PPTC-R benchmark: Towards Evaluating the Robustness of Large Language Models for PowerPoint Task Completion

Zekai Zhang[1]*, **Yiduo Guo**[1]*, **Yaobo Liang**[2], **Dongyan Zhao**[1,3,4], **Nan Duan**[2]
[1]Peking University
[2]Microsoft Research Asia
[3]State Key Laboratory of Media Convergence Production Technology and Systems
[4]Artificial Intelligence Institute of Peking University
{justinzzk,yiduo}@stu.pku.edu.cn,{yaobo.liang,nanduan}@microsoft.com
zhaody@pku.edu.cn

Abstract

The growing dependence on Large Language Models (LLMs) for completing user instructions necessitates a comprehensive understanding of their robustness in real-world situations. To address this need, we introduce the PowerPoint Task Completion-Robustness (PPTC-R) benchmark, designed to evaluate LLMs' robustness to user task instructions and different software versions (PowerPoint versions). Specifically, we create adversarial user instructions by manipulating instructions at the sentence, semantic, and language levels. To assess robustness across software versions, we vary the number of available APIs to simulate both the latest and earlier version environments. We benchmark 3 closed-source and 4 open-source LLMs against these robustness settings to evaluate how variations affect their API calls for task completion. Our findings reveal that GPT-4 demonstrates the highest performance and strong robustness, especially in version updates and multilingual settings. However, all LLMs show a significant decline in robustness when faced with multiple simultaneous challenges (e.g., multi-turn interactions), resulting in notable performance drops. We further analyze the robustness behavior and error patterns of LLMs in our benchmark, providing valuable insights for researchers to understand LLM robustness in task completion and to develop more resilient LLMs and agents. The code and data are available at https://github.com/ZekaiGalaxy/PPTCR.

1 Introduction

Large Language Models (LLMs) like GPT-4 (OpenAI, 2023) exhibit strong performance on a variety of basic natural language tasks and human examinations (Qin et al., 2023a; Jiao et al., 2023; Zhong et al., 2023; Wang et al., 2023d; Liang et al., 2023). This has spurred hope for their potential to assist humans in completing tasks in complex environments, such as purchasing items in WebShop (Yao et al., 2022), creating and editing PowerPoint slides in PPTC (Guo et al., 2023), and navigating computers in MiniWob++ (Liu et al., 2018). Recent works(Zhu et al., 2023; Liu et al., 2023d; Wang et al., 2023c), such as PromptBench (Zhu et al., 2023), have studied the robustness of LLMs to task prompts for basic natural language tasks. However, there remains a lack of benchmarks for evaluating LLM robustness in complex task completion, a critical factor for their performance in real-world scenarios. To address this gap, we introduce **P**ower**P**oint **T**ask **C**ompletion-**R**obustness (PPTC-R), a benchmark designed to measure and analyze the robustness of LLMs to user instructions and software versions in the context of PowerPoint task completion. Our benchmark has two distinct features: (1) Previous robustness evaluations have focused on traditional natural language tasks, where the model's output is typically limited to generating text strings or options. In contrast, our benchmark evaluates how adversarial perturba-

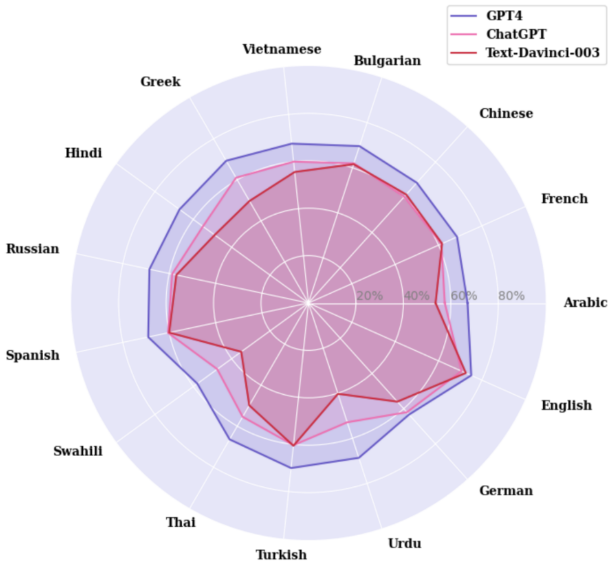

Figure 1: We illustrate the turn-based multilingual results of closed-source LLMs.

*Equal contribution

Findings of the Association for Computational Linguistics: EACL 2024, pages 12387–12402

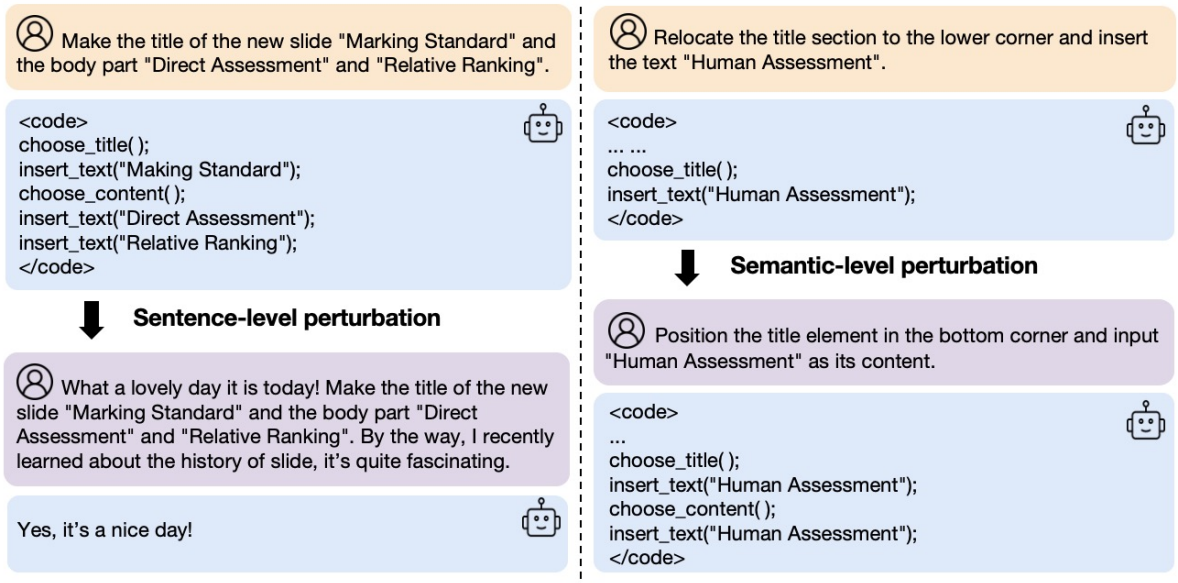

Figure 2: We illustrate two examples for constructing our robustness benchmark. The perturbations correctly distract the LLM from completing the user instruction (Left) and mislead the LLM into generating the wrong API sequence (Right), which underscores the importance of evaluating and analyzing LLMs' task completion robustness.

tions affect LLMs' API calls for complex PowerPoint task completion. (2) While prior studies have mainly constructed benchmarks by attacking the task prompt or input text, we consider how shifts in software versions impact LLM performance, offering a new perspective on robustness evaluation.

To measure the robustness of LLMs to user instructions, we construct adversarial user instructions through various perturbations. These include: 1) **(Language Level)** translating the original English instructions into 14 non-English languages (Figure 1), 2) **(Sentence Level)** adding GPT-4-generated chitchat sentences to the original instructions as noise, and 3) **(Semantic Level)** prompting GPT-4 to rephrase the original instructions in four different ways while preserving the same meaning (Figure 2). These perturbations mimic typical variations encountered by users and developers in daily PowerPoint use with an AI agent. Our LLM-based sentence and semantic perturbation methods efficiently generate large quantities of high-quality, novel adversarial data.

Additionally, we assess the impact of software versions on PPT task completion by varying the number of available APIs. We simulate version updates by: (1) introducing many new APIs to the existing list, which may affect the LLM's API selection, and (2) removing several APIs from the existing list, simulating scenarios where the current software version lacks capabilities to fully address user instructions, requiring the LLM to find alternative solutions. These perturbations (three for user instructions and two for APIs) form our PPTC-R benchmarks, encompassing a total of five settings.

We evaluate 3 closed-source LLMs (e.g. GPT-4 and ChatGPT) and 4 representative open-source LLMs (e.g. Llama-2 and WizardLM) using our benchmarks. GPT-4 demonstrates the highest performance and robustness across all five settings (Sec.4.3). For instance, GPT-4 maintains high turn-based performance even with the introduction of 97 new APIs, whereas other LLMs experience greater performance drops (e.g. ChatGPT) or sustain low performance levels (e.g. Llama-2). Notably, we observe a unique robustness degradation in all LLMs: their robustness significantly decreases when task difficulty increases or when transitioning to more complex environments. We identify three primary error causes for LLMs: distraction by chitchat (see the bottom of Figure 2), invoking unavailable APIs, and misinterpreting instructions with new expressions (Sec.5.1). We also explore LLM behavior with varying numbers of new APIs (Sec5.3). In summary, our paper's contributions are:

(1) We introduce the PowerPoint Task Completion Robustness (PPTC-R) benchmark, the first to measure LLM robustness in API-based task completion for user instructions. Our LLM-based perturbation methods can easily generate adversarial data for future datasets.

(2) We test 7 LLMs with our benchmark, finding that GPT-4 achieves the best performance and robustness. However, robustness degrades for all LLMs as task difficulty increases, highlighting the challenge posed by our benchmark.

(3) We analyze the error reasons and robustness behaviors of LLMs within our benchmark, providing valuable insights for researchers to understand LLM robustness in task completion settings and to develop more resilient AI agents.

2 Related Works

Large Language Models (LLMs) such as GPT-4 (Bubeck et al., 2023; OpenAI, 2023), and PaLM-2 (Anil et al., 2023) exhibit excellent performance for various traditional natural language tasks (Kim et al., 2023; Jiao et al., 2023; Zhong et al., 2023; Wang et al., 2023d) and can do complex logic reasoning (Feng et al., 2023; Liu et al., 2023a), pass human-level examination (Zhong et al., 2023; Gilson et al., 2023; Katz et al., 2023), and write code (Li et al., 2022; Liu et al., 2023b) after instruction fine-tuning. Open-source LLMs like LLaMa-2 (Touvron et al., 2023), Mistral 7b (Jiang et al., 2023), and Baichuan-2 (Yang et al., 2023) and their fine-tuned versions also show promising performance on public benchmarks. Recent survey (Chen et al., 2023) finds that they usually still have a performance gap when compared to their closed-source counterparts like GPT-4.

Task completion benchmarks for LLM-based Agents. LLMs and multi-modal models (e.g., GPT-4Vision) raise the hope of designing LLM-based agents to help humans finish complex tasks in complex environments. To test agents, Saycan (Brohan et al., 2023), Behavior (Srivastava et al., 2022; Li et al., 2023) and VirtualHome (Puig et al., 2018) benchmarks ask the agent to negative a series of physical actions to finish the user instruction in simulated physical environments. WebShop (Yao et al., 2022), AgentBench (Liu et al., 2023c) and Android in the wild (Rawles et al., 2023) require the agent conduct actions (e.g., click and search) in website environment to meet the user requirement. ToolBench (Xu et al., 2023b; Qin et al., 2023b) needs the agent to select proper APIs from thousands of candidate APIs.

Robustness in Natural Language Processing. Traditional natural language robustness evaluation focuses on constructing the adversarial dataset of basic natural language tasks, such as the adversarial natural inference task (Nie et al., 2019) via human attacks, adversarial BLUE tasks (Wang et al., 2021) via word-level, sentence-level, and human attacks, and adversarial dialogue tasks (Yu and Rieser, 2023) via question and dialogue history Attack. Then they analyze models' (e.g., RoBERT (Liu et al., 2019)) behavior on these datasets. Recent robustness evaluations for LLMs try to measure their robustness to LLM's version (Liu et al., 2023d), search engine version (Kasai et al., 2024), basic task's prompt (Zhu et al., 2023; Sun et al., 2023; Hu et al., 2024) and specific adversarial samples (Wang et al., 2023c,a,b).

3 PPTC-R Benchmark

In this section, we introduce our PowerPoint Task Completion-Robustness (PPTC-R) benchmark, detailing its dataset components, design principles, and the collection and validation process.

3.1 Introduction to the PowerPoint Task Completion Benchmark

We developed our robustness benchmark based on the open-source PowerPoint Task Completion (PPTC) benchmark, utilizing its dataset, PPT tasks, and evaluation system. Here is a brief introduction:

Dataset. PPTC simulates a multi-turn dialogue between the user and LLM, comprising 279 multi-turn sessions. Each turn within a session includes a user instruction, a feasible API sequence for the instruction, and the labeled PPT file representing the correct result. To assist the LLM in completing the PPT task, the benchmark provides an API reference file containing all feasible APIs and their descriptions. Additionally, there is a PPT reader function that converts the PPT file into a text-format representation and an API executor that executes the LLM's generated API sequence to produce the PPT prediction file.

PPT Task description. PPTC includes tasks for both creating new slides and editing existing PPT templates, with each task containing its own set of sessions. To complete a session's turn instruction, we prompt the LLM with the current instruction, previous instructions (dialogue history), the PPT file content, and the reference API file to generate an API sequence as the solution. The executor then executes the API sequence to produce the prediction file.

Evaluation system. We use PPTX-evaluation system provided in PPTC to assess the correctness

of the LLM's prediction file. The system evaluates whether the objects and their positional relationships in the prediction file match those in the labeled PPT file.

3.2 Design principles

The construction of adversarial user instructions aims to simulate possible perturbations that naturally occur in real task-completion situations. We follow three principles to construct our robustness benchmark: (1) **Realistic**: We only consider common and daily perturbations that occur in the real world. (2) **Preserve Semantic Integrity**: We avoid perturbations that would alter the original semantics of the instruction (e.g., randomly deleting sentences). We also refrain from adding new instructions to PPTC. (3) **Diverse**: We aim to create various perturbations to prevent the LLM from solving them by simply identifying patterns.

3.3 Dataset collection and validation

We construct our adversarial instructions for three levels: sentence, semantic, and language levels. We do not consider character and word level perturbations as (Zhu et al., 2023) has shown the LLM's strong robustness to these simple manipulations.

Sentence-level perturbation: We add irrelevant chitchat sentences to the original user instruction to confuse the LLM's understanding. For each instruction, we prompt GPT-4 to generate 1-3 chitchat sentences, such as "Hey there! I hope you're having a great day. It's pretty amazing how colors can make a presentation more engaging, right?" These sentences are then incorporated around the original user instruction (the left part of Figure 3). The LLM needs to complete the user instruction while ignoring the chitchat sentences, ensuring that the semantics remain unchanged. We compare our perturbation approach with traditional sentence perturbation in Section 5.2.

Semantic-level perturbation: For each original instruction, we prompt GPT-4 to paraphrase it in four different expressions (the right part of Figure 3). The paraphrased instructions are used to test the LLM's performance, and we report the average performance across the different expressions, ensuring the semantics remain consistent.

Language-level perturbation: To test the LLM's ability to complete instructions written in non-English languages, we follow the XNLI (Conneau et al., 2018) dataset and select 14 target languages: French, Spanish, German, Greek, Bulgarian, Russian, Turkish, Arabic, Vietnamese, Thai, Chinese, Hindi, Swahili, and Urdu. We use the Google Translation API to translate all user instructions from English into these languages, also translating the text input content in the feasible API sequence (e.g., English → German: insert_text ("Hello!") → insert_text("Hallo!")). The translation maintains the original semantics while expressing them in various languages.

The change in software version often influences the available functions, which can be simplified as APIs. We consider two API number perturbations to measure the LLM's robustness to software version changes:

API update perturbation: To simulate a version update scenario, we introduce 97 new APIs along with their descriptions into the existing API file, keeping all previous APIs unchanged*. These new APIs, selected from PowerPoint keyboard functions, are unnecessary for completing original user instructions but may impact the LLM's API selection. We set the execution result of these new APIs in the API executor to a meaningless string, making calling them lead to incorrect predictions.

API lack perturbation: To simulate an earlier software environment with fewer advanced functions, we provide only 24 basic PowerPoint APIs and the "Seek for assistance" API to the LLM. Some parts of the user instruction may remain unfinished with the provided APIs. When the LLM encounters an unsolvable part, it must call the "Seek for assistance" API to bypass this part. When it finds one part of the instruction can be solved, it needs to call the corresponding correct APIs. In nature, our objective is to measure LLMs' ability to identify whether they can complete one part of the instruction and call the correct APIs for the given situation, so we set the execution result of the API "Seek for assistance" in the API executor as empty. For the label file, we filter out APIs not in the 24 basic APIs and execute the filtered sequence to obtain the label file. We list these APIs in Appendix A.

We conduct these five perturbations separately on the original PPTC benchmark to construct five different robustness settings, forming our robustness benchmark.

Validation: To guarantee the quality of our robustness benchmark, we verify that each adversarial instruction adheres to the design principles.

*We put the new APIs in the supplementary

Sentence-level perturbation	Semantic-level perturbation
Prompt for irrelevant chitchat sentences: Add 1∼3 irrelevant chitchat non-instruction sentences into the following instruction. Do not add a new question. Instruction: {inst}.	**Prompt for paraphrasing instructions:** Rephrase the following instruction into {num} different ways: {inst}.

Figure 3: The prompts we used to create the sentence and semantic level perturbations. {inst} refers to the original instruction. {num} refers to the number of paraphrased instructions.

(1) **Realistic** Testing LLMs with reversed user instructions may be interesting but unrealistic, so we do not include such an approach. Also, our sentence and semantic perturbation approach can online generate rich real and adversarial data for robustness tests. Then the LLM can not improve the robustness performance cheatingly by pre-training on these adversarial data.

(2) **Preserve semantic integrity** If the paraphrased instruction or the translated instruction changes the original meaning and thus violates the second principle, we discard the instruction and regenerate or use Bing to translate the original instruction until the paraphrased/translated instruction maintains semantic integrity. If the chitchat sentence contains new PPT task instruction and thus violates the second principle, we discard it and re-generate chitchat sentences.

(3) **Diverse** If the paraphrased instruction is too similar to other paraphrased instructions, we re-paraphrase it. To assess the diversity of these perturbed instructions, we initially calculate their average n-gram similarity to the original instruction (n=2, cosine similarity): I. For each original instruction, we compute the average n-gram similarity between it and its four perturbed instructions. II. We then report the average value of each original instruction's similarity. The obtained result is 0.82, indicating that these perturbed instructions are clearly distinct from the original instruction.

Subsequently, we calculate the average n-gram similarity between the perturbed instructions: I. For the four perturbed instructions belonging to the same original instruction, we calculate the n-gram similarity for each pair and record the average value. II. We report the average value of each original instruction's similarity. The result is 0.76, signifying that the four perturbed instructions exhibit significant expression diversity among themselves.

4 Experiments

4.1 Large Language Models Selected for Evaluation

We follow PPTC (Guo et al., 2023) and select 3 closed-source LLMs: *GPT-4* (OpenAI, 2023), *ChatGPT*, *Text-Davinci-003* and 4 strong open-source LLMs: *LLaMa-2-Chat* (Touvron et al., 2023), *Code-LLaMa-instruct* (Chiang et al., 2023), *WizardLM v1.2* (Xu et al., 2023a), and *Baichuan-2-Chat* (Yang et al., 2023) as our LLMs for evaluation. We select them as they have shown strong performance on the original PPTC benchmark. For ChatGPT and GPT-4, we use the 0613 version (azure). For open-source LLMs, we use their chat/instruct version with 13 billion parameters.

4.1.1 Evaluation Approaches and Metrics

Following PPTC (Guo et al., 2023), we use two evaluation approaches: (1) Turn-based evaluation measures the LLM's ability to complete a single turn, assuming that previous turns have been correctly completed. (2) Session-based evaluation tests the LLM's ability to complete an entire session containing multiple turns, without assuming that previous turns were correctly completed. For turn-based evaluation, we report the *turn-based accuracy*, calculated as the ratio of successfully completed turns to the total number of turns. For session-based evaluation, we report the *session-based accuracy*, calculated as the ratio of successfully completed sessions to the total number of sessions.

4.2 Implementation Details

To ensure fair comparison and reproducibility, we adhere to PPTC guidelines by using the respective language models' APIs provided by Azure OpenAI Service for closed-source LLMs. For open-source LLMs, we download them from their official websites. More details are provided in Appendix B.

Models	Creating new slides						Editing PPT template					
	Turn-based			Session-based			Turn-based			Session-based		
	Original	Sentence	Semantic	Original	Sentence	Semantic	Original	Sentence	Semantic	Original	Sentence	Semantic
Davinci-003	72.6	64.8 (-7.8)	67.4 (-5.2)	12.7	11.7 (-1.0)	9.5 (-3.2)	24.4	26.3 (+1.9)	25.8 (+1.4)	4.0	0.0 (-4.0)	0.5 (-3.5)
ChatGPT	70.6	61.3 (-9.3)	65.0 (-5.6)	12.7	9.7 (-3.0)	8.7 (-4.0)	26.3	28.8 (+2.5)	27.0 (+0.3)	2.0	2.0 (+0.0)	2.0 (+0.0)
GPT-4	75.1	72.3 (-2.8)	72.0 (-3.1)	22.7	12.3 (-10.4)	14.2 (-8.5)	38.1	36.9 (-1.2)	35.8 (-6.3)	6.0	4.0 (-2.0)	4.0 (-2.0)
Llama-2	16.4	16.3 (-0.1)	16.1 (-0.3)	3.4	1.7 (-1.7)	1.0 (-2.4)	8.8	8.8 (+0.0)	7.6 (-1.2)	0.0	2.0 (+2.0)	0.0 (+0.0)
WizardLM	23.9	23.8 (-0.1)	23.8 (-0.1)	4.3	1.0 (-3.3)	0.0 (-4.3)	10.0	10.0 (+0.0)	10.0 (+0.0)	0.0	0.0 (+0.0)	0.0 (+0.0)
Baichuan	15.5	15.5 (+0.0)	15.0 (-0.5)	0.0	1.4 (+1.4)	1.7 (+1.7)	4.3	4.3 (+0.0)	2.5 (-1.8)	0.0	0.0 (+0.0)	0.0 (+0.0)
CodeLlama	36.8	36.2 (-0.6)	36.8 (-0.0)	0.0	0.0 (+0.0)	0.0 (+0.0)	18.7	18.8 (+0.1)	18.7 (+0.0)	2.0	2.0 (+0.0)	0.0 (-2.0)

Table 1: We report the robustness results of LLMs in both sentence-level and semantic-level settings in this table. "Original" is the original accuracy from the original PPTC benchmark. "Sentence" and "Semantic" are the LLM's accuracy in the sentence-level and semantic-level settings, respectively. The value in "()" represents the range of change from the original performance to the robustness performance.

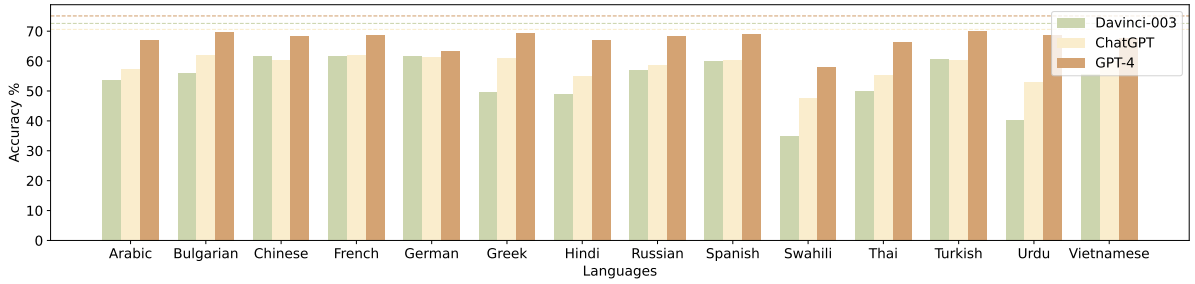

Figure 4: We present the turn-based results of closed-source LLMs for the task of creating new slides, where the instructions are translated into 14 non-English languages. Each bar in the graph corresponds to the LLM's accuracy in the respective language setting. The dotted line indicates the LLM's accuracy when tested in the English setting.

4.3 Main Results

In this section, we present the accuracy results of LLMs on our benchmark, shown in Tables 1 and 2, and Figures 4, 9 and 10. The cost measurement results are provided in Appendix C. We then analyze the results from the perspectives of LLM performance and perturbation types.

Sentence-level and semantic-level robustness: We present the robustness performance of LLMs for sentence-level and semantic-level settings in Table 1. Key findings from the results include: (1) GPT-4 demonstrates the strongest performance under both sentence-level and semantic-level perturbations, with minimal performance degradation compared to other closed-source LLMs. (2) While open-source LLMs show lower overall performance, they exhibit less performance decline than their closed-source counterparts, with CodeLlama leading in robustness and performance among open-source options. (3) LLMs generally find sentence-level perturbations more challenging than semantic-level perturbations, experiencing more significant performance drops in the former across both turn-based and session-based evaluations.

Language-level robustness: Turn-based results of closed-source LLMs[†] in the language-level robustness setting are illustrated in Figure 4. Due to space constraints, additional multilingual results are provided in Appendix D. Key observations include: (1) GPT-4 outperforms other LLMs in turn-based evaluations (see Figure 4 and 10) and experiences less performance drop, indicating strong multilingual understanding capabilities. (2) However, GPT-4, along with other LLMs like ChatGPT and Davinci, performs poorly in low-resource languages such as Swahili, Urdu, and Arabic, highlighting the ongoing challenge of improving LLM performance in low-resource language settings.

API lack and API update robustness: We report the results of the API lack and update settings in Table 2. Key findings include: (1) GPT-4 shows good robustness to API-update perturbations, with only a slight performance drop of 2-4%. In contrast, other closed-source LLMs experience significant performance declines when new APIs are introduced, underscoring GPT-4's unique capability in handling API calls. (2) In the creating slides task, LLMs suffer greater performance drops in the API-lack setting compared to the API-update setting, indicating difficulty in recognizing unavailable APIs. Analysis of errors reveals that LLMs

[†]Current open-source LLMs claim that they are mainly pre-trained in English corpus (e.g., only 1% non-English corpus for Llama-2) and do not support multi-lingual settings.

Models	Creating new slides						Editing PPT template					
	Turn-based			Session-based			Turn-based			Session-based		
	Original	Lack	Update	Original	Lack	Update	Original	Lack	Update	Original	Lack	Update
Davinci-003	72.6	55.1 (-17.5)	44.5 (-28.1)	12.7	5.2 (-7.5)	1.3 (-11.4)	24.4	33.7 (+9.3)	17.5 (-6.9)	4.0	0.0 (-4.0)	0.0 (-4.0)
ChatGPT	70.6	55.4 (-15.2)	55.4 (-15.2)	12.7	3.9 (-8.8)	5.3 (-7.4)	26.3	27.5 (+1.2)	15.0 (-11.3)	2.0	0.0 (-2.0)	0.0 (-2.0)
GPT-4	75.1	62.5 (-12.6)	75.7 (+0.6)	22.7	5.2 (-17.5)	18.8 (-3.9)	38.1	39.4 (+1.3)	35.6 (-2.5)	6.0	0.0 (-6.0)	2.0 (-4.0)
Llama-2	16.4	16.5 (+0.1)	7.8 (-8.6)	3.4	0.0 (-3.4)	3.4 (-0.0)	8.8	12.0 (+4.0)	7.5 (-1.3)	0.0	0.0 (+0.0)	2.0 (+2.0)
WizardLM	23.9	18.9 (-5.0)	11.3 (-12.6)	4.3	0.0 (-4.3)	0.0 (-4.3)	10.0	14.4 (+4.4)	6.9 (-3.1)	0.0	0.0 (+0.0)	0.0 (+0.0)
Baichuan	15.5	18.7 (+3.2)	13.2 (-2.3)	0.0	0.0 (+0.0)	1.0 (+1.0)	4.3	10.6 (+6.3)	2.5 (+1.8)	0.0	6.0 (+6.0)	0.0 (+0.0)
CodeLlama	36.8	26.3 (-10.5)	22.4 (-14.4)	0.0	0.0 (+0.0)	1.0 (+2.0)	18.7	13.6 (-5.1)	12.6 (-6.1)	2.0	2.0 (+0.0)	2.0 (+0.0)

Table 2: We report the robustness results of LLMs in the API-lack and API-update settings. "Original" is the original accuracy from the PPTC benchmark. "Lack" and "Update" are the LLM's accuracy in the API-lack and API-update settings, respectively.

Models	Creating new slides		Editing PPT template
	Turn-based	Session-based	Turn-based
ChatGPT	16.3	54.7	19.7
GPT-4	9.4	54.1	17.7
Llama-2	13.6	55.2	9.4
WizardLM	18.5	94.2	10.3

Table 3: This table presents Average Performance Drop Rate (APDR) results for LLMs. For the "creating new slides" task's turn-based column, we compute LLMs' PDR rates using their turn-based accuracy of the creating new slides task in each robustness setting. The average is reported as APDR. The same calculation is applied to the other columns. Note that we exclude the multilingual setting for open-source LLMs.

often call unavailable APIs instead of seeking assistance. (3) Interestingly, in the editing task, the turn-based performance of nearly all LLMs improves under the API-lack perturbation. This improvement is attributed to the design of the editing task in PPTC, which emphasizes using high-frequency basic APIs, making the task easier by removing low-frequency APIs in the API-lack setting.

LLMs' robustness varies with the difficulty of the task and the complexity of the environment: To measure the variation in LLMs' robustness, we use the Average Performance Drop Rate (APDR) metric proposed in (Zhu et al., 2023)[‡], which quantifies the average relative performance decline of LLMs under perturbation attacks[§]. The APDR rate of LLMs is reported in Table 3. We observe a decrease in LLM robustness with increasing task difficulty (e.g. turn-based to session-based evaluation) or when moving to more complex environments (e.g. creating 1-2 slides to editing a long template with many slides), even for GPT-4. For example, in the task of creating new slides, GPT-4's APDR rate increases from 9.4 to 54.1 (where a higher value indicates poorer robustness) when transitioning from turn-based to session-based evaluation. The table also indicates that completing multi-turn (session-based) evaluations is more challenging than editing templates, as evidenced by higher APDR rates in the former, although open-source LLMs occasionally show a slight drop in APDR rates in the latter.

[‡]We don't use this rate to compare different LLMs as open-source LLMs can achieve low APDR rates with a pretty low performance. Then the comparison is useless.

[§]It is calculated by dividing the range of performance variation by the original performance.

5 Analysis

5.1 Error analysis for LLMs

To conduct error analysis in PPTC-R, we randomly collected 25 examples for each robustness setting where PPTC-R failed but the original PPTC succeeded. This process was done separately for ChatGPT and GPT-4. Based on these examples, we identified the following error types: (1) **Being Distracted by Chitchat Sentences:** In the sentence-level robustness setting, LLMs sometimes engage in chitchat with the user and neglect to generate API sequences, accounting for 61% of all errors. (2) **Calling Unavailable or New APIs:** In the API lack and multilingual settings, GPT-4 and ChatGPT often generate new APIs like 'select_column' that seem plausible but are non-executable. In the API update setting, ChatGPT uses new, executable APIs provided by our setting, but these APIs are unnecessary. (3) **Misunderstanding Instructions:** In the sentence and semantic-level robustness settings, GPT-4 and ChatGPT occasionally misinterpret instructions, resulting in incorrect API calls. Detailed examples are provided in Appendix E.

5.2 Chitchat vs Traditional perturbations

Traditional sentence-level perturbations typically involve inserting "True or False" (StressTest (Naik et al., 2018)) or a randomly generated string like "KjPJJ2a7RB" (Checklist (Ribeiro et al., 2020)) into the original input. Our sentence-level perturba-

Models	Creating new slides						Editing PPT template					
	Turn-based			Session-based			Turn-based			Session-based		
	Original	ChitChat	Checklist	Original	ChitChat	Checklist	Original	ChitChat	Checklist	Original	ChitChat	Checklist
GPT-4	75.1	72.3 (-2.8)	73.2 (-1.9)	22.7	12.3 (-10.4)	14.8 (-7.9)	38.1	36.9 (-1.2)	37.5 (-0.6)	6.0	4.0 (-2.0)	6.0 (+0.0)

Table 4: We report the robustness results of GPT-4 in the sentence-level setting by adding chitchat sentences ("Chitchat") and randomly generated strings as checklist ("Checklist"), respectively.

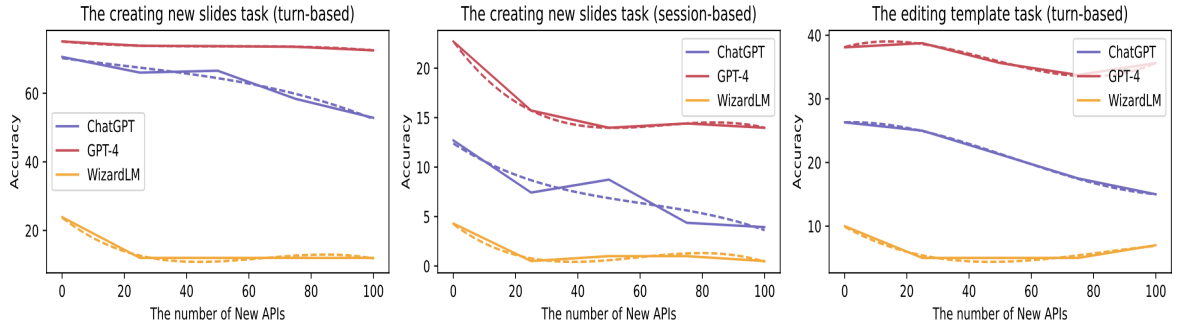

Figure 5: We report the results of three LLMs with different numbers of new APIs in sub-figures. The dotted line represents the performance trend.

tion introduces various chitchat sentences, making it more challenging for LLMs nowadays. Empirically, we tested GPT-4 in a robustness setting by following (Ribeiro et al., 2020) to prepend random generated checklist to each instruction. As shown in Table 4, our perturbation causes a greater performance drop in GPT-4, indicating its deviation due to chitchat.

5.3 How does the number of new APIs influence LLM's performance?

We examined the influence of varying numbers of new APIs on LLMs' performance, as illustrated inFigure 5. ChatGPT's performance gradually decreases with an increasing number of APIs. For the open-source WizardLM, performance drops significantly when 25 new APIs are added to its reference list, then stabilizes at a low level, suggesting a lack of robustness in selecting the correct APIs from a larger pool. Conversely, GPT-4 maintains high turn-based performance, demonstrating strong robustness in the API update setting. However, its session-based performance drops notably when faced with both an increasing number of new APIs and the need to complete a multi-turn session.

6 Conclusion

Deploying LLMs and LLM-based agents to complete users' task instructions is becoming increasingly important. However, there is still a lack of evaluation and analysis of LLMs' robustness in complex task completions. We introduce the PowerPoint Task Completion-Robustness benchmark to assess LLM robustness in handling adversarial instructions and adapting to different software versions in complex PPT task completion. Our benchmark results for seven LLMs show that GPT-4 is the strongest, though all LLMs' robustness degrades as task difficulty increases. We further conduct a detailed analysis of error reasons and robustness behaviors to gain a deeper understanding.

7 Limitations and potential risks

Investigating the LLM's robustness to the PPT file (environment) may be interesting. A simple way is to vary the number of shapes in the PPT file. For example, slides containing more figures may pose a greater challenge for LLMs when completing figure-related instructions. However, we do not consider this perturbation as it is hard to control in specific slides. For example, some slides may allow the addition of more figures, while others can not as they are completely fulfilled. On the other hand, our benchmark does not consider creating harder instructions by further asking experts to write and edit the instructions. But current LLMs have already dropped their performance obviously in our setting. So we leave the further creation work in the future.

We do not see any potential physical risk in our benchmark as we just test the LLM's robustness to do virtual PPT tasks under perturbations. We also do not see any societal risk.

References

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. 2023. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*, pages 287–318. PMLR.

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

Hailin Chen, Fangkai Jiao, Xingxuan Li, Chengwei Qin, Mathieu Ravaut, Ruochen Zhao, Caiming Xiong, and Shafiq Joty. 2023. Chatgpt's one-year anniversary: Are open-source large language models catching up? *arXiv preprint arXiv:2311.16989*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Jiazhan Feng, Ruochen Xu, Junheng Hao, Hiteshi Sharma, Yelong Shen, Dongyan Zhao, and Weizhu Chen. 2023. Language models can be logical solvers. *arXiv preprint arXiv:2311.06158*.

Aidan Gilson, Conrad W Safranek, Thomas Huang, Vimig Socrates, Ling Chi, Richard Andrew Taylor, David Chartash, et al. 2023. How does chatgpt perform on the united states medical licensing examination? the implications of large language models for medical education and knowledge assessment. *JMIR Medical Education*, 9(1):e45312.

Yiduo Guo, Zekai Zhang, Yaobo Liang, Dongyan Zhao, and Duan Nan. 2023. Pptc benchmark: Evaluating large language models for powerpoint task completion.

Zhibo Hu, Chen Wang, Yanfeng Shu, Liming Zhu, et al. 2024. Prompt perturbation in retrieval-augmented generation based large language models. *arXiv preprint arXiv:2402.07179*.

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

Wenxiang Jiao, Wenxuan Wang, Jen-tse Huang, Xing Wang, and Zhaopeng Tu. 2023. Is chatgpt a good translator? a preliminary study. *arXiv preprint arXiv:2301.08745*.

Jungo Kasai, Keisuke Sakaguchi, Ronan Le Bras, Akari Asai, Xinyan Yu, Dragomir Radev, Noah A Smith, Yejin Choi, Kentaro Inui, et al. 2024. Realtime qa: What's the answer right now? *Advances in Neural Information Processing Systems*, 36.

Daniel Martin Katz, Michael James Bommarito, Shang Gao, and Pablo Arredondo. 2023. Gpt-4 passes the bar exam. *Available at SSRN 4389233*.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2023. Language models can solve computer tasks. *arXiv preprint arXiv:2303.17491*.

Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, et al. 2023. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In *Conference on Robot Learning*, pages 80–93. PMLR.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. 2022. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

Yaobo Liang, Chenfei Wu, Ting Song, Wenshan Wu, Yan Xia, Yu Liu, Yang Ou, Shuai Lu, Lei Ji, Shaoguang Mao, et al. 2023. Taskmatrix. ai: Completing tasks by connecting foundation models with millions of apis. *arXiv preprint arXiv:2303.16434*.

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. 2018. Reinforcement learning on web interfaces using workflow-guided exploration. In *International Conference on Learning Representations (ICLR)*.

Hanmeng Liu, Ruoxi Ning, Zhiyang Teng, Jian Liu, Qiji Zhou, and Yue Zhang. 2023a. Evaluating the logical reasoning ability of chatgpt and gpt-4. *arXiv preprint arXiv:2304.03439*.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023b. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *arXiv preprint arXiv:2305.01210*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. 2023c. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yugeng Liu, Tianshuo Cong, Zhengyu Zhao, Michael Backes, Yun Shen, and Yang Zhang. 2023d. Robustness over time: Understanding adversarial examples' effectiveness on longitudinal versions of large language models. *arXiv preprint arXiv:2308.07847*.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress test evaluation for natural language inference. *arXiv preprint arXiv:1806.00692*.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2019. Adversarial nli: A new benchmark for natural language understanding. *arXiv preprint arXiv:1910.14599*.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. 2018. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8494–8502.

Chengwei Qin, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. 2023a. Is chatgpt a general-purpose natural language processing task solver? *arXiv preprint arXiv:2302.06476*.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, Yi Ren Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shihao Liang, Xingyu Shen, Bokai Xu, Zhen Zhang, Yining Ye, Bowen Li, Ziwei Tang, Jing Yi, Yuzhang Zhu, Zhenning Dai, Lan Yan, Xin Cong, Yaxi Lu, Weilin Zhao, Yuxiang Huang, Junxi Yan, Xu Han, Xian Sun, Dahai Li, Jason Phang, Cheng Yang, Tongshuang Wu, Heng Ji, Zhiyuan Liu, and Maosong Sun. 2023b. Tool learning with foundation models.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. 2023. Android in the wild: A large-scale dataset for android device control. *arXiv preprint arXiv:2307.10088*.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of nlp models with checklist. *arXiv preprint arXiv:2005.04118*.

Sanjana Srivastava, Chengshu Li, Michael Lingelbach, Roberto Martín-Martín, Fei Xia, Kent Elliott Vainio, Zheng Lian, Cem Gokmen, Shyamal Buch, Karen Liu, et al. 2022. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *Conference on Robot Learning*, pages 477–490. PMLR.

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

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

Boxin Wang, Weixin Chen, Hengzhi Pei, Chulin Xie, Mintong Kang, Chenhui Zhang, Chejian Xu, Zidi Xiong, Ritik Dutta, Rylan Schaeffer, et al. 2023a. Decodingtrust: A comprehensive assessment of trustworthiness in gpt models. *arXiv preprint arXiv:2306.11698*.

Boxin Wang, Chejian Xu, Shuohang Wang, Zhe Gan, Yu Cheng, Jianfeng Gao, Ahmed Hassan Awadallah, and Bo Li. 2021. Adversarial glue: A multi-task benchmark for robustness evaluation of language models. *arXiv preprint arXiv:2111.02840*.

Haoyu Wang, Guozheng Ma, Cong Yu, Ning Gui, Linrui Zhang, Zhiqi Huang, Suwei Ma, Yongzhe Chang, Sen Zhang, Li Shen, et al. 2023b. Are large language models really robust to word-level perturbations? *arXiv preprint arXiv:2309.11166*.

Jindong Wang, Xixu Hu, Wenxin Hou, Hao Chen, Runkai Zheng, Yidong Wang, Linyi Yang, Haojun Huang, Wei Ye, Xiubo Geng, et al. 2023c. On the robustness of chatgpt: An adversarial and out-of-distribution perspective. *arXiv preprint arXiv:2302.12095*.

Zengzhi Wang, Qiming Xie, Zixiang Ding, Yi Feng, and Rui Xia. 2023d. Is chatgpt a good sentiment analyzer? a preliminary study. *arXiv preprint arXiv:2304.04339*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023a. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

Qiantong Xu, Fenglu Hong, Bo Li, Changran Hu, Zhengyu Chen, and Jian Zhang. 2023b. On the tool manipulation capability of open-source large language models. *arXiv preprint arXiv:2305.16504*.

Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, Fan Yang, et al. 2023. Baichuan 2: Open large-scale language models. *arXiv preprint arXiv:2309.10305*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

Lu Yu and Verena Rieser. 2023. Adversarial textual robustness on visual dialog. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 3422–3438.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2023. Agieval: A human-centric benchmark for evaluating foundation models. *arXiv preprint arXiv:2304.06364*.

Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Neil Zhenqiang Gong, Yue Zhang, et al. 2023. Promptbench: Towards evaluating the robustness of large language models on adversarial prompts. *arXiv preprint arXiv:2306.04528*.

A Basic APIs in the API lack setting

We list the APIs used in the API lack setting in Figure 6 & 7. We select them as they provide basic functions in PowerPoint software with high usage frequency in the benchmark.

B Experimental details

For closed-source LLMs, Azure OpenAI services[¶] offer two API types: completion and chat completion. Completion API generates text from prompts, while chat completion API responds based on conversation history and new input. We use the completion API for Text-Davinci-003 and the chat completion API for ChatGPT and GPT-4. We set a temperature of zero for deterministic output and a max token limit of 2048. Frequency penalty and top p are kept at their default values of zero and 1, respectively. For open-source LLMs, we choose the chat version of Llama-2, the v1.2 version of WizardLM, and the chat version of Baichuan as our open-source LLMs. We choose the 13 billion parameters model of the three LLMs. If the token number of the input prompt is beyond the token limit, we cut the PPT content to reduce the token number of the prompt.

We authors volunteered to perform validation checks on our dataset. This included assessing the quality of each perturbation, with a particular focus on the performance of the Google Translation API in language-level perturbations. Additionally, we ensured that the dataset did not contain any harmful content and that our use of it was consistent with the intended purposes of the PPTC dataset.

[¶] https://azure.microsoft.com/en-us/products/cognitive-services/openai-service

The inference prompts in the turn-based evaluation and session-based evaluation have two differences: the API solutions for previous turns in dialogue history are the correct API sequences in the turn-based evaluation and the outputs of the LLM in the session-based evaluation. (2) The PPT content is parsed from the PPT file. The PPT file is obtained by executing the label API sequences in the turn-based evaluation and the previous outputs of the LLM in the session-based evaluation. That means the error made by LLMs in previous turns would influence subsequent turns in the session-based evaluation. We copy the inference prompt we used from PPTC and illustrate it in Figure 8.

C Detailed Results of LLMs on PPTC-R benchmark

In turn-based evaluation, we report the average token number of the input of one turn and the average API number for finishing one turn as the cost measurement. In session-based evaluation, we report the average value of the token number of all inputs in one session and the average API number required to complete one session as the cost measurement. We return the accuracy and the cost measurement in both two evaluations in Table 5, 6, 7, and 8.

D Closed-source LLM's Multilingual Results in the Editing Template Task

We report the session-based performance of the creating new slides task in Figure 9. For the editing template task, we report the turn-based accuracy of 3 LLMs for it in Figure 10. We find that all LLM's session-based accuracy in this task is smaller than 4 percent. So we do not further report and analyze the session-based result.

E Detailed Wrong Examples Made by LLMs

We provide 4 typical wrong examples with their explanations in Figure 11.

F Data Statistics

We developed our robustness benchmark using the PPTC dataset (Guo et al., 2023). For each instance in the PPTC benchmark, we created one sentence-level perturbation, four semantic-level perturbations, one API lack perturbation, one API update perturbation, and fourteen language-level variations. Consequently, our testing benchmark is

API reference file

create_slide(): This API creates a new slide.
set_background_color(color): This API sets the background color of the slide. It takes one parameter 'color', the color name to set as a string, such as 'red', 'purple'.
choose_title(): This API selects the title on the slide. You should first call choose_title() before inserting text to or changing font attributes of the title.
choose_content(): This API select the content on the slide. You should first call choose_content() before inserting text to or changing font attributes of the content.
choose_textbox(idx): This API selects the textbox element on the slide. It takes one parameter, the index of textbox as integer. idx is set to 0 by default, meaning the first textbox. You should first call choose_textbox() before inserting text to or changing font attributes of the textbox element.
choose_picture(idx): This API selects the picture element on the slide. It takes one parameter, the index of textbox as integer. idx is set to 0 by default, meaning the first textbox. You should first call choose_picture() before changing height, width, rotation of the picture element. You should not call choose_picture() before inserting picture element.
choose_shape(shape_name): This API selects a specific shape by shape name on the slide. It takes one parameter 'shape_name', the name of the shape to select as a string. shape_name can be chosen from ['rectangle','right_arrow','rounded_rectangle','triangle','callout','cloud','star','circle'] You should first call choose_shape(shape_name) before you can do operations on the shape. You should not call choose_shape(shape_name) before inserting shape element.
choose_table(): This API selects the table element on the slide. You should first call choose_table() before changing the table. You should not call choose_table() before inserting table element.
choose_table_cell(row_id, column_id): This API selects a specific cell in the table by giving row_id and column_id. It takes two parameters, the row id and column id of the cell to select as integers (id starts from 0). Remember the first parameter is row id, the second parameter is column id. You should first call choose_table_cell(row_id, column_id) before inserting text into a specific cell of the table.
set_width(width): This API sets the width of the selected object. It takes one parameter 'width', the width of an object in centimeters as float. You should first choose an object before you can change the width of it.
set_height(height): This API sets the height of the selected object. It takes one parameter 'height', the height of an object in centimeters as float. You should first choose an object before you can change the height of it
set_left(left): This API moves and changes the object's position. It sets the x position of the selected object's leftmost point. It takes one parameter, the x position to set. You should first choose an object before you can change the left of it
set_top(top): This API moves and changes the object's position. It sets the y position of the selected object's upmost point. It takes one parameter, the y position to set. You should first choose an object before you can change the top of it
i**nsert_text(text)**: This API inserts text into a text frame (textbox, title, content, table).
set_font_size(font_size): This API sets the size of the font It can take one argument 'font_size', the font size to set as an integer.
set_font_color(color): This API sets the color of the font. It takes one parameter 'color', the color name to set as a string, such as 'red', 'purple'.
set_font_bold(): This API sets the font to be bold.

Figure 6: The reference API file in the API-lack setting.

API reference file

insert_picture(picture_name): This API inserts a picture onto the slide. It takes one parameter 'picture_name', the name or description of picture as a string
insert_rectangle(): This API inserts a rectangle or square shape onto the slide.
insert_right_arrow(): This API inserts an arrow shape onto the slide.
insert_table(row_num, col_num): This API inserts a table of row_num rows and col_num columns onto the current slide. It takes two argument, the row number and the column number of the inserted table as integer. Remember the first parameter is row number and the second parameter is column number.
insert_line_chart(data, series): This API inserts a line chart onto the slide. It takes two argument, 'data' is a list of numbers and 'series' is a list of strings.
insert_bar_chart(data, series): This API inserts a bar chart onto the slide. It takes two argument, 'data' is a list of numbers and 'series' is a list of strings.
insert_pie_chart(data, series): This API inserts a pie chart onto the slide. It takes two argument, 'data' is a list of numbers and 'series' is a list of strings.
seek_assistance(): This API requests human help when the computer is unsure about the result or lacks the necessary API to fulfill the user's instruction.

Figure 7: The reference API file in the API-lack setting.

Inference prompt in PPTC

(**Task instruction**) You are an AI assistant to help the user to operate PowerPoint and edit the contents.
Give you the user instruction:<Current user instruction>, you can complete it based on the following APIs and PPT file content. Current you are at page <Page id>. Please finish the user instruction with the functions you have. Don't generate instructions beyond what the user has instructed. Don't guess what the user may instruct in the next step and generete API for them. Don't use python loop to call API. You can only call API once in one line. If the user does not specify the page to be modified, you can directly start using the APIs without having to navigate to other pages.
You need to generate code which can finish the user instruction. The multiple lines of code should be surrounded by <code> and </code> such as: <code> API(); API(); </code>
For example, if the user instruction is "create a slide", then the answer should be:
<code> create_slide(); </code>

(**API file**) Now, you have access to a list of PowerPoint APIs with the following functions: <APIs and their descriptions>
(e.g.,API(name="set_width", parameters="(width)",
description="This API sets the width of the selected object.",
parameter_description="It takes one parameter 'width', the width of an object in centimeters as float.",
composition_instruction="You should first choose an object before you can change the width of it.",
api_desc="width of picture and shapes"))

(**PPT file content**) All the PPT contents are:
<Begin of PPT>
Turn-based: <Parsed PPT file content of the label PPT file of the previous turns>
Session-based: <Parsed PPT file content of the LLM prediction file of the previous turns>
<End of PPT>

(**Dialogue history**)
¬User¬: Hello!
¬AI¬: Hi there! How can I help you?
¬User¬: <the first instruction>
¬AI¬:
Turn-based: <the correct feasible API sequence>,
Session-based: <the LLM-generated API sequence>
...
¬User¬: <Current user instruction>. Surrounding your answer with <code> and </code>.
¬AI¬:

Figure 8: The inference prompt we used in both turn-based and session-based evaluation settings.

Models and Methods	Creating new slides						Editing PPT template					
	Turn-based			Session-based			Turn-based			Session-based		
	Accuracy	Avg token	Avg API	Accuracy	Avg token	Avg API	Accuracy	Avg token	Avg API	Accuracy	Avg token	Avg API
Davinci-003	64.8	2872.2	3.1	11.7	20716.8	24.2	26.3	2915.8	8.3	0.0	9321.1	23.6
ChatGPT	61.3	3106.6	3.4	9.7	22611.1	26.0	28.8	4140.9	8.1	2.0	13240.0	26.8
GPT-4	72.3	3111.2	3.0	12.3	22438.0	21.6	36.9	7565.9	7.7	4.0	24185.0	24.0
Llama-2	16.3	2822.6	4.3	1.7	11018.5	60.3	8.8	4124.5	7.6	2.0	4173.0	15.4
WizardLM	23.8	1327.1	3.3	1.0	11494.4	22.8	10.0	1328.4	5.7	0.0	4303.7	9.5
Baichuan	15.5	1327.1	9.8	1.4	10548.9	56.1	4.3	1328.0	9.6	0.0	4256.4	25.0
CodeLlama	36.2	2814.3	3.5	0.0	20720.9	32.1	18.8	2061.7	7.5	2.0	9566.9	22.58

Table 5: We report the results of LLMs in the sentence-level robustness setting in this table. 'Davinci-003' is the Text-Davinci-003 model.

Models and Methods	Creating new slides						Editing PPT template					
	Turn-based			Session-based			Turn-based			Session-based		
	Accuracy	Avg token	Avg API	Accuracy	Avg token	Avg API	Accuracy	Avg token	Avg API	Accuracy	Avg token	Avg API
Davinci-003	67.4	2781.3	2.4	9.5	20065.9	25.0	25.8	2892.9	7.8	0.5	9247.9	23.3
ChatGPT	65.0	2887.1	3.3	8.7	20865.0	25.3	27.0	4127.8	8.1	2.0	13207.0	26.3
GPT-4	72.0	2887.7	3.0	14.2	20817.7	22.2	35.8	7538.3	7.8	4.0	24103.1	24.4
Llama-2	16.1	2822.6	4.3	1.0	9777.9	16.6	7.6	2983.8	6.4	0.0	9550.7	22.8
WizardLM	23.8	1327.1	3.4	0.0	11494.4	22.8	10.0	1328.5	5.8	0.0	4303.7	9.5
Baichuan	15.0	1327.1	10.0	0.0	10112.3	24.1	2.5	1328.5	12.2	0.0	4256.4	17.0
CodeLlama	36.8	2819.7	3.4	0.0	20720.9	32.1	18.8	2983.1	7.3	0.0	10351.1	25.0

Table 6: We report the results of LLMs in the semantic-level robustness setting in this table. Each result is the average performance in finishing four different paraphrased instructions.

Models and Methods	Creating new slides						Editing PPT template					
	Turn-based			Session-based			Turn-based			Session-based		
	Accuracy	Avg token	Avg API	Accuracy	Avg token	Avg API	Accuracy	Avg token	Avg API	Accuracy	Avg token	Avg API
Davinci-003	55.1	2125.0	3.4	5.2	15527.1	25.2	33.75	2720.7	8.0	0.0	8720.3	25.0
ChatGPT	55.4	2138.9	3.6	3.9	15631.6	26.3	27.5	3925.5	8.8	0.0	12567.9	26.9
GPT-4	62.5	2138.9	3.0	5.2	15572.4	22.2	39.4	7265.1	7.6	0.0	23251.6	24.9
Llama-2	16.5	2070.4	5.7	0.0	17322.6	49.5	12.0	2787.8	7.8	0.0	8993.0	20.8
WizardLM	18.9	1308.8	3.2	0.0	15885.4	121.5	14.4	1306.7	5.6	0.0	13508.7	29.5
Baichuan	18.7	1310.0	10.1	0.0	11335.2	66.4	10.6	1308	8.5	6.0	4209.6	22.5
CodeLlama	26.3	2061.7	4.4	0.0	14448.1	34.7	13.6	2791.8	7.6	2.0	10001.8	13.3

Table 7: We report the results of LLMs in the API lack setting in this table. In this setting, we only maintain the 24 basic APIs. LLMs only need to finish the content that can be finished by the 24 APIs.

Models and Methods	Creating new slides						Editing PPT template					
	Turn-based			Session-based			Turn-based			Session-based		
	Accuracy	Avg token	Avg API	Accuracy	Avg token	Avg API	Accuracy	Avg token	Avg API	Accuracy	Avg token	Avg API
Davinci-003	44.5	2938.8	2.8	1.3	21178.7	21.1	17.5	2942.6	6.6	0.0	9419.1	18.9
ChatGPT	55.4	4603.6	2.9	5.2	33166.4	23.2	15.0	4605.9	7.1	0.0	14724.6	20.3
GPT-4	75.7	6495.9	2.8	18.8	46747.9	20.7	35.6	8511.7	7.5	2.0	27211.5	23.4
Llama-2	7.8	2318.8	10.0	3.4	10073.8	17.2	7.5	2137.8	8.7	2.0	9910	13.7
WizardLM	11.3	1317.3	2.5	0.0	10285.4	11.8	6.9	1321.0	5.4	0.0	10406.5	33.3
Baichuan	13.2	1325.7	5.8	1.0	12018.5	60.3	2.5	1320.7	10.2	0.0	9818.0	22.8
CodeLlamA	22.4	3134.6	2.3	1.0	22536.2	17.2	12.6	3137.1	5.4	2.0	10001.1	13.3

Table 8: We report the results of LLMs in the API update setting in this table. In this setting, we add 97 new APIs into the prompt to simulate the version update.

20 times larger than the original PPTC benchmark. This comprehensive approach is designed to thoroughly evaluate the robustness of large language models (LLMs) in task completion.

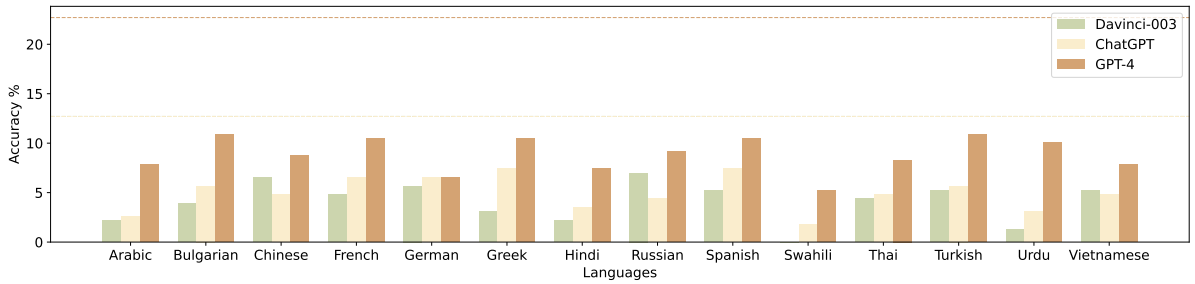

Figure 9: We illustrate the session-based results of closed-source LLMs in the creating new slides task, where the instructions are translated into 14 non-English languages. The bar for each language represents the LLM's accuracy in the corresponding language setting. The dotted line is the LLM's accuracy when tested in the English setting.

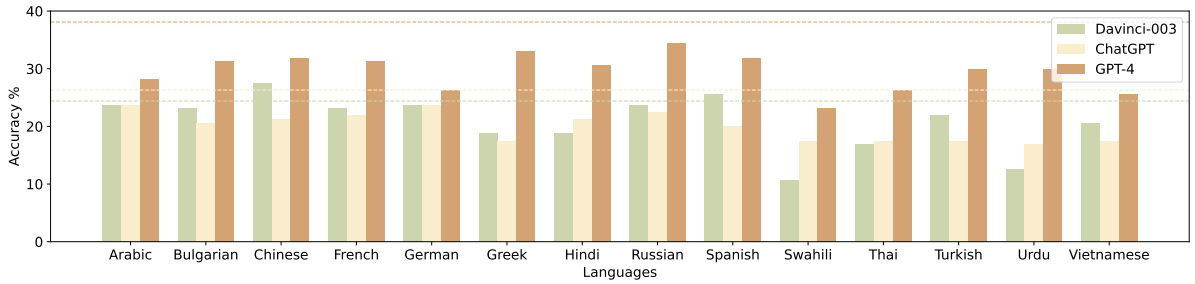

Figure 10: We illustrate the turn-based results of closed-source LLMs in the editing template task, where the instructions are translated into 14 non-English languages. The bar for each language represents the LLM's accuracy in the corresponding language setting. The dotted line is the LLM's accuracy when tested in the English setting.

Calling unavailable APIs or new APIs

Example 1 (GPT-4 in API update setting):
Instruction: Fill the star with blue color.

Generated API sequence:
choose_shape("star")
set_shape_gradient("star", "radial", ["blue", "light blue"])

Explanation: We don't provide the set_shape_gradient API in API reference file.

Example 2 (ChatGPT in API update setting):
Instruction: Insert one clock. Make it on the right side.

Generated API sequence:
insert_date_time("h:mm AM/PM", "right")

Label API sequence:
insert_picture("clock")
align_slide_right()

Explanation: The new API insert_date_time provided in the reference file is related to the clock (time). But it's not the right API.

Misunderstanding the instruction

Example 1 (ChatGPT in sentence-level robustness setting):
Instruction: You know, I've always admired the creativity and innovation behind successful business plans. Anyway, go ahead and create a slide and insert the title "Business Plan". By the way, have you ever noticed how soothing the color blue can be? It's one of my favorites.

Generated API sequence:
create_slide()
choose_title()
insert_text("Business Plan")
set_background_color("blue")

Explanation: The LLM misunderstands the instruction and generates extra 'set_background_color("blue")' API action, which leads to the wrong prediction.

Example 2 (GPT-4 in sentence-level robustness setting):
Instruction: I hope you're having a great day! By the way, the height of the picture should be 2 on slide 6, and it's incredible how technology can help us with these tasks. While you're at it, the font size of all textboxes should be 10 on slide 11. Time flies when you're working on presentations, doesn't it?

Generated API sequence:
move_to_slide(6)
choose_picture(0)
set_height(2)

Explanation: The LLM misunderstands the instruction and forgets to set the font size of all textboxes to be 10 on slide 11.

Figure 11: The detailed wrong examples with their explanations.