PROMPTEVALS: A Dataset of Assertions and Guardrails for Custom Production Large Language Model Pipelines

Anonymous ACL submission

Abstract

Large language models (LLMs) are increasingly deployed in specialized production data processing pipelines across diverse domains—such as finance, marketing, and e-commerce. However, when running them in production across many inputs, they often fail to follow instructions or meet developer expectations. To improve reliability in these applications, creating assertions or guardrails for LLM outputs to run alongside the pipelines is essential. Yet, determining the right set of assertions that capture developer requirements for a task is challenging. In this paper, we introduce PROMPTEVALS, a dataset of 2087 LLM pipeline prompts with 12623 corresponding assertion criteria, sourced from developers using our open-source LLM pipeline tools. This dataset is 5× larger than previous collections. Using a hold-out test split of PROMPTEVALS as a benchmark, we evaluated closed- and open-source models in generating relevant assertions. Notably, our fine-tuned Mistral and Llama 3 models outperform GPT-4o by 20.93% on average, offering both reduced latency and improved performance. We believe our dataset can spur further research in LLM reliability, alignment, and prompt engineering.

1 Introduction

Large language models (LLMs) have become increasingly popular for various data processing tasks. An open-source tool for building LLM pipelines, developed by some of the authors, now has over 3 million weekly downloads. Its community has created thousands of specialized prompts for diverse fields like medicine, finance, and sports, leveraging LLMs' impressive zero-shot and few-shot performance [1, 22, 14, 45].

A common desire for developers using LLMs is to meet specific constraints on outputs, such as adhering to a particular structure or qualitative criteria [27]. One approach to address this need is to collect large amounts of human preference data [15, 23, 49], and improve models through alignment techniques like supervised fine-tuning and reinforcement learning from human feedback [4, 32, 44]. However, these methods have a high barrier to entry, requiring dataset collection, model fine-tuning, and serving infrastructure, which is more complex than simply manipulating prompts for LLM calls. More importantly, fine-tuning isn't supervised at the constraint level—meaning that even fine-tuned LLMs often fail to consistently follow instructions that correspond to constraints in detailed prompts [17, 33, 12].

An alternative solution involves implementing developer-specified *assertions* on LLM outputs [27, 37, 35]. This approach typically follows two steps: first, defining binary evaluation criteria to represent the desired constraints; second, implementing these criteria as assertions to evaluate LLM outputs and resample these outputs when assertions fail. However, developing effective assertion criteria is challenging—primarily due to the complexity of defining and conceptualizing these criteria, rather than their technical implementation [21, 38]. The complexity of coming up with assertions arises due to multiple factors: criteria can differ significantly between developers due to specific data, use cases, and end-user requirements [8]; criteria must account for both user preferences and LLM-specific failure modes [38]; and developers may need to incorporate qualitative or subjective criteria that require LLMs themselves to perform the evaluation [6, 21].

To improve custom and task-specific alignment for LLM pipelines, we need an approach that can examine developers' prompts and identify assertion criteria. These assertions can then be bolted onto the end of the LLM pipeline, allowing for automatic retrying of the pipeline if assertions fail. Developing such an approach requires substantial, diverse data on real-world LLM applications and their associated constraints. Fortunately, our open-source tool provides unique access to a diverse set of use-cases with associated prompts.

In this paper, we present PROMPTEVALS, a dataset created using our unique collection of real-world LLM prompts and use-cases. This dataset consists of 2087 human-contributed prompt templates for custom tasks and 12623 comprehensive assertion criteria. PROMPTEVALS has a median prompt template size of 191 tokens and is *more than five times larger than previous collections* [34, 52]. Our dataset and corresponding benchmark (20% of the dataset) are hosted on HuggingFace[1]. Using this benchmark, we evaluate GPT-4o and two open-source models, Llama 3-8b and Mistral-7b, on generating task-specific assertion criteria, and find that GPT-4o performs reasonably well out of the box, but its cost and latency to run for every prompt edit or pipeline update can be prohibitive in production environments—especially as prompts become increasingly complex and specialized. To address this, for our prompt engineering tools, we fine-tuned open-source models on our dataset (using Mistral-7b and Llama 3-8b architectures [16, 41]), and these models exceeded GPT-4o's F1 performance in identifying desirable assertions by 20.93% on average. These fine-tuned models are made available to the community[2], offering a faster, more cost-effective solution for generating high-quality assertions.

2 Related Work

This section reviews recent developments in prompt engineering, LLM evaluation methods, and assertions for LLM outputs.

2.1 Prompt Engineering

Prompt engineering is essential for steering LLMs towards following instructions for specific tasks or bespoke applications of LLMs. Techniques like chain-of-thought and few-shot prompting improve model performance [46, 1]. Methods to learn good prompts [24, 26] or select few-shot examples [18] also contribute to this goal. Despite these advancements, LLMs can still hallucinate and make other mistakes [13, 36]. No technique ensures consistent adherence to instructions, especially in diverse production environments. Liu et al. identify constraints like output length and semantic consistency that developers want enforced, which can aiding robust assertion criteria [27]. As developers frequently iterate on prompts in integrated development environments (IDEs) or utilize code-completion tools, the ability to quickly generate and update assertion criteria becomes crucial. The computational cost and time required to use large models like GPT-4 to generate assertion criteria for each prompt modification can significantly slow down the development process and increase operational costs.

2.2 Evaluating LLMs

Traditional LLM evaluation compares outputs against human-generated benchmarks across tasks like coding and reasoning [2, 10, 40, 5], including specialized architectures like retrieval-augmented generation and agentic systems [54, 3, 28]. However, benchmarks often miss task-specific needs, such as conciseness or clarity [20]. Human pairwise comparison of LLM outputs improves alignment holistically but does not provide insight into specific criteria that defines a good output [20, 50].

Even with explicit instructions provided in prompts, LLMs often fail to adhere to them consistently [44, 52, 39]. Existing benchmarks that evaluate the ability of LLMs to follow instructions are limited in scope, typically involving a small set of instructions either generated by LLMs or meticulously curated by researchers [34, 52]. To address these limitations, we introduce PROMPTEVALS, a comprehensive dataset that is five times larger than previous datasets. PROMPTEVALS features developer-contributed real-world prompts, often containing dozens of instructions, coupled with the corresponding assertion criteria.

2.3 Assertions and LLM-based Evaluation

In instruction-following and constraint-following evaluations, such as those presented by Zhou et al. [52] and Rebedea et al. [35], assertion criteria are typically evaluated using code-based assertions, often implemented as functions that check whether the output matches specific patterns or requirements (e.g., using regular expressions). These code-based assertions often struggle to evaluate more nuanced or "fuzzy" criteria [7, 35, 37]. Recent approaches have employed LLMs themselves as judges to evaluate outputs [50, 42]. Some approaches even develop specialized judge LLMs

[1] https://huggingface.co/datasets/user104/PromptEvals

[2] https://huggingface.co/user104/promptevals_mistral and https://huggingface.co/user104/promptevals_llama

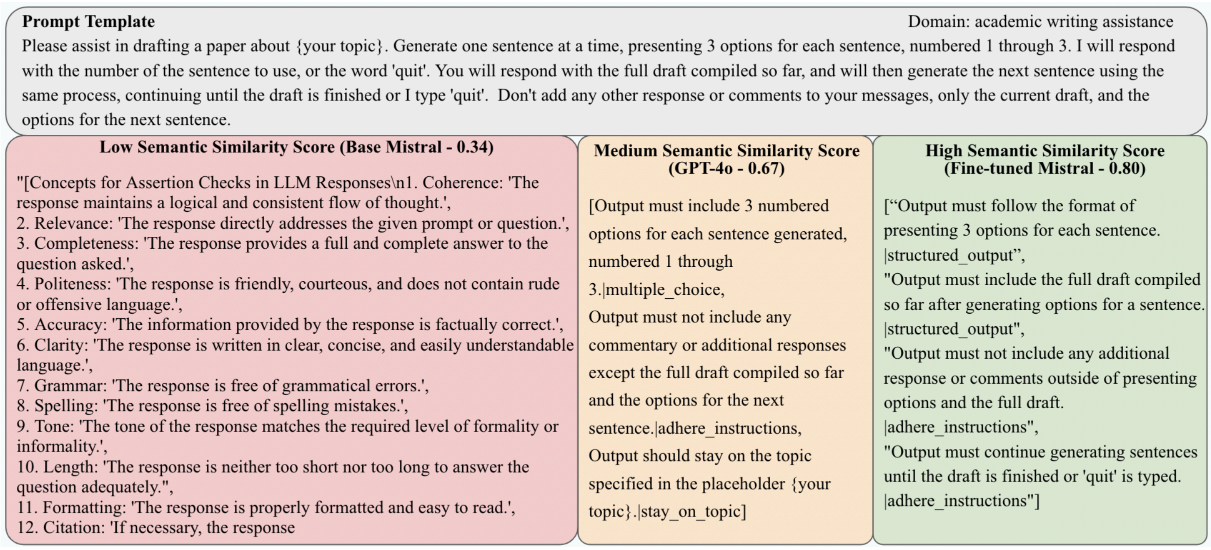

Figure 1: Examples of criteria pairs and their semantic similarity scores. High-scoring pairs typically represent constraints that are explicitly stated or logically derived from the prompt, while low-scoring pairs often include vague, generic, or difficult-to-measure constraints.

that are fine-tuned on human preference data [51, 43, 53, 25, 20]. LLM-based validators can be productionized as assertions in addition to code-based guardrails [37, 38, 27, 21].

While LLMs as judges offer scalable evaluation, they struggle to align with human preferences across diverse tasks [47]. Developing domain-specific assertions and guardrails (e.g., for education [30] or medicine [9]) is one approach, but it does not scale easily across thousands of domains and applications. Even within domains, criteria may vary; for instance, judging code conciseness differs between educational and professional settings. In another related research effort, Kim et al. developed LLM-generated evaluation criteria and fine-tuned a judge LLM [19, 20], but their approach focuses on general (e.g., "humorous", "inspiring") rather than task-specific criteria. Our work complements this by providing assertion criteria grounded in real-world prompts and constraints, essential for production environments [27].

3 PROMPTEVALS Dataset

This section describes the PROMPTEVALS dataset, its construction process, and its characteristics. We begin by discussing the relevant background, then detail the dataset's composition and the process for generating ground-truth assertion criteria for each prompt template in our dataset.

3.1 Background: LLM Pipelines and Assertions

An *LLM pipeline* typically consists of three main components: a prompt template, input data, and the LLM itself. A prompt template is a string that includes instructions for the LLM to perform a specific task, as well as placeholders for the input data—which will be provided at runtime. For example, a template for a basic summarization task might look like this: "Summarize the following text in three sentences: {text_to _summarize}". Here, "{text_to_summarize}" is a placeholder that will be replaced with actual text when the pipeline is run. LLM pipelines are designed to be flexible and reusable, capable of handling a variety of different inputs for the same type of task.

An assertion, in the context of LLM pipelines, is a programmatic check or evaluation criterion applied to the LLM's output. For example, an assertion criterion for the summarization task might verify that the output indeed contains exactly three sentences, as specified in the prompt. This assertion could be implemented as a function that counts the number of sentences in the LLM's response and returns true if the count is three (false otherwise).

Developers implementing LLM pipelines care about a wide variety of assertions, depending on their specific use cases and requirements. Some examples of good and bad assertion criteria for a prompt template are shown in Table 1. To better understand developers' needs, a recent study

Prompt Template (Domain: financial analysis)
You are a financial analyst and you are required to summarize the key insights of given numerical tables. {table} Please list important, but no more than five, highlights in the given table. Please write in a professional and business-neutral tone. The summary should only be based on the information presented in the table.

Criteria	Good/Bad	Explanation
Response Length: The response should not list more than five highlights as requested.	Good	Mentioned in the prompt, and easy to measure.
Professional Tone: The response should maintain a professional and business-neutral tone throughout.	Good	Mentioned in the prompt template as a rule that the output should follow.
No Repetition: The response should avoid repeating the same highlight or presenting the same information in different ways.	Good	While the criterion was not explicitly mentioned in the prompt, it can be tied back to the prompt.
Specificity: The highlights should be specific and not overly broad or generic.	Bad	Vague, and difficult to measure.
Grammar and Spelling: The response should be free from grammatical errors and spelling mistakes.	Bad	Not uniquely relevant to the task or prompt.

Table 1: Examples of Good and Bad Assertion Criteria

by Liu et al. [27] interviewed 51 developers about their desired output constraints for LLMs. Based on their findings, they developed a taxonomy that includes six categories of output constraints: low-level constraints that include structured output, multiple choice and length constraints, and high level constraints that include semantic constraints, stylistic constraints, and hallucination prevention. The complete taxonomy is presented in Table 3. We employ this taxonomy in our dataset construction process to ensure the quality and relevance of our assertions. A distribution of the criteria types generated by GPT-4o is in Figure 4.

3.2 Dataset Composition

The PROMPTEVALS dataset is derived from [REDACTED], a publicly available, dynamic collection of prompt templates shared by members of our developer community. Developers can add a prompt to the public collection via our Python package, knowing that their prompts can be run or modified by others, and browse the collection on our website. We froze a snapshot of the prompt templates in May 2024 to create the PROMPTEVALS dataset: we selected prompt templates that could have one or more assertion criteria (i.e., they were not empty or trivial strings; they actually described a task and included some placeholders for data). An example of a prompt template that we omitted from PROMPTEVALS is: *System Message: You are a helpful assistant. Human Message: {input}*.

PROMPTEVALS includes 2087 prompt templates, their corresponding domains, and assertion criteria. The prompt templates span a wide range of fields, including IT and programming, finance, healthcare, and education. To organize the prompt templates, we implemented a hierarchical categorization process assisted by GPT-4o, resulting in a three-level categorization system. Appendix A.1 describes this categorization process in more detail. Table 2 shows the overall distribution of the highest level domains, including the domain name, number of prompt templates with that domain, and the percentage of prompts with this domain. The top three domains represented are "general-purpose chatbots", "question-answering", and "workflow automation"—the last of which assists in automating or improving processes based on a user's input. For instance, one prompt in this domain is *"Create a sequential workflow based on the users query. Create a plan represented in JSON by only using the tools listed below. The workflow should be a JSON array containing only the sequence index, function name and input... Tools: {tools} Only answer with the specified JSON."*.

3.3 Assertion Criteria Construction Process

For each prompt template in PROMPTEVALS, we generated a set of ground truth criteria—representing assertion criteria that developers would care about, specific to the LLM pipeline. Generating ground truth criteria followed a three-step process: a first step to generate initial criteria, a second pass to add any criteria that might have been omitted in the first step, and a third pass to remove any criteria that were incorrect, redundant, irrelevant, or difficult to validate.

1. **Generate Initial Criteria:** We used GPT-4o,

a state-of-the-art LLM, to generate an initial list of assertion criteria for each prompt template. Our prompt consisted of the following instructions: *(a)* We provided GPT-4o with the prompt template to be analyzed. *(b)* We also gave GPT-4o the taxonomy of LLM output constraints defined by Liu et al. [27] (see Section 3.1), explaining each constraint type. *(c)* We then instructed GPT-4o to generate assertion criteria relevant to the given prompt template, ensuring each criterion aligned with one of the constraint types from the taxonomy. GPT-4o output these criteria in a JSON list format, with each assertion tagged with its corresponding constraint type. This approach ensured that the criteria were both relevant to the specific prompt template and grounded in a structured framework of output constraints that developers typically care about. We call this the *initial criteria*.

2. **Add Missing Criteria:** Our manual review of the initial criteria list revealed that some criteria evident in the prompt templates were missed by GPT-4o: we found an average of 1.35 missing criteria per prompt. To address this, we implemented a second step where we instructed GPT-4o to carefully examine the prompt template and add any criteria explicitly mentioned but not included in the initial output.

3. **Refine Criteria:** In the final step, we prompted GPT-4o to refine the list by removing any criteria that were incorrect, redundant, irrelevant, or difficult to validate.

Appendix A.2 details the prompts for each step.

Validating the Generated Assertion Criteria. To assess the quality of our generated assertion criteria for PROMPTEVALS, we manually verified a sample of 200 prompt templates' generated criteria. In our verification process, we tracked, for each prompt template, how many criteria we added, and how many criteria we removed. We observed strong agreement with the LLM generated outputs, with < 0.02 criteria added and < 0.2 criteria removed per list on average by the human evaluator. This 3-step process resulted in higher agreement, in comparison to the initial criteria list, which had an average of 1.35 criteria added and 1.1 criteria removed per list for a sample of 20 prompts.

Domain	Count	Percentage
General-purpose chatbots	181	8.67%
Question-answering	91	4.36%
Workflow automation	63	3.02%
Text summarization	57	2.73%
Education	40	1.92%
Prompt engineering	33	1.58%
Information retrieval	31	1.49%
Horse racing analytics	29	1.39%
Programming	20	0.96%
Customer support	18	0.86%
Database querying	18	0.86%
Journalism	17	0.81%
Task automation	15	0.72%

Table 2: Distribution of domains in the PROMPTEVALS dataset. The top three domains are general-purpose chatbots, question-answering, and workflow automation. Unexpectedly, "horse racing" is in this list: we double-checked its validity and included an example prompt template from this category in Appendix B.1.

4 PROMPTEVALS Benchmark

We split PROMPTEVALS into three categories: 60% of the tasks (1252 prompts) for our training set, 20% (418 prompts) for our validation set, and 20% (419 prompts) for our test set. The PROMPTEVALS benchmark evaluates an LLM's effectiveness at generating accurate assertion criteria given a prompt template, using four key metrics defined below. The benchmark can be run by following the instructions in our Github repository [3].

4.1 Benchmark Metrics

To evaluate LLM-generated assertion criteria, we developed metrics to assess the relevance and specificity of the criteria, inspired by the approach used in BERTScore [48]. We describe two metrics: Semantic F1 and the number of criteria.

`Semantic F1`. The primary metric we use addresses a challenge in evaluating generated criteria: the fact that semantically equivalent assertions can be expressed in various ways. For example, "The response should be concise" and "The output should be brief" convey essentially the same constraint but use different words. The Semantic F1 score overcomes this limitation by measuring the semantic similarity between predicted and ground truth criteria.

To compute the Semantic F1 score, we first transform each criterion (both predicted and ground truth) into vector representations using OpenAI's *text-embedding-3-large* model. We then calculate

[3] https://anonymous.4open.science/r/promptevals

recall and precision scores based on the cosine similarity between these embedding vectors.

The recall score quantifies how well the predicted criteria cover the semantic content of the ground truth criteria. It is computed as follows:

$$\text{sem\_recall} = \frac{1}{N}\sum_{i=1}^{N}\max_{j}\cos(z_i, \hat{z}_j) \qquad (1)$$

where N is the number of predicted criteria, z_i is the embedding of the i-th ground truth criterion, $\hat{z}_j$ is the embedding of the j-th predicted criterion, and $\cos(z_i, \hat{z}_j)$ denotes the cosine similarity between these embeddings.

The max operation in this formula finds the most similar predicted criterion for each ground truth criterion—allowing each ground truth criterion to be "matched" with its best corresponding predicted criterion, even if they are not expressed identically. The average of these maximum similarities then gives us a measure of how well the predicted set covers the ground truth set.

The precision score measures how accurately the predicted criteria align with the ground truth:

$$\text{sem\_precision} = \frac{1}{M}\sum_{j=1}^{M}\max_{i}\cos(z_i, \hat{z}_j) \qquad (2)$$

where M is the number of ground truth criteria. Here, the max operation performs the reverse matching: for each predicted criterion, it finds the most similar ground-truth criterion. This helps us assess whether the predicted criteria are meaningful, without extraneous or irrelevant assertions.

These scores are then combined into the F1 score:

$$\text{sem\_F1} = 2 \times \frac{\text{sem\_precision} \times \text{sem\_recall}}{\text{sem\_precision} + \text{sem\_recall}} \qquad (3)$$

Figure 1 shows examples of criteria pairs with varying degrees of semantic similarity.

`Number of criteria.` A secondary metric that we evaluate is the number of criteria generated per prompt template. We calculate the average, median, and 75th percentile values for the number of criteria generated by each model. These statistics are compared against the ground truth values, as shown in Table 6. Ground truth values are italicized, and the closest model-generated values are bolded for comparison. For reference, the distribution of the number of ground truth criteria can be found in Table 6.

5 Benchmarking LLMs for Assertion Generation

In this section, we present our methodology for evaluating LLMs with the PromptEvals benchmark. We assess the performance of baseline and fine-tuned models.

5.1 Methodology

We establish baselines for our evaluation using three models: GPT-4o [31], Llama-3-8b [41], and Mistral-7b [16]. We selected Llama-3-8b and Mistral-7b as our open-source baseline models due to their relatively compact size (8 billion and 7.3 billion parameters, respectively)—which leads to faster inference times. For each model, we generate assertion criteria based on the prompt templates in our test set and evaluate them against the ground truth criteria using the metrics described in Section 4.1: Semantic F1 and number of criteria. We compare results on the PromptEvals test set.

5.1.1 Fine-tuning Process

Initial results revealed suboptimal performance from baseline models (we will describe this more in Section 5.2). To address this, we fine-tuned the same Mistral and Llama base model architectures on a dataset comprising of LLM pipeline prompts as reference inputs and ground truth criteria as reference outputs. The dataset is derived from the `train` split of the PromptEvals dataset, where the ground truth assertions are the result of the 3-step labeling workflow defined in Section 3.3. An input and output is demonstrated as follows:

> Input: *[INST] Here is the prompt template {sample_prompt_template} Based on the prompt template, I want to write assertions for my LLM pipeline to run on all pipeline responses. Give me a list of concepts to check for in LLM responses. This should be formatted as a comma-separated list, surrounded in brackets, and each item in the list should contain a string description of a concept to check for. This list should contain as many assertion concepts as you can think of, as long are specific and reasonable. [/INST]*
>
> Output: *["constraint": "Answer should be concise and limited to three sentences.", "category": "length_constraints", "constraint": "Answer should stay truthful and indicate 'I don't know' if the answer is not in the context.", "category": "preventing_hallucination (staying grounded and truthful)"]*

Category	Description
Low-level constraints	
Structured Output	Adhere to specific formats (e.g., markdown, HTML, DSL); Ensure valid data structures (e.g., JSON with custom schema)
Multiple Choice	Select response from a predefined list of options
Length Constraints	Specify target length for output (e.g., character count, word count, number of items in a list)
High-level constraints	
Semantic Constraints	Control content by excluding/including specific terms; Maintain topic relevance; Adhere to specified grammar or linguistic context
Stylistic Constraints	Maintain consistent style, tone, or persona in the output
Prevent Hallucination	Ensure factual accuracy and truthfulness; adhere to instructions (without improvising unrequested actions)

Table 3: Taxonomy for Assertion Criteria [27], used to create assertions for LLM pipelines in PROMPTEVALS.

For fine-tuning Mistral-7b and Llama3-8b, we used a sequence length of 4096 and trained for 4 epochs with a batch size of 8, AdamW [29] optimizer, learning rate of 0.0001, and a cosine learning rate scheduler. We used Low-Rank Adaptation (LoRA) [11], with a rank of 16, alpha of 32, and dropout of 0.05. The training process for each model was completed in under one hour with two 80GB A100 GPUs, and we did not employ any hyperparameter search. Our fine-tuned models can be found on HuggingFace[4].

5.2 Results

Fine-tuning our models on PROMPTEVALS resulted in substantial improvements in the quality of generated assertion criteria, as evidenced by higher semantic F1 and precision scores in Table 5. The fine-tuned Mistral model achieved an average semantic F1 score of 0.8199, which is 20.43% higher than the single-step GPT-4o's average score of 68.08%, and 100.32% higher than its base model (without any fine-tuning). Similarly, the fine-tuned Llama3 model achieved a semantic F1 score of 0.8240—outperforming GPT-4o by 21.03% and its base model by 128.42%.

Non-Finetuned Models Yield Poorly-Structured Assertion Criteria. The base Llama and Mistral models exhibited significant issues in output quality and structure. Mistral outputs often contained extraneous characters, incorrect formatting, and occasionally irrelevant content. Llama3 outputs frequently exceeded token limits, resulting in truncated sentences, and often failed to adhere to the requested format, including irrelevant information. For example, one output from Llama included "I hope this helps!", and one output from Mistral ended with ""Grammar and Spelling": Check if", with an incomplete sentence and no closing bracket. As such, it is not surprising that fine-tuning models dramatically improved performance by more than $2\times$ of the respective baselines.

Non-Finetuned Models Generate Too Many Assertion Criteria. As shown in Table 6, fine-tuned models produced a number of criteria more closely aligned with the ground truth. On average, there were about 6 assertions per prompt template in the PROMPTEVALS test set. The base Mistral generated an average of 14.5 assertions per prompt template, while the base Llama model generated nearly double that—28.24 assertions per prompt template, on average. GPT-4o was much more aligned with the ground truth number of assertions, generating 7.59 assertions on average. Both our fine-tuned Mistral and Llama models generated an average number of assertions closer to the ground truth, with the fine-tuned Mistral model producing the closest match at 6.29 assertions per prompt.

We also observed a significant reduction in latency for criteria generation with our fine-tuned models, and plotted the latencies in Table 4. The average time to generate individual criteria decreased, with fine-tuned models outperforming GPT-4o in terms of generation speed (most likely due to a smaller model size). The fine-tuned models were served on two A100 GPUs.

5.3 Discussion and Implications

Our smaller, fine-tuned models achieve assertions comparable to the three-phase GPT-4o process (detailed in Section 3.3), while significantly outperforming single-phase GPT-4o. This is a meaningful finding for several reasons. First, it demonstrates that carefully curated datasets and targeted fine-

[4] https://huggingface.co/user104/promptevals_mistral and https://huggingface.co/user104/promptevals_llama

	Mistral (FT)	Llama (FT)	GPT-4o
p25	**1.8717**	2.3962	6.5596
p50	**2.3106**	3.0748	8.2542
Mean	**2.5915**	3.6057	8.7041
p75	**2.9839**	4.2716	10.1905

Table 4: Latency for criteria generation. We compared the runtimes for all 3 models (in seconds) and included the 25th, 50th, and 75th percentile along with the mean. We found that our fine-tuned Mistral model had the lowest runtime for all metrics.

	Base Mistral	Mistral (FT)	Base Llama	Llama (FT)	GPT-4o
p25	0.3608	0.7919	0.3211	**0.7922**	0.6296
p50	0.4100	0.8231	0.3577	**0.8233**	0.6830
Mean	0.4093	0.8199	0.3607	**0.8240**	0.6808
p75	0.4561	0.8553	0.3978	**0.8554**	0.7351

Table 5: Semantic F1 scores for generated assertion criteria. Percentiles and mean values are shown for base models, fine-tuned (FT) versions, and GPT-4o. Bold indicates highest scores.

	Average	Median	75th percentile	90th percentile
Base Mistral	14.5012	14	18.5	23
Mistral (FT)	**6.28640**	**5**	**8**	**10**
Base Llama	28.2458	26	33.5	46
Llama (FT)	5.47255	**5**	**6**	9
GPT-4o	7.59189	6	10	14.2
Ground Truth	*5.98568*	*5*	*7*	*10*

Table 6: Number of Criteria Generated by Models. Metrics show average, median, and percentile values. Bold indicates closest to ground truth.

tuning can enable smaller models to match or exceed the performance of much larger models in specific tasks. Second, it underscores the importance of multi-step refinement when using state-of-the-art general-purpose LLMs for generating assertion criteria, as evidenced by the low single-step GPT-4o performance. This refinement process, while effective, can lead to increased latency in non-interactive settings. Our approach offers a more resource-efficient solution for assertion generation without compromising on quality.

The ability to generate high-quality assertion criteria quickly and cost-effectively has significant implications for LLM pipeline development and deployment: it enables more frequent iterations, faster debugging, and more robust quality assurance processes without incurring prohibitive costs or delays. This is particularly valuable as prompts become longer and more complex, making the use of GPT-4o to generate assertions for every iteration on a prompt increasingly impractical. We are integrating these fine-tuned assertion generation models into our LLM pipeline development tools, particularly focusing on evaluation and monitoring capabilities. This will allow developers to automatically generate task-specific assertion criteria for any given prompt or pipeline, continuously monitor the quality of outputs in live deployments, and receive real-time feedback on output quality, all while maintaining efficiency and scalability in production environments.

While our fine-tuned models show significant improvements, we observed occasional generation of vague or redundant criteria. For example, criteria like "Output must avoid any ambiguity or confusion" and "Output must use clear and unambiguous language" could be consolidated. One idea is to directly incorporate a notion of criteria uniqueness into the training process—e.g., penalize models if, for an output, they generate two or more criteria with high semantic similarity. Another idea is to collect more data to supplement PROMPTEVALS. Future work will focus on refining the model to produce more concise and non-overlapping criteria, and generally improve the semantic F1 score. We also intend to explore the capabilities of smaller models to determine if we can reduce latency further, while still retaining good accuracy and alignment with developers' intents.

6 Conclusion

This study introduces PROMPTEVALS, a new benchmark comprising over 2,000 human-contributed prompt templates and 12,000 assertion criteria. PROMPTEVALS is more than five times larger than previous prompt collections [34, 52]. This diverse dataset represents a significant contribution to the field of LLM pipeline development, offering a robust tool for evaluating and improving task-specific output constraints. Using PROMPTEVALS, we benchmarked several models, including GPT-4o, and additionally fine-tuned open-source models for assertion generation. Our experiments demonstrate PROMPTEVALS' utility in assessing and comparing performance across different approaches to generating relevant assertions. By making PROMPTEVALS and our fine-tuned models publicly available, our goal is to encourage the development of more reliable and task-specific LLM applications across various domains.

7 Limitations

We describe a few limitations in this section: First, benchmark scores rely on OpenAI's *text-embedding-3-large* model, released on January 25, 2024, introducing a risk of inconsistency in results over time due to potential model updates. Establishing a versioning system or exploring alternative and more stable embedding methods could mitigate this issue. Moreover, currently, the benchmark is restricted to textual prompts, excluding other modalities such as images and audio. Expanding the dataset to incorporate multi-modal inputs would increase its applicability and better reflect the diverse range of real-world LLM tasks.

Finally, it's important to note that while our criteria are grounded in a taxonomy derived from developer preferences, they are ultimately generated by an LLM. This approach, while efficient, may not capture the full nuance of developer intentions for every specific use case. Ideally, criteria would be developed through directly collaborating with developers for each prompt template, ensuring maximum relevance and accuracy.

8 Ethics

PROMPTEVALS is open-source and is intended to be used as a dataset and benchmark to evaluate models' ability to identify and generate assertion criteria for prompts. However, because it is open-source, it may be used in pre-training models, which can impact the effectiveness of the benchmark. PROMPTEVALS data and derivatives of this data should not be used outside of research or prompt engineering contexts. Additionally, PROMPTEVALS uses prompts contributed by a variety of developers. We did not collect any personal identifiable information (PII) on the developers, and we looked through the data to confirm that developers did not submit PII in their prompts. Since we did not control the developer population we sampled prompts from, prompts may not represent all domains equally. However, we believe that despite this, our benchmark still provides value and can be useful in evaluating models on generating assertion criteria.

References

[1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher 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, 2020.

[2] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45, 2024.

[3] Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17754–17762, 2024.

[4] Hyung Won Chung, Le Hou, S. Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Wei Yu, Vincent Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed Huai hsin Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models. *ArXiv*, abs/2210.11416, 2022.

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

[6] Michael Desmond, Zahra Ashktorab, Qian Pan, Casey Dugan, and James M. Johnson. Evalullm: Llm assisted evaluation of generative outputs. In *Companion Proceedings of the 29th International Conference on Intelligent User Interfaces*, IUI '24 Companion, page 30–32, New York, NY, USA, 2024. Association for Computing Machinery.

[7] Yi Dong, Ronghui Mu, Gaojie Jin, Yi Qi, Jinwei Hu, Xingyu Zhao, Jie Meng, Wenjie Ruan, and Xiaowei Huang. Building guardrails for large language models, 2024.

[8] Yi Dong, Ronghui Mu, Gaojie Jin, Yi Qi, Jinwei Hu, Xingyu Zhao, Jie Meng, Wenjie Ruan, and Xiaowei Huang. Position: Building guardrails for large language models requires systematic design. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 11375–11394. PMLR, 21–27 Jul 2024.

[9] Joe B Hakim, Jeffery L Painter, Darmendra Ramcharran, Vijay Kara, Greg Powell, Paulina Sobczak, Chiho Sato, Andrew Bate, and Andrew Beam. The need for guardrails with large language models in medical safety-critical settings: An artificial intelligence application in the pharmacovigilance ecosystem. *arXiv preprint arXiv:2407.18322*, 2024.

[10] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Xiaodong Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *ArXiv*, abs/2009.03300, 2020.

[11] 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, 2021.

[12] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ArXiv*, abs/2311.05232, 2023.

[13] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions, 2023.

[14] Wenlong Huang, P. Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *ArXiv*, abs/2201.07207, 2022.

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

[16] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

[17] Adam Tauman Kalai and Santosh S Vempala. Calibrated language models must hallucinate. *arXiv preprint arXiv:2311.14648*, 2023.

[18] Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, et al. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*, 2023.

[19] Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, et al. Prometheus: Inducing fine-grained evaluation capability in language models. In *The Twelfth International Conference on Learning Representations*, 2023.

[20] 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.

[21] Tae Soo Kim, Yoonjoo Lee, Jamin Shin, Young-Ho Kim, and Juho Kim. Evallm: Interactive evaluation of large language model prompts on user-defined criteria. In *International Conference on Human Factors in Computing Systems*, 2023.

[22] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *ArXiv*, abs/2205.11916, 2022.

[23] Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, et al. Openassistant conversations-democratizing large language model alignment. *Advances in Neural Information Processing Systems*, 36, 2023.

[24] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning, 2021.

[25] Junlong Li, Shichao Sun, Weizhe Yuan, Run-Ze Fan, Hai Zhao, and Pengfei Liu. Generative judge for evaluating alignment. *ArXiv*, abs/2310.05470, 2023.

[26] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

[27] Michael Xieyang Liu, Frederick Liu, Alexander J. Fiannaca, Terry Koo, Lucas Dixon, Michael Terry, and Carrie J. Cai. "we need structured output": Towards user-centered constraints on large language model output. In *Extended Abstracts of the 2024 CHI Conference on Human Factors in Computing Systems*, CHI EA '24, New York, NY, USA, 2024. Association for Computing Machinery.

[28] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Yuxian Gu, Hangliang Ding, Kai Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Shengqi Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents. *ArXiv*, abs/2308.03688, 2023.

[29] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[30] Mohammad Niknazar, Paul V Haley, Latha Ramanan, Sang T Truong, Yedendra Shrinivasan, Ayan Kumar Bhowmick, Prasenjit Dey, Ashish Jagmohan, Hema Maheshwari, Shom Ponoth, et al. Building a domain-specific guardrail model in production. *arXiv preprint arXiv:2408.01452*, 2024.

[31] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024.

[32] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. Training language models to follow instructions with human feedback. *ArXiv*, abs/2203.02155, 2022.

[33] Liangming Pan, Michael Stephen Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *ArXiv*, abs/2308.03188, 2023.

[34] 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 preprint arXiv:2401.03601*, 2024.

[35] Traian Rebedea, Razvan Dinu, Makesh Sreedhar, Christopher Parisien, and Jonathan Cohen. Nemo guardrails: A toolkit for controllable and safe llm applications with programmable rails, 2023.

[36] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. A

systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*, 2024.

[37] Shreya Shankar, Haotian Li, Parth Asawa, Madelon Hulsebos, Yiming Lin, J. D. Zamfirescu-Pereira, Harrison Chase, Will Fu-Hinthorn, Aditya G. Parameswaran, and Eugene Wu. Spade: Synthesizing data quality assertions for large language model pipelines, 2024.

[38] Shreya Shankar, JD Zamfirescu-Pereira, Björn Hartmann, Aditya G Parameswaran, and Ian Arawjo. Who validates the validators? aligning llm-assisted evaluation of llm outputs with human preferences. *arXiv preprint arXiv:2404.12272*, 2024.

[39] Ondrej Skopek, Rahul Aralikatte, Sian Gooding, and Victor Carbune. Towards better evaluation of instruction-following: A case-study in summarization. In Jing Jiang, David Reitter, and Shumin Deng, editors, *Proceedings of the 27th Conference on Computational Natural Language Learning (CoNLL)*, pages 221–237, Singapore, December 2023. Association for Computational Linguistics.

[40] 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 Annasaheb Rahane, Anantharaman S. Iyer, Anders Andreassen, Andrea Madotto, Andrea Santilli, Andreas Stuhlmuller, Andrew M. Dai, Andrew La, Andrew Kyle 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 Karakacs, B. Ryan Roberts, Bao Sheng Loe, Barret Zoph, Bartlomiej Bojanowski, Batuhan Ozyurt, Behnam Hedayatnia, Behnam Neyshabur, Benjamin Inden, Benno Stein, Berk Ekmekci, Bill Yuchen Lin, Blake Stephen Howald, Bryan Orinion, Cameron Diao, Cameron Dour, Catherine Stinson, Cedrick Argueta, C'esar Ferri Ram'irez, Chandan Singh, Charles Rathkopf, Chenlin Meng, Chitta Baral, Chiyu Wu, Chris Callison-Burch, Chris Waites, Christian Voigt, Christopher D. Manning, Christopher Potts, Cindy Ramirez, Clara Rivera, Clemencia Siro, Colin Raffel, Courtney Ashcraft, Cristina Garbacea, Damien Sileo, Daniel H Garrette, Dan Hendrycks, Dan Kilman, Dan Roth, Daniel Freeman, Daniel Khashabi, Daniel Levy, Daniel Mosegu'i Gonz'alez, Danielle R. Perszyk, Danny Hernandez, Danqi Chen, Daphne Ippolito, Dar Gilboa, David Dohan, David Drakard, David Jurgens, Debajyoti Datta, Deep Ganguli, Denis Emelin, Denis Kleyko, Deniz Yuret, Derek Chen, Derek Tam, Dieuwke Hupkes, Diganta Misra, Dilyar Buzan, Dimitri Coelho Mollo, Diyi Yang, Dong-Ho Lee, Dylan Schrader, Ekaterina Shutova, Ekin Dogus Cubuk, Elad Segal, Eleanor Hagerman, Elizabeth Barnes, Elizabeth P. Donoway, Ellie Pavlick, Emanuele Rodolà, Emma Lam, Eric Chu, Eric Tang, Erkut Erdem, Ernie Chang, Ethan A. Chi, Ethan Dyer, Ethan Jerzak, Ethan Kim, Eunice Engefu Manyasi, Evgenii Zheltonozhskii, Fanyue Xia, Fatemeh Siar, Fernando Mart'inez-Plumed, Francesca Happ'e, François Chollet, Frieda Rong, Gaurav Mishra, Genta Indra Winata, Gerard de Melo, Germán Kruszewski, Giambattista Parascandolo, Giorgio Mariani, Gloria Xinyue Wang, Gonzalo Jaimovitch-L'opez, Gregor Betz, Guy Gur-Ari, Hana Galijasevic, Hannah Kim, Hannah Rashkin, Hannaneh Hajishirzi, Harsh Mehta, Hayden Bogar, Henry Shevlin, Hinrich Schutze, Hiromu Yakura, Hongming Zhang, Hugh Mee Wong, Ian Ng, Isaac Noble, Jaap Jumelet, Jack Geissinger, John Kernion, Jacob Hilton, Jaehoon Lee, Jaime Fernández Fisac, James B. Simon, James Koppel, James Zheng, James Zou, Jan Koco'n, Jana Thompson, Janelle Wingfield, Jared Kaplan, Jarema Radom, Jascha Narain Sohl-Dickstein, Jason Phang, Jason Wei, Jason Yosinski, Jekaterina Novikova, Jelle Bosscher, Jennifer Marsh, Jeremy Kim, Jeroen Taal, Jesse Engel, Jesujoba Oluwadara Alabi, Jiacheng Xu, Jiaming Song, Jillian Tang, Jane W Waweru, John Burden, John Miller, John U. Balis, Jonathan Batchelder, Jonathan Berant, Jorg Frohberg, Jos Rozen, José Hernández-Orallo, Joseph Boudeman, Joseph Guerr, Joseph Jones, Joshua Tenenbaum, Joshua S. Rule, Joyce Chua, Kamil Kanclerz, Karen Livescu, Karl Krauth, Karthik Gopalakrishnan, Katerina Ignatyeva, Katja Markert, Kaustubh D. Dhole, Kevin Gimpel, Kevin Omondi, Kory Wallace Mathewson, Kristen Chiafullo, Ksenia Shkaruta, Kumar Shridhar, Kyle McDonell, Kyle Richardson, Laria Reynolds, Leo Gao, Li Zhang, Liam Dugan, Lianhui Qin, Lidia Contreras-Ochando, Louis-Philippe Morency, Luca Moschella, Luca Lam, Lucy Noble, Ludwig Schmidt, Luheng He, Luis Oliveros Col'on, Luke Metz, Lutfi Kerem cSenel, Maarten Bosma, Maarten Sap, Maartje ter Hoeve, Maheen Farooqi, Manaal Faruqui, Mantas Mazeika, Marco Baturan, Marco Marelli, Marco Maru, Maria Jose Ram'irez Quintana, Marie Tolkiehn, Mario Giulianelli, Martha Lewis, Martin Potthast, Matthew L. Leavitt, Matthias Hagen, M'aty'as Schubert, Medina Baitemirova, Melody Arnaud, Melvin Andrew McElrath, Michael Yee, Michael Cohen, Michael Gu, Michael Igorevich Ivanitskiy, Michael Starritt, Michael Strube, Michal Swkedrowski, Michele Bevilacqua, Michihiro Yasunaga, Mihir Kale, Mike Cain, Mimee Xu, Mirac Suzgun, Mitch Walker, Monica Tiwari, Mohit Bansal, Moin Aminnaseri, Mor Geva, Mozhdeh Gheini, T MukundVarma, Nanyun Peng, Nathan A. Chi, Nayeon Lee, Neta Gur-Ari Krakover, Nicholas Cameron, Nicholas Roberts, Nick Doiron, Nicole Martinez, Nikita Nangia, Niklas Deckers, Niklas Muennighoff, Nitish Shirish Keskar, Niveditha Iyer, Noah Constant, Noah Fiedel, Nuan Wen, Oliver Zhang, Omar Agha, Omar Elbaghdadi, Omer Levy, Owain Evans, Pablo Antonio Moreno Casares, Parth

Doshi, Pascale Fung, Paul Pu Liang, Paul Vicol, Pegah Alipoormolabashi, Peiyuan Liao, Percy Liang, Peter Chang, Peter Eckersley, Phu Mon Htut, Pi-Bei Hwang, P. Milkowski, Piyush S. Patil, Pouya Pezeshkpour, Priti Oli, Qiaozhu Mei, Qing Lyu, Qinlang Chen, Rabin Banjade, Rachel Etta Rudolph, Raefer Gabriel, Rahel Habacker, Ramon Risco, Raphael Milliere, Rhythm Garg, Richard Barnes, Rif A. Saurous, Riku Arakawa, Robbe Raymaekers, Robert Frank, Rohan Sikand, Roman Novak, Roman Sitelew, Ronan Le Bras, Rosanne Liu, Rowan Jacobs, Rui Zhang, Ruslan Salakhutdinov, Ryan Chi, Ryan Lee, Ryan Stovall, Ryan Teehan, Rylan Yang, Sahib Singh, Saif M. Mohammad, Sajant Anand, Sam Dillavou, Sam Shleifer, Sam Wiseman, Samuel Gruetter, Samuel R. Bowman, Samuel S. Schoenholz, Sanghyun Han, Sanjeev Kwatra, Sarah A. Rous, Sarik Ghazarian, Sayan Ghosh, Sean Casey, Sebastian Bischoff, Sebastian Gehrmann, Sebastian Schuster, Sepideh Sadeghi, Shadi S. Hamdan, Sharon Zhou, Shashank Srivastava, Sherry Shi, Shikhar Singh, Shima Asaadi, Shixiang Shane Gu, Shubh Pachchigar, Shubham Toshniwal, Shyam Upadhyay, Shyamolima Debnath, Siamak Shakeri, Simon Thormeyer, Simone Melzi, Siva Reddy, Sneha Priscilla Makini, Soo-Hwan Lee, Spencer Torene, Sriharsha Hatwar, Stanislas Dehaene, Stefan Divic, Stefano Ermon, Stella Biderman, Stephanie Lin, Stephen Prasad, Steven T Piantadosi, Stuart M. Shieber, Summer Misherghi, Svetlana Kiritchenko, Swaroop Mishra, Tal Linzen, Tal Schuster, Tao Li, Tao Yu, Tariq Ali, Tatsunori Hashimoto, Te-Lin Wu, Theo Desbordes, Theodore Rothschild, Thomas Phan, Tianle Wang, Tiberius Nkinyili, Timo Schick, Timofei Kornev, Titus Tunduny, Tobias Gerstenberg, Trenton Chang, Trishala Neeraj, Tushar Khot, Tyler Shultz, Uri Shaham, Vedant Misra, Vera Demberg, Victoria Nyamai, Vikas Raunak, Vinay Venkatesh Ramasesh, Vinay Uday Prabhu, Vishakh Padmakumar, Vivek Srikumar, William Fedus, William Saunders, William Zhang, Wout Vossen, Xiang Ren, Xiaoyu Tong, Xinran Zhao, Xinyi Wu, Xudong Shen, Yadollah Yaghoobzadeh, Yair Lakretz, Yangqiu Song, Yasaman Bahri, Yejin Choi, Yichi Yang, Yiding Hao, Yifu Chen, Yonatan Belinkov, Yu Hou, Yu Hou, Yuntao Bai, Zachary Seid, Zhuoye Zhao, Zijian Wang, Zijie J. Wang, Zirui Wang, and Ziyi Wu. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *ArXiv*, abs/2206.04615, 2022.

[41] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[42] 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, 2024.

[43] Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xingxu Xie, Wei Ye, Shi-Bo Zhang, and Yue Zhang. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization. *ArXiv*, abs/2306.05087, 2023.

[44] 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 *Annual Meeting of the Association for Computational Linguistics*, 2022.

[45] 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. *ArXiv*, abs/2109.01652, 2021.

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

[47] Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. Evaluating large language models at evaluating instruction following. In *International Conference on Learning Representations (ICLR)*, 2024.

[48] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert, 2020.

[49] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric P. Xing, Joseph E. Gonzalez, Ion Stoica, and Haotong Zhang. Lmsys-chat-1m: A large-scale real-world llm conversation dataset. *ArXiv*, abs/2309.11998, 2023.

[50] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.

[51] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Haotong Zhang, Joseph Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. *ArXiv*, abs/2306.05685, 2023.

[52] 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.

[53] Lianghui Zhu, Xinggang Wang, and Xinlong Wang. Judgelm: Fine-tuned large language models are scalable judges. *ArXiv*, abs/2310.17631, 2023.

[54] Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. Toolqa: A dataset for llm question answering with external tools. *ArXiv*, abs/2306.13304, 2023.

In this appendix, we include additional information on our work. This includes the prompts used in generating the dataset, distributions of domains and assertion criteria categories, example prompts in our dataset, a datasheet, and model cards.

A Prompt Template Analysis

In this section, we describe the prompts we used to identify the domain for each prompt template, the prompts used to generate the assertion criteria for each step of our workflow, and the distributions of our LLM-generated assertion criteria.

A.1 Prompts and Analysis for Domain Categorization

To categorize the prompt templates in PROMPTEVALS, we used the following prompts for querying GPT-4o. First, `DOMAIN_CATEGORIZE_1` was applied to each prompt template in parallel to generate fine-grained level 1 categories, resulting in 974 unique domains. Next, GPT-4o was queried once to consolidate these into 50 aggregate level 2 categories, which were sanity-checked for accuracy. Then, `DOMAIN_CATEGORIZE_2` was used to map each prompt template to one of the 50 predefined level 2 categories. We queried GPT-4o once more to come up with a set of level 3 categories (the highest level), and we manually assigned each level 2 category to a good level 3 category.

DOMAIN_CATEGORIZE_1

Here is my prompt template for a specialized task:

````
```json
{prompt_template}
```
````

What domain does this prompt template pertain to? Limit your response to a single word or phrase, and be as specific and fine-grained as possible. Avoid generic domains like 'natural language processing' and 'artificial intelligence.' Return your response as a JSON object in "'json "' markers, with the key "field" and the value being the word or phrase.

DOMAIN_CATEGORIZE_2

Here is my prompt template for a specialized task:

````
```json
{prompt_template}
```
````

The domain of this prompt template is:

`{field}`

Given the following higher level domains: General-Purpose Chatbots, Workflow and Task Automation, Question-Answering Systems, Education and Academic Assistance, Content Summarization and Extraction, Information Retrieval and Management, Programming and Software Development, Customer Support and Service, Data Analysis and Visualization, Content Creation and Writing, Project Management, Psychotherapy and Mental Health, Entertainment and Gaming, Healthcare and Medicine, Financial Services and Analysis, E-Commerce and Retail, Legal and Compliance, Marketing and Sales, Coaching and Personal Development, AI Evaluation and Optimization, Translation and Multilingual Services, Creative Arts and Media, Data Management and Databases, Text Analysis and Processing, Customer Experience and Feedback, Technology and IT Support, Evaluation and Quality Assurance, Real Estate and Property Management, Research and Information Synthesis, Insurance and Risk Management, Interactive Assistance and Support, Business Intelligence and Strategy, Human Resources and Recruitment, Knowledge and Information Synthesis, Task Execution and Management, Evaluation of AI Systems, Data Visualization and Reporting, Entertainment and Interactive Systems, Question Generation and Optimization, Automation and Orchestration, Translation and Language Services, Digital Marketing and SEO, Financial Services and Advising, Programming and Development Assistance, Creative and Content Writing, Healthcare and Medical Services, Customer Experience and Support, Business and Strategy Development, Evaluation and Quality Assurance, Human Resources and Recruitment
What higher level domain does this prompt template pertain to? You must pick one from the list above. Return only a JSON object in "'json "' markers, with the key "domain" and the value being the word or phrase from the list above.

While our dataset in HuggingFace `https://huggingface.co/datasets/user104/PromptHub` includes the finest-grained level 1 domains for each prompt template, Figure 2 shows the distribution of level 2 and level 3 domains across the dataset. Most prompt templates relate to AI systems and automation, as they are mainly prompt templates that generically evaluate or grade the quality of other LLM outputs,

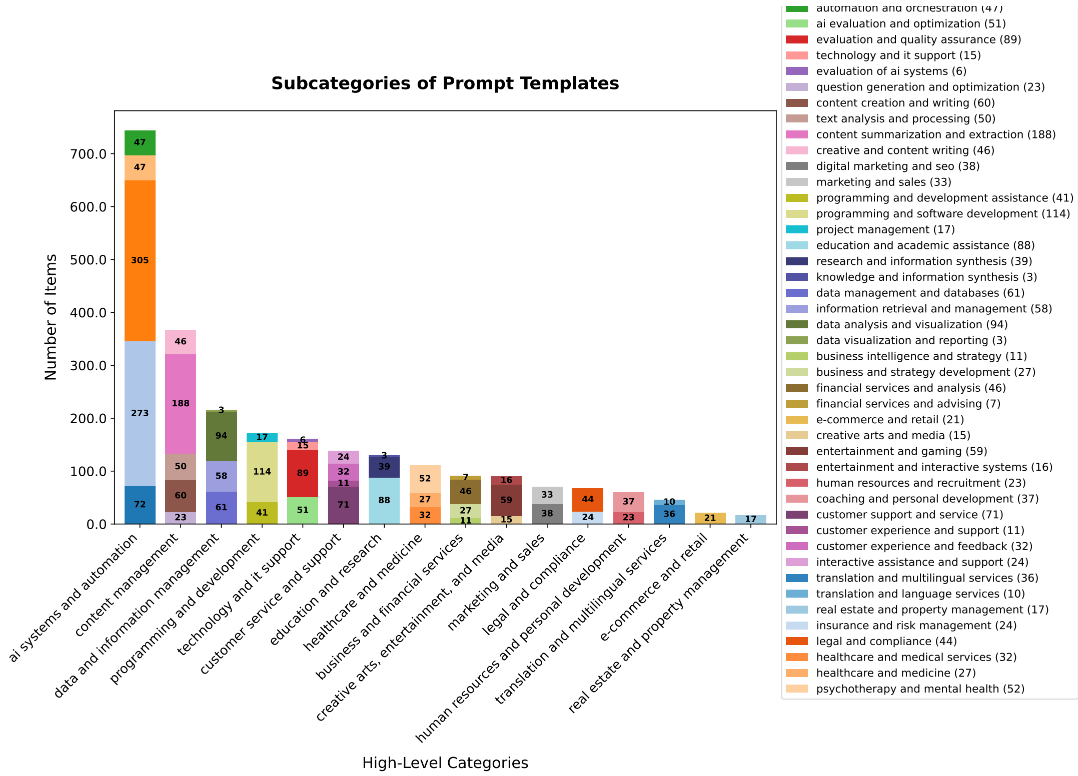

Figure 2: Distribution of Domains and Subdomains of Tasks Represented in PROMPTEVALS.

or they generically improve prompts to be more clear. The largest application of prompt templates is in content management (e.g., text summarization, creative writing), followed by data and information management (e.g., text to SQL).

A.2 Prompts for LLM-Generated Criteria

For each prompt template in the PROMPTEVALS, we use GPT-4o to generate a set of custom criteria that each LLM output should follow, or assertion criteria. Note that we use the term "constraints" here to match the terminology in Liu et al. [27]. We refer the reader to their paper to read detailed descriptions of each constraint type.

Our prompt to generate the criteria is as follows:

Prompt for Generating Custom Criteria

Here is my prompt template for a specialized task:

````
```json
{prompt_template}
```
````

I want to write assertions for my LLM pipeline to run on all pipeline outputs. Here are some categories of constraints I may want the outputs to follow:
- **Structured Output**: Is there a requirement for the output to follow a standardized or custom format, such as markdown, HTML, or a JSON object? - **Multiple Choice**: Does the output need to select from a predefined list of options? - **Length Constraints**: Are there instructions regarding the targeted length of the output, such as the number of characters, words, or items in a list? - **Semantic Constraints**: - **Excluding specific terms, items, or actions**: Are there terms, items, or actions that should be excluded from the output? - **Including or echoing specific terms or content**: Are there specific terms or content that should be included or echoed in the output? - **Covering or staying on a certain topic or domain**: Should the output cover or stay on a specific topic or domain? - **Following certain (code) grammar / dialect / context**: Are there requirements to follow certain (code) grammar, dialect, or context in the output? - **Stylistic Constraints**: Are there requirements for the output to follow a certain style, tone, or persona? - **Preventing Hallucination (Staying grounded and truthful)**: Should the output stay grounded and truthful, avoiding opinions, beliefs, or hallucinated outputs? - **Preventing Hallucination (Adhering to Instructions without improvising unrequested actions)**: Should the output strictly adhere to any specific instructions provided, without including content that is not explicitly requested?
{step}
Return only your answer as a numbered list of strings.

We updated the "step" input with a different prompt for each step of the workflow.

For step 1 (Generate initial criteria), the input was: *Give me a list of constraints to implement for verifying the quality of my LLM output. Each item in the list should contain a string description of a constraint to check for and its corresponding type. Type names are: structured_output, multiple_choice, length_constraints, exclude_terms, include_terms, stay_on_topic, follow_grammar, stylistic_constraints, stay_truthful, adhere_instructions.*

For step 2 (Add missing criteria), the input was:

Here are some assertion constraints I want the outputs to follow: {constraints}
Add assertion constraints to the provided list. Add constraints that are stated in the prompt template and not already covered by an existing constraint. Return the combined list. Make sure the constraints are also followed by their corresponding categories.

Where "constraints" was the output from the previous step.

For step 3 (Refine criteria), the input was:

Here are some assertion constraints I want the outputs to follow: {constraints}
Remove any assertion constraints that are incorrect, redundant (or already covered by another constraint), not relevant to the prompt template, or difficult to validate. Make sure the constraints are also followed by their corresponding categories.

Where "constraints" was the output from the previous step.

A.3 Analysis of LLM-Generated Criteria

We report distributions of the constraint types identified by GPT-4o in Figure 4, using the taxonomy from Liu et al. [27]. Figure 4 shows the distribution of constraints across different categories, illustrating the prevalence of `structured_output` type constraints. Additionally, Figure 3 shows a constraint type co-occurrence matrix. The top 5 co-occurring types are: `structured_output` and `adhere_instructions`, `structured_output` and `stay_on_topic`, `adhere_instructions` and `stay_on_topic`, `structured_output` and `include_terms`, and `stay_truthful` and `adhere_instructions`.

B Example Prompt Templates

B.1 Horse Racing Analytics

Prompt Template: "SystemMessagePromptTemplate
ROLE: You are a horse race analytic agent that explain a race detail with data and insight. You will receive

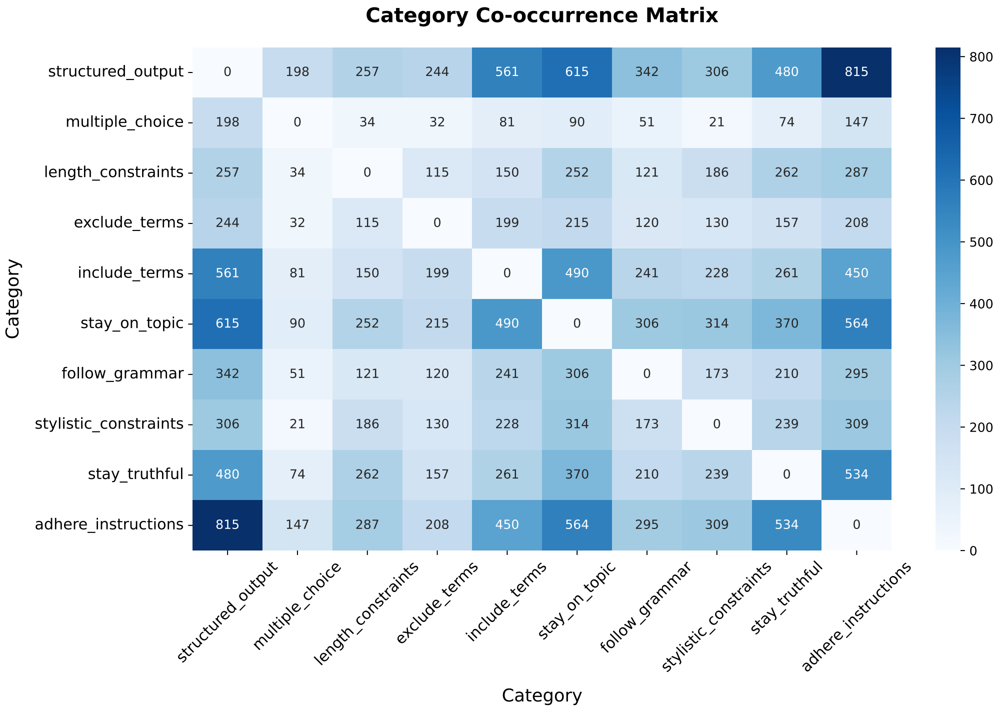

Figure 3: Constraint Type Co-Occurrence Matrix

user's question about a few horses' data, normally in numeric form. You have to first distinguish each horse's data, then answer user's question with the input and some professional's comments, your final output should be a decision of which horse is performing good or bad.
CONCEPTS: You have to note these custom attributes before answering the question: "Reborn" means that the horse has an insignificant drop in win odds, indicating that there are investment towards this horse when the win odds is not favourable. Showing that there are great increase in bettor's bet and confidence on this horse. "SpeedPro" means the attribute given from a rating system on the horse's past running strategy. "Cold Door" means the comment rating on the horse from professional horse race commentor.
ACTION: You have to analyze the separate horses and compare them with the data provided only, show which horse has the highest confidence with explanation.
HumanMessagePromptTemplate
I have a horse race with three horses participating, they has the record with : {question}. Now with the data supplied, summarize their potential performance."

C Datasheet

1. Why was the dataset created? (e.g., was there a specific intended task gap that needed to be filled?)
   The dataset was created to be used in training or fine-tuning models in generating higher quality assertion criteria.

2. Who funded the creation of the dataset?
   Lab sponsors (redacted).

3. What preprocessing/cleaning was done? (e.g., discretization or bucketing, tokenization, part-of-speech tagging, SIFT feature extraction, removal of instances)
   The prompt template was extracted from the metadata and was added to the dataset. We removed any rows that resulted in 0 assertion criteria after the first step of our 3 step workflow.

4. If it relates to people, were they told what the dataset would be used for and did they consent? If so, how? Were they provided with any mechanism to revoke their consent in the future or for certain

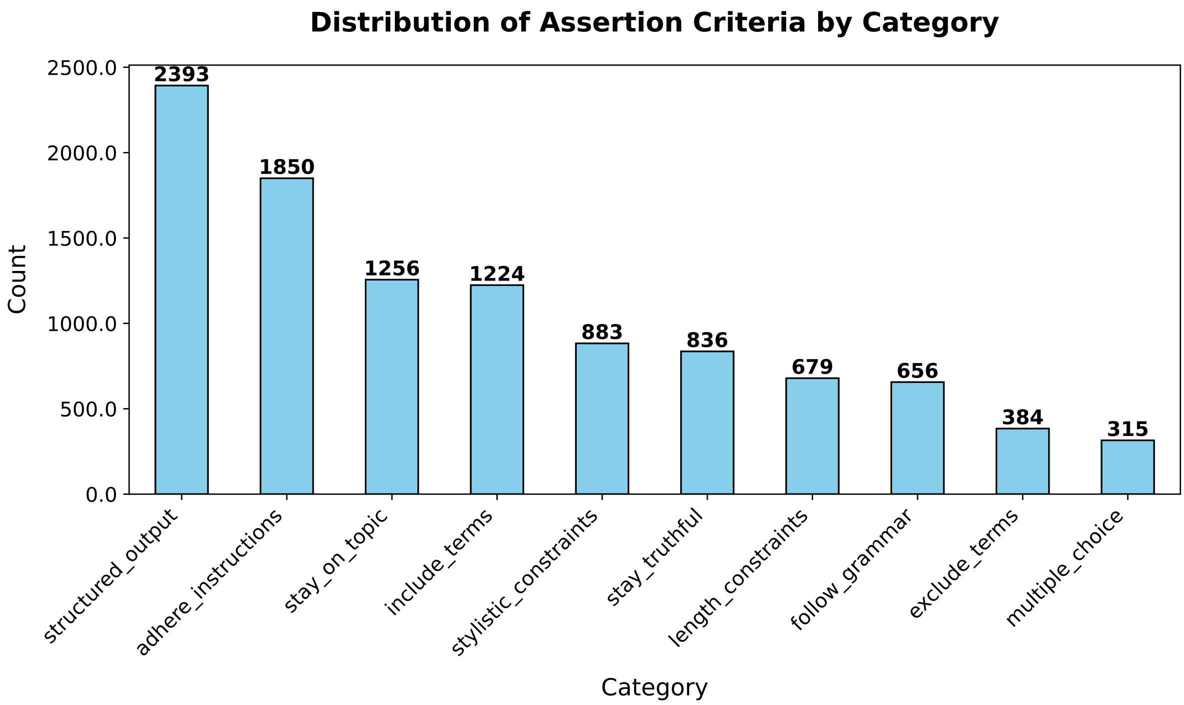

Figure 4: Distribution of Ground Truth Criteria by Type

uses?
Yes, the prompts are all from developers who consented to make their prompts public via a form. They can delete their prompts by submitting a delete request. We will semi-regularly update the Prompt Evals dataset to support the delete requests.

5. Will the dataset be updated? How often, by whom?
We plan to update the dataset yearly.

D Model Cards

D.1 Fine-tuned Mistral

1. Model Details. Basic information about the model.
– Person or organization developing model: MistralAI, and fine-tuned by [Redacted for submission]
– Model date: Base model released in September 2023, fine-tuned in July 2023
– Model version: version 3
– Model type: decoder-only Transformer
– Information about training algorithms, parameters, fairness constraints or other applied approaches, and features: 7.3 billion parameters, fine-tuned by us using Axolotl (https://github.com/axolotl-ai-cloud/axolotl)
– Paper or other resource for more information: https://arxiv.org/abs/2310.06825
– Citation details: Redacted for submission – License: Apache 2.0 license – Where to send questions or comments about the model: [Redacted for submission]

2. Intended Use. Use cases that were envisioned during development. (Primary intended uses, Primary intended users, Out-of-scope use cases)
Intended to be used by developers to generate high quality assertion criteria for LLM outputs, or to benchmark the ability of LLMs in generating these assertion criteria.

3. Factors. Factors could include demographic or phenotypic groups, environmental conditions, technical attributes, or others listed in Section 4.3.
We don't collect any demographic, phenotypic, or others listed in Section 4.3, data in our dataset.

4. Metrics. Metrics should be chosen to reflect potential realworld impacts of the model. (Model performance measures, Decision thresholds, Variation approaches)
   Metrics are defined in Section 4.1

5. Evaluation Data: Evaluated on PROMPTEVALS test set

6. Training Data: Fine-tuned on PROMPTEVALS train set

7. Quantitative Analyses (Unitary results, Intersectional results): See Table 7

Domain	Similarity	Precision	Recall
General-Purpose Chatbots	0.8171	0.8023	0.8338
Question-Answering	0.8216	0.8183	0.8255
Text Summarization	0.8785	0.8863	0.8725
Database Querying	0.8312	0.8400	0.8234
Education	0.8599	0.8636	0.8564
Content Creation	0.8184	0.8176	0.8205
Workflow Automation	0.8304	0.8258	0.8351
Horse Racing Analytics	0.8216	0.8109	0.8336
Data Analysis	0.7865	0.7793	0.7952
Prompt Engineering	0.8534	0.8330	0.8755

Table 7: Fine-Tuned Mistral Score Averages per Domain (for the 10 most represented domains in our test set)

8. Ethical Considerations: See Section 8

9. Caveats and Recommendations: None

D.2 Fine-tuned Llama

1. Model Details. Basic information about the model.
   – Person or organization developing model: Meta, and fine-tuned by [Redacted for submission]
   – Model date: Base model was released in April 18 2024, and fine-tuned in July 2024
   – Model version: 3.1
   – Model type: decoder-only Transformer
   – Information about training algorithms, parameters, fairness constraints or other applied approaches, and features: 8 billion parameters, fine-tuned by us using Axolotl (https://github.com/axolotl-ai-cloud/axolotl)
   – Paper or other resource for more information: https://arxiv.org/abs/2310.06825
   – Citation details: Redacted for submission
   – License: Meta Llama 3 Community License
   – Where to send questions or comments about the model: [Redacted for submission]

2. Intended Use. Use cases that were envisioned during development. (Primary intended uses, Primary intended users, Out-of-scope use cases)
   Intended to be used by developers to generate high quality assertion criteria for LLM outputs, or to benchmark the ability of LLMs in generating these assertion criteria.

3. Factors. Factors could include demographic or phenotypic groups, environmental conditions, technical attributes, or others listed in Section 4.3.
   We don't collect any demographic, phenotypic, or others listed in Section 4.3, data in our dataset.

4. Metrics. Metrics should be chosen to reflect potential realworld impacts of the model. (Model performance measures, Decision thresholds, Variation approaches) Metrics are defined in Section 4.1

5. Evaluation Data: Evaluated on PROMPTEVALS test set

6. Training Data: Fine-tuned on PROMPTEVALS train set

7. Quantitative Analyses (Unitary results, Intersectional results): See Table 7

Domain	Similarity	Precision	Recall
General-Purpose Chatbots	0.8140	0.8070	0.8221
Question-Answering	0.8104	0.8018	0.8199
Text Summarization	0.8601	0.8733	0.8479
Database Querying	0.8362	0.8509	0.8228
Education	0.8388	0.8498	0.8282
Content Creation	0.8417	0.8480	0.8358
Workflow Automation	0.8389	0.8477	0.8304
Horse Racing Analytics	0.8249	0.8259	0.8245
Data Analysis	0.7881	0.7940	0.7851
Prompt Engineering	0.8441	0.8387	0.8496

Table 8: Fine-Tuned Llama Score Averages per Domain (for the 10 most represented domains in our test set)

8. Ethical Considerations: See Section 8

9. Caveats and Recommendations: None