
Responsible Task Automation: Empowering Large Language Models as Responsible Task Automators

Zhizheng Zhang* Xiaoyi Zhang* Wenxuan Xie Yan Lu
Microsoft Research Asia
{zhizzhang, xiaoyizhang, wenxie, yanlu}@microsoft.com

Abstract

The recent success of Large Language Models (LLMs) signifies an impressive stride towards artificial general intelligence. They have shown a promising prospect in automatically completing tasks upon user instructions, functioning as brain-like coordinators. The associated risks will be revealed as we delegate an increasing number of tasks to machines for automated completion. A big question emerges: how can we make machines behave responsibly when helping humans automate tasks as personal copilots? In this paper, we explore this question in depth from the perspectives of feasibility, completeness and security. In specific, we present Responsible Task Automation (ResponsibleTA) as a fundamental framework to facilitate responsible collaboration between LLM-based coordinators and executors for task automation with three empowered capabilities: 1) predicting the feasibility of the commands for executors; 2) verifying the completeness of executors; 3) enhancing the security (e.g., the protection of users’ privacy). We further propose and compare two paradigms for implementing the first two capabilities. One is to leverage the generic knowledge of LLMs themselves via prompt engineering while the other is to adopt domain-specific learnable models. Moreover, we introduce a local memory mechanism for achieving the third capability. We evaluate our proposed ResponsibleTA on UI task automation and hope it could bring more attentions in ensuring LLMs more responsible in diverse scenarios.

1 Introduction

Recent advanced Large Language Models (LLMs) [3, 28, 27, 6, 7, 10, 32, 40] exhibit powerful language understanding, reasoning, generation, generalization and alignment capabilities in a vast suite of real-world scenarios. LLMs capture generic knowledge on open-domain tasks by scaling up deep learning, which marks a significant milestone in the advancement towards artificial general intelligence. Beyond language tasks, LLMs are empowered with multi-modal perception and generation capabilities through collaboration with domain-specific models [39, 30, 36]. They have been revolutionizing the field of task automation by connecting LLMs to various domain-specific models or APIs, in which LLMs serve as brain-like coordinators while domain-specific models or APIs function as action executors [18, 33, 10].

Adopting LLMs to constructing a general-purpose copilot for automating diverse tasks is still in an incipient exploration stage. Taking the interaction with UIs as an example, browsing and interacting with various websites and APPs to achieve their diverse intentions (*e.g.*, searching in websites, online shopping, setting changing, *etc.*) are almost indispensable for many people’s daily life. Some of them require a hierarchical action architecture [2, 1] where a high-level instruction from human known as task-level goal needs to be decomposed into a series of low-level step-wise instructions for actual execution. Such complex multi-step tasks requires each step to be reasonably planned and reliably

*Equal contribution.

executed in line with human intentions. This actually poses significant challenges in the compatibility between LLM-based coordinators and their executors without intervention in their training. Their reliable collaboration requires LLM-based coordinators to be very familiar with the capabilities of executors, and to replan the task execution when necessary upon the command completeness of executors. Existing works [36, 39, 30, 18] just tell LLMs the powers of executors with model/API descriptions via heavy prompt engineering. However, these descriptions are commonly manually written and summarizing in nature, which are inadequate to describe the executors’ capabilities and cannot reflect the execution completeness for LLMs. In addition, LLMs are also possible to have insufficient awareness of the task goals and environments, thus issuing unreasonable instructions to executors. These will expose more and more risks as an increasing number of tasks are delegated to machines for automated completion, imposing urgent demands on address the reliability issue.

In this work, we propose ResponsibleTA as a fundamental multi-modal framework to empower LLMs as responsible task automators in three aspects: 1) **Feasibility**: ResponsibleTA predicts the feasibility of low-level commands generated by the LLM-based coordinator and return the results to LLMs for replanning before action execution. This capability aims to minimize the risks and time consumption of having the executors perform unachievable instructions, making task automation more controllable and efficient. 2) **Completeness**: ResponsibleTA checks the execution results of low-level commands step-by-step and provides feedbacks timely for the LLM-based coordinator to allow it reschedule next steps more reasonably. This capability can improve the success rates of task automated completion. 3) **Security**: ResponsibleTA augments LLMs with edge-deployed memories, which allows user sensitive information are hidden during the interaction with cloud-deployed LLMs and are only stored and used locally on users’ devices. This capability reduces or avoids the transmission of users’ sensitive information between the cloud-deployed LLMs and edge-deployed executors, thus lowering the leakage risk of users’ privacy. Empowering LLMs with these three capabilities, ResponsibleTA achieves automatic verification before and after each execution step, which improves not only the success rate but also the efficiency of automated task completion with more thoughtful security guarantees for users.

In addition to the framework design of ResponsibleTA, we investigate the implementations of three core functionalities of ResponsibleTA as an empirical study in a practical application scenario (*i.e.*, UI task automation). Its goal is to automatically ground target UI elements and interact with them via automatic click and input operations based on task instructions from users, which is of high demands in both academia and industry. For feasibility and completeness, we propose and compare two technical paradigms for their implementations. One paradigm is to leverage generic knowledge in LLMs themselves via prompt engineering. The other paradigm is to train domain-specific models specifically responsible for these two functionalities as external assistants of the coordinators and executors. We observe that leveraging LLMs themselves is inferior to adopting domain-specific models, demonstrating that domain-specific knowledge is crucial for enhancing the reliability of LLMs in task automation. For security, we introduce a local memory and propose a mechanism design to use it for endowing ResponsibleTA with security guarantees.

The contributions of this work can be summarized as follows:

- We present a fundamental framework, dubbed as ResponsibleTA, which empowers LLMs with the capabilities of feasibility prediction, completeness verification and security protection for automatically completing tasks in a responsible manner.
- We propose and compare two technical paradigms for implementing the feasibility prediction and completeness verification functionalities of ResponsibleTA in UI task automation scenarios. One is to leverage internal knowledge of LLMs themselves while the other is to train domain-specific models as external assistants.
- We introduce a local memory mechanism to endow ResponsibleTA with the capability of security protection for user privacy, and showcase its effectiveness in some practical representative cases.

2 Related Works

2.1 Development of Large Language Models

Starting from the language model Bert [9], the pretraining-finetuning scheme is a common practice in many natural language processing (NLP) and computer vision (CV) problems. By bringing data and models to a larger scale, GPT-3 [3] demonstrates that LLMs obtain the ability of in-context learning,

where LLMs can quickly adapt to a new task given only a few examples as prompt. Furthermore, InstructGPT [29] finetunes GPT-3 with a dataset of labeler demonstrations to make LLMs follow users’ intent. Recently, ChatGPT [28] and GPT-4 [27] demonstrate excellent ability in generating complete answers to users’ natural language questions. Considering that these LLMs only deal with language tokens, Kosmos-1 [12] extends its training set to image-text data and shows capability in visual question answering and multimodal dialog. In the meanwhile, there are concurrent LLMs such as PaLM [6, 7] and PaLM-E [10], and open-source efforts such as LLaMA [32] and OPT [40]. Despite the strong performance, answers generated by LLMs are not always reliable. Our work aims to empower LLMs to be reliable in the area of task automation, in which it is critical to avoid erroneous executions.

2.2 Large Language Models for Task Automation

LLMs are able to serve as actors or coordinators/planners for digital or physical AI task automation. When functioning as actors, the outputs of LLMs are executable actions [25, 1], which is limited to natural language processing tasks. Towards broader applications in physical [13, 1, 20, 10, 18], simulated physical [34, 35] and digital [30, 36, 39, 18] environments, LLMs usually act as a brain-like coordinator to process human high-level instructions into step-wise executable machine commands and hand them over to domain-specific models/APIs for actual execution. Along this route, LLMs have been opening up infinite possibilities for task automation and putting forward higher requirements for the reliability of automated systems meanwhile. The knowledge defects or biases of LLMs and the gap between LLMs and executors both possibly lead to potential risks in the failure of task completion or even causing harm to users [11]. Previous work [1] attempts to address the executability issue by training an external model to critic the outputs of LLM. It delivered success in simple robotic environment [1] but is demonstrated hard to be applied to complex environments with more objects and diverse actions [31, 37]. Besides, [35, 37] track the execution results for the follow-up planning of LLMs. To the best of our knowledge, we are the first to systematically take into account the reliability of LLM-based task automation from feasibility, completeness and security perspectives.

3 Method

3.1 ResponsibleTA Framework

ResponsibleTA is a fundamental multi-modal framework for empowering LLMs as responsible task automators. The goal is to comprehensively enhance the reliability when using LLMs as coordinators while domain-specific models/APIs as executors for automatically completing tasks in according with human instructions. It achieves reliability enhancement with three empowered capabilities: feasibility prediction, completeness verification and security protection.

As shown in Figure 1, ResponsibleTA takes high-level instructions (*i.e.*, task goals) from human as inputs and parses them into low-level (*i.e.*, step-wise) commands with a pretrained LLM for task planning. To ensure the low-level command is reasonably generated and correctly executed at each step, we introduce a feasibility predictor and a completeness verifier in ResponsibleTA. Before each execution, the feasibility predictor assesses whether the generated low-level command is executable in practice upon the current screenshot and this command itself. It will ask the LLM-based coordinator for a replanning via prompt engineering when the command is judged as infeasible, otherwise feed the command to the executor once feasible. When reaching the pre-set maximum number of replanning attempts, the LLM will terminate the current task and offer feedbacks for users. After each execution, the completeness verifier checks whether the execution result aligns with the goal of given low-level command. Replanning will be launched by LLM once an invalid or unfulfilled execution is detected. Note that we input the UI element information on the screen to the LLM in linguistic form through a trained screen parsing model only when replanning is needed, so as to protect personal information on users’ screens as much as possible. (See supplementary for more details.) Moreover, we integrate a security protector into ResponsibleTA to allow user-sensitive information to be stored locally. With this module, users are allowed to replace all sensitive information by their predefined placeholders, and translates them back when sending the commands to edge-deployed executors for local execution. As a result, ResponsibleTA enhances the internal alignment between the coordinator and executors and provides security guarantees for users as a responsible task automation framework.

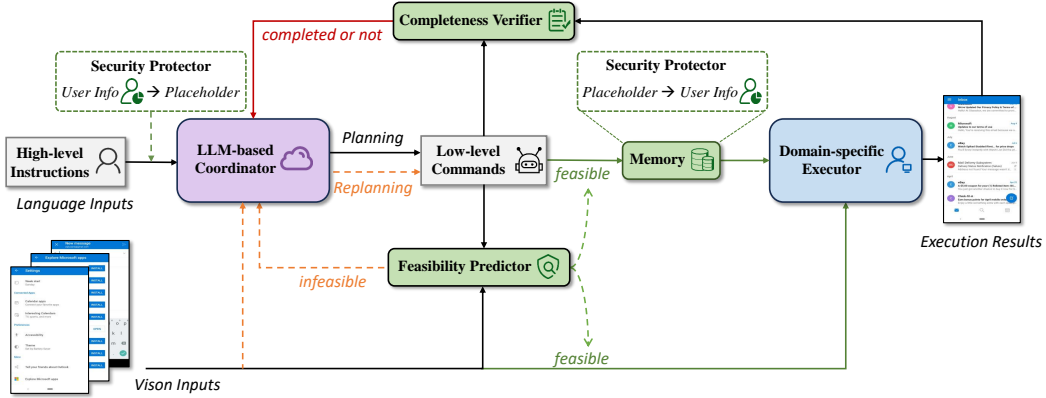


Figure 1: The framework of the proposed ResponsibleTA. It augments the cloud-deployed LLM-based coordinator with feasibility protector, completeness verifier and a local memory, achieving its responsible collaboration with the domain-specific executor. They are all detailed in the main text.

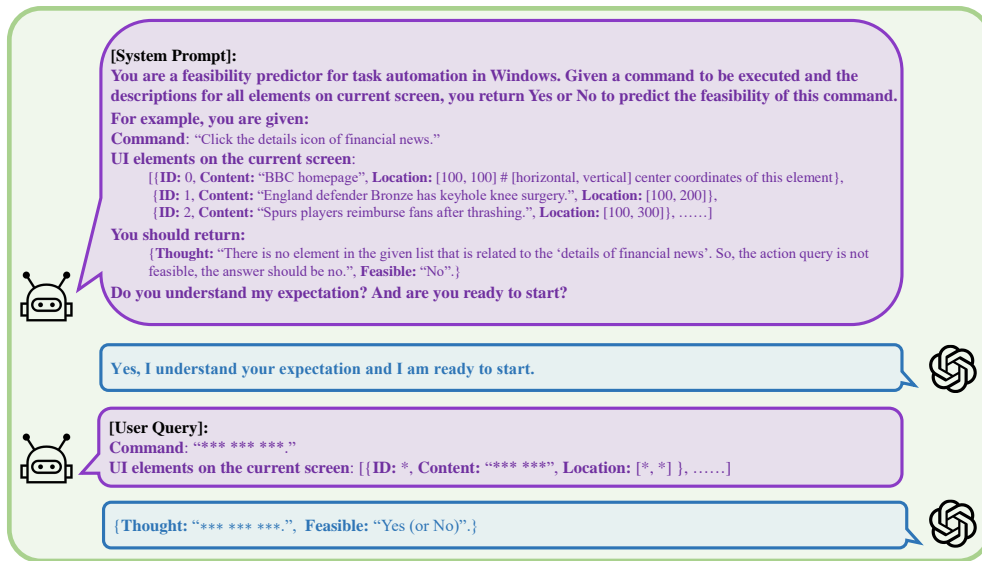


Figure 2: Illustration of our prompt engineering based paradigm for implementing the feasibility predictor in our proposed ResponsibleTA.

We employ UI task automation as a practical application scenario for in-depth investigation and experiments, so as to 1) showcase the effectiveness of ResponsibleTA intuitively and 2) state the implementations of three core modules in ResponsibleTA clearly. Its goal is to automate the interaction tasks between human and UIs (*e.g.*, searching in websites, playing media, online shopping, changing settings, *etc.*). We train a domain-specific executor to automatically ground the target UI element as its spatial coordinates for clicking or information inputting with a given low-level command and the corresponding screenshot as the inputs. Since this is not the emphasis in this work, we place the detailed model design and training of the executor in the experiment section and supplementary.

3.2 Feasibility Predictor

The feasibility predictor takes the low-level command and the current screenshot as inputs to predict the feasibility of this command. It contributes to avoid the execution of a infeasible or dangerous command by asking for replanning once infeasible. Here, we introduce two technical paradigms for its implementation, and compare their effectiveness in the following experiment section.

Prompt engineering based paradigm. This paradigm is designed to leverage the internal generic knowledge of a LLM for achieving feasibility prediction. We first design a system prompt as the launching prompt to tell LLM our task goal, expected outputs together with some specific demonstrations. Considering that most of advanced LLMs (*e.g.*, ChatGPT, GPT-4) have not integrated or released their input interfaces for other modalities except for language, we train a screen parsing model to translate the screenshots into a series of UI elements where each element is represented in linguistic description with its *index*, *text content*, *location* and *type*. We organize the linguistic descriptions of all UI elements at the current screenshot as a structured form (*i.e.*, dictionary), and input it together with the low-level command at the current step as the user queries for each task instance. The system prompt and the demonstrations of user queries of our proposed feasibility predictor are illustrated in Figure 2 in detail.

Domain-specific model based paradigm. This paradigm aims to address the feasibility prediction by training a domain-specific model, *i.e.*, an external expert. Here, we devise an end-to-end multi-modal model with its architecture shown in Figure 3. Our Domain-Specific Feasibility Predictor, referred to as DSFP, consists of a transformer-based vision encoder and a transformer-based language decoder. Given an image $\mathbf{X} \in \mathbb{R}^{H \times W \times C}$, the vision encoder captures the features of \mathbf{X} and embeds it into n_v d -dimensional tokens denoted by $\{\mathbf{v}_i | \mathbf{v}_i \in \mathbb{R}^d, 1 \leq i \leq n_v\}$. The low-level command is tokenized to be n_t d -dimensional text tokens denoted by $\{\mathbf{t}_i | \mathbf{t}_i \in \mathbb{R}^d, 1 \leq i \leq n_t\}$. Then the sets $\{\mathbf{v}\}$ and $\{\mathbf{t}\}$ are fed into the language decoder to generate the prediction results.

To make the above architecture design unified to other tasks, such as completeness verification introduced subsequently, we model the outputs of its decoder in a structured linguistic form as “<task_prompt> {results} </task_prompt>” following [14, 38]. Here, “<task_prompt>” and “</task_prompt>” denotes the start and end in the linguistic sequence for the results, respectively. And “{results}” denotes the actual contents of the results. For the feasibility predictor of ResponsibleTA, “<task_prompt>” and “</task_prompt>” are instantiated to be “<s_feasibility>” and “</s_feasibility>”, respectively. The “{results}” could be 0 or 1 where 0 represents “infeasible” while 1 denotes “feasible”.

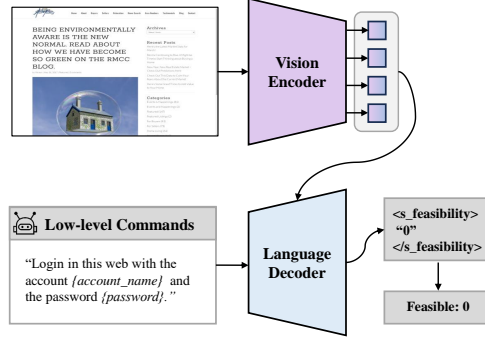


Figure 3: Illustration of our domain-specific model based paradigm for implementing the feasibility predictor in the proposed ResponsibleTA.

3.3 Completeness Verifier

The completeness verifier in ResponsibleTA takes the executed command and the screenshot after execution as its inputs to assess the completeness of this command. It in fact plays the role of handling the internal alignment between the LLM-based coordinator and the executor to ensure the reliability of their collaboration. For each step, the verifier provides a feedback for the LLM-based coordinator to tell whether the execution result of the executor aligns with its expectation at this step so that it can remedy erroneous or biased execution timely via a replanning operation when needed. It forms a loop to endow ResponsibleTA with the capability of self-correction. We showcase its effectiveness in the case study in the experiment section.

We also propose and compare two technical paradigms for implementing this module, which are similar with those of the feasibility predictor introduced as above. For conciseness, we only state the differences with the feasibility predictor here and place more details in our supplementary. For the prompt engineering based paradigm, we share the screen parsing model with that in the feasibility predictor, and update the system prompt and user queries in line with the goal of completeness verification. Their contents are detailed in the supplementary. For the domain-specific model based paradigm, we adopt the same model architecture and the output format with those of the feasibility predictor, but feed the executed command and the screenshot after execution as its inputs. Here, “<task_prompt>” and “</task_prompt>” are instantiated to be “<s_completeness>” and “</s_completeness>”, respectively. The “{results}” could be 0 or 1 where 0 represents “incomplete” while 1 denotes “complete”.

Table 1: The results of quantitative evaluation for the proposed feasibility prediction and completeness verification modules. “AP”, “Acc” and “F1” are short for the average precision, accuracy and F1 score metrics, respectively. The superscript “+” denotes the result evaluated on a larger test dataset.

Models	Feasibility Prediction			Completeness Verification		
	Acc (%)	AP	F1	Acc (%)	AP	F1
LLM-based (ChatGPT)	65.7	0.631	0.546	61.1	0.564	0.704
LLM-based (GPT-4)	68.9	0.669	0.583	62.9	0.575	0.721
DSM-based	74.8	0.818	0.671	83.8	0.803	0.833
DSM-based ⁺	75.3	0.823	0.678	83.5	0.804	0.829

3.4 Security Protector

Many tasks that users need to be automated involve their private information. For example, online shopping or computer configuration may require the credit card, account and password information of users. The security protector in ResponsibleTA is designed to enable the sensitive information from users to be stored and used locally, obviating the transmission of them to the cloud-deployed LLM and reducing information leaky risks. To this end, the security protector uses a NER model (*e.g.*, Bert-NER [19]) to automatically detect the sensitive information in users’ instructions, and replace them with a information placeholder denoted by “*{info_name}*”. Then, it stores “*{info_name}*” and its corresponding real contents using a dictionary structure into an edge-deployed memory. When received a command from LLM, the security protector translates the information placeholder back to its original contents, then sends the translated command to executors for local execution. For example, when users ask ResponsibleTA to change the shipping address in a shopping website, their old and new address are represented as “*{old_address}*” and “*{new_address}*”, respectively, when interacting with the cloud-deployed LLMs, and are translated back for local executors.

4 Experiments

4.1 Datasets and Implementation Details

Feasibility prediction dataset. We collect data for feasibility prediction from Common Crawl², an open repository of web crawl data. We sample about 40K web pages from this open repository, yielding 1.1M image-text pairs. Among them, 1M pairs are used for training while the remaining 100K pairs are for testing. For each web page, we extract leaf elements and assign captions to them from HTML entries such as *inner-text*, *value*, *alt*, *aria-label*, *label-for*, and *placeholder*. Subsequently, we employ a randomized algorithm to generate low-level instructions for elements by leveraging their captions. Example instructions include “select the *{element_caption}* item”, “click the item to the right of *{element_caption}*”, “enter *{words}* into *{element_caption}*”, “scroll until *{element_caption}*”, *etc.* All of these instructions are deemed as feasible. Infeasible instructions are generated based on fake element captions that do not appear in the current web page.

Completeness verification dataset. This dataset is also based on publicly available web pages. This dataset also leverages the transitions between two web pages, *i.e.*, it jumps to page B by clicking some element in page A. Based on the transitions we are able to define a positive example of completeness as a 3-tuple [screenshot A, *{element_caption}*, screenshot B]. We generate negative examples by substituting the 3-tuple with fake items. Overall, this dataset includes 113K web pages with 1.2M image-text pairs for training and 6K web pages with 60K image-text pairs for testing.

The data for task automation executor is based on the feasibility prediction dataset. We place more details about the dataset configuration and model implementation in our supplementary.

4.2 Quantitative Results

We evaluate afore-introduced two technical paradigms for implementing the feasibility predictor and completeness verifier in ResponsibleTA. The experiment results are in Table 1. Due to the cost

²<https://commoncrawl.org/the-data/>

Table 2: The case study results of task automation in the real world. The “*Baseline*” denotes the model consisting of a LLM-based coordinator and a task automation executor, without our proposed feasibility prediction and completeness verification modules. The “+*Fea.*” and “+*Com.*” refers to adding the feasibility prediction module and adding the completeness module, respectively. We represent each execution result with its “*Progress*” and “*End Status*”. Here, the “*Progress*” is shown in the form of the number of “*valid steps / total execution steps (human expert steps)*”, in which “*human expert steps*” refers to the step number of completing given instruction by a human expert. For the “*End Status*”, ✓ means the task goal has been achieved in the end while ✗ indicates it has not.

No.	High-level Instruction	Baseline		Baseline+Fea.		Baseline+Fea.+Com.	
		Progress	End Status	Progress	End Status	Progress	End Status
1	Open football news in bbc.com.	4/4 (4)	✓	4/4 (4)	✓	4/4 (4)	✓
2	Find the view setting page in Outlook.	4/4 (4)	✓	4/4 (4)	✓	4/4 (4)	✓
3	Navigate to the language setting in my Windows11.	3/3 (3)	✓	3/3 (3)	✓	3/3 (3)	✓
4	Find the system setting for text size in my Windows11.	1/3 (3)	✗	3/3 (3)	✓	3/3 (3)	✓
5	Help me open the latest received e-mail in my Outlook.	2/3 (3)	✗	3/3 (3)	✓	3/3 (3)	✓
6	Go to github.com and check issues that mentioned me, already logged in.	2/5 (4)	✗	4/4 (4)	✓	4/5 (4)	✓
7	Log in Instacart with username {username} and password {password}.	5/6 (6)	✗	6/6 (6)	✓	6/6 (6)	✓
8	Go to Amazon and add a pair of gloves into the shopping cart.	4/6 (6)	✗	6/6 (6)	✓	6/6 (6)	✓
9	Go to Amazon and add the cheapest charger into the shopping cart.	4/7 (9)	✗	5/5 (9)	✗	9/9 (9)	✓
10	Add my Costco’s loyalty card number {card_num} in the website {web_url}.	2/6 (6)	✗	3/3 (6)	✗	6/7 (6)	✓
11	Create a meeting at 2023/04/15 14:00-14:30 in Outlook.	3/6 (8)	✗	3/3 (8)	✗	3/3 (8)	✗
12	Search the Cpython repo and download its zip file in github.com.	5/7 (8)	✗	6/6 (8)	✗	6/6 (8)	✗

and efficiency of testing with LLMs, we randomly sample 5K image-text pairs from the test splits on feasibility prediction and completeness verification for evaluating the prompt engineering based paradigms. As for domain-specific models based paradigms, we evaluate them on the same 5K image-text pairs (abbreviated as DSM in Table 1) as well as on the entire test splits (abbreviated as DSM⁺ in Table 1). As shown in Table 1, we can find that *DSM-based* and *DSM-based*⁺ perform very closely under these two test settings. This indicates our sampled 5K image-text pairs are diverse and representative enough for evaluation.

Accuracy of feasibility prediction. As shown in Table 1, GPT-4 is more powerful than ChatGPT for feasibility prediction. And the *DSM-based* feasibility predictor outperforms the *LLM-based* (*GPT-4*) one by 5.9%, 0.149, 0.088 in accuracy, average precision and F1 score, respectively, on feasibility prediction. These results demonstrate the superior performance of the domain-specific models based paradigm relative to prompt engineering based paradigm on feasibility prediction.

Accuracy of completeness verification. From the in Table 1, we observe that *DSM-based* feasibility predictor outperforms the *LLM-based* (*GPT-4*) one by 20.9%, 0.228, 0.112 in accuracy, average precision and F1 score, respectively, on completeness verification, showing a similar tendency with that on feasibility prediction. It also indicates the performance superiority of adopting a domain-specific model as the completeness verifier in ResponsibleTA.

As above, we can find the domain-specific model based paradigms perform consistently better than prompt engineering based paradigms on implementing these two functionalities of ResponsibleTA. Despite this, the advantage of prompt engineering based paradigms is that they do not require the collection of specific data for training, offering better flexibility in practical deployment.

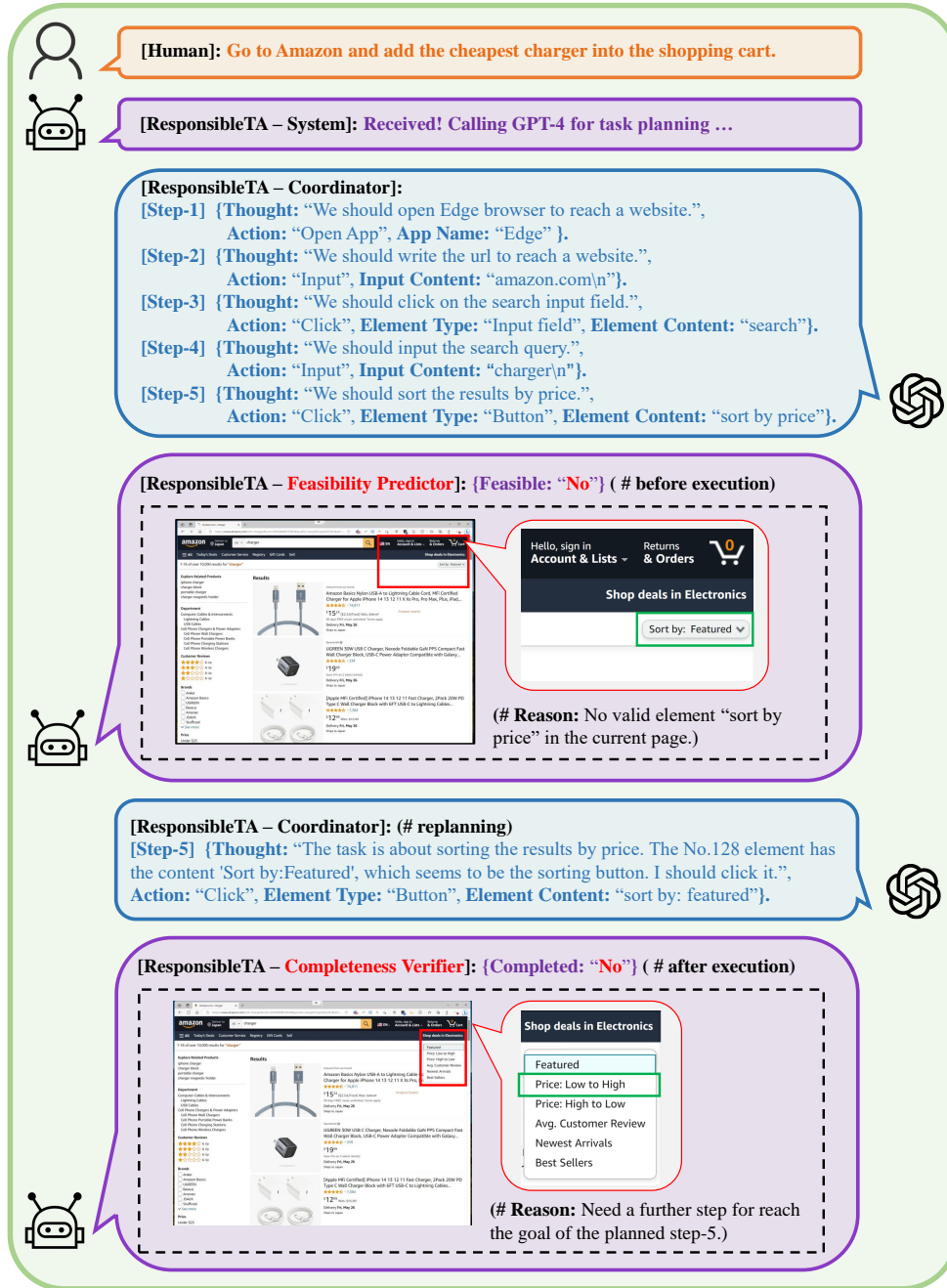


Figure 4: Detailed case study about how our proposed feasibility predictor and completeness verifier in ResponsibleTA remedy the failure case to achieve success on the No.9 task in Table 2. The 6-th to 9-th steps are omitted for brevity. GPT-4 [27] is used as the LLM-based coordinator.

4.3 Case Study and Demonstration

We analyze the performance and behaviors of our proposed ResponsibleTA via real-world case study on 12 tasks, considering that there are no fledged benchmarks in this research field yet. Besides, we conduct ablation study on these tasks by comparing *Baseline*, *Baseline+Fea.* and *Baseline+Fea.+Com.* where *Baseline+Fea.+Com.* is the complete version of ResponsibleTA. Their configurations are detailed in the caption of Table 2. The security predictor is installed in all models.

Ablation study and analysis. We report the specific completion progress and the final success status over all 12 real-world tasks in Table 2. As for the success ratio, we can find that *Baseline* does not reach the final success in 9 out of 12 tasks. For these 9 tasks, 5 of them are successfully remedied by the feasibility predictor on its own. On top of it, the completeness verifier help turn 2 of them to the final success in addition. It can be seen that the feasibility predictor and the completeness verifier in ResponsibleTA can significantly improve the success ratio of task automation by providing feedbacks for LLM-based coordinator so that it can replan timely. *How our ResponsibleTA achieves this* will be elaborated in detail in the subsequent text.

Besides the benefits in improving the success rate, our proposed ResponsibleTA can effectively reduce the number of invalid instruction executions thanks to its feasibility predictor. This conclusion is drawn by comparing the valid steps and total execution steps for *Baseline* and *Baseline+Fea.* models. Thus, our ResponsibleTA can avoid the risks when executing those invalid steps.

Moreover, we further verify the role of ResponsibleTA on security protection by comparing using placeholders and real user information in the No.7 and No.10 tasks. We experimentally find that launching the security protector neither affects the task success rate nor increases the number of invalid executions on these two cases for all three models. It can effectively obviate the need of uploading user sensitive information to the cloud.

How ResponsibleTA remedies failure cases? Here, we take a close look into a specific case for detailed analysis on how ResponsibleTA remedies failure cases to achieve success in the end. We illustrate specific execution processes of the first 4 steps for the No.9 task in Figure 4. The execution for remaining steps and the corresponding prompts are omitted here for brevity. As shown in Figure 4, the LLM-based coordinator generates feasible low-level commands for the first 4 steps, and the corresponding execution is smooth for all three models. For the 5-th step, the coordinator originally thinks we should click the button with the content of “sort by price”. It is seemingly a reasonable command but does not match the current web page in fact, because there is no matched button. At this point, the *Baseline* model executes this infeasible command in a seemingly random manner, leading to 4 valid steps in total. When ResponsibleTA has the feasibility predictor, this module will return “infeasible” signal to the coordinator so that it can replan upon additionally provided screen page information. After the replanning, the coordinator gives a feasible command, *i.e.*, click the “Sort by: Featured” button, and yields one more valid step. Despite this, *Baseline+Fea.* still fails for the next step since it does not realize that another click operation is further needed to achieve the goal of sorting by price. This can be effectively remedied by the complete version of ResponsibleTA thanks to its completeness verifier. With the assist of this module, ResponsibleTA add such click operation for the “Price: Low to High” button before executing subsequent steps via another replanning. In this way, it adjusts timely and contributes to the eventual success.

5 Limitation

We have to admit that it is difficult, even impossible, to cover all aspects of a newly devised system in a single academic paper. This paper make contributions and deliver insights from the perspectives of framework design and technical paradigm comparison. Its remaining limitations lie in two aspects: 1) benchmark construction; 2) specific model design/iteration aimed at pursuing high performance for this new domain. We here call on the community to address them with our work as a start point.

6 Conclusion and Broader Impact

In this paper, we present a fundamental multi-modal framework, ResponsibleTA, for empowering LLMs as responsible task automators. In specific, we enhance LLMs with three core capabilities, *i.e.*, feasibility prediction, completeness verification and security protection. Moreover, we propose and compare different technical paradigms for implementing these core functionalities. We experimentally observe that domain-specific models deliver superior performance compared to prompt engineering based solutions on their implementations, while the later one does not require collecting domain-specific data for model training. Besides, we also demonstrate the effectiveness of our proposed ResponsibleTA and provide explanations on how our ResponsibleTA works intuitively through case study. As for the broader impact, we hope this work can inspire more excellent works on benchmark construction, method design and functionality extension on top of our this work in the future.

References

- [1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.
- [2] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.
- [3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, volume 33, pages 1877–1901, 2020.
- [4] Ting Chen, Saurabh Saxena, Lala Li, David J Fleet, and Geoffrey Hinton. Pix2seq: A language modeling framework for object detection. *ICLR*, 2022.
- [5] Ting Chen, Saurabh Saxena, Lala Li, Tsung-Yi Lin, David J Fleet, and Geoffrey E Hinton. A unified sequence interface for vision tasks. *Advances in Neural Information Processing Systems*, 35:31333–31346, 2022.
- [6] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.
- [7] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.
- [8] MDN contributors. Dom, 2021. https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction.
- [9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186, 2019.
- [10] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.
- [11] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. More than you’ve asked for: A comprehensive analysis of novel prompt injection threats to application-integrated large language models. *arXiv preprint arXiv:2302.12173*, 2023.
- [12] Shaohan Huang, Li Dong, Wenhui Wang, Yaru Hao, Saksham Singhal, Shuming Ma, Tengchao Lv, Lei Cui, Owais Khan Mohammed, Qiang Liu, et al. Language is not all you need: Aligning perception with language models. *arXiv preprint arXiv:2302.14045*, 2023.
- [13] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pages 9118–9147. PMLR, 2022.
- [14] Geewook Kim, Teakgyu Hong, Moonbin Yim, JeongYeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoon Yun, Dongyoon Han, and Seunghyun Park. Ocr-free document understanding transformer. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVIII*, pages 498–517. Springer, 2022.
- [15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [16] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

- [17] Chenxia Li, Weiwei Liu, Ruoyu Guo, Xiaoting Yin, Kaitao Jiang, Yongkun Du, Yuning Du, Lingfeng Zhu, Baohua Lai, Xiaoguang Hu, Dianhai Yu, and Yanjun Ma. Pp-ocrv3: More attempts for the improvement of ultra lightweight ocr system, 2022.
- [18] Yaobo Liang, Chenfei Wu, Ting Song, Wenshan Wu, Yan Xia, Yu Liu, Yang Ou, Shuai Lu, Lei Ji, Shaoguang Mao, et al. Taskmatrix.ai: Completing tasks by connecting foundation models with millions of apis. *arXiv preprint arXiv:2303.16434*, 2023.
- [19] David Lim. dslim/bert-base-ner, 2023. <https://huggingface.co/dslim/bert-base-NER>.
- [20] Bill Yuchen Lin, Chengsong Huang, Qian Liu, Wenda Gu, Sam Sommerer, and Xiang Ren. On grounded planning for embodied tasks with language models. *AAAI*, 2023.
- [21] Thomas F. Liu, Mark Craft, Jason Situ, Ersin Yumer, Radomir Mech, and Ranjitha Kumar. Learning design semantics for mobile apps. In *The 31st Annual ACM Symposium on User Interface Software and Technology*, UIST '18, pages 569–579, New York, NY, USA, 2018. ACM.
- [22] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.
- [23] Chengqi Lyu, Wenwei Zhang, Haiyan Huang, Yue Zhou, Yudong Wang, Yanyi Liu, Shilong Zhang, and Kai Chen. Rtmddet: An empirical study of designing real-time object detectors, 2022.
- [24] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018.
- [25] Vincent Micheli and François Fleuret. Language models are few-shot butlers. *arXiv preprint arXiv:2104.07972*, 2021.
- [26] Microsoft. Microsoft ui automation, 2023. <https://learn.microsoft.com/en-us/dotnet/framework/ui-automation/ui-automation-overview>.
- [27] OpenAI. Gpt-4, 2023. <https://openai.com/research/gpt-4>.
- [28] OpenAI. Introducing chatgpt, 2023. <https://openai.com/blog/chatgpt>.
- [29] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, et al. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.
- [30] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. *arXiv preprint arXiv:2303.17580*, 2023.
- [31] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworlde: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.
- [32] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- [33] Sai Vemprala, Rogerio Bonatti, Arthur Bucker, and Ashish Kapoor. Chatgpt for robotics: Design principles and model abilities. Technical Report MSR-TR-2023-8, Microsoft, February 2023.
- [34] Ryan Volumn, Sudha Rao, Michael Xu, Gabriel DesGarnes, Chris Brockett, Benjamin Van Durme, Olivia Deng, Akanksha Malhotra, and William B Dolan. Craft an iron sword: Dynamically generating interactive game characters by prompting large language models tuned on code. In *Proceedings of the 3rd Wordplay: When Language Meets Games Workshop (Wordplay 2022)*, pages 25–43, 2022.

- [35] Zihao Wang, Shaofei Cai, Anji Liu, Xiaojian Ma, and Yitao Liang. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. *arXiv preprint arXiv:2302.01560*, 2023.
- [36] Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*, 2023.
- [37] Yue Wu, So Yeon Min, Yonatan Bisk, Ruslan Salakhutdinov, Amos Azaria, Yuanzhi Li, Tom Mitchell, and Shrimai Prabhunoye. Plan, eliminate, and track—language models are good teachers for embodied agents. *arXiv preprint arXiv:2305.02412*, 2023.
- [38] Zhengyuan Yang, Zhe Gan, Jianfeng Wang, Xiaowei Hu, Faisal Ahmed, Zicheng Liu, Yumao Lu, and Lijuan Wang. Unitab: Unifying text and box outputs for grounded vision-language modeling. In *Computer Vision—ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXVI*, pages 521–539. Springer, 2022.
- [39] Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. Mm-react: Prompting chatgpt for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*, 2023.
- [40] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.