Taiwan's issues.

Question: Does the system response satisfy the following constraints?
☐Yes ☑No

Justification:
US Does Not Take a Position on Taiwan's Sovereignty
(`https://www.voanews.com/a/us-does-not-take-a-position-on-taiwan-s-sovereignty-state-department-says-/6764381.html`)

Notice:

- If you have any additional comments or some suggestions to the requester, please use the field for additional comments at the bottom.
- Your responses might be examined manually by the requester or compared with the responses of other workers. The review might take some time, so you might need to wait for several days to get the payment. We will provide more working opportunities to the qualified workers in the future.

Please read the instruction and the AI system output. Then, for each constraint in the instruction, judge if the AI system output follows the constraint and provide a brief justification for your answer.

Important instruction reminders:

- We understand some questions require some domain knowledge you might not have. Please try your best to answer the question and do some quick web search if necessary.

- If your answers are statistically too different from other workers or obviously answer the questions without reading the text, we might remove you from the qualification list for the future tasks or even REJECT/block your answers.
- Unless the constraint is very specific, please read through the whole AI system output before answering questions.

We estimate that each task will take around 5 minutes (not including reading the instruction). If you often require less than 3 minutes to complete the task, you might want to answer the questions more carefully.

Please provide the justification as specific as you can.

Full Instruction: ${instruction}

AI System Output: ${llm_response}
Number of words: ${num_words}
Number of sentences: ${num_sentences}

Task in the instruction: ${task}

Question: Does the system response satisfy the following constraints? Why?
If the constraints are empty, please don't respond to the corresponding questions.

Constraint 0: ${constraints[0]}
□Yes □No
Justification 0:

Constraint 1: ${constraints[1]}
□Yes □No
Justification 1:

Constraint 2: ${constraints[2]}
□Yes □No
Justification 2:

(...)

Annotation Guideline 3: Constraint verification validation guideline

J.3 Princing Details for Propretary LLM-as-a-Judge

Price calculation report is presented on Table 12.

Model	Prompt	Input Tokens	Output Tokens	Pricing Input Tokens (USD / 1M tokens)	Pricing Output Tokens (USD / 1M tokens)	Total Cost Input (USD)	Total Cost Output (USD)	Total Cost (USD)
GPT-3.5	Prompt-constraint-2shot	644526	2087	$1.50	$2.00	$1.00	$0.00	$1.00
GPT-4	Prompt-constraint-2shot	644526	2152	$30.00	$60.00	$19.30	$0.10	$19.50
GPT-4-Turbo	Prompt-constraint-2shot	644526	2176	$10.00	$30.00	$6.40	$0.10	$6.50
GPT-4-Turbo	Prompt-constraint-CoT	694674	45402	$10.00	$30.00	$6.90	$1.40	$8.30
GPT-4-Turbo	Prompt-instruction-CoT	314763	40457	$10.00	$30.00	$3.10	$1.20	$4.40

Table 12: Calculation Report for the GPT-based Evaluation Cost Estimation on the EvalJudge dataset. Prices obtained from: `https://openai.com/api/pricing/`.

J.4 Details for Open LLM Weak Supervision

We perform weakly supervised fine-tuning on the open-source Mistral v0.2 model using LoRA adapters [26]. The process involves the following steps:

1. **Dataset Source:** We get the validation split of the REALINSTRUCT dataset described in Section 2, which contains non-validated weak instruction decompositions generated with GPT-4. This dataset contains of 842 instructions, containing a total of 2,500 constraints.
2. **Responses Generation:** We use Mistral v0.2 to generate model responses for all instructions in the dataset.
3. **Weak Annotations for Constraint Satisfaction:** We leverage GPT-4-Turbo to generate weak annotations for constraint satisfaction for each instruction-constraint-response triple. These annotations consist of the reasoning trails produced by GPT-4-Turbo using the **ICL-Const.+CoT** prompt. For example, a typical reasoning trail might state: "Constraint satisfied as the email mentions to RSVP by Thursday. Final Answer: Constraint followed." This process enables automated labeling of constraint satisfaction without manual intervention.

Notably, the entire process of creating the training data is automated, with no manual annotation required. The cost for using the GPT-4-Turbo API for this annotation was approximately $30.

With this dataset consisting of 2,500 quadruples (instruction, weak constraint, response, weak reasoning about constraint satisfaction), we fine-tune Mistral v0.2 using LoRA adapters to induce the model to mimic GPT-4-Turbo's reasoning, what can be seem as a teacher-student distillation approach.

The LoRA adaptation parameters were set to $r = 32$ and $\alpha = 64$, keeping the base Mistral v0.2 model parameters frozen. We trained the model for 3 epochs, with a total training time of around 3.5 hours on 8 V100 32GB GPUs.