

Evaluating Robustness of Large Language Models in Enterprise Applications: Benchmarks for Perturbation Consistency Across Formats and Languages

Anonymous ACL submission

Abstract

Enterprise LLM applications require consistently high quality and reliable performance across diverse scenarios, demanding robustness to minor variations. Existing research shows that even small prompt changes can lead to substantial differences in output, but has mainly focused on a narrow set of perturbations with small academic datasets, limiting their relevance to real-world applications. To address this, we present a comprehensive benchmark suite that evaluates robustness across multiple perturbation types, including general text edits (e.g., punctuation, whitespace), formatting changes (e.g., JSON, YAML), multilingual and cross-lingual inputs, and positional variations in instructions. Evaluating 11 models ranging from 4B to 120B+ parameters, we find that minor perturbations reduce performance by up to 40 percentage points on key enterprise metrics. Critically, we demonstrate that the relationship between model size and robustness is more nuanced than conventional assumptions suggest: an 8B parameter model (Minstral 3 8B) outperforms most larger models, while another 8B model (Llama 3.1 8B) performs worst overall.

1 Introduction

Large language models (LLMs) are increasingly being used in enterprise environments to support high-stakes customer-facing applications such as virtual assistants, support automation, and content generation. However, users vary widely in their writing and understanding of model capabilities. In addition, prompt engineering practices often lack standardized guidelines or evaluation frameworks. As a result, we lack a clear understanding of how minor differences in prompt wording, structure, formatting or language affect model behavior and output quality. This variability introduces risks in consistency and performance across production use cases. Prior work [Agrawal et al. \(2025\)](#); [Chaudhary et al. \(2024\)](#); [Singh et al. \(2024\)](#); [Wang et al.](#)

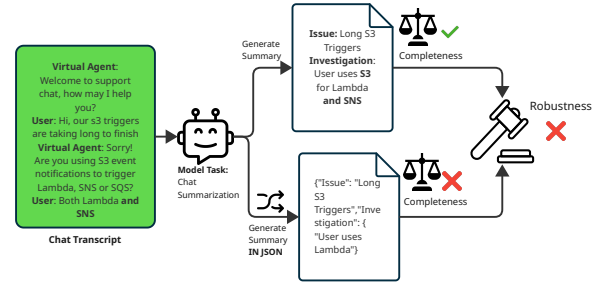


Figure 1: **JSON Format Perturbation on a Chat Summarization Task** The model performs a chat summarization task under two prompt conditions: standard (natural language) and perturbed (structured JSON format). We evaluate the outputs by measuring the quality (e.g., completeness) of each summary and assess robustness based on the consistency of model behavior across the two prompts.

(2023) has largely focused on sentence level perturbations (e.g. word swaps or paraphrasing) typically evaluated in narrow academic benchmarks. These studies do not adequately address the performance impact of prompt variation in large-scale, real-world enterprise contexts and on key metrics. We investigate how minimal variations in prompts, such as changes in word order, formatting, punctuation, or language, affect the robustness of LLMs. Here, robustness is defined as the model’s ability to consistently generate outputs of similar quality when slight input modifications are applied.

To evaluate this, we first construct realistic enterprise datasets that reflect the complexity and noise typically associated with enterprise tasks—characteristics often underrepresented in existing academic benchmarks. We then establish a baseline by repeatedly running inference on identical prompts to measure consistency. Next, we introduce independent common enterprise variations—such as rewording, translation into other languages, tone shifts, and changes in output format (e.g., YAML versus plain text)—one at a time. As seen in Figure 1, for each variation, we compare

067 the new response with the baseline to identify any
068 differences, and then calculate a final robustness
069 score based on both content changes and shifts in
070 relevant use-case metrics (e.g. Completeness, Faith-
071 fulness). Each evaluation is run separately to iso-
072 late the effect of each factor on model behavior and
073 performance.

074 Our central hypothesis is that in order to provide
075 value in enterprise tasks, a model should main-
076 tain output consistency, task alignment and output
077 quality when faced with non-substantive variations
078 in prompt structure or language. Achieving such
079 robustness is critical for scalable and dependable
080 LLM deployment across teams, languages, and
081 real-world applications.

082 We observe that models are least robust to mul-
083 tilingual and cross-lingual perturbations (mean
084 scores of 79.18 and 82.06 respectively), followed
085 by positional changes (76.33). General per-
086 turbations (variation in whitespace, punctuation,
087 etc) prove most manageable (85.51). Critically,
088 we find that the relationship between model size
089 and robustness is non-linear: while within-family
090 scaling consistently improves robustness (GPT 5
091 nano to full: +6.66 points), training methodology
092 can override size entirely—Ministral 3 8B outper-
093 forms GPT OSS 120B by 2.11 points, while an-
094 other 8B model (Llama 3.1 8B) performs worst
095 overall. This finding has significant practical impli-
096 cations for enterprise model selection. When using
097 fixed prompt templates, models robust to positional
098 changes are preferable. Tasks requiring specific
099 file formats benefit from models resilient to format
100 perturbations. Multilingual applications demand
101 models with strong multilingual robustness—and,
102 ideally, language-specific strengths. By mapping
103 task requirements to model capabilities, our analy-
104 sis supports informed model selection tailored to
105 real-world enterprise constraints. We hope these
106 findings help enterprise system designers more ef-
107 fectively integrate language models into their work-
108 flows by offering actionable insights into model
109 selection and system design, grounded in an under-
110 standing of each model’s strengths and limitations
111 under common real-world perturbations.

112 2 Related Work

113 Recent studies have systematically examined the
114 robustness of large language models (LLMs) to
115 various input perturbations, with a particular fo-
116 cus on task instructions, prompts, and evaluation

reliability.

Agrawal et al. (2025) and Chaudhary et al. (2024) show that even minor lexical changes—such as typos or substitutions—can significantly degrade human-aligned metrics like coherence, fluency, and naturalness, often leading LLMs to emphasize edge-case behaviors inappropriately. They propose mitigation strategies like denoising, perplexity smoothing, and alignment techniques. Other studies address more naturalistic or synthetic text variations. Singh et al. (2024) find that LLMs exhibit general robustness to mild semantic-preserving noise (e.g. OCR errors, keyboard typos, word splits, and synonym swaps), but performance drops as perturbations become more complex or compounded. Similarly, Wang et al. (2023) focus on common lexical variations in user inputs, including word swaps, misspellings, and synonym replacements, revealing that minor lexical changes can reduce accuracy.

In the domain of task-specific robustness, Chen et al. (2024a) introduce 18 perturbation types relevant to code generation, including stylistic (e.g. active/passive voice) and structural (e.g. in docstrings) changes, reporting up to 21.2% performance degradation. Wang and Zhao (2024) evaluate LLM robustness to 9 adversarial textual perturbations - covering lexical (e.g. homophones, typos, leetspeak), syntactic (e.g. clefts, compound variations) and semantic (e.g. red herrings, CheckList, StressTest) levels - on reasoning datasets. The results show that larger models are generally more resilient, but commonsense and arithmetic tasks are highly sensitive due to reliance on contextual knowledge and precise computation. Yang et al. (2025) observe that LLMs struggle to generate renderable structured formats (YAML, JSON, CSV etc). In particular, models struggle more with generating these formats than with converting from one to another.

Other work investigates prompt sensitivity at scale. Zhu et al. (2024) show that even small prompt perturbations, ranging from character-level (typos) to semantic-level (paraphrasing), can cause large performance drops, often transferable across models. Cao et al. (2024) introduce metrics for format adherence and structural correctness and highlights substantial limitations in LLMs’ ability to reliably generate and translate between formats across 44 tasks and 18 formats.

In addition to lexical and task specific robustness, existing work demonstrates the importance of

prompt ordering (Chen et al., 2024b; Zheng et al., 2023b; Tang et al., 2024). Zeng et al. (2025) find that LLM performance varies based on constraint ordering in prompts. Specifically, they find that LLMs generally perform better when constraints are ordered from hard to easy.

While these studies offer valuable insights, they are typically limited to isolated perturbation types or narrow academic domains that do not represent real-world enterprise scenarios. Our work introduces a unified and comprehensive benchmark that evaluates LLM robustness across a broader range of realistic enterprise related perturbation types—including multilingual, positional and output-format variations.

3 Proposed Method

To systematically evaluate the robustness of language models in practical enterprise settings, we introduce a benchmark that spans four representative enterprise tasks:

Case Summarization – Given an enterprise support case (e.g., an issue reported by a customer or support agent), the model generates a concise summary highlighting the main problem, key events, and resolution steps. **Chat Summarization** – The model summarizes a multi-turn conversation between a user and a support agent, capturing the user’s issue, the agent’s responses, and the overall outcome. **Question and Answer (QnA)** – The model answers a user question by retrieving and synthesizing information from a provided set of reference documents. **Entity Slot-filling** – The model extracts entities from a natural language query and populates a predefined schema by filling in the appropriate slots.

For each task (see task prompt templates in Appendix A.4), we create realistic input datasets and prompt the model on both the original inputs and their perturbed counterparts across five categories of perturbations: *General*, *Positional*, *Format*, *Multilingual*, and *Cross-Lingual*.

We evaluate robustness by comparing model outputs on original vs. perturbed inputs using a combination of content similarity and task-specific quality metrics. This two-pronged evaluation allows us to capture not only whether the output changes meaningfully, but also whether those changes impact task performance.

ID	Issue
1	AWS IAM policy error on assume-role chaining
2	BitLocker recovery prompt after Group Policy update
3	Zscaler proxy blocks SharePoint file sync
4	CrowdStrike policy not assigned to new OU

Table 1: Sample rows from the dataset used in benchmarking.

3.1 Dataset Creation

We synthetically construct a base dataset comprising 420 IT-related issues reflecting common enterprise software failures using GPT 4.1. Each issue served as a seed for generating inputs across multiple downstream enterprise tasks. The base issues were manually curated and covered a range of realistic scenarios, such as authentication failures, policy misconfigurations, and connectivity disruptions. Table 1 presents a few representative examples.

For each use case, we synthetically generate task-specific inputs from each issue in the base set e.g. case details for case summarization (See Table 7), chat transcripts for chat summarization (See Table 8), queries and reference documents for QnA (See Table 9) and query-entity pairs (with expected entity lists) for entity slot-filling also using GPT 4.1(See Table 10).

3.1.1 Perturbation Types

We introduce 5 categories of perturbations designed to test different aspects of LLM robustness.

General Perturbations simulate natural variation in user inputs through word- and sentence-level changes. Subtypes include: Whitespace, Spelling, Punctuation, Casing, Paraphrasing of entire sentences, and Tone modifications.

Positional Perturbations alter the order of sections within the prompt template to test instruction order sensitivity. We define 4 subtypes: Context (task input such as case details is moved in the prompt), Output Instructions (reordering or relocating formatting instructions like JSON guidelines), Reordering (changing the order of documents or bullet list items), and Mixed, which includes multiple positional changes from the other subtypes.

Format Perturbations involve modifying the final output format instructions and comparing it

255	to a free response format. Formats include JSON,	307
256	HTML, XML, Markdown and YAML.	308
257	Multilingual Perturbations assess robustness	
258	across different input languages. Prompts are trans-	309
259	lated from English into the target language, and	310
260	the model is instructed to respond in that same lan-	311
261	guage. Languages include Spanish, French, French	312
262	Canadian, German, Dutch, Italian, Japanese and	313
263	Brazilian Portuguese.	314
264	Cross-Lingual Perturbations test generaliza-	315
265	tion across mismatched input and output languages.	
266	Each task’s prompt remains in English, while the	316
267	contextual input (e.g. case details) is translated into	317
268	one of the target languages, and the model is asked	318
269	to respond in another. This creates $n \times n$ combina-	319
270	tions, where n is the number of languages (here, 9),	320
271	covering all possible input-output language pairs.	321
272	Baselines. As a baseline, the same model is	322
273	called twice with the identical prompt to estimate	323
274	natural variation in generation. This allows us to	324
275	distinguish sensitivity to perturbations from inher-	325
276	ent sampling variance.	
277	Each of these perturbation types is utilized in	
278	one of two distinct application scenarios. User-	
279	input perturbations: These included variations	
280	in tone, spelling, and language. They were ap-	
281	plied directly to task inputs such as user queries	
282	or case details i.e only parts of the input that an	
283	end-user would be able to change in a real-world	
284	scenario. For multilingual evaluation, the English	
285	dataset was translated into each target language for	
286	all tasks. Prompt-template perturbations: These	
287	involved altering prompt structure or formatting	
288	(e.g., changing instruction order or visual layout).	
289	Such modifications were applied dynamically dur-	
290	ing inference within our evaluation framework as	
291	prompt template modifications.	
292	These perturbations were chosen to reflect real-	
293	world challenges that enterprise applications face:	
294	General perturbations test resilience to the natu-	
295	ral variation in human communication and input	
296	quality variation; Positional perturbations evalu-	
297	ate whether models maintain consistent perfor-	
298	mance when prompt engineering decisions change	
299	or when different teams structure prompts differ-	
300	ently; Format perturbations assess the model’s abil-	
301	ity to reliably produce structured outputs required	
302	for downstream system integration; Multilingual	
303	perturbations ensure consistent performance across	
304	global customer bases; and Cross-lingual pertur-	
305	bations test the model’s ability to handle mixed-	
306	language scenarios common in international busi-	
	ness contexts where input documents may be in	307
	one language while outputs are required in another.	308
	4 Evaluation Metrics	309
	We assess robustness using a two-stage evaluation	310
	process that focuses on content similarity and task-	311
	specific response quality. Using this content sim-	312
	ilarity and response quality, we report an overall	313
	score along with the content and quality deltas that	314
	contribute to it.	315
	4.1 Content Similarity	316
	While several similarity metrics were explored	317
	(see Appendix A.2), we adopt LLM-as-a-Judge	318
	(Zheng et al., 2023a) as the primary method. In this	319
	setup, a large language model is prompted to eval-	320
	uate whether the perturbed response differs mean-	321
	ingfully from the original, focusing on high-level	322
	semantic consistency , such as the preservation of	323
	key entities and facts, rather than superficial lexical	324
	overlap. The scoring scale is defined as follows:	325
	<ul style="list-style-type: none"> • 3 - The responses are semantically identical; 	326
	no meaningful difference in content.	327
	<ul style="list-style-type: none"> • 2 - Minor but noticeable differences in mean- 	328
	ing; some content may be altered or omitted.	329
	<ul style="list-style-type: none"> • 1 - Major differences in meaning; the core 	330
	message or factual content is changed.	331
	(See ‘Robustness Judge Prompt’ in Appendix	332
	A.4).	333
	4.2 Quality Metrics	334
	If the similarity score is less than 3, indicating a	335
	meaningful shift in content, we evaluate both re-	336
	sponses using task-specific quality metrics , and	337
	compute the delta to determine if their quality dif-	338
	fers. These quality metrics represent how well the	339
	model has performed its use-case task based on	340
	specific criteria. Below are the quality metrics used	341
	for each task:	342
	<ul style="list-style-type: none"> • Case Summarization: Faithfulness, Com- 	343
	pleteness	344
	<ul style="list-style-type: none"> • Chat Summarization: Faithfulness, Com- 	345
	pleteness	346
	<ul style="list-style-type: none"> • Q&A: Faithfulness, Completeness, Rele- 	347
	vance, Citations, Conciseness	348
	<ul style="list-style-type: none"> • Entity Slot-filling: Exact-match F1 score and 	349
	ROUGE-L on extracted entities	350

Faithfulness represents the extent to which the model’s response adheres to the information provided in the input prompt and context, without including unsupported claims, factual errors, or hallucinations.

Completeness represents the degree to which the model response covers all important and useful content that was provide in the context in the final response.

Relevance evaluates how well the model’s response remains directly connected to the user’s query, avoiding repetition, redundancy, and the inclusion of unrelated information.

Citations assesses whether factual claims in the model’s response are properly supported by, and attributed to, the appropriate sources within the provided documents.

Conciseness checks whether the model response is succinct, to the point, and avoids excessive rambling.

Entity Match is defined as the F1 score between entities in the candidate and response, where entities are key-value pairs extracted from user input according to a schema. Keys must match exactly. For values, we require exact matches if the value has predefined choices in the schema, and use a ROUGE-L F1 threshold of 0.7 for free-text (description) fields.

These quality metrics were selected based on task-specific indicators of successful LLM behavior that could correlate with user satisfaction. For example, proper citation is important in question and answering tasks where a customer may want to refer to certain documents based on the model’s response for additional information. This would not be as relevant in another task like summarization. We find that completeness and faithfulness are important for all tasks as model responses with hallucinations are always considered problematic and incomplete responses are generally unhelpful to the user. For a task like entity slot-filling where there is exactly one correct answer, we use exact match instead of LLM as a Judge for reliability.

All of the above metrics (except entity match) are represented as Judge Prompts within an LLM-as-a-Judge setting (See prompts in A.4). We prompt the Judge Model to rate each model response on a scale of 1-3 for the given metric and use GPT 4.1 as the Judge Model.

4.3 Overall Score

To quantify the effects of perturbations, we compute content delta, quality delta and an overall robustness score derived from the content similarity and task-specific quality scores:

Content Delta measures how often a perturbation causes a meaningful change in response content. Given a dataset of N samples, we apply a specific perturbation (e.g., case change or synonym replacement) to generate N perturbed versions. For each original-perturbed pair, the LLM judge assigns a similarity score (1–3). Content Delta is calculated as the proportion of samples with a score < 3 .

Example: Consider 100 original-perturbed pairs. If 30 receive a similarity score of 2 and 20 receive a score of 1, then 50 out of 100 pairs show meaningful content differences. Thus, the Content Delta is:

$$\Delta_{Content} = \frac{30 + 20}{100} = 0.5$$

Quality Delta captures the proportion of content-shifted pairs where the perturbation also affects task-specific output quality. We focus only on the subset of pairs with a similarity score < 3 . For each such pair, we evaluate the original and perturbed responses using the appropriate quality metrics. If any metric changes—regardless of direction—we count it as a quality change.

Example: Continuing the example above, suppose that among the 50 content-shifted pairs, 15 show a change in at least one quality metric (e.g., a drop in faithfulness or a gain in relevance). Then the Quality Delta is:

$$\Delta_{Quality} = \frac{15}{50} = 0.3$$

Finally, we compute the **overall robustness score** using the following formula:

$$Overall\ Score = 1 - (\Delta_{Content} \times \Delta_{Quality}) \quad (1)$$

This formulation penalizes both frequent and impactful content changes. A higher score indicates greater robustness, reflecting both the likelihood and severity of performance degradation under input perturbations.

5 Experiments

5.1 Models

The benchmark was evaluated on 11 models spanning a range of sizes (4B to 120B+ parameters) and

architectures. We include both open-source and proprietary models: GPT 5 family (Nano, Mini, 5.2), GPT 4.1, GPT OSS (20B, 120B), Gemma 3 (4B, 12B), Ministral 3 8B Instruct 2512, Mistral Small 3.2 24B Instruct 2506, and Llama 3.1 8B Instruct.

For each model, parameters were chosen to ensure sufficient token capacity, reproducibility, and coherent responses. A minimum token limit of 6,000 was used for non-reasoning models and 12,000 for reasoning models. All models were run with a temperature of 0 and a fixed seed (where these parameters were supported) to maximize determinism. All reasoning models were run on with the lowest reasoning setting available. All results report mean and standard deviation across 5 independent runs.

5.2 Overall Results Analysis

Table 2 presents the main results across all models and perturbation categories, with cell coloring indicating robustness levels. Several key findings emerge:

Top performers. GPT 5.2 achieves the highest overall robustness score (91.01 ± 0.07), followed by GPT 4.1 (90.20 ± 0.60) and Ministral 3 8B (89.40 ± 0.73). These models maintain consistent performance across perturbation types, with Content Deltas below 30%. The heatmap reveals that top models show uniformly green (robust) coloring across categories.

Bottom performers. Llama 3.1 8B performs worst overall (69.52 ± 0.30), with a Content Delta of 64.36%—indicating that nearly two-thirds of perturbed inputs produce meaningfully different outputs. The heatmap highlights catastrophic failures (red cells) for this model on multilingual (56.97) and crosslingual (53.40) perturbations.

Perturbation difficulty hierarchy. Across all models, we observe a consistent difficulty ordering visible in the column-wise color patterns: baseline scores are uniformly high (green), while positional and multilingual columns show the most orange/yellow cells. Table 3 quantifies this: multilingual and positional perturbations cause the largest performance drops (-15.14 and -17.99 points respectively), followed by crosslingual (-12.26) and format (-10.28). General perturbations prove most manageable (-8.81 points).

5.3 Model Size Analysis

Conventional wisdom suggests that larger models exhibit greater robustness. Our results partially confirm but significantly complicate this view.

Within-family scaling helps. The GPT 5 family shows robustness improvements from scaling: nano (84.35 ± 0.14) \rightarrow mini (88.55 ± 0.15) \rightarrow 5.2 (91.01 ± 0.07), an overall improvement of 6.66-point with consistently low variance. Similarly, Gemma 3 improves from 4B (75.92 ± 0.20) to 12B (79.48 ± 2.03), and GPT OSS from 20B (82.03 ± 5.34) to 120B (87.29 ± 2.54).

Architecture and training dominate scale. The most striking finding is the 19.88-point gap between two 8B models: Ministral 3 8B (89.40 ± 0.73) dramatically outperforms Llama 3.1 8B (69.52 ± 0.30). Furthermore, Ministral 3 8B outperforms the 120B GPT OSS model by 2.11 points. This demonstrates that training methodology, data quality, and architecture choices can override substantial size advantages.

Variance patterns. The GPT OSS family exhibits notably high variance (± 5.34 for 20B, ± 2.54 for 120B), suggesting less stable behavior compared to the GPT 5 family (all $< \pm 0.15$). This variance is particularly pronounced for positional perturbations (see Figure 2).

5.4 Perturbation Category Analysis

Positional Perturbations. Figure 2 reveals dramatic differences in sensitivity to instruction ordering. The GPT 5 family shows stability (83.04.87–91.22) with minimal variance (± 0.17 – 0.62). In contrast, GPT OSS models exhibit extremely high variance: ± 18.93 for the 20B model and ± 20.91 for the 120B model, indicating fundamentally unstable positional handling despite large scale. Smaller open-source models consistently struggle (Llama 3.1 8B: 63.33 ± 0.52 , Gemma 3 4B: 66.26 ± 0.16).

Format Perturbations. Table 5.4 reveals format-specific failure modes invisible in aggregate scores. XML proves catastrophically difficult: Gemma 3 12B fails entirely to produce valid XML (N/A), Mistral Small 24B achieves only 35.00 with extreme variance (± 22.36), and Gemma 3 4B scores 60.57 ± 2.75 . Notably, Ministral 3 8B—despite ranking third overall—fails on YAML (58.03 ± 0.95). Top models (GPT 5.2, GPT 4.1) maintain uniform performance across all formats with low variance.

Multilingual Perturbations. Table 4 shows

Model	Size	Overall Score \uparrow	Content Delta \downarrow	Quality Delta \downarrow	Baseline	General	Positional	Format	Multilingual	Crosslingual
GPT 5.2	Large	91.01 ± 0.07	14.34 ± 0.09	63.07 ± 0.69	93.27 ± 0.48	90.68 ± 0.15	91.22 ± 0.17	91.25 ± 0.43	89.06 ± 0.14	90.56 ± 0.09
GPT 4.1	Large	90.20 ± 0.60	23.33 ± 1.91	45.78 ± 2.49	95.31 ± 0.65	90.07 ± 0.24	86.89 ± 0.62	91.00 ± 0.34	86.82 ± 0.59	91.36 ± 1.58
Ministral 3 8B	8B	89.40 ± 0.73	29.37 ± 1.98	34.23 ± 2.51	98.32 ± 0.48	92.49 ± 0.25	78.92 ± 0.34	89.45 ± 0.31	89.12 ± 0.04	87.66 ± 4.61
GPT 5 Mini	Med	88.55 ± 0.15	19.36 ± 0.06	59.90 ± 0.88	93.86 ± 0.70	87.96 ± 0.50	87.87 ± 0.22	86.36 ± 0.29	85.74 ± 0.17	89.53 ± 0.10
GPT OSS 20B	120B	87.29 ± 2.54	28.79 ± 2.24	43.78 ± 3.28	93.54 ± 0.36	88.21 ± 1.87	73.00 ± 20.91	89.10 ± 0.71	85.78 ± 0.31	88.53 ± 1.17
GPT 5 Nano	Small	84.35 ± 0.14	31.61 ± 0.31	50.23 ± 0.80	92.95 ± 0.60	85.31 ± 0.30	83.04 ± 0.23	82.06 ± 0.29	77.67 ± 0.30	85.04 ± 0.24
Mistral Small 3.2 24B	24B	82.45 ± 0.12	41.33 ± 0.19	41.37 ± 0.57	96.58 ± 0.26	84.50 ± 0.23	74.31 ± 0.26	85.26 ± 0.13	75.22 ± 0.20	78.82 ± 0.06
GPT OSS 20B	20B	82.03 ± 5.34	42.28 ± 4.26	42.20 ± 6.36	92.09 ± 2.01	83.26 ± 6.51	66.29 ± 18.93	85.20 ± 6.50	82.07 ± 0.17	83.28 ± 2.89
Gemma 3 12B	12B	79.48 ± 2.03	41.27 ± 1.63	48.24 ± 2.51	95.32 ± 0.83	82.88 ± 0.17	68.53 ± 9.47	75.49 ± 6.94	75.25 ± 1.82	79.83 ± 1.69
Gemma 3 4B	4B	75.92 ± 0.20	51.08 ± 0.20	45.90 ± 0.45	93.45 ± 0.56	79.15 ± 0.27	66.26 ± 0.16	74.78 ± 0.39	67.24 ± 0.24	74.63 ± 0.16
Llama 3.1 8B	8B	69.52 ± 0.30	64.36 ± 0.47	45.45 ± 0.71	92.80 ± 1.01	76.15 ± 0.43	63.33 ± 0.52	74.49 ± 0.44	56.97 ± 0.40	53.40 ± 0.46
Average	—	83.65 ± 1.11	35.19 ± 1.21	47.28 ± 1.93	94.32 ± 0.72	85.51 ± 0.99	76.33 ± 4.71	84.04 ± 1.53	79.18 ± 0.40	82.06 ± 1.19

Table 2: Overall Scores (mean \pm std over 5 runs) per model across perturbation categories. Cell colors indicate robustness level for category scores: ≥ 90 , 85–90, 80–85, 75–80, 70–75, 65–70, 60–65, < 60 . Content and Quality Delta columns are uncolored as lower values are better. Models sorted by Overall Score; best per column in **bold**.

Category	Mean Score	Mean Drop from Base	Mean Std (Dev)
Baseline	94.32	—	0.72
General	85.51	-8.81	0.99
Format	84.04	-10.28	1.53
Crosslingual	82.06	-12.26	1.19
Positional	76.33	-17.99	4.71
Multilingual	79.18	-15.14	0.40

Table 3: Perturbation category difficulty analysis. Categories ordered by mean overall score drop from baseline. Higher standard deviation across models indicates less predictable difficulty. Multilingual and positional perturbations are universally challenging; general perturbations are manageable for most models.

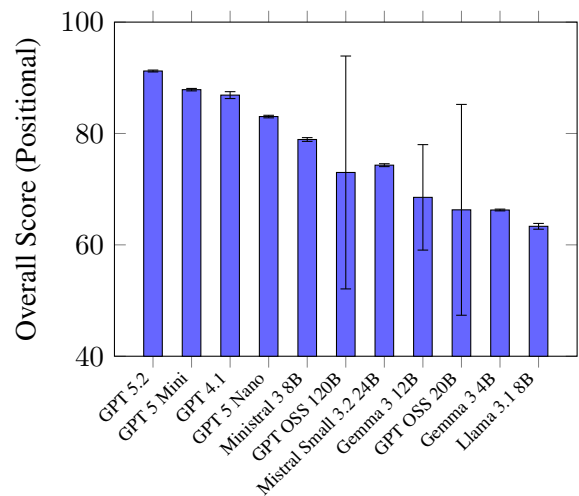


Figure 2: Overall Scores on positional perturbations by model (mean \pm std). The GPT 5 family demonstrates consistent advantages with low variance. Note the extremely high variance for GPT OSS 20B (± 18.93) and GPT OSS 120B (± 20.91), indicating unstable positional handling. Smaller open-source models (Llama 3.1 8B, Gemma 3 4B) show significant sensitivity to instruction ordering.

Quality Delta by language (lower is better). Japanese and Dutch emerge as especially challenging languages (mean delta: 44.32 and 44.31 respectively), followed by German (43.87). In particular, 6 out of the 11 models evaluated exhibit the largest quality delta on Japanese. Ministral 3 8B achieves remarkably uniform cross-language performance with the lowest deltas (26.15-31.31, range of only 5.16 points), while GPT 5.2 shows the highest quality deltas despite strong overall performance, suggesting a quality-consistency tradeoff.

5.5 Universal vs. Model-Specific Difficulty

Table 3 analyzes which perturbation types are universally challenging versus model-dependent.

Universally hard: Positional (-17.99), multilingual (-15.14), and crosslingual (-12.26) appear to be the most challenging perturbations across models.

Model-dependent: Format perturbations show moderate mean difficulty but extreme variance on specific formats (XML, YAML). A model’s format

robustness cannot be predicted from overall performance—Ministral 3 8B’s YAML failure demonstrates this clearly.

Universally manageable: General perturbations cause the smallest drops (-8.81) indicating predictable, manageable difficulty across all models tested.

6 Conclusion

We introduced a comprehensive benchmark evaluating LLM robustness across 11 models, four enterprise tasks, and five perturbation categories. Our evaluation of models ranging from 4B to 120B+

Model	German	Spanish	French	Italian	Japanese	Dutch	Portuguese	Brazilian Portuguese
GPT 5.2	63.34±1.84	57.37±3.24	56.79±2.37	60.46±1.54	64.77 ±2.88	63.75±2.20	62.96±2.41	61.93±1.56
GPT 4.1	37.45±1.50	33.65±0.58	33.47±1.47	40.01 ±2.32	29.14±2.60	39.29±1.45	38.48±2.80	35.44±2.73
GPT 5 Mini	53.40±1.06	54.28±1.13	50.52±1.24	56.75±2.36	51.20±0.95	55.66±1.23	56.93 ±0.56	54.50±1.06
GPT 5 Nano	54.45 ±1.64	47.22±0.66	45.21±1.94	48.76±1.03	45.50±1.73	53.54±1.14	49.44±2.26	47.48±1.02
Ministral 3 8B	26.69±1.28	27.64±1.41	27.96±0.54	27.54±1.12	31.31 ±1.39	26.15±1.41	29.48±1.25	27.90±1.58
GPT OSS 120B	39.01±0.61	35.07±1.05	33.82±1.04	40.05±1.90	40.35±1.47	43.67 ±1.06	35.80±1.65	36.06±1.34
Mistral Small 3.2 24B	43.04±1.37	36.67±0.59	41.23±0.47	38.69±0.66	45.74 ±0.84	37.90±0.81	37.41±0.48	36.43±1.27
GPT OSS 20B	33.17±1.66	28.60±0.95	28.50±0.83	30.87±1.07	33.88 ±0.42	31.23±0.94	29.00±1.18	29.39±1.30
Gemma 3 12B	40.77±4.17	43.26±8.52	36.81±0.66	40.64±1.04	44.88 ±1.02	40.74±0.90	39.67±2.00	40.36±2.78
Gemma 3 4B	44.81±0.38	48.62 ±0.75	40.89±0.37	45.32±0.95	45.95±0.36	47.51±0.79	42.10±0.67	43.36±1.09
Llama 3.1 8B	46.40±0.81	44.73±1.24	44.90±0.84	46.40±0.75	54.75 ±0.95	47.99±0.36	44.03±1.28	45.88±1.57
<i>Mean</i>	43.87	41.56	40.01	43.23	44.32	44.31	42.30	41.70

Table 4: Multilingual Quality Delta by language (mean ± std; lower is better, indicates less quality change when perturbed). **Bold** in data rows indicates worst (highest) delta for that model.

Model	JSON	YAML	XML	Markdown	HTML
GPT 5.2	91.47±0.42	91.52±0.62	91.54±0.52	90.16 ±0.43	91.57±0.63
GPT 4.1	91.88±1.35	91.11±0.45	91.50±0.65	89.03 ±0.40	91.53±0.48
Ministral 3 8B	92.03±0.28	58.03 ±0.95	90.60±1.06	96.30±0.40	88.37±0.39
GPT OSS 120B	89.64±0.49	89.62±0.14	89.33±0.74	88.50±1.59	88.45 ±1.02
Mistral Small 3.2 24B	85.56±0.25	87.16±0.43	35.00 ±22.36	86.36±0.45	82.16±0.30
GPT OSS 20B	88.55±0.84	80.48 ±17.30	88.48±0.77	82.80±9.39	86.85±2.51
Gemma 3 12B	73.74 ±5.70	79.41±5.97	N/A	74.20±9.22	75.16±6.69
Gemma 3 4B	75.57±0.72	79.14±0.79	60.57 ±2.75	73.20±0.41	73.77±0.80
Llama 3.1 8B	72.12 ±0.46	77.02±0.84	73.56±1.55	76.36±0.68	73.33±0.75

Table 5: Overall Scores across format perturbation scores (mean ± std). N/A indicates complete failure to produce valid output. Note the high variance for Mistral Small 24B on XML (±22.36), indicating inconsistent behavior. **Bold** in data rows indicates lowest robustness for that model.

parameters yields several key findings:

Perturbation hierarchy. Models are least robust to multilingual and positional perturbations (15–17 point drops from baseline), moderately affected by format and crosslingual changes (10–12 points), and most resilient to general perturbations (9 points). This hierarchy is consistent across model sizes.

Size-robustness relationship. Conventional wisdom that “bigger is better” receives partial support but warrants qualification. Within model families, scaling tends to improve robustness (GPT 5: +6.66 points from nano to 5.2). However, training methodology appears to play a substantial role: Ministral 3 8B outperforms the 120B GPT OSS model, while Llama 3.1 8B—at identical 8B scale—performs worst overall. The 19.88-point gap between these 8B models demonstrates that architecture and training quality matter more than raw parameters.

Format-specific vulnerabilities. Even high-performing models exhibit catastrophic failures on specific formats. XML proves problematic for multiple models regardless of size, and Ministral 3

8B—despite ranking third overall—fails on YAML generation. Format robustness cannot be predicted from overall performance.

Language challenges. Japanese consistently emerges as the most difficult language across models. Crosslingual tasks (mismatched input/output languages) prove particularly challenging, with English-to-other-language generation showing unexpected weaknesses.

These findings offer actionable guidance for enterprise deployment: (1) robustness testing should be incorporated into model selection, particularly for multilingual and structured-output applications; (2) model size alone is an insufficient predictor of robustness—evaluation on specific perturbation types is necessary; (3) format requirements should be tested individually, as aggregate performance masks format-specific failures.

7 Limitations

While our benchmark provides a comprehensive evaluation of LLM robustness in enterprise scenarios, several limitations should be acknowledged:

LLM-as-a-Judge: Our primary quality evalua-

tion metrics for three of the four use cases rely on using GPT 4.1 to assess model responses. NLP experts tuned these judge prompts to ensure maximal consistency between human expert evaluation and LLM evaluation for the given metrics. While this approach enables scalable and consistent evaluation across a large dataset, it may overlook subtle errors that a human expert would detect. The LLM-as-a-judge paradigm, though increasingly common, is not a perfect substitute for human annotation, especially for nuanced or domain-specific tasks.

Information Loss in Aggregation: To summarize robustness across tasks, models, and perturbation types, we aggregate results using overall scores and deltas. This aggregation can obscure important granular differences, such as specific failure modes or language-specific weaknesses, and may mask outlier behaviors that are critical in certain enterprise contexts. This consideration is particularly important when analyzing quality deltas and differences, as a net change of -6% at the aggregate level does not clarify whether most individual records experienced minor quality declines or whether a mix of substantial decreases and offsetting increases contributed to the aggregate value.

Non-Determinism in Model Outputs: Despite setting fixed random seeds and using temperature 0 for most models, we observed non-deterministic outputs across repeated runs. For some models (e.g., OpenAI reasoning models), temperature cannot be configured which may increase output variability. As a result, our robustness measurements may underestimate the true variance in model behavior, and some observed differences could be due to inherent sampling noise rather than prompt perturbations alone.

Translation Quality and Verification: For multilingual perturbations, only the base prompt translations were reviewed and verified by linguistic experts familiar with each language being tested. The contextual data (e.g., case details, chat transcripts) was translated automatically by LLMs without human verification. This introduces the possibility of translation errors or artifacts that could affect model performance independently of the intended perturbation.

Limited Close-Source Model Evaluations: Due to budget constraints and the size of our experiments, we were only able to evaluate our robustness benchmarks on a small group of closed-source models from one company. Future work would ideally evaluate more frontier models from a wider range

of labs to provide a more comprehensive view of robustness across the current LLM ecosystem.

Single Domain Focus: The vast majority of the 420 base issues in our benchmark are IT-related software failures. This narrow domain focus may not capture robustness challenges present in other enterprise sectors such as healthcare, finance, legal, or manufacturing. Different domains may exhibit unique types of noise, jargon, and perturbation patterns that could reveal different robustness characteristics in LLMs.

Limited Task Diversity: The benchmark focuses on only 4 enterprise tasks: case summarization, chat summarization, question answering, and entity slot-filling. While these tasks are representative of common enterprise applications, they do not cover the full spectrum of LLM use cases in production environments, such as code generation, complex multi-step reasoning chains, data analysis, or creative content generation.

Isolated Perturbation Testing: Each perturbation type is evaluated independently to isolate its specific effect on model behavior. However, real-world scenarios often involve multiple simultaneous perturbations—for example, a user might submit a misspelled query in a non-English language with unusual formatting. The compounding effects of multiple perturbations may reveal different robustness patterns than those observed when perturbations are applied in isolation.

Limited Run Count: Despite setting fixed random seeds and using temperature 0 for most models, we observed non-deterministic outputs across repeated runs and therefore conducted 5 independent runs per configuration. However, this may be insufficient to fully capture the variance in non-deterministic behaviors, particularly for models exhibiting high variance such as the GPT OSS family. A larger number of runs could provide more stable estimates of robustness, especially for models with inherent output variability.

Fixed Temperature Settings: All models were evaluated at temperature 0 (where supported) to maximize determinism and isolate the effects of perturbations from sampling variance. However, this does not reflect all real-world deployment scenarios where higher temperatures are often used to generate more diverse or creative outputs. We chose temperature 0 because higher temperatures would introduce additional output variance that could confound our robustness measurements—making it difficult to distinguish between

perturbation-induced changes and temperature-induced stochasticity. Nevertheless, enterprises deploying models with higher temperature settings may observe different robustness characteristics than those reported here.

Reasoning Model Limitations: For reasoning-capable models, we used the lowest reasoning setting available to maintain consistency and control costs. This decision was informed by the relatively straightforward nature of our enterprise tasks, which do not require extensive chain-of-thought reasoning or complex problem decomposition. However, this configuration may not reflect optimal or typical deployment settings in production environments where higher reasoning settings might be employed for more complex tasks. Enterprises using higher reasoning settings may experience different robustness-performance tradeoffs than those captured in our evaluation.

Language Coverage: Our multilingual and cross-lingual evaluations include 8 languages: Spanish, French, French Canadian, German, Dutch, Italian, Japanese, and Brazilian Portuguese. While this provides meaningful coverage of major European and Asian languages, it excludes other widely-used languages in global enterprises such as Chinese, Hindi, Arabic, Korean, and various African and Southeast Asian languages. The robustness patterns we observe may not generalize to these underrepresented languages, which may have different linguistic characteristics, resource availability, and model training exposure.

References

- Aryan Agrawal, Lisa Alazraki, Shahin Honarvar, and Marek Rei. 2025. [Enhancing llm robustness to perturbed instructions: An empirical study](#). In *Proceedings of the Thirteenth International Conference on Learning Representations*.
- Boxi Cao, Mengjie Ren, Hongyu Lin, Xianpei Han, Feng Zhang, Junfeng Zhan, and Le Sun. 2024. [Structeval: Deepen and broaden large language model assessment via structured evaluation](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5300–5318. Association for Computational Linguistics.
- Manav Chaudhary, Harshit Gupta, Savita Bhat, and Vasudeva Varma. 2024. [Towards understanding the robustness of LLM-based evaluations under perturbations](#). In *Proceedings of the 21st International Conference on Natural Language Processing (ICON)*, pages 197–205, AU-KBC Research Centre, Chennai, India. NLP Association of India (NLP AI).
- Junkai Chen, Zhenhao Li, Xing Hu, and Xin Xia. 2024a. [Nlperturbator: Studying the robustness of code llms to natural language variations](#). *Preprint*, arXiv:2406.19783.
- Xinyi Chen, Baohao Liao, Jirui Qi, Panagiotis Eustratiadis, Christof Monz, Arianna Bisazza, and Maarten de Rijke. 2024b. [The SIFo benchmark: Investigating the sequential instruction following ability of large language models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1691–1706, Miami, Florida, USA. Association for Computational Linguistics.
- Ayush Singh, Navpreet Singh, and Shubham Vatsal. 2024. [Robustness of llms to perturbations in text](#). *Preprint*, arXiv:2407.08989.
- Raphael Tang, Crystina Zhang, Xueguang Ma, Jimmy Lin, and Ferhan Ture. 2024. [Found in the middle: Permutation self-consistency improves listwise ranking in large language models](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2327–2340, Mexico City, Mexico. Association for Computational Linguistics.
- Haoyu Wang, Guozheng Ma, Cong Yu, Ning Gui, Linrui Zhang, Zhiqi Huang, Suwei Ma, Yongzhe Chang, Sen Zhang, Li Shen, Xueqian Wang, Peilin Zhao, and Dacheng Tao. 2023. [Are large language models really robust to word-level perturbations?](#) *Preprint*, arXiv:2309.11166.
- Yuqing Wang and Yun Zhao. 2024. [Rupbench: Benchmarking reasoning under perturbations for robustness evaluation in large language models](#). *Preprint*, arXiv:2406.11020.
- Jialin Yang, Dongfu Jiang, Lipeng He, Sherman Siu, Yuxuan Zhang, Disen Liao, Zhuofeng Li, Huaye Zeng, Yiming Jia, Haozhe Wang, Benjamin Schneider, Chi Ruan, Wentao Ma, Zhiheng Lyu, Yifei Wang, Yi Lu, Quy Duc Do, Ziyang Jiang, Ping Nie, and Wenhu Chen. 2025. [Structeval: Benchmarking llms’ capabilities to generate structural outputs](#). *Preprint*, arXiv:2505.20139.
- Jie Zeng, Qianyu He, Qingyu Ren, Jiaqing Liang, Weikang Zhou, Zeye Sun, Fei Yu, and Yanghua Xiao. 2025. [Order matters: Investigate the position bias in multi-constraint instruction following](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 12479–12492, Vienna, Austria. Association for Computational Linguistics.
- 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. 2023a. [Judging llm-as-a-judge with mt-bench and chatbot arena](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc.

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. 2023b. [Judging llm-as-a-judge with mt-bench and chatbot arena](#). In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Yue Zhang, Neil Zhenqiang Gong, and Xing Xie. 2024. [Promptrobust: Towards evaluating the robustness of large language models on adversarial prompts](#). Preprint, arXiv:2306.04528.

A Appendix

A.1 Model Versions

We use the following versions of closed-source models in our experiments:

Model	Version Used
GPT 5.2	2025-12-11
GPT 5 Mini	2025-08-07
GPT 5 Nano	2025-08-07
GPT 4.1	2025-04-14

Table 6: Closed-source model versions used in evaluation.

A.2 Similarity Metrics

In addition to using an LLM as a judge to assess similarity between perturbed and unperturbed prompts, we also experimented with computing the following supplementary metrics. While these are not used in the core analysis presented in this paper, they are available for reference when running the benchmark. We found that these methods did not capture variation in core meaning and semantic relevance as well as LLM-as-a-Judge.

(1) **ROUGE-L** computes the longest common subsequence (LCS) between the original and perturbed responses. This metric reflects lexical similarity by capturing both precision and recall over sequence alignment, offering a surface-level comparison useful for detecting changes in phrasing or structure.

(2) **Semantic similarity**, measured using the all-MiniLM-L6-v2 sentence transformer model. Each response is embedded into a dense vector space, and cosine similarity is calculated to quantify semantic overlap. This approach enables detection of meaning-preserving paraphrases and subtle shifts in content, even when lexical choices differ.

Note that most of the prompts for each enterprise task expects a JSON-formatted output. To compute similarity in these cases, we first align the keys between the original and perturbed outputs, then calculate similarity scores on the corresponding values. We refer to these methods as **JSONRouge** and **JSONSimilarity**.

A.3 AI Usage Disclosure

We utilized AI assistants (Claude and ChatGPT) to support various aspects of this paper’s preparation, including paraphrasing and refining sections for clarity, generating LaTeX formatting for tables and figures, proofreading and spell-checking, and conducting code review.

A.4 Prompts

Case Summarization Task Prompt

User:

You are given the following case that needs to be summarized:
`{{case_details}}`
 Be sure to include a few sections to make it easy for the user to scan.

Include the following sections:

Main Problem

- It should be the main issue that the case is about.

Key Events

- It should be a list of all the key events that took place over the course of the case

Final Resolution

- Any conclusion notes about the case that led to its resolution.

Only generate content for the above sections.

Output in the following JSON format with keys in English and values in English:

```
{ "Main Problem": <string>, "Key Events": <list[string]>, "Final Resolution": <string>}
```

Entity Slot-Filling Task Prompt

User:

Here is a JSON schema of entities that may be able to be extracted from the case.

```
{{entities}}
```

Here is the text where entities may be extracted from :

Issue: {{issue}}

Question: {{question}}

Your task is to extract all of the entities that can be filled from the above entity schema from the text.

Please provide your output in the following JSON format:

```
{
  "entities": [
    {
      "name": <entity_name>,
      "value": <entity_value>
    }
  ]
}
```

Chat Summarization Task Prompt

User:

You are given the following chat transcript that needs to be summarized:

```
{{chat_transcript}}
```

Be sure to include a few sections to make it easy for the user to scan.

Include the following sections:

Main Problem

- It should be the main issue that the case is about.

Key Events

- It should be a list of all the key events that took place over the course of the case

Final Resolution

- Any conclusion notes about the case that led to its resolution.

People Involved

- All the people and names of individuals who are involved in the chat

Sentiment

- The overall sentiment of the person interacting with the agent

Overall Summary

- A final 2-3 sentence summary to explain what went on during the chat

Only generate content for the above sections.

Output in the following JSON format with keys in English and values in English:

```
{ "Main Problem": <string>, "Key Events":
<list[string]>, "Final Resolution":
<string>, "People Involved": []}
```

Question Answering Task Prompt

User:

You are given the following documents where an answer may be found.

Doc 1:
{{kb_document_1}}

Doc 2:
{{kb_document_2}}

Doc 3:
{{kb_document_3}}

Doc 4:
{{kb_document_4}}

The user has asked the following question:

```
{{query}}
```

Your job is to answer the user's question using **ONLY** the documents that are available to you.

Do not answer any questions that are not relevant to the above context.

This means that even if you did know the answer from your prior knowledge, you should not answer the question.

Response Formatting

Make sure that each sentence is cited appropriately using the following format (id).

For example:

Banks make money (doc_1). Using that money, employees can purchase ice cream from the cafeteria. The cafeteria is at building B (doc_2).

In the above case, doc_1 is where the "banks make money" statement comes from, while doc_2 contains information about ice cream and the cafeteria's location.

NEVER say a sentence that cannot be grounded in one of the above documents.

Output Structure

Provide your output in the following format with JSON keys in English and values in English:

```
{ "output_response": <string with inline
citations>, "references": <list[string]> }
```

Robustness Judge Prompt

System:

You are an expert focused on evaluating how closely a candidate response aligns with a reference response.

User:

Please consider the following factors in your evaluation:

Does the content presented in the candidate response match the content of the reference response?

Are there any significant differences in the details and entities presented in each response?

Details and entities refer to any proper nouns and specific pieces of information.

<BEGIN_REFERENCE_RESPONSE>

{{reference_response}}

<END_REFERENCE_RESPONSE>

<BEGIN_CANDIDATE_RESPONSE>

{{candidate_response}}

<END_CANDIDATE_RESPONSE>

Rating Criteria:

3 - Strongly Aligned: The candidate response contains the same content as the reference response. Both responses include the same details and entities. Paraphrasing is fine as long as the details and entities are the same.

2 - Moderately Aligned: The candidate responses conveys the same ideas as the reference response, but there are some differences in the specific details and entities included in each response.

1 - Not Aligned: The candidate response conveys a different message than the reference response.

The details and entities in the candidate response do not match the details and entities in the reference response.

OUTPUT FORMAT:

Please provide the response in a valid JSON format as shown below:

```
{  "rating":    <your score here>,  
  "explanation":  "<explanation of how  
closely the responses align, mention any  
notable discrepancies here>"}
```

Completeness Judge Prompt (Case Summarization)

User:

You are an expert judge tasked with judging the completeness of an AI generated summary of case details. To make your assessment, the following will be provided below. Evaluation criteria, Rating criteria, AI generated summary, Case Details, Output Format. Follow the steps below to judge the completeness of the AI generated summary.

AI GENERATED SUMMARY:

{{response}}

First, identify the sections in the AI generated summary above.

CASE DETAILS:

{{case_details}}

Second, compare the AI generated summary to the case details.

For each section of the summary, consider whether the content of the section captures the relevant details and entities from the case details.

Entities refer to things like case numbers, contact information, account numbers, and other case specific information.

Key entities are the entities that are most critical for understanding the case.

Now you will rate the completeness of the AI generated summary using the EVALUATION CRITERIA and RATING CRITERIA below.

EVALUATION CRITERIA:

- The summary includes key details and the most relevant entities from the case details
- The summary does NOT need to include every detail from the case details
- For each section of the summary, that section content includes the key entities for the details it describes.

RATING CRITERIA:

Rate the completeness of the generated summary on a scale of 1-3.

1 - the summary sections contain very few or none of the relevant details and entities from the case details.

2 - the summary sections contains some relevant details and entities from the case details but miss the most relevant details and entities for some sections of summary.

3 - all sections of the summary include the most relevant details and entities. The summary contains many key details and entities from the case details.

OUTPUT FORMAT:

Please provide the response in a valid JSON format as shown below:

```
{  
  "explanation":  
<your_explanation_for_the_rating_here>,  
  "rating":  "<your_score_here>"}
```

Faithfulness Judge Prompt (Chat Summarization)

User:

You are an expert judge tasked with judging the faithfulness of an AI generated summary of a chat transcript.

To make your assessment, the following will be provided below.

- Evaluation criteria
- Rating criteria
- AI generated summary
- Chat Transcript
- Output Format

Use the information below to judge the faithfulness of the AI generated summary.

EVALUATION CRITERIA:

- The summary only includes details and information provided in the chat transcript
- The summary only includes assumptions that can be reasonably implied from the chat transcript
- The summary DOES NOT misinterpret or misrepresent information from the chat transcript
- Failure to include relevant details from the chat transcript DOES NOT affect faithfulness

RATING CRITERIA:

Rate the faithfulness of the generated summary on a scale of 1-3. Remember that failure to include relevant details should NOT affect the rating and should NOT be mentioned in the explanation.

1 - the summary contains major hallucinations, includes details not mentioned in the chat transcript, or is completely unrelated to the chat transcript.

2 - the summary contains minor hallucinations such as misinterpreting or misrepresenting details from the chat transcript.

3 - the summary contains no hallucinations and the summary is grounded in the chat transcript. All interpretations made by the summary are have some basis in the chat transcript and are logical assumptions.

AI GENERATED SUMMARY:

{{response}}

CHAT TRANSCRIPT:

{{chat_transcript}}

OUTPUT FORMAT:

Please provide the response in a valid JSON format as shown below:

```
{
  "explanation":
  <your_explanation_for_the_rating_here>,
  "rating": "<your_score_here>"
}
```

Relevance Judge Prompt (Question Answering)

User:

You have the following documents which are related to the user query: doc_1, doc_2, doc_3 and etc.

DOCUMENTS

Doc 1

{{kb_1}}

Doc 2

{{kb_2}}

Doc 3

{{kb_3}}

Doc 4

{{kb_4}}

END_DOCUMENTS

You are also provided with the below user query:

QUERY

{{query}}

END_QUERY

Your task is to grade the response generated by the agent to the user query.

The response is provided below:

AGENT_RESPONSE

{{response}}

END_AGENT_RESPONSE

Relevance Judge Prompt (Question Answering) cont'd

****Your Task****

Evaluate the relevance of the response to the user query.

Relevant means closely connected or appropriate to what is being done or considered. Information may be incorrect or inaccurate, that is not related to relevance. To do this task, start by meticulously analyzing the agent's response against the user query and documents. The analysis should be comprehensive and cover all the information in the response that is relevant to the user query. Then proceed by assigning one of the scores below:

- 3: In this case, all the information provided in the agent response, every single sentence needs to be analyzed, are relevant. Furthermore, at least 50% of the response needs to be directly addressing the user query. This means that all the information in the response should be relevant in some sense but we allow for a maximum half of the response not to be directly relevant. Any evidence of relevance should be included in your analysis.

- 2: In this case, all the information provided in the agent response, every single sentence needs to be analyzed, are relevant. However, more than half of the response is not directly addressing the user's query.

- 1: In this case, the response contains information not relevant to the user query.

EXAMPLE

USER_QUERY

user: Is the sky blue?

END_USER_QUERY

AGENT_RESPONSE

Yes, the sky often appears blue during the day. This is due to a phenomenon called Rayleigh scattering. When sunlight enters Earth's atmosphere, it collides with molecules and small particles in the air. Blue light is scattered in all directions by the gases and particles in the atmosphere more than other colors because it travels as shorter, smaller waves. This scattered blue light is what we see when we look up at the sky.

END_AGENT_RESPONSE

Analysis: The response starts by directly responding to the user query. However, it proceeds by four sentence explaining why the sky is blue which is not directly addressing the query. Therefore, the entire response is relevant but only the first sentence is directly addressing the user's query. Therefore, based on the descriptions, it is acceptable.

Rating: Acceptable

END_EXAMPLE

Relevance Judge Prompt (Question Answering) cont'd

EXAMPLE

USER_QUERY

user: Is the sky blue?

END_USER_QUERY

AGENT_RESPONSE

Yes, the sky often appears blue during the day.

END_AGENT_RESPONSE

Analysis: The response directly answers the user query. It does not contain any extra information and therefore, is ideal according to the descriptions.

Rating: Ideal

END_EXAMPLE

EXAMPLE

USER_QUERY

user: Is the sky blue?

END_USER_QUERY

AGENT_RESPONSE

Yes, the sky often appears blue during the day. This is due to a phenomenon called Rayleigh scattering. When sunlight enters Earth's atmosphere, it collides with molecules and small particles in the air. Blue light is scattered in all directions by the gases and particles in the atmosphere more than other colors because it travels as shorter, smaller waves. This scattered blue light is what we see when we look up at the sky. We should note that blue is the favourite colour among most successful men and represents calmness and power inspired by the sky.

END_AGENT_RESPONSE

Analysis: The response starts by directly responding to the user query. However, it proceeds by four sentence explaining why the sky is blue which is not directly addressing the query. Furthermore, it contains an irrelevant sentence at the end. Therefore, the response is unacceptable because of the irrelevant sentence.

Rating: Unacceptable

END_EXAMPLE

OUTPUT FORMAT:

Please provide the response in a valid JSON format as shown below:

```
{
    "explanation":
    <your_explanation_for_the_rating_here>,
    "rating": "<your_score_here>"
}
```

Conciseness Judge Prompt (Question Answering)

User:

You are provided following documents served as the knowledge base for the user's question:

****DOCUMENTS****

Doc 1

{{kb_1}}

Doc 2

{{kb_2}}

Doc 3

{{kb_3}}

Doc 4

{{kb_4}}

****END_DOCUMENTS****

You are also provided with the below user query:

****QUERY****

{{query}}

****END_QUERY****

And you have the following response from an agent to answer the user's query.

****AGENT_RESPONSE****

{{response}}

****END_AGENT_RESPONSE****

****Your Task****

Your task is to evaluate if the agent's response is sufficiently concise.

- For more specific queries: The response should include the necessary details from the retrieved context to fulfill the user's intent or answer the question.
- For generic queries: The response should contain the topically relevant details from the retrieved context. Example of generic queries "401k", "benefits", "access" etc

Conciseness Judge Prompt (Question Answering) cont'd

Review the response carefully and consider these factors (in the order of importance):

1. Redundancy: Is there repetitive information or extra (non-informative or empathetic) language?
 2. Answer length: General questions should be answered with a general overview with no specific details. Specific queries should not be too long, but can be longer than those of the generic queries and are more detailed.
 3. Specificity: Does the level of detail match the question - do specific queries have specific answers and do generic queries have more generic answers? Are there overly specific details for a general question?
 4. Response to generic queries: When a query is ambiguous, each bullet point in the response should not describe what the document is about, but rather the information contained therein. If the query is specific, rate this factor as "good"
- For each factor, rate it as:

- Good: No issues found
- Fair: Minor issues that don't significantly impact concision
- Poor: Significant issues that make the response less concise

Based on your evaluation, assign one of the following scores:

- 3: All factors are rated as Good, or at most one is Fair
- 2: Most factors are Good or Fair, with at most one Poor
- 1: Two or more factors are rated as Poor

OUTPUT FORMAT:

Please provide the response in a valid JSON format as shown below:

```
{
  "explanation":
  <your_explanation_for_the_rating_here>,
  "rating": "<your_score_here>"
}
```

Citation Judge Prompt (Question Answering)

User:

You are tasked with judging whether a response contains appropriate citations given the reference documents.

****REFERENCE_DOCUMENTS****

Doc 1 {{kb_1}}

Doc 2 {{kb_2}}

Doc 3 {{kb_3}}

Doc 4 {{kb_4}}

****END_REFERENCE_DOCUMENTS****

****RESPONSE****

{{response}}

****END_RESPONSE****

****EVALUATION_CRITERIA****

Consider each sentence in the RESPONSE.

If the sentence makes a factual claim, check whether it has a corresponding citation at the end.

If the sentence does contain a citation, refer to that document and check if the factual claim is supported by information in the corresponding reference document.

For each sentence, if it requires a citation and the citation is supported by the reference document, that sentence is considered acceptable.

If the sentence requires a citation but the citation is not supported by the reference document, that sentence is considered unacceptable.

If the sentence does not require a citation, that sentence is considered acceptable.

Once you have evaluated each sentence, determine the overall score for the response given the guidelines below:

- 3: All sentences are rated as acceptable
- 2: The majority of the sentences are rated as acceptable.
- 1: Less than half of the sentences are rated as acceptable

****END_EVALUATION_CRITERIA****

OUTPUT FORMAT:

Please provide the response in a valid JSON format as shown below:

```
{  
    "explanation":  
<your_explanation_for_the_rating_here>,  
    "rating": "<your_score_here>"  
}
```

A.5 Dataset Samples

We show below samples from our dataset for each use case.

905

906

907

908

ID	Case Details
160	Terraform apply fails due to missing state lock release. Multiple apply attempts blocked with error "Resource is locked by another process". <i>Root cause:</i> Dangling lock after interrupted process. <i>Resolution:</i> Used 'terraform force-unlock', released lock, deployments unblocked.
321	ECR image scan not updating after new image push. Customer's new images pushed to Amazon ECR show "Scan status: Not scanned" despite scan-on-push enabled and manual scans. Found to be caused by the lifecycle policy deleting images before scan finishes. <i>Resolution:</i> Increased retention period in policy; scan completes and status updates as expected.
369	OAuth authentication failing due to redirect URI mismatch after promoting to production. Production logins failed with "redirect_uri_mismatch", worked in staging. <i>Root cause:</i> OAuth config for production missing authorized redirect URI. <i>Resolution:</i> Added correct URI; logins now work in both environments.

Table 7: Case Summarization - Sample Case Details. Case details are truncated for readability.

ID	Chat Transcript
160	<p>[15:02] Virtual Agent: Hello! I am here to help you with your Terraform issues.</p> <p>[15:03] Jeremy Clark: Hi, my terraform apply keeps failing. It says something about a state lock not being released.</p> <p>[15:04] Virtual Agent: It sounds like the state lock may not have released properly due to the interruption. Would you like to try unlocking the state file automatically?</p> <p>[15:05] Jeremy Clark: Yes, please.</p> <p>[15:06] System: Nicole Daniels has joined the chat.</p> <p>[15:06] Nicole Daniels: Hi Jeremy, I see you're running into an S3 state lock issue. Could you please share the exact error message you're seeing?</p> <p>[15:06] Jeremy Clark: Sure. It says: Error acquiring the state lock: Error message: state file is locked by another process.</p>
321	<p>[11:35] Virtual Agent: Hello! Welcome to AWS Support Chat.</p> <p>[11:36] Sameer Patel: Hi, I pushed a new image to my ECR repo, but the image scan status isn't updating. It still shows the previous scan result.</p> <p>[11:37] Virtual Agent: Are image scanning settings enabled for "api-backend"?</p> <p>[11:37] Sameer Patel: Yes, scanOnPush is enabled for this repo.</p> <p>[11:38] Virtual Agent: Got it. Let me connect you with a live agent for further investigation.</p> <p>[11:38] Lauren Choi: Hi Sameer, this is Lauren from AWS Support. I see you're having trouble with ECR image scan results not updating after a new image push.</p>
369	<p>[09:15] Virtual Agent: Hello! How can I assist you today?</p> <p>[09:16] Marissa Gilbert: Hi, I'm running into an OAuth redirect URI mismatch error in our staging environment, but prod is working fine.</p> <p>[09:17] Virtual Agent: Understood. Are you using the same OAuth provider for both environments?</p> <p>[09:18] Marissa Gilbert: Yes, Google OAuth for both, but with separate client IDs for staging and production.</p> <p>[09:19] Marissa Gilbert: I just checked Google Cloud Console. For staging I have "https://staging.ourapp.com/oauth/callback" registered. The app is using that exact URL.</p> <p>[09:29] Marissa Gilbert: Thanks for your help!</p> <p>[09:29] Virtual Agent: You're welcome! Have a great day.</p>

Table 8: Chat Summarization - Sample Chat Transcripts. Utterances truncated readability.

ID	Question	Knowledge Base Documents
160	I'm encountering Terraform state lock errors, and I'm not sure if it's due to my network being unreliable or because multiple operations are interfering with each other. How can I effectively diagnose and resolve these issues?	KB 1 Title: Troubleshooting Terraform Apply Failure Due to State Lock Not Released. KB 2 Title: Understanding and Preventing Terraform State Lock Contention Errors. KB 3 Title: Diagnosing and Resolving Orphaned Terraform State Locks in Remote Backends.
321	I'm noticing that after pushing a new image to ECR, my scan results don't update or still show outdated vulnerabilities. What could be causing this issue even though my repository has scan-on-push enabled?	KB 1 Title : ECR Image Scan Not Updating After New Push. KB 2 Title : Resolving ECR Image Scan Results Showing Old Vulnerabilities After New Image Push. KB 3 Title : Understanding ECR Image Scan Behavior with Immutable Tags and Multi-Architecture Images
369	I'm encountering intermittent 'redirect_uri_mismatch' errors during deployments across my staging and production environments, and the issue seems to occur inconsistently for different users. Can you explain the common causes and suggest measures to both resolve and prevent these errors?	KB 1 Title : Resolving OAuth Redirect URI Mismatch Between Staging and Production. KB 2 Title : Preventing OAuth Redirect URI Issues When Switching Between Staging and Production Environments. KB 3 Title : Understanding Security Implications of OAuth Redirect URI Mismatch Across Staging and Production.

Table 9: Question Answering - Sample Questions and Knowledge Base Document Titles. KB Document content removed for readability

ID	Question	Entity Schema (name, type: description)	Non-null Extracted Entities
160	I tried running a Terraform apply operation, but it keeps failing because of a missing state lock release. I'm using Terraform version 1.0.11 with a remote backend. It's happening in our staging environment. Can you help me understand why this is occurring?	error_message (<i>string</i> : The error message received during the operation) resource_name (<i>string</i> : Name of the resource being managed) operation_type (<i>choice</i> [apply, plan, destroy]: Type of operation being performed) state_lock_status (<i>choice</i> [locked, unlocked]: Status of the state lock) terraform_version (<i>string</i> : Version of Terraform being used) backend_type (<i>choice</i> [local, remote]: Type of backend used for storing state) retry_attempts (<i>number</i> : Number of retry attempts made) timeout_duration (<i>number</i> : Duration of timeout for the operation) user_permissions (<i>string</i> : Permissions of the user executing the operation) environment (<i>choice</i> [development, staging, production]: Environment in which the operation is being performed) issue_details (<i>other</i> : Detailed description of the issue)	error_message : missing state lock release operation_type : apply terraform_version : 1.0.11 backend_type : remote environment : staging issue_details : apply fails due to missing state lock release
321	Hi, I pushed a new Docker image to our ECR repository named "app-backend" last night, but the scan status is still showing as "pending." This is in the us-west-2 region. Could you help me understand why the scan isn't updating? My AWS account ID is 123456789012. Thanks!	repository_name (<i>string</i> : Name of the ECR repository) image_tag (<i>string</i> : Tag of the Docker image) scan_status (<i>choice</i> [pending, in-progress, complete, failed]: Status of image scan) scan_results (<i>other</i> : Results of the image scan) push_timestamp (<i>string</i> : Timestamp when the image was pushed) scan_timestamp (<i>string</i> : Timestamp when the scan was performed) scan_trigger (<i>choice</i> [manual, automatic]: Trigger for the image scan) error_message (<i>string</i> : Error message if scan failed) aws_region (<i>choice</i> [us-east-1, us-west-2, eu-west-1, ap-southeast-1]: Region of ECR repository) account_id (<i>string</i> : AWS account ID)	repository_name : app-backend scan_status : pending aws_region : us-west-2 account_id : 123456789012

Table 10: Entity Slot-Filling - Sample questions, entity schema (with types and descriptions), and ground-truth extracted entities.