Do LLMs have an Anti-exception Reasoning Ability for Planning

Anonymous ACL submission

Abstract

Human-like planning aims to predict an action sequence given a task. The existing studies have demonstrated the potentials of Large Language Models (LLMs) upon human-like planning. However, it has not been verified whether LLMs are capable of overcoming an exceptional situation. Therefore, we carry out a preliminary study on Anti-Exception Planning (AEP) task. Specifically, we build AEP datasets using semi-artificial and automatic labeling approaches. On this basis, we evaluate AEP performance of different LLMs (Vicuna, Qwen, LLaMA, GPT-4o and DeepSeek-R1) within the Generation-Retrieval-Ranker (GRR) framework. In addition, we propose a reverse engineering approach to enhance GRR. Experiments show that LLMs tackle exceptions less effectively. The success rate of exception attack is up to 93.64% at worst, although the reverse engineering-based GRR yields substantial improvements. We will make all datasets publicly available to support future studies.

1 Introduction

Human-like planning is required to infer a stepwise action sequence (namely **plan**) that enables the accomplishment of a specific task (e.g., "*brewing coffee*"), where each **action** is embodied with a sentence (e.g., "*grinding coffee beans*") (Huang et al., 2022; Lin et al., 2023; Guo et al., 2024). The recent studies have proven the potentials of LLMs in human-like planning (Ahn et al., 2022; Zhao et al., 2023; Shen et al., 2023; Hao et al., 2023; Yao et al., 2023; Wu et al., 2023; Yuan et al., 2023; Yang et al., 2023; Tang et al., 2023; Li et al., 2023; Guo et al., 2024; Liu et al., 2025; Wen et al., 2025; Hao et al., 2025; Li et al., 2025).

Inspired by the autonomous intelligence (LeCun, 2022), this paper extends the aforementioned research framework by supplementing an AEP task. AEP purposely imposes an exception upon the plan (namely **exception attack**), and meanwhile asks for solutions to handle or bypass the exception. We show a pair of exception and solution in (1).

(1) **Exception**: *Coffeemaker is broken.*
Solution: *Repair the coffeemaker.*

It is difficult to systematically study AEP due to the absence of an applicable dataset. To address the issue, we construct AEP datasets using the open-grounded planning benchmark corpus (Guo et al., 2024). Considering that semi-artificial data labeling is time-consuming, we develop a dual-agent progressive labeling model. It enables efficient and low-cost data annotation (2.5K instances per hour).

We evaluate the anti-exception capabilities of different LLMs, including Vicuna (Zheng et al., 2023), LLaMA3.1-8B (Grattafiori et al., 2024), Qwen2.5-7B (Yang et al., 2024), DeepSeek-R1 (Guo et al., 2025) and GPT-4o (Hurst et al., 2024). Evaluation is performed at the zero-shot setting without using Supervised Fine-Tuning (SFT). Two anti-exception solution acquisition frameworks are used, including GRR and reverse engineering-based GRR. Experiments on AEP datasets demonstrate the crucial aspects as follows.

- LLMs are less capable of generating practicable solutions to tackle exceptions. The high success rates of exception attacks (Section 4) expose the significant challenge in autonomous intelligence enhancement.
- The reverse engineering approach substantially improves the performance of anti-exception solution acquisition.

2 Task Definition of AEP

Assume that $\mathcal{T}$ is a task objective, $\mathcal{P}$ is known to be an effective plan for accomplishing $\mathcal{T}$, and $\mathcal{P}$ consists of n step-wise actions, i.e., $\mathcal{P}$=[$\mathcal{A}_1$...$\mathcal{A}_n$]. Thus, AEP imposes an exception $\mathcal{E}$ upon the i-th action $\mathcal{A}_i$, and asks for the solution $\mathcal{S}$ to handle $\mathcal{E}$

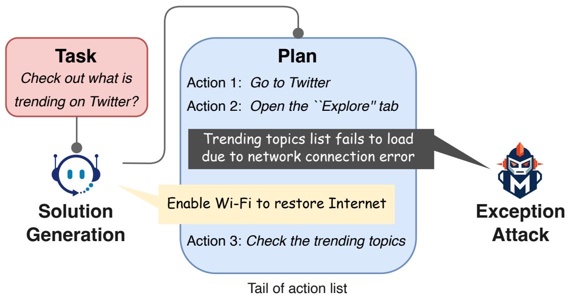

Figure 1: Example of AEP attacked from the tail.

before conducting $\mathcal{A}_i$. We show an AEP example in Figure 1, where the exception $\mathcal{E}$ is used to disable the third action $\mathcal{A}_3$ in $\mathcal{P}$.

In this paper, we limit AEP to a game that suffers from a tail-end exception attack, in which the exception is uniformly imposed on the final action of $\mathcal{P}$. This game avoids exception propagation to a wide range of subsequent actions. Note that exception propagation causes redundant solutions and, more seriously, the confusion on the alignment between exceptions and solutions. This makes it difficult to precisely evaluate AEP models.

3 AEP Corpus

We construct AEP corpus using the chapter "WikiHow" (Zhang et al., 2020) of the publicly-shared benchmark OGP [1] (Guo et al., 2024), which contains about 7.5K pairs of tasks and plans, as well as a large action space that holds nearly 39K executable actions. Given an OGP sample (i.e., "task-plan" pair), we produce its aligned AEP instance by labeling exceptions and solutions for the tail-end action of plan. Both semi-artificial and automatic labeling strategies are used as follows.

Semi-artificial Labeling— For a tail-end action $\mathcal{A}_n$, we prompt GPT-4o (Hurst et al., 2024) to generate m exceptions for disabling $\mathcal{A}_n$. Both the task $\mathcal{T}$ and preceding actions (i.e., $[\mathcal{A}_1 ... \mathcal{A}_{n-1}]$) in plan $\mathcal{P}$ are used as context during prompting GPT-4o.

The annotators who major in linguistics assess the quality of exceptions in accordance with 1) their relevance to $\mathcal{T}$ and $\mathcal{P}$, and 2) interference effect and reasonability. The quality level is labeled with scores raning from 0 to 3, where a score of "2" indicates a qualified exception that meets the quality criteria of relevance and practicability. A score of "3" aligns with an exception that is not only qualified but credible, where reliable evidence has been found to prove its interpretability. In the same way, we produce solutions for each qualified exception, where quality criteria are revised as 1) relevance to the exception, 2) practicability in handling the exception, and 3) interpretability. We provide all details of semi-artificial labeling in Appendix A, including the assessment criteria, annotation scheme, and training programme, etc.

By semi-artificial labeling, we produce about 4.4K AEP samples for 200 tasks that derive from 19 categories of life skills. There are 94.51% examples labeled as the qualified cases. Two groups of well-trained annotators (3 members per group) engage in quality assessment, who achieve a Kappa value of 85.38% for agreement.

Automatic Labeling— Semi-artificial labeling is time-consuming (10 tasks per hour). This makes it difficult to efficiently construct a larger AEP dataset. To solve the problem, we train two agents to perform automatic labeling. One agent (namely generator G_α) serves to generate AEP samples (i.e., "exception-solution" pairs). The other agent (viz., assessor G_β) marks AEP samples with 0-3 scores for quality. A two-stage training process is conducted to obtain agents. First, teacher-student knowledge distillation (Hu et al., 2023) is applied for pre-training, where G_α and G_β learn from GPT-4o in generating and assessing AEP samples, respectively. Further, we perform Supervised Fine-Tuning (SFT) to optimize agents, where the semi-artificially labeled AEP samples are used.

Due to the involvement of closed-source GPT-4o, the above automatic labeling suffers from the increasing cost. To solve the problem, we propose a progressive labeling approach as follows.

- **Initialization**: We initialize the generator G_α and assessor G_β by aforementioned distillation and SFT. Qwen-2.5 (7B) is used to form G_α and G_β. We also initialize a data pool $\mathcal{D}$. It is loaded with the qualified AEP samples (score$\geq$2) obtained by semi-artificial labeling.

- **Data Expansion**: We select $\mathcal{K}$ "task-plan" pairs from OGP. G_α produces AEP samples for the "task-plan" pairs. G_β marks AEP samples for quality (0-3 scores). The qualified samples are adopted to expand the pool $\mathcal{D}$.

- **Relearning**: Using the expanded data pool $\mathcal{D}$, we fine-tune the generator G_α once again.

[1]OGP is the Open Grounded Planning dataset. It supports the human-like planning on multi-domain tasks such as *Life Skills*, *Robot* and *Tools*. https://github.com/Shiguang-Guo/Open-Grounded-Planning/tree/master/datasets.

Approch	#Full-size	#Qualified
Semi-artificial	4,371	4,131
GPT-4o	11,657	8,362
Agents (G_α+G_β)	17,277	14,751

Table 1: Statistics in all AEP datasets. **Semi-artificial** denotes the dataset obtained by semi-artificial labeling. **GPT-4o** refers to the dataset produced by GPT-4o, which is also used for initialization during progressive labeling. **Agents** align with the dataset constructed by two agents G_α and G_β in the 10-iteration progressive labeling process. **Full-size** is the number of all "exception-solution" pairs in a dataset, while **Qualified** is the number of qualified instances (score$\geq$2).

- **Iteration**: We iteratively expand $\mathcal{D}$ and use it to fine-tune the generator G_α. The goal is to progressively enhance G_α.

During initialization, we use 500 tasks (11.6K exception-solution AEP instances) for distillation, while 200 tasks (4.1K AEP instances) for SFT. We select 100 new tasks ($\mathcal{K}$=100) from OGP for each iteration of progressive labeling. The labeling process is excecuted for 10 iterations in total. Table 1 provides the statistics in all datasets. Appendix B details the prompts used for distillation.

4 Grounded AEP Models

We follow Guo et al. (2024) to ensure groundedness when executing AEP task. Accordingly, an AEP model is forcibly to adopt the solution $\mathcal{S}$ that does exist in the solution space $\check{\mathcal{C}}$. We build $\check{\mathcal{C}}$ by expanding the action space $\mathcal{C}$ of OGP with all the qualified solutions in our AEP datasets. Neither solutions nor actions in $\check{\mathcal{C}}$ are given any kind of marks to expose their particularity. This enables the black-box testing, and thus increases the challenge of AEP. In other words, an AEP model will struggle with not only distracting solutions but irrelevant actions.

We construct our grounded AEP model with the Generation-Retrieval-Ranker (**GRR**) framework (Huang et al., 2024; Lee et al., 2024). Specifically, it runs as follows:

- **Generation**: For an exception $\mathcal{E}$, we prompt LLM to generate a solution $\mathcal{S}$. Besides of $\mathcal{E}$, the input of LLM also comprises the task $\mathcal{T}$ and historical actions $[\mathcal{A}_1...\mathcal{A}_{n-1}]$ of plan $\mathcal{P}$, which serve as hints to imply the scenario.

- **Retrieval**: Using $\mathcal{S}$ as query, we retrieve k most relevant candidates from solution space $\check{\mathcal{C}}$. Both semantically-similar solutions or actions in $\check{\mathcal{C}}$ may emerge as candidates.

- **Ranker**: We rank the candidates according to their semantic consistency with $\mathcal{S}$, and adopt the top-1 candidate in the ranking list. BGE-based similarity (Chen et al., 2024) is computed for semantic consistency analysis.

GRR fails to effectively ensure groundedness. Instead, the free-style solution generation may occur in GRR because solution space $\check{\mathcal{C}}$ is not exposed to LLM. This easily causes the invalidation of solution retrieval and ranking. However, it is actually hard to expose the whole space $\check{\mathcal{C}}$ due to the large data it contains (41K solutions and actions; 9 tokens per case in average). More importantly, even if $\check{\mathcal{C}}$ can be fed into LLM, the big data in $\check{\mathcal{C}}$ makes it difficult to perform out-of-redundancy prompting.

To address the issue, we develop a Reverse Engineering based GRR (**RE-GRR**). It runs as follows.

- **Reverse Engineering**: RE-GRR regards each case in $\check{\mathcal{C}}$ as a potential solution, and uses LLM to reversely generate the most possible exception that can be handled by this solution. This allows a referential "exception-solution" mapping table $\mathcal{B}$ to be built over $\check{\mathcal{C}}$, where exceptions in $\mathcal{B}$ are considered as entries.

- **Reference Retrieval**: Given an exception $\mathcal{E}$ during performing AEP (i.e., testing stage), RE-GRR uses $\mathcal{E}$ as query to retrieve $\check{k}$ similar exceptions from the entries of $\mathcal{B}$. By exploring the one-to-one mapping relationship between exceptions and solutions in $\mathcal{B}$, RE-GRR fishes out $\check{k}$ referential solutions from $\mathcal{B}$.

- **Reference based GRR**: RE-GRR feeds $\check{k}$ referential solutions into LLM, and prompts it to generate the most possible solution according to the referential cases. In RE-GRR, the retriever and ranker of GRR are not changed.

All the details of GRR and RE-GRR (e.g., LLM-oriented prompting, retriever, ranker and parameter settings of k and $\check{k}$) are presented in Appendix C.

5 Experimentation

In our experiments, a variety of grounded AEP models are evaluated, which use different LLMs as solution generators. We intend to explore the varied anti-exception capabilities of LLMs during solution

Model	**SFT**	Semi-artificial Labeling			GPT-4o Dataset			Progressive Labeling		
		$\text{Set}_{\geq 1}$	$\text{Set}_{\geq 2}$	Set_3	$\text{Set}_{\geq 1}$	$\text{Set}_{\geq 2}$	Set_3	$\text{Set}_{\geq 1}$	$\text{Set}_{\geq 2}$	Set_3
Generation-Retrieval-Ranker (**GRR**)										
Vicuna-v1.5 (7B)	w/o	55.93	58.08	83.73	77.75	81.13	93.64	65.98	68.05	86.36
Qwen-2.5 (7B)	w/o	45.26	46.23	78.56	66.14	70.24	89.02	57.78	60.03	82.59
LLaMA-3.1 (8B)	w/o	43.86	45.91	78.23	62.94	68.51	88.45	57.16	59.35	82.63
GPT-4o	w/o	34.16	35.45	72.19	51.84	58.02	84.69	51.87	54.27	80.48
DeepSeek-R1	w/o	32.43	33.62	75.86	28.39	35.99	75.06	56.41	58.83	81.34
Qwen-2.5 (7B)	w/	33.84	34.91	**71.66**	48.80	55.59	84.07	46.90	49.21	77.95
Reverse Engineering based GRR (**RE-GRR**)										
Vicuna-v1.5 (7B)	w/o	41.59	43.75	85.02	58.67	66.91	88.34	63.33	66.67	86.80
Qwen-2.5 (7B)	w/o	23.60	25.75	75.22	30.40	40.39	76.72	45.71	48.32	76.78
LLaMA-3.1 (8B)	w/o	28.56	30.28	78.88	33.73	44.36	78.82	49.21	52.64	80.30
GPT-4o	w/o	**19.91**	21.32	72.40	26.54	35.51	72.86	43.62	46.47	76.67
DeepSeek-R1	w/o	23.81	25.00	74.78	27.92	35.14	**69.85**	47.20	49.97	**76.56**
Qwen-2.5 (7B)	w/	19.94	**21.12**	**71.66**	**26.00**	**35.10**	72.76	**43.38**	**46.09**	76.58

Table 2: Performance on different AEP datasets. Symbol "w/" denotes that SFT is conducted, while "w/o" not.

generation. Therefore, the performance comparison among AEP models is carried out within the same framework (either GRR or RE-GRR), where other components like retriever and ranker (except generator) are identical. Success rate γ of exception attack is used as the evaluation metric. It is calculated as the proportion of successful exception attacks in all AEP samples, where a success attack aligns with an ineffective solution (i.e., an out-of-vocabulary solution). A higher γ reflects a less strong anti-exception ability (Appendix D).

Table 2 shows the performance of all grounded AEP models, where three AEP datasets are used for evaluation, including the ones obtained by semi-artificial labeling, GPT-4o and progressive labeling respectively. The columns of $\text{Set}_{\geq 1}$, $\text{Set}_{\geq 2}$ and Set_3 in Table 2 denote the data subsets involved in the experiments. Their contents are as follows.

- Set_3 only contains AEP samples that are not only practicable but credible (score=3).
- $\text{Set}_{\geq 2}$ expands Set_3 with samples that are practicable but uncertain in credibility (score$\geq$2).
- $\text{Set}_{\geq 1}$ contains all the samples that hold a pair of relevant exception and solution, regardless of whether they are practicable (score$\geq$1).

It can be observed from Table 2 that, unfortunately, all LLMs achieves unsatisfactory performance in the most rigorous test that uses Set_3. Although success rates of exception attacks have been brought down by LLMs in the relatively simple tests (e.g., on $\text{Set}_{\geq 2}$), they are still no less than 33.62% in the GRR framework. Nevertheless, we found some encouraging results as follows.

- Performance of LLMs on the semi-arcificially labeled data is slightly comparable to that of progressive labeling. This gives a chance to enhance AEP by SFT using a larger number of automatically-produced instances.
- SFT works. It is proven by the performance of fine-tuned Qwen-2.5 (7B) in Table 2, where SFT is conducted using 8,362 AEP instances produced by GPT-4o.
- Re-GRR allows LLMs to achieve better performance (lower γ). This implies the possibility of methodology-based AEP enhancement.

6 Conclusion

We provide a preliminary study of anti-exception solution acquisition for human-like planning. Experiments show that LLMs are less capable of generating practicable and credible solutions, revealing the challenge in autonomous plan refinement. Nevertheless, it has proven that the optimized framework like RE-GRR achieves substantial improvements, illustrating the potential of methodological innovation. In addition, the progressive labeling approach and its resultant AEP dataset are supportive for the investigation of SFT-based approaches.

Limitations

The fine-tuned light-weight LLM Qwen-2.5 (7B) shows comparable performance with GPT-4o and DeepSeek-R1. Though, we suspect that the optimal performance of fine-tunable LLMs hasn't yet been reached. This suspicion derives from the consideration that the most proper size of observable AEP instances for fine-tuning is not explored. Therefore, if a future study intends to use light-weight LLMs to form ideal baselines or backbones, a larger scale of training data needs to be produced. This will cause additional efforts in manual data labeling or cost in GPT-4o based automatic labeling. An alternative strategy is to use our progressive labeling approach, which enables the production of unlimited size of training data. This potentially contributes to pursuing the optimal performance of fine-tunable LLMs. In this case, the comparison experiment with closed-source LLMs like DeepSeek-R1 needs to be reformed as the test set is possibly changed.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, and 26 others. 2022. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. *ArXiv preprint*, abs/2204.01691.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Shiguang Guo, Ziliang Deng, Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. 2024. Open grounded planning: Challenges and benchmark construction. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4982–5003, Bangkok, Thailand. Association for Computational Linguistics.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023. Reasoning with Language Model is Planning with World Model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173, Singapore. Association for Computational Linguistics.

Yilun Hao, Yang Zhang, and Chuchu Fan. 2025. Planning anything with rigor: General-purpose zero-shot planning with LLM-based formalized programming. In *The Thirteenth International Conference on Learning Representations*.

Chengming Hu, Xuan Li, Dan Liu, Haolun Wu, Xi Chen, Ju Wang, and Xue Liu. 2023. Teacher-student architecture for knowledge distillation: A survey. *arXiv preprint arXiv:2308.04268*.

Tenghao Huang, Dongwon Jung, and Muhao Chen. 2024. Planning and editing what you retrieve for enhanced tool learning. *arXiv preprint arXiv:2404.00450*.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 9118–9147. PMLR.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Yann LeCun. 2022. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62.

Myeonghwa Lee, Seonho An, and Min-Soo Kim. 2024. PlanRAG: A plan-then-retrieval augmented generation for generative large language models as decision makers. 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 6537–6555, Mexico City, Mexico. Association for Computational Linguistics.

Ao Li, Yuexiang Xie, Songze Li, Fugee Tsung, Bolin Ding, and Yaliang Li. 2025. Agent-oriented planning in multi-agent systems. In *The Thirteenth International Conference on Learning Representations*.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. API-Bank: A Comprehensive Benchmark for Tool-Augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 3102–3116. Association for Computational Linguistics.

Bill Yuchen Lin, Chengsong Huang, Qian Liu, Wenda Gu, Sam Sommerer, and Xiang Ren. 2023. On Grounded Planning for Embodied Tasks with Language Models. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pages 13192–13200. AAAI Press.

Yanming Liu, Xinyue Peng, Jiannan Cao, Shi Bo, Yuwei Zhang, Xuhong Zhang, Sheng Cheng, Xun Wang, Jianwei Yin, and Tianyu Du. 2025. Tool-planner: Task planning with clusters across multiple tools. In *The Thirteenth International Conference on Learning Representations*.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face. *ArXiv preprint*, abs/2303.17580.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. 2023. ToolAlpaca: Generalized Tool Learning for Language Models with 3000 Simulated Cases. *ArXiv preprint*, abs/2306.05301.

Jiaxin Wen, Jian Guan, Hongning Wang, Wei Wu, and Minlie Huang. 2025. Codeplan: Unlocking reasoning potential in large language models by scaling code-form planning. In *The Thirteenth International Conference on Learning Representations*.

Zhenyu Wu, Ziwei Wang, Xiuwei Xu, Jiwen Lu, and Haibin Yan. 2023. Embodied Task Planning with Large Language Models. *ArXiv preprint*, abs/2307.01848.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. 2023. GPT4Tools: Teaching Large Language Model to Use Tools via Self-instruction. *ArXiv preprint*, abs/2305.18752.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Quan Yuan, Mehran Kazemi, Xin Xu, Isaac Noble, Vaiva Imbrasaite, and Deepak Ramachandran. 2023. TaskLAMA: Probing the Complex Task Understanding of Language Models. *ArXiv preprint*, abs/2308.15299.

Li Zhang, Qing Lyu, and Chris Callison-Burch. 2020. Reasoning about goals, steps, and temporal ordering with WikiHow. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4630–4639, Online. Association for Computational Linguistics.

Zirui Zhao, Wee Sun Lee, and David Hsu. 2023. Large Language Models as Commonsense Knowledge for Large-Scale Task Planning. *ArXiv preprint*, abs/2305.14078.

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. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

```
INSTRUCTION:
You are an expert in task analysis and problem identification. Your role is to generate realistic, domain-appropriate exceptions that might occur during the execution of the final step of a task.
exception is related to both task topic and previous actions in the task.
exception prevents the execution of the final step but can be resolved with a single additional step.
IMPORTANT:
- The exception must occur right before attempting the final step, not during earlier steps.
- All previous steps have been successfully completed
- The exception happens when the person is about to execute the final step
- The exception directly prevents the immediate execution of the final step
- The exception does not relate to problems during previous steps
FORMATTING REQUIREMENTS:
1. Keep descriptions concise and brief (maximum 15-20 words)
2. Focus on one clear issue per exception
3. Avoid using "and" or "or" connectors that introduce multiple issues
4. Use simple, direct language without unnecessary details
OUTPUT FORMAT:
Return a JSON object with the following structure:
{
  "exceptions": [
   {
    "exception_description": "Brief description of the exception that occurred"
   },
   ...additional exceptions...
  ]
}
Generate 2-5 different exceptions with diverse characteristics. Each exception should:
1. Be directly related to the specific final step
2. Be appropriate for the task domain and categories
3. Present a realistic obstacle that prevents the immediate completion of the final step
4. Be diverse in nature (different types of problems)
5. Be specific and focused on a single issue
--------------------------------------------------------------------------------
QUERY:
Task: {title}
Method: {method}
Categories: {category}
Previous Steps (already completed):
{previous_steps}
Final Step: {last_step}
```

Figure 2: Prompt of Generating Exception.

```
INSTRUCTION:
You are an expert problem solver with extensive knowledge across various domains. Your role is to generate effective single-step solutions to resolve exceptions that occur during task execution.
Solution is related to the exception and resolves it in a single step, allowing the final task step to be executed.

FORMATTING REQUIREMENTS:
1. Keep solution descriptions concise and brief (maximum 15-20 words)
2. Each solution must be a single, atomic action (not a combination of actions)
3. Avoid using "and" or "or" connectors that suggest multiple actions
4. Use imperative, direct language (start with a verb)
5. Focus on one clear, specific action per solution
IMPORTANT: Each solution must directly address and resolve the specific exception mentioned in the query.
Do not provide general solutions for other possible exceptions or problems.
Focus only on solving the exact exception described in the query.
OUTPUT FORMAT:
Return a JSON object with the following structure:
{
"solutions": [
{
"solution_description": "Brief description of the single-step solution",
},
...additional solutions...
]
}
Generate 2-5 different solutions with diverse approaches. Each solution should:
1. Be a single atomic action - not a sequence of actions
2. Be directly relevant to resolving the specific exception
3. Be realistic and practical
4. Be specific and actionable
5. Be diverse (different approaches to solving the same problem)
--------------------------------------------------------------------------------
QUERY:
Task Title: {title}
Task Method: {method}
Task Categories: {categories}
Previous Steps (already completed):
{previous_steps}
Final Step (that needs to be executed after solving the exception):
{last_step}
exception: {exception_desc}
```

Figure 3: Prompt of Generating Solutions.

A Details of Semi-Artificial Labeling

A.1 Prompt for Constructing AEP Datasets

Our prompts comprise two components, including *instruction* and *query*. All prompts are configured for zero-shot settings, and the output format is restricted to JSON. Figure 2 and 3 illustrate the prompt templates for exception generation and the corresponding solution generation, respectively.

A.2 Assessment Criteria

To enable the manual quality assessment on the generated AEP instances, we establish specific 4-level quality criteria as follows, where the criterion for assessing exceptions is exhibited first, and then the criterion for solutions.

 # Criterion for Exceptions

- 0 point: The exception is irrelevant to the "*task-plan*" pair. Irrelevance denotes the topic-level difference or the distinction of entities.

- 1 point: The exception is related to the "task-plan" pair, though it fails to disrupt the execution of the tail-end action of the plan.

- 2 point: The exception is related to the "task-plan" pair. More importantly, it can disrupt the tail-end action (i.e., a potentially effective exception attack). Though the effectiveness is determined intuitively because annotators are unaware of the whole background knowledge about it, and cannot find exact evidence to claim the interpretability.

- 3 point: In addition to meeting the requirement for 2 points, the exception needs to be logically justified. More importantly, it can be tackled by a single-step solution instead of inducing a chain reaction.

Criterion for Solutions

- 0 point: Irrelevant solution to the exception.

- 1 point: The solution is related to the exception. Though it is obviously ineffective or impractical to solve the exception in a single step, the tail-end action of the plan cannot be executed afterward.

- 2 point: The solution can effectively handle the exception in a single step, and thus enables the tail-end step to proceed without further intervention. More importantly, it is practicable but not unrealistic in practice. Nevertheless, the practicability is determined based on intuition. There is a lack of evidence provided by annotators to claim interpretability.

- 3 point: In addition to meeting the requirement for 2 points, the solution needs to be logically justified. More importantly, it is determined as the optimal solution by comparing multiple solution candidates.

A.3 Annotation Scheme

We recruit 6 annotators who major in linguistics and conduct a structured training phase followed by up to three rounds of trial annotation. Trial annotations proceeds as follows:

- If the average Kappa value reaches at least 75%, we proceed to the formal annotation phase.

- If the threshold is not met after three rounds, a new group of annotators is recruited.

Annotators are compensated $0.27 per sample during both the trial and formal annotation stages. In the formal annotation stage, the six annotators are divided into two groups of three, with each group assigned to label the same set of data. This setup enables the calculation of inter-annotator agreement within each group to assess labeling consistency.

After the initial trial phase, we conduct two rounds of formal annotation, with each group labeling 50 tasks per round. This results in a total of 200 annotated tasks, which constitute part of the semi-artificially labeled dataset. After each round, all annotated tasks are jointly reviewed by the three annotators in each group. Annotators receive $0.14 for each reviewed sample.

Annotators are compensated based on their agreement with the adjudicated results:

- Highest agreement: 50% of the total compensation.

- Second-highest agreement: 30%.

- Lowest agreement: 20%.

A.4 Background of Annotators

Number	Gender	Age	Major	Grade
1	female	21	English	junior
2	female	21	English	junior
3	female	21	English	junior
4	female	21	English	junior
5	female	21	English	junior
6	female	21	English	junior

Table 3: Annotator background.

Number	Iter1	Iter2	Iter3	Time
1	58.43	77.33	83.56	1.57
2	64.04	68.00	76.71	1.42
3	70.79	68.00	72.60	1.57
4	79.78	70.67	83.56	1.50
5	57.30	74.67	79.45	1.57
6	56.17	80.00	67.12	1.57

Table 4: Trial annotation details.

Table 3 outlines the demographic and academic profiles of the six annotators, including gender, age, academic discipline, and educational level.

Table 4 reports annotator agreement with the ground truth, quantified by Kappa value, alongside the mean annotation time per instance (in minutes) during the trial phase.

A.5 Q&A

We document the questions raised by annotators during trial annotation, together with our corresponding responses.

```
INTRODUCTION:
You are a professional task execution evaluator. Please rate the
exception scenario according to the following criteria:

SCORING CRITERIA (0-3 points)
- 0 point: The exception is completely unrelated to the task topic
- 1 point: The exception is related to the task topic, but unrelated to
the historical steps
- 2 point: The exception is related to both the task topic and the
historical steps
- 3 point: Building on the 2-point requirements, the exception is
reasonable and would genuinely prevent the execution of the final step,
but could be resolved by adding a single intermediate step
Please return only a JSON format numerical score, for example:
{
  "score": 2
}
Do not explain your reasoning, only return the score. It must be an
integer between 0-3.
--------------------------------------------------------------------------------
QUERY:
Task Title: {title}
Task Method: {method}
Task Categories: {categories}
Previous Steps (already completed):
{previous_steps}
Final Step (to be executed after resolving the exception): {last_step}
exception: {exception}
```

Figure 4: Prompt of scoring exception.

```
INTRODUCTION:
You are a professional solution evaluator. Please rate ALL solutions for
resolving an exception according to the following criteria:
SCORING CRITERIA (0-3 points)
- 0 point: The solution is unrelated to the exception
- 1 point: The solution is related to the exception but cannot fully
resolve it in a single step
- 2 point: The solution is related to the exception and can completely
resolve it in a single step, allowing the execution of the original last
step to complete the task.
- 3 point: Building on the 2-point requirements, this is the optimal
solution among all alternatives
IMPORTANT: Among all solutions for a single exception, at most one
solution can receive 3 points(There may not be 3 points).
If multiple solutions could potentially qualify for 3 points, you must
select only the absolute best one to receive 3 points.
All other solutions can receive a maximum of 2 points. It is not
necessary to have a solution with a score of 3. If you believe that none
of them are the optimal solution, then the highest score is only 2
points.
OUTPUT FORMAT
Return a JSON object with scores for each solution, using the solution
index as the key:
{
  "scores": {
    "0": 3,
    "1": 2,
    "2": 1,
    ...
  }
}
Do not explain your reasoning, only return the scores. Each score must
be an integer between 0-3.
--------------------------------------------------------------------------------
QUERY:
Task Title: {title}
Task Method: {method}
Task Categories: {categories}
Previous Steps (already completed):
{previous_steps}
Final Step (to be executed after resolving the exception): {last_step}
exception: {exception}

Solutions to evaluate:
{solutions}
```

Figure 5: Prompt of scoring solutions

Q1: When the procedure is unclear, it can be confusing, for example, regarding hair perming. If someone has never had a perm before, they might not be familiar with the subsequent care steps.

A1: Indeed, there are many instances involving relatively uncommon world knowledge. Annotators are encouraged to rely on their general knowledge and judgment when assigning scores. If they are unable to resolve an issue, they may use a search engine as a supplementary resource.

Q2: It is difficult to determine which solution is optimal. The evaluation feels highly subjective and lacks sufficient supporting information.

A2: It is not strictly necessary to select the optimal solution; if it is hard to judge, assigning a maximum of 2 points is acceptable. However, annotators are still encouraged to identify the best solution when possible and assign it 3 points.

Q3: Some of the options seem like the same method phrased in four different ways. Can I just mark all of them as 2 points in this case?

A3: Yes. Treat this as if there is no single optimal solution—simply assign 2 points to all of them.

Q4: You mentioned that the criteria for assigning 2 points to an "exception" are strict—it's only when none of the provided solutions can resolve the exception in a single step. However, taking the sunscreen article as an example, the last "exception" was "clothes getting dirty." While the solutions listed could address the issue, dirty clothes don't inherently prevent achieving the core goal of "sun protection" (they might mainly affect aesthetics or willingness to wear them). Thus, I believe this exception wouldn't render the final step unexecutable, so I'd keep the score at 2. This is where I'm conflicted: based on your additional clarification, it seems like a 3, but based on yesterday's scoring rules, I'm inclined to stick with 2. Am I misunderstanding something here?

A4: Avoid over-interpretation during scoring. In this task, "dirty sunscreen clothing" does qualify as an exception that disrupts the original final step (disregard tangential factors like core

```
INSTRUCTION:
You are an expert in task analysis and exception resolution. Your role is to analyze whether a given action could be a solution to a specific exception that might occur during a task.

INSTRUCTIONS:
1. Carefully analyze the action in the context of the task.
2. Determine if this action could reasonably solve a specific exception.
3. If the action could be a solution, describe a specific exception that this action would address in 10-15 words maximum.

IMPORTANT GUIDELINES:
- Be realistic about whether the action actually solves a exception or is just a regular task step.
- Focus on realistic, concrete exceptions that could naturally occur during the task.
- Your response should be a concise exception description, nothing else.
- Do not include any explanations, analysis, or additional text.

OUTPUT FORMAT:
"A concise description of the specific exception this action would solve"
Please analyze the following action and determine if it could be a solution to an exception in a task.
--------------------------------------------------------------------------------
QUERY:
Task Context:
- Task Title: {title}
- Task Categories: {categories}
- Previous Steps: {previous_steps}
- Final Step: {last_step}
Action: {action}
```

Figure 6: Prompt of reverse engineering.

```
INSTRUCTION:
You are an expert in exception resolution and solution selection. Your role is to analyze a set of reference solutions from similar exceptions and select or create the best solution for the current exception.
Please help me analyze and select the best solution:

1) First, evaluate each reference solution. Which ones would be most helpful for addressing the current exception? Explain your reasoning briefly for each.
2) Based on your analysis, select the most appropriate solution from the reference solutions.
SCORING CRITERIA (0-3 points)
- 0 point: The solution is unrelated to the exception
- 1 point: The solution is related to the exception but cannot fully resolve it in a single step
- 2 point: The solution is related to the exception and can completely resolve it in a single step, allowing the execution of the original last step to complete the task.
- 3 point: Building on the 2-point requirements, this is the optimal solution among all alternatives
You should choose the solution that you believe can achieve the highest score.
OUTPUT FORMAT:
Return a JSON object with the following structure(You must strictly output your choices in the given JSON format):
{
"solution": "the selected solution"
}
After outputting the solution you have chosen, do not provide any further explanation.
--------------------------------------------------------------------------------
QUERY:
- Title: {title}
- Categories: {categories}
- Previous Steps (already completed):
{previous_steps}
- Final Step (to be executed after resolving the accident): {last_step}
- Current Exception: {exception}
- Reference Solutions:
{reference_solutions}
```

Figure 7: Prompt of selecting reference solutions.

purpose or aesthetics). If any solution can resolve it in a single step, assign 3 points; if none can, assign 2.

B Details of Distillation

We adopt Qwen2.5-7B as the base model for training the distilled models, G_α and G_β. The supervised fine-tuning (SFT) dataset is constructed from the AEP dataset, which includes both the GPT-4o-generated dataset and a semi-artificial labeling dataset.

For G_α, the training objective is to generate high-quality exceptions and corresponding solutions. For G_β, the objective is to produce reliable scoring outputs. Both models are first pretrained on GPT-4o-generated data, followed by fine-tuning with high-quality semi-artificial labeling data.

C Details of AEP-oriented GRR

For G_α, the training data comprises instruction-query pairs as depicted in Figure 2 and Figure 3. For G_β, the instruction-query format is illustrated in Figure 4 and 5. In both cases, the models are trained to generate outputs in strict JSON format to ensure structural consistency and avoid arbitrary or malformed responses.

To enable large language models (LLMs) to generate solutions for exceptions, we propose two methods: GRR and RE-GRR.

GRR Method: We prompt the LLM to generate an initial solution, which is subsequently used to retrieve a similar solution from a pre-constructed solution library. The generation prompt is shown in Figure 8.

For the retrieval component, we use BGE-M3 as the embedding model. We precompute embeddings for all actions and solutions in the library. The

INSTRUCTION:
You are an expert exception solver with extensive knowledge across various domains. Your role is to generate the most effective single-step solution to resolve an exception that occurs during task execution.
Solution is related to the exception and resolves it in a single step, allowing the final task step to be executed.
FORMATTING REQUIREMENTS
1. Keep solution description concise and brief (maximum 15-20 words)
2. The solution must be a single, atomic action (not a combination of actions)
3. Avoid using "and" or "or" connectors that suggest multiple actions
4. Use imperative, direct language (start with a verb)
5. Focus on one clear, specific action
DIRECT RELEVANCE
IMPORTANT: Your solution MUST directly address and resolve the specific exception mentioned in the query.
DO NOT provide general solutions for other possible exceptions or problems.
Focus only on solving the exact exception described in the query.
SCORING CRITERIA (0-3 points)
- 0 point: The solution is unrelated to the exception
- 1 point: The solution is related to the exception but cannot fully resolve it in a single step
- 2 point: The solution is related to the exception and can completely resolve it in a single step, allowing the execution of the original last step to complete the task.
- 3 point: Building on the 2-point requirements, this is the optimal solution among all alternatives
You should generate the solution that you believe can achieve the highest score.
OUTPUT FORMAT
Return a JSON object with the following structure:
{
"solution": "Brief description of the single-step solution"
}
The solution should:
1. Be a single atomic action - not a sequence of actions
2. Be directly relevant to resolving the specific exception
3. Be realistic and practical
4. Be specific and actionable
5. Be the most effective approach to solving the exception

QUERY:
Task Title: {title}
Task Categories: {categories}
Previous Steps (already completed):
{previous_steps}
Final Step (to be executed after resolving the exception): {last_step}
exception: {exception}

Figure 8: Prompt of GRR.

generated solution is also embedded using the same model, and we compute cosine similarity to retrieve the top-5 ($k = 5$) most similar solutions. The most similar solution is then selected as the result.

RE-GRR Method: We first use GPT-4o to generate a set of potential exceptions that each solution might address (one-to-one mapping relationship between exceptions and solutions). The prompt for this step is shown in Figure 6. These generated exceptions are then embedded using BGE-M3 to form an exception-to-solution retrieval index. During inference, the embedding of the current exception (generated via BGE-M3) is used to retrieve the top-5 ($k = 5$) most similar generated exceptions and their corresponding solutions.

To enhance decision quality, we prompt the LLMs to select the most appropriate solution from these five candidates. The model is also instructed to articulate its reasoning process, thereby improving the reliability of the selection. The selection prompt is shown in Figure 7.

D Details of Evaluating

To facilitate evaluation, we construct three types of sets for each dataset: $\mathrm{Set}_{\geq 1}$, $\mathrm{Set}_{\geq 2}$, and Set_{3}. A solution with a score of at least 1 is added to $\mathrm{Set}_{\geq 1}$, and similarly for the other sets. Notably, the sets are constructed separately for each exception. This design ensures that LLM-generated solutions are not only relevant to the corresponding exceptions but also satisfy the specified scoring criteria. Here is an example for AEP:

```
{
  "title": "How to Watch Sports on Apple
      TV",
  "steps": [
    "Establish your viewing criteria.",
    "Review your free options.",
    "Review your paid options.",
    "Review third-party apps.",
    "Check out the \"Sling TV\" app."
  ],
  "exceptions_and_solutions": [
    {
      "exception_description": "Sling TV
           app not installed on Apple TV
           device",
      "single_step_solution": [
        {
          "solution_description": "
              Install Sling TV app via
              App Store on Apple TV", "
              score": 3
        },
        {
          "solution_description": "Use
              Siri voice command to
              search and install Sling
              TV", "score": 2
        },
        {
          "solution_description": "
              Redownload Sling TV from
              Purchased section in App
              Store", "score": 1
        },
        {
          "solution_description": "
              Install Sling TV via Apple
               TV remote app on paired
              iPhone", "score": 2
        }
      ],
```

```
        "selected_action": "Install Sling
            TV app via App Store on Apple
            TV",
    }, ...
    ]
}
```

In this example, we design the exception *"Sling TV app not installed on Apple TV device"* for the step *"Check out the 'Sling TV' app"*. This exception hinders the successful execution of the final step. To resolve this exception, we generate multiple candidate solutions and assign scores to each based on predefined evaluation criteria. Each solution is then added to the corresponding exception-specific set according to its score.

After obtaining a solution via either the GRR or RE-GRR method, we evaluate its effectiveness by calculating the win rate across different sets. If the generated solution does not match any of our designed solutions (ground-truth), it is classified as a win for the exception (i.e., the solution is invalid). If the solution exists in the set and has a score of 1, then both $\text{Set}_{\geq 2}$ and $\text{Set}_{\geq 3}$ are considered wins. If the score is 2, then $\text{Set}_{\geq 3}$ alone is considered a win. Notably, a higher win rate indicates poorer solution quality and weaker model capability in handling exceptions.