OFFICEBENCH: Benchmarking Language Agents across Multiple Applications for Office Automation

Anonymous ACL submission

Abstract

Office automation significantly enhances human productivity by automatically finishing routine tasks in the workflow. Beyond the basic information extraction studied in much of the prior document AI literature, the office automation research should be extended to more realistic office tasks which require to integrate various information sources in the office system and produce outputs through a series of decision-making processes. We introduce OFFICEBENCH, one of the first office automation benchmarks for evaluating current LLM agents' capability to address the office tasks in realistic office workflows. OFFICEBENCH requires LLM agents to perform feasible long-horizon planning, proficiently switch between applications in a timely manner, and accurately ground their actions within a large combined action space, based on the contextual demands of the workflow. Applying our customized evaluation methods on each task, we find that GPT-4 Omni achieves the highest pass rate of 47.00%, demonstrating a decent performance in handling office tasks. However, this is still far below the human performance and accuracy standards required by real-world office workflows. We further observe that most issues are related to operation redundancy and hallucinations, as well as limitations in switching between multiple applications, which may provide valuable insights for developing effective agent frameworks for office automation. Code and data will be released upon acceptance.

1 Introduction

Office automation plays a pivotal role in interacting with diverse environments to accomplish complex goals set by users. In the rapidly evolving landscape of workplace technology, the integration of office automation into daily tasks represents a critical advancement with the potential to significantly enhance human efficiency and transform traditional workflows (Aghion et al., 2023; Filippi et al., 2023).

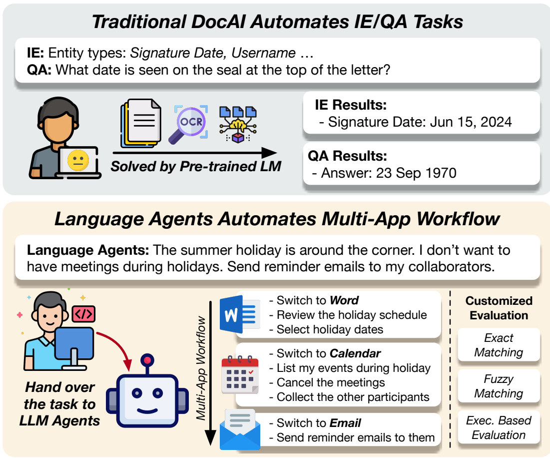


Figure 1: **OFFICEBENCH is one of the first office automation benchmarks for language agents.** We assess the ability of language agents to perform complex office workflows across multiple applications using customized evaluation methods, such as Exact Matching, Fuzzy Matching, and Execution-based Evaluation.

By automating routine and time-consuming tasks, office automation systems free up human workers to focus on more strategic and creative aspects of their roles (Howcroft and Taylor, 2023).

Towards the ambitious goal of automating office work, numerous efforts have been made from both industry and academia (Binmakhashen and Mahmoud, 2019; Cui et al., 2021). One common direction is Document AI which automates the fundamental tasks, such as information extraction and question answering, by pre-trained language models (Xu et al., 2020; Wang et al., 2021; Garncarek et al., 2021; Xu et al., 2021; Hong et al., 2022; Huang et al., 2022; Perot et al., 2023). Following this direction, many benchmarks include structured documents with detailed annotations, requiring language models to understand the rich structure and extract the key information or respond to the specific questions posed within these documents. (Jaume et al., 2019; Park et al., 2019; Mathew et al., 2021; Wang et al., 2023c).

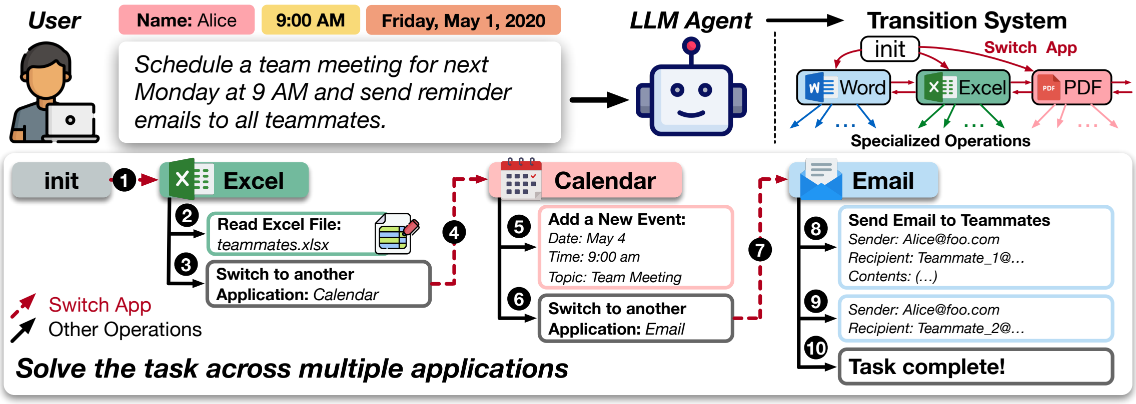


Figure 2: **Illustration of the workflow in OFFICEBENCH** where the LLM agent leverages the operations from multiple applications to systematically construct an operation chain that addresses the office tasks effectively. The framework is formulated as a transition system where the current application serves as the *state* and the operations serve as the *transitions*. Specialized operations, such as `read_file` and `send_email`, perform specific tasks.

However, a realistic office environment extends far beyond basic extraction tasks. Prior works on structured document understanding are only part of the office automation pipeline. For example, extracting data from PDF invoices is just the beginning; the full process involves integrating this data into financial software, flagging discrepancies, and generating payment reminders. It is necessary to develop and evaluate an entire office automation framework that seamlessly integrates various information within the office system, ensuring the output aligns with logical planning processes. With the impressive planning and reasoning capabilities of large language models (LLMs) (Achiam et al., 2023; Team et al., 2023; Reid et al., 2024), language agents powered by them are expected to construct feasible operation chains to execute the typical workflows for human labors, including but not limited to the information extraction tasks.

To this end, we propose **OFFICEBENCH**, as one of the first office automation evaluation benchmarks for LLM agents. By deploying agents within simulated human labor workflows that replicate the complexity and variability of modern office environments, this benchmark is instrumental in assessing the ability of language agents to handle a variety of office tasks across different applications. The OFFICEBENCH benchmark operates within a Docker environment pre-installed with office applications such as `Word`, `Excel`, `calendar`, and `email clients` to simulate various scenarios, including sending emails, editing tables, and scheduling events. With the consideration of multiple applications, LLM agents are required to demonstrate their proficiency in *switching* between applications on time, and grounding their actions accurately from a *large combined action space* based on the contextual demands of the workflow. Furthermore, OFFICEBENCH incorporates various evaluation methods including exact matching, fuzzy matching, and execution-based evaluation customized for each test example in the benchmark, allowing for the results of agent actions to be assessed in various file formats. This feature is critical in precisely assessing the agent ability to follow the user task instruction individually.

We extensively evaluate state-of-the-art LLMs as language agents in following natural language commands and perform various office tasks across multiple applications. We evaluate the proprietary GPT-3.5, GPT-4 (Achiam et al., 2023), Gemini-1.0 (Team et al., 2023), Gemini-1.5 (Reid et al., 2024), and open-weights Llama 3 (Meta, 2024), Qwen 2 (Bai et al., 2023). The experimental results indicate that GPT-4 Omni achieves the highest pass rate of 47.00%, showcasing a decent performance of current LLMs in handling office automation tasks. However, this is still well below the accuracy standards required by real-world office workflows, highlighting the need for continued research to further explore the limits of language agents powered by LLMs. We further conduct ablation study on application switching in multi-application scenarios, and analyze the failure cases. We identify issues related to operation redundancy and hallucinations, as well as limitations of current LLM in complex planning across multiple applications.

With our proposed OFFICEBENCH benchmark, we would like to shed new light on more robust and effective language agents, facilitating the development of advanced automation for realistic, everyday tasks, and breaking down invisible barriers in modern workspace, including those related to disability, education, and cultural differences.

2 Related Work

Language Agent Benchmarks Previous studies typically assess LLM agents in focused domains, such as arithmetic reasoning, which targets correct solutions, and tool-use, evaluating agents' proficiency in employing tools (Yang et al., 2018; Cobbe et al., 2021; Xu et al., 2023; Liu et al., 2023; Wang et al., 2023a; Ma et al., 2024). The most recent evaluation benchmarks have increasingly focused on real-world scenarios, including web and OS environments (Deng et al., 2023; Zhou et al., 2023; Koh et al., 2024; Lù et al., 2024; Xie et al., 2024), wheere they enable agents to interact with web/OS interfaces using keyboard and mouse actions. Different from these prior works, OFFICEBENCH is an agent evaluation benchmark specifically designed to assess LLM abilities within real-world workflows, requiring the operation of multiple software applications to complete tasks. OFFICEBENCH encompasses a larger action space and demands LLMs to possess the capability in switching between software applications as needed. It is also one of the first benchmarks to offer customized evaluation methods tailored to different software and individual tasks, ensuring a precise assessment. OFFICEBENCH provides an extensible and cost-effective evaluation platform, supporting the addition of new applications and tasks compatible with the Bash environment with less manual effort than the complex simulators in OSWorld annotated with around 1800 human hours (Xie et al., 2024).

Document AI Benchmarks Document AI focuses on various structured documents, including invoices, receipts, forms, and tables. Previous studies primarily focus on the information extraction tasks on these documents, assessing the capability of language models in understanding the textual contents and rich structural information. CORD (Park et al., 2019) and SPOIE (Huang et al., 2019), FUNSD (Jaume et al., 2019) and VRDU (Wang et al., 2023c) incorporate grocery receipts or multi-domain forms for entity extraction tasks. DocVQA (Mathew et al., 2021) formulates the structured document understanding as an extractive QA task. Realistic office scenarios involve more comprehensive workflows with multiple applications. The information extraction or question answering tasks are only parts of the complex workflow. Our proposed OFFICEBENCH go beyond the document-based benchmarks and evaluate the powerful LLM agents in calling different applications for general office automation.

We further compare OFFICEBENCH with recent benchmarks in Document AI and language agents from different perspectives in Appendix A. OFFICEBENCH excels in cross-application scenarios, offering a diverse suite of precisely curated customized evaluation functions for each task. Additionally, it supports a larger action space and provides more extensible task annotation and environment creation capabilities.

3 OFFICEBENCH: Modeling Office Works Across Multiple Applications

Office automation must be capable of complex planning and reasoning to construct an applicable chain of actions for solving real-world tasks. While LLMs have demonstrated satisfactory performance in single-application scenarios (Wang et al., 2023b; Zhou et al., 2023; Deng et al., 2023; Lù et al., 2024), comprehending the diverse execution environments including various applications and effectively managing a vast action space for realistic tasks remains a challenge. To evaluate LLM agent performance on office automation in multi-application scenarios, in OFFICEBENCH benchmark, we develop a realistic and extensible framework designed to simulate office work scenarios which incorporates applications such as `Word`, `Excel`, `PDF`, `Shell` and `email client`. The framework also supports a large set of valid actions applicable to these applications. LLM agents should smartly leverage the applications supported in the environment with the valid actions by utilizing their advanced planning and reasoning abilities. In this section, we present the overall framework of OFFICEBENCH, detailing the multi-application environment and the workflow of the automation system with the large action space.

3.1 Multi-Application Environment

We formulate an office task for autonomous agents as a task description T with a variety of applications commonly used in office scenarios, such as `Word`, `Excel`, `PDF`, and `Shell`. Each application is defined by a distinct set of operations tailored to the specific usage. These operations are denoted as $A = \{\alpha_1, ..., \alpha_n\}$, where A represents the applica-

Applications	Operation Examples
`System` (2)	`switch_app, submit_task`
`Word` (4)	`convert_to_pdf, write_to_file`
`Excel` (5)	`set_cell, read_excel_file`
`PDF` (3)	`convert_to_doc, read_pdf_file`
`Calendar` (3)	`create_event, delete_event`
`Email` (3)	`list_emails, send_email`
`OCR` (1)	`recognize_text`
`ChatGPT` (1)	`query_chatgpt`
`Shell` (1)	`run_command`

Table 1: **Applications and their corresponding operations** implemented in OFFICEBENCH for simulating a realistic office scenario. The number in the brackets are the total number of the operations in this application. See Appendix B for details.

tion, and each α_i represents an individual operation within this application's environment. For example, in the context of the `Excel` application, we design specialized operations such as `read_excel_file` and `set_cell_content`, which are explicitly designed to interact with spreadsheet data.

Overall, as shown in Table 1, we have designed a total of 9 applications within the multi-application environment of OFFICEBENCH to simulate a realistic office work scenario. These include specialized applications, such as `Word`, `Excel`, `PDF`, `Calendar`, `Email`, `OCR`, `ChatGPT`, `Shell`, and an auxiliary application `System`. We develop various basic operations for each application as listed in Table 1. In OFFICEBENCH, the LLM agents are able to leverage the operations from multiple applications to systematically construct an operation chain that addresses the office tasks effectively.

Application Transition In a single-application environment, it is relatively straightforward to consistently engage with only one application, querying the LLM agents for subsequent actions based on interaction feedback with that application. However, when it comes to a multi-application environment, it is necessary to design approaches to coordinate the various applications. Drawing inspiration from the idea of operating systems, we introduce an auxiliary application named `System`, serving as a foundational platform that coordinates other specialized applications. This `System` application is crucial as it includes the operation `switch_app`, which is designed to manage the seamless transition between multiple execution environments tailored for various applications. Once the agent self-identifies that it has already obtained what it expects from an application, then it can use the `switch_app` operation to change to another one. For example, when solving a task "*Send emails to the participants of the meeting today.*", an LLM agent needs to switch to `Email` after acquiring participants information from the `Calendar`.

Operation Observation We integrate the operation outputs into the observation space of the LLM agents, formatting these outputs textually so that the agents can directly learn from the rich signals they contain. Given the variety of operations across different applications in OFFICEBENCH, we handle each case individually. In simpler cases, we directly print the outputs for LLM agents. For example, when calling `run_command` with the `Shell` application, we simply copy the terminal outputs to the execution history. In cases involving structured data, we decode the structure and retain the essential information in the textual outputs. For example, when calling `read_excel_file` with the `Excel` application, we list the cell values along with their indices in the format (i, j) : `Value` where i, j are the row and column indices, and `Value` is the content of the specified cell. Refer to Appendix C for the formalized outputs in more details.

3.2 Autonomous Workflow

Based on the multi-application environment and supported action space of OFFICEBENCH, we formulate the autonomous workflow as a transition system $\mathcal{E} = \{\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}\}$, standing for state space $\mathcal{S}$, action space $\mathcal{A}$, observation space $\mathcal{O}$, and transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$, as shown in Figure 2. We set the currently selected application as the *state* in the transition system and introduce the restricted action space for each application. We further specify the observation space and termination condition of the agent system in this section.

Restricted Action Space The current application A in use determines the set of actions that are currently valid. Given the extensive range of operations across various applications, we restrict the action space to the specialized operations within A. Additionally, we include the `switch_app` and `submit` operation in the action space, allowing the LLM agent to switch to another application or submit the task when necessary. Specifically, supposing the application at timestamp t is A_t, the action space is formulated as, $\{\alpha_{t1}, \alpha_{t2}, ..., \alpha_{tm}\} \cup \{\texttt{switch\_app}, \texttt{submit}\}$, where $\{\alpha_{t1}, \alpha_{t2}, ..., \alpha_{tm}\}$ are valid actions under the application A_t.

Observation Space In OFFICEBENCH, we provide the LLM agents with the full execution history in the prompt as the observation so the LLM agents can determine the next action based on the previous actions and their observed outputs, leading the system to transition to the next state. Specifically, at timestamp t, the execution history of previous actions is represented as $H_t = [(A_1, \alpha_1, o_1), ..., (A_{t-1}, \alpha_{t-1}, o_{t-1})]$, where (A_i, α_i, o_i) denotes the application, action, and the observed outputs at timestamp i, respectively. The observed outputs of each action are introduced in Section 3 and listed in Table 5. The LLM agent predicts the next action among the restricted action space based on the history H_t and triggers a transition function to proceed to the next state.

Termination Condition LLM agents are designed to iteratively predict and execute operations as a transition system until the given task is completed. In the `System` application, we implement an operation, `submit_task`, as a symbol of normal termination. Nevertheless, due to the limitations of current LLM agents, we have identified two additional conditions that necessitate terminating the agent system prematurely – *Operation Stagnation:* If an LLM agent continuously generates the same operation multiple times, we interpret this as a failure. Specifically, if this repetition occurs 5 times consecutively, we terminate the system and classify the task as unsuccessful in OFFICEBENCH. *Iteration Overflow:* Given the constraints on resources, it is necessary to limit the number of iterations an LLM agent can perform. Therefore, we set a maximum step limit as 50 to prevent excessive resource use and ensure timely task completion.

3.3 Implementation Details

We build OFFICEBENCH in a Docker environment with pre-installed applications and use Python libraries to automate the operations. We create a file system for the documents, emails, and calendar events involved in the tasks. We formulate each user's emails as ".eml" files under a specific directory (e.g. `/emails/[username]/`). Similarly, we save the user's calendar events as a ".ics" file (e.g. `/calendar/[username].ics`). We save the other ordinary documents in `/data/`. After the agents finish the task, we save the entire file system and run customized evaluation to check the correctness.

4 Benchmark Annotation and Evaluation

In OFFICEBENCH, we construct a comprehensive suite of 300 tasks to evaluate the performance of LLM agents in office automation. For each task, we synthesize documents, emails, and calendar events involved in the tasks to simulate a realistic scenario. We also design customized evaluation methods, including the exact and fuzzy matching, and the execution-based evaluation. We outline the annotation process and describe our comprehensive evaluation framework in this section.

4.1 Task Annotation

OFFICEBENCH evaluates the capability of LLM agents in managing multiple applications with the three categories of tasks, Single App, Two Apps, and Three Apps, specifying the number of involved apps. Among these tasks, the difficulty level increases with more applications involved. Overall, we annotate 93, 95, and 112 tasks in these three categories, respectively.

Single App Tasks In the Single App category, tasks are relatively easier. The LLM agents select one application in the beginning, adhere to it, and plan an operation chain to solve the task. With these simpler tasks, we aim to investigate whether the LLM agents are able to understand the functionality of the elementary operations in each application.

Two Apps Tasks In the other two categories, Two/Three Apps, LLM agents need to switch to another application once they self-identify that they have already obtained what it expects from the current application. When annotating tasks in the Two Apps category, we request annotators to brainstorm realistic and diverse tasks relevant to every pair of applications in OFFICEBENCH. For example, when integrating `PDF` and `Email`, we design a task "*Extract a notification from a business travel image and send emails to Bob and Tom*".

Three Apps Tasks In order to further evaluate LLM agents with more challenging tasks, we expand the tasks in the Two Apps category with one more relevant application while ensuring the validity of the combination. In this way, we annotate more complex tasks in the Three Apps category. For example, we already annotate a task of Two Apps (`Excel` and `Calendar`): "*Schedule a team training session for all participants from the Excel file and create calendar events for each member*".

We add a relevant application `Email`, requesting LLM agents to email the training details to each participant in the following steps. Despite the seemingly simple addition, the tasks in the Three Apps category present a greater challenge to LLM agents, requiring them to adeptly manage dynamic switching between the applications.

4.2 Data Synthesis

We aim to simulate a realistic office environment in OFFICEBENCH. A delicate file system is indispensable. We synthesize the documents of various formats, emails, and calendar events for each of the tasks in our benchmark. To eliminate the sensitive privacy issues, we resort to ChatGPT[1] and random generators instead of using real data. Specifically, we query ChatGPT to generate paragraphs on specific topics as needed, and run Python programs to generate random numbers. For example, to synthesize exam scores for a class, we initially query ChatGPT to generate a list of common student names and then assign each student a random score ranging from 0 to 100. When it comes to files with special formats, such as images, PDFs, we use the HTML format as an intermediary. In particular, we first edit the HTML files to involve rich layout structure and then convert it to the requested formats. Similarly, for emails and calendar events, we fill in the fields in the special data structure with synthesize contents. Finally, we copy the involved data to the OFFICEBENCH Docker environment for the evaluation of the LLM agents.

4.3 Evaluation Framework

To evaluate LLM agents within the simulated office workflow of OFFICEBENCH, it is crucial to develop a precise and reliable method for assessing the output files produced by these agents after planning and execution. Given the diversity of the office work tasks, the task metrics may greatly vary due to the different task requirements and involved applications. Following Zhou et al. (2023); Xie et al. (2024), we incorporate the exact matching, fuzzy matching, and execution-based methods into the evaluator of OFFICEBENCH (See Appendix D for detailed examples).

Exact Matching & Fuzzy Matching In the exact matching setting, we utilize our annotated ground-truth outputs of the tasks as references and assess whether LLM agent's final outputs match them exactly. For example, given a task "*Bob got 98 points in the final exam. Add his score in final-exam.xlsx.*", we add a new row for Bob and his score in the specified file, *final-exam.xlsx*, and compare the file processed by the LLM agent with the ground-truth annotation. However, when evaluating more complex tasks, it becomes challenging to design strict criteria for the correct answer. For example, consider the task: "*Add a meeting to Bob's calendar from 10:30 am to 11:00 am tomorrow.*" In this case, we employ a fuzzy matching function to assess accuracy. This function checks the correctness of the timestamps in the calendar event and verifies that the event subject includes the keyword *meeting*. We disregard other details of the event, adopting a more flexible criterion for correctness.

Execution-based Evaluation In addition to exact and fuzzy matching, we incorporate execution-based evaluation methods to address more complicated scenarios. Specifically, we run a short code snippet to verify the correctness of results from the LLM agent since the expected results are not unique. Take the task "*Set up a meeting for Alice and Bob tomorrow when they are both free.*" as an example. This requires the LLM agent to check Alice's and Bob's schedules to pinpoint a mutually available time slot. To validate the result, we implement a code snippet that checks if the meeting is scheduled in both Alice's and Bob's calendars and ensures there are no overlapping commitments or time conflicts.

5 Experiments

With our proposed OFFICEBENCH, we evaluate the office automation capability of the proprietary LLMs, including Gemini-1.0 (Team et al., 2023), Gemini-1.5 (Reid et al., 2024), GPT-3.5 (Achiam et al., 2023), and GPT-4 (Achiam et al., 2023), and the open-weights LLMs, including Llama 3 (Meta, 2024) and Qwen 2 (Bai et al., 2023), as these models are among the highest-ranking LLMs available (Beeching et al., 2023). We also ask two computer science graduate students to perform these task and report the human performance (See Appendix F for error analysis for human annotators).

In OFFICEBENCH, the LLM agents need to interact with the multiple applications available in the environment, construct a feasible operation chain, and accomplish the task step by step. We adopt the end-to-end prompting approach to guide LLMs in

[1]https://chatgpt.com/

LLM Agents	Single App (93)	Two Apps (95)	Three Apps (112)	Overall (300)
Proprietary Models				
Gemni-1.0 Pro (Latest update: Feb 2024)	24.73	13.68	0.89	12.33
Gemni-1.5 Flash (Latest update: May 2024)	34.41	24.21	0.89	18.67
Gemni-1.5 Pro (Latest update: May 2024)	41.94	32.63	7.14	26.00
GPT-3.5 Turbo (0125)	8.60	7.45	0.89	5.35
GPT-4 Turbo (2024-04-09)	56.99	50.63	11.61	38.00
GPT-4 Omni (2024-05-13)	**64.52**	**60.00**	**21.43**	**47.00**
Open-weights Models				
Llama 3 (70B-Instruct)	**39.79**	**41.05**	5.36	**27.33**
Qwen 2 (72B-Instruct)	30.23	28.42	**8.04**	21.16
Human Performance	96.00	96.00	88.00	93.33

Table 2: **Pass rates (%) on agent automation tasks** from OFFICEBENCH for the proprietary models, Gemini-1.0, Gemini-1.5, GPT-3.5, GPT-4, and the open-sourced models, Llama 3 and Qwen 2. We divide the tasks into "Single/Two/Three App(s)", specifying the number of applications required by the tasks; we also report the overall performance; the number in the brackets denotes the number of tasks in each subcategory. **Bold** denotes the best performance among the proprietary or the open-weights models.

planning and executing workflows autonomously, without the need for manually selected demonstrations. In this way, we eliminate the biases introduced by the cherry-picking demonstrations and guarantee the reliability and robustness of the experimental results on OFFICEBENCH. We leverage our designed customized evaluation methods discussed in Section 4.3 for each test task to verify if the outcomes from the LLM agents pass. We use *pass rate*, $\frac{\#\text{Pass Examples}}{\#\text{All Examples}}$, as our final metrics.

5.1 Main Results

We demonstrate the experimental results of the LLM agents in Table 2. We present both the overall performance and fine-grained performance of the evaluated LLM agents across the subcategories of "Single/Two/Three App(s)". We separate the LLMs into two groups: proprietary models and open-weights models. Within each group, the best-performing model is highlighted in bold. Table 2 shows that GPT-4 Omni and Llama 3 lead their respective groups, achieving overall pass rates of 47.00% and 27.33% for proprietary and open-weights models, respectively. These decent results show the basic capability of current LLM agents in solving office automation tasks, while there is still a huge gap to the human performance. We also observe that the open-weight Llama 3 even surpasses the proprietary Gemini-1.5, underlining that open-weight models are not necessarily worse than the proprietary models.

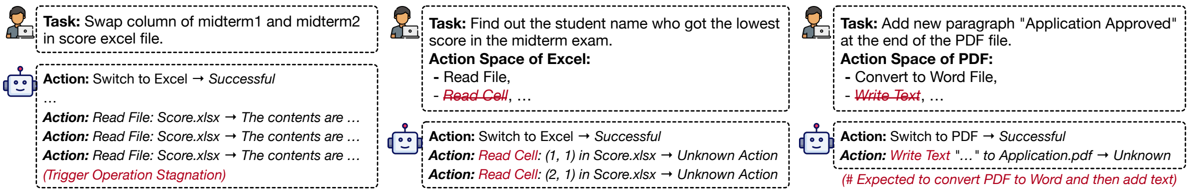
Specifically, we observe that performance diminishes greatly when tasks require interactions between multiple applications, underscoring the inherent complexities associated with more intricate tasks. The state-of-the-art LLM agent, GPT-4 Omni, can only achieve 21.43% in the subcategory of "Three Apps", indicating a dramatic performance drop compared with "Two Apps" and "Single App" subcategories. We attribute this tendency to the limited capability of LLMs in tackling complex workflows with multiple applications, including the data formats specific to each application and the planning with different applications. Refer to Section 5.3 for the detailed error analysis.

LLM Agents	Single	Multiple	Overall ↑	# Token ↓
GPT-4O (2024-05-13)				
- Use App Switch	**64.52**	**39.13**	**47.00**	**1439.82**
- List All Operations	63.44	35.75	44.33	2177.51
Llama 3 (70B-Instruct)				
- Use App Switch	**39.79**	21.74	**27.33**	**1181.28**
- List All Operations	29.03	**24.15**	25.57	1630.15

Table 3: **Evaluation results (%) of the ablation study** for application switching on OFFICEBENCH. We investigate the performance of GPT-4 Omni and Llama 3 when using the `switch_app` operation (*Use App Switch*) or listing all operations in the prompt (*List All Operations*) to manage the environment with multiple applications.

5.2 Ablation Study for Application Switching

We highlight the complex workflows with multiple applications in our proposed OFFICEBENCH which can well simulate the realistic office environment and investigate the planning and reasoning capabilities of LLMs in the complex workflow. In our designed framework, the LLM agents can switch between different applications by calling the `switch_app` operation and get access to the action space specific to the target application. We denote this method as *Use App Switch*.



(a) Stagnation at Redundant Operations **(b) Hallucinations in Action Space** **(c) Complex Planning across Applications**

Figure 3: **Typical failure cases of the LLM agents** when solving office automation tasks in OFFICEBENCH. We highlight the repeated redundant operations in (a), the hallucinated actions in (b), and the planning failure in (c). We omit the other contexts in the prompts and responses due to space limitation. The contents on the right side of the arrow "→" are the observation of the action.

In the ablation study, we follow the vanilla prompting method which lists all the operations regardless of the corresponding applications in the prompt. We denote this method as *List All Operations*. We investigate the performance of GPT-4 Omni and Llama 3 under these two settings as they are the top-performing proprietary and open-weights models in OFFICEBENCH, respectively.

We report the performance and also calculate the average number of tokens used per iteration in cases that terminates normally[2] in Table 3. We observe that the application switching mechanism outperforms its counterpart, enabling LLM agents to effectively manage multiple applications within complex workflows. This enhancement can be attributed to the more concise natural language instructions and the constrained action space in the prompts. The action space of next step is largely constrained to the operations of the current application via the application switching operation (refer to Section 3.2 for details).

5.3 Error Analysis

We further conduct error analysis on the outcomes from the LLM agents and summarize the typical failure cases in Figure 5.3.

Stagnation at Redundant Operations As illustrated in Figure 5.3 (a), although the activation of the `read_file` operation to examine the spreadsheet's contents is initially successful, the LLM agent persistently repeats this operation. This occurs despite the feedback provided from previous actions, leading to an operational stagnation.

Hallucinations in Action Prediction LLM agents are susceptible to hallucinating actions not pre-defined in the given action space. As illustrated in Figure 5.3 (b), we dynamically limit the action space to include only the operations pertinent to the currently selected application (see Section 3.2). However, under such a narrowed subset of the entire action space, we still frequently observe that LLM agents tend to hallucinate non-existent actions, resulting in non-executable commands. These malformed actions not only fail to achieve the expected outcomes but also lead to a significant API calling or local inference costs.

Complex Planning across Applications In addition to the hallucinations discussed earlier, another type of non-executable actions can occur when LLM agents are tasked with complex workflows involving multiple applications. As shown in Figure 5.3 (c), LLM agents are instructed to edit a PDF file. However, due to a lack of knowledge that editing a `PDF` file typically involves first converting it to a `Word` document, making the necessary edits, and then converting it back to `PDF`, the agents mistakenly attempt direct edits on the `PDF`. This step is beyond the pre-defined action space, thereby resulting in a malformed action error.

6 Conclusion

We propose OFFICEBENCH, one of the first office automation benchmarks for language agents. We simulate a realistic execution environment and extensively evaluate the capability of current powerful LLM agents in solving tasks across different applications. Our findings highlight the efficacy of application switching in managing operations from multiple applications, and identify the limitations of LLMs in tackling cross-application workflows. With OFFICEBENCH, we aim to to advance the development of more robust and effective language agents for comprehensive office automation.

Limitation

In this paper, we propose OFFICEBENCH as one of the first office automation benchmarks for lan-

[2]We exclude the cases that terminate due to *Operation Stagnation* or *Iteration Overflow*, which introduces meaningless wasted tokens.

guage agents. While the system comprehensively analyzes the capability of current LLMs in planning complex workflow involving multiple applications in office automation, we anticipate that a wider range of applications will further expand our benchmark's usage in more application scenarios. Additionally, we are exploring the potential of instruction tuning for language models specifically tailored to office automation tasks, aiming to boost their performance on OFFICEBENCH.

Ethical Statement

In our proposed OFFICEBENCH benchmark, we only incorporate synthesized data in the file systems and all names of individuals and companies are fictitious and generated by ChatGPT. Therefore, we do not anticipate any major ethical concerns.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Philippe Aghion, Céline Antonin, Simon Bunel, and Xavier Jaravel. 2023. The effects of automation on labor demand. a survey of the recent literature. *Robots and Ai. A new economic era*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. 2023. Open llm leaderboard. https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard.

Galal M Binmakhashen and Sabri A Mahmoud. 2019. Document layout analysis: a comprehensive survey. *ACM Computing Surveys (CSUR)*, 52(6):1–36.

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

Lei Cui, Yiheng Xu, Tengchao Lv, and Furu Wei. 2021. Document ai: Benchmarks, models and applications. *arXiv preprint arXiv:2111.08609*.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36.

Emilia Filippi, Mariasole Banno, and Sandro Trento. 2023. Automation technologies and their impact on employment: A review, synthesis and future research agenda. *Technological Forecasting and Social Change*, 191:122448.

Łukasz Garncarek, Rafał Powalski, Tomasz Stanisławek, Bartosz Topolski, Piotr Halama, Michał Turski, and Filip Graliński. 2021. Lambert: Layout-aware language modeling for information extraction. In *International Conference on Document Analysis and Recognition*, pages 532–547. Springer.

Teakgyu Hong, Donghyun Kim, Mingi Ji, Wonseok Hwang, Daehyun Nam, and Sungrae Park. 2022. Bros: A pre-trained language model focusing on text and layout for better key information extraction from documents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 10767–10775.

Debra Howcroft and Phil Taylor. 2023. Automation and the future of work: A social shaping of technology approach. *New Technology, Work and Employment*, 38(2):351–370.

Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 2022. Layoutlmv3: Pre-training for document ai with unified text and image masking. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 4083–4091.

Zheng Huang, Kai Chen, Jianhua He, Xiang Bai, Dimosthenis Karatzas, Shijian Lu, and CV Jawahar. 2019. Icdar2019 competition on scanned receipt ocr and information extraction. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1516–1520. IEEE.

Guillaume Jaume, Hazim Kemal Ekenel, and Jean-Philippe Thiran. 2019. Funsd: A dataset for form understanding in noisy scanned documents. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, volume 2, pages 1–6. IEEE.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. 2024. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. *arXiv preprint arXiv:2401.13649*.

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

Xing Han Lù, Zdeněk Kasner, and Siva Reddy. 2024. Weblinx: Real-world website navigation with multi-turn dialogue. *arXiv preprint arXiv:2402.05930*.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. 2024. Agentboard: An analytical evaluation board of multi-turn llm agents. *arXiv preprint arXiv:2401.13178*.

Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. 2021. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2200–2209.

Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date.

Seunghyun Park, Seung Shin, Bado Lee, Junyeop Lee, Jaeheung Surh, Minjoon Seo, and Hwalsuk Lee. 2019. Cord: a consolidated receipt dataset for post-ocr parsing. In *Workshop on Document Intelligence at NeurIPS 2019*.

Vincent Perot, Kai Kang, Florian Luisier, Guolong Su, Xiaoyu Sun, Ramya Sree Boppana, Zilong Wang, Jiaqi Mu, Hao Zhang, and Nan Hua. 2023. Lmdx: Language model-based document information extraction and localization. *arXiv preprint arXiv:2309.10952*.

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. Scienceworld: Is your agent smarter than a 5th grader? *arXiv preprint arXiv:2203.07540*.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. 2023a. Mint: Evaluating llms in multi-turn interaction with tools and language feedback. In *The Twelfth International Conference on Learning Representations*.

Zilong Wang, Yiheng Xu, Lei Cui, Jingbo Shang, and Furu Wei. 2021. Layoutreader: Pre-training of text and layout for reading order detection. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4735–4744.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, et al. 2023b. Chain-of-table: Evolving tables in the reasoning chain for table understanding. In *The Twelfth International Conference on Learning Representations*.

Zilong Wang, Yichao Zhou, Wei Wei, Chen-Yu Lee, and Sandeep Tata. 2023c. Vrdu: A benchmark for visually-rich document understanding. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5184–5193.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *arXiv preprint arXiv:2404.07972*.

Binfeng Xu, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu, and Dongkuan Xu. 2023. Rewoo: Decoupling reasoning from observations for efficient augmented language models. *arXiv preprint arXiv:2305.18323*.

Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, et al. 2021. Layoutlmv2: Multi-modal pre-training for visually-rich document understanding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2579–2591.

Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2020. Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1192–1200.

John Yang, Akshara Prabhakar, Karthik Narasimhan, and Shunyu Yao. 2024. Intercode: Standardizing and benchmarking interactive coding with execution feedback. *Advances in Neural Information Processing Systems*, 36.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

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.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. 2023. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*.

Appendix

A Comparison with Recent Benchmarks

As shown in Table 4, OfficeBench excels in cross-application scenarios, offering a diverse suite of precisely curated customized evaluation functions for each task. Additionally, it supports a larger action space and provides more extensible task annotation and environment creation capabilities.

B Applications and Operations

We list all the applications and their corresponding operations in Table 5. We simulate a realistic execution environment for evaluating LLM agents in office automation tasks.

C Observation Formats

We illustrate the observation formats of the representative operations in OfficeBench in Figure 4.

D Evaluation Methods

We provide more examples of evaluation methods used in OfficeBench in Table 6.

E OfficeBench Prompts

We provide prompt examples used in our experiments.

- Prompt for application switching in Figure 5.
- Prompt for planning next operation based on the trajectory in Figure 6.
- Prompt of *List All Operations* used in the ablation study in Figure 7.

F Error Analysis for Human Annotators

The errors by human annotators mostly come from the misunderstanding of the task description or negligence in operations. For example, when solving "*Bob was invited to party hold by Jane Doe. Please send an email from Jane to Bob to notify Bob, and make a poster welcome.jpg for Bob*", one annotator ignored the email sending requests and only created the poster. Another example is the task "*How many quarters did Bob win a scholarship? A scholarship is awarded only when a student's GPA exceeds 3.9.*", where one annotator miscounted the number of quarters.

Benchmarks	Office Automation	# Supported Actions	Planning	Cross App.	Extensible	Customized Task Eval.
Document AI Benchmarks						
FUNSD (Jaume et al., 2019)	✓	-	✗	✗	✓	✗
CORD (Park et al., 2019)	✓	-	✗	✗	✓	✗
SROIE (Huang et al., 2019)	✓	-	✗	✗	✓	✗
VRDU (Wang et al., 2023c)	✓	-	✗	✗	✓	✗
DocVQA (Mathew et al., 2021)	✓	-	✗	✗	✓	✗
Language Agent Benchmarks						
ALFWorld (Shridhar et al., 2020)	✗	9	✓	✗	✓	✗
WebShop (Yao et al., 2022)	✗	8	✓	✗	✓	✗
ScienceWorld (Wang et al., 2022)	✗	25	✓	✗	✓	✗
InterCode (Yang et al., 2024)	✗	1	✓	✗	✗	✗
Mind2Web (Deng et al., 2023)	✗	3	✓	✗	✓	✗
WebArena (Zhou et al., 2023)	✗	12	✓	✗	✓	✓
WebLINX (Lù et al., 2024)	✗	15	✓	✗	✓	✗
OSWorld (Xie et al., 2024)	✓	13	✓	✓	✗	✓
OfficeBench	✓	23	✓	✓	✓	✓

Table 4: **Comparison with recent benchmarks** in document AI and language agent evaluation. It highlights several key strengths of OfficeBench. OfficeBench excels in cross-application scenarios (**Cross-App.**), offering a diverse suite of precisely curated customized evaluation functions for each task (**# Customized Task Eval.**). Additionally, it supports a larger action space (**# Supp. Actions**) and provides more extensible task annotation and environment creation capabilities (**Extensible**).

Application	Operations	Arguments	Explanation
`System`	`switch_app` `submit`	`target_app` `None`	Switch to the target application Finish the operation and submit the results

`Word`	`create_new_file`
	`convert_to_pdf`
	`read_doc_file`
	`write_to_file`
		`new_file_path`
		`doc_file_path, pdf_file_path`
		`file_path`
		`file_path, contents`
			Create a new empty doc file
			Convert a given doc file into a pdf file
			Read the contents of a doc file
			Write text to the doc file

`Excel`	`create_new_file`
	`set_cell_content`
	`delete_cell_content`
	`read_excel_file`
	`convert_to_pdf`
		`new_file_path`
		`file_path, cell_index, content`
		`file_path, cell_index`
		`file_path`
		`excel_file_path, pdf_file_path`
			Create a new empty excel file
			Set a specified cell value in an excel file
			Delete a specified cell in an excel file
			Read the contents of an excel file
			Convert a given excel file into a pdf file

`PDF`	`convert_to_image`
	`convert_to_doc`
	`read_pdf_file`
		`pdf_file_path, image_file_path`
		`pdf_file_path, doc_file_path`
		`file_path`
			Convert a given pdf file into an image
			Convert a given pdf file into a doc file
			Read the contents of a pdf file with a PDFParser

`Calendar`	`create_event`
	`delete_event`
	`list_event`
		`username, event_info`
		`username, event_id`
		`username`
			Create a new calendar event to the specified user
			Delete a calendar event for the specified user
			List all the calendar events for the specified user

`Email`	`list_emails`
	`read_email`
	`send_email`
		`username`
		`username, email_id`
		`sender, receiver, email_contents`
			List all the emails for the specified user
			Read a specified email for the user
			Send an email from one user to another one

`OCR`	`recognize_text`	`image_file_path`	Use an OCR engine to recognize the text in an image
`ChatGPT`	`query_chatgpt`	`query`	Submit a query to ChatGPT
`Shell`	`run_command`	`shell_command`	Run a shell command

Table 5: **Applications and their corresponding operations** implemented in OfficeBench.

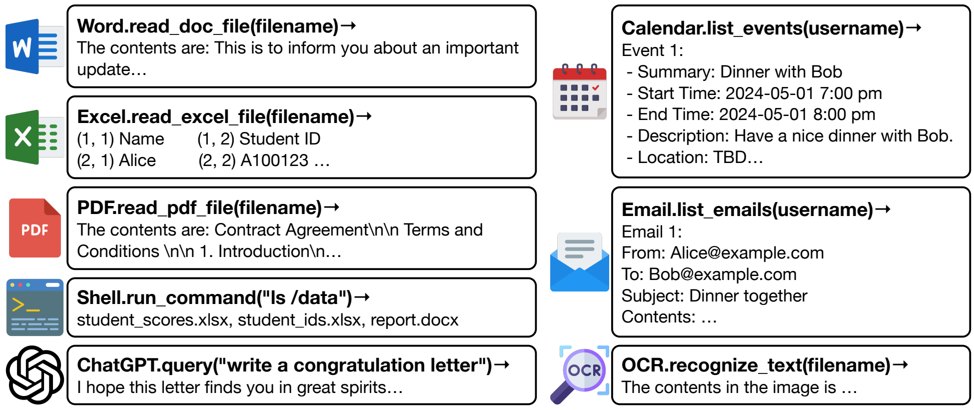


Figure 4: **Observation formats of representative operations** implemented in OfficeBench.

Type	Task Examples	Evaluation Functions
Exact	Change Carol's midterm1 score to 98 in score excel file	`excel_cell_value`(index=(21,2), content="98")
	Add a paragraph "Approved!" to the end of Application.docx.	`exact_match`(reference="application_w_para.docx")
Fuzzy	Add a meeting to Bob's calendar at 10:30 a.m to 11:00 a.m on 5/17/2024.	`contain_text`(text=["DTSTART:20240517T1030", "DTEND:20240517T1100", "meeting"])
	Check car trading records and only copy the information about my car into car_records.xlsx, skipping other cars.	`contain_text`(text="Civic") && `not_contain_text`(text="BMW")
	Summarize content from the notification image into one notification pdf file named notification.pdf.	`file_exist`(file="notification.pdf")
	Delete the result files from last month.	`file_not_exist`(file="April.docx")
Exec.	Find a common time for Bob and Tom for dinner at 5/1/2024.	`no_overlap`("Bob.ics") && `no_overlap`("Tom.ics") && `common_event`("Bob.ics", "Tom.ics", event="dinner")
	Randomly assign each student to class 1 to 5 in class member excel file.	`excel_cell_comparator`(index=(2,2), comparator="lambda x: x in ['1', '2', '3', '4', '5']")

Table 6: **Evaluation methods and task examples in OfficeBench**. We design three types of evaluation methods, Exact Matching, Fuzzy Matching, and Execution-based Evaluation to accurately validate the results of the LLM agents. We skip a few arguments in the evaluation functions due to space limitation.

```
========================================================== System ==========================================================
Today is 2020-05-01 (Friday). The current time is 10:00 AM. You are an AI assistant for user Bob.
You can help solve the task step by step.
You can interact with an operation system and use apps to solve the task.
You must follow the instructions and use the given json format to call APIs.
You can only generate one action at a time.
You can find files for your task in `/testbed/data`.
You have following apps installed in the system:
 - calendar: an app to manage daily events on calendar.
 - excel: an app to manipulate excel files, including reading, writing, etc.
 - ocr: an app to recognize text from images.
 - pdf: an app to manipulate pdf files, including format conversion and file reading.
 - shell: an app to run shell commands in the system.
 - word: an app to manipulate word files, including reading, writing, converting, etc.
 - email: an app to manage emails, such as sending and reading emails.
 - llm: an app to interact with the large language model to answer questions, generate text, etc.

========================================================== Prompt ==========================================================
##Task: Add a meeting to Bob's calendar at 5/17/2024 10:30 a.m to 11:00 a.m
##Available apps: ['calendar', 'excel', 'ocr', 'pdf', 'shell', 'word', 'email', 'llm']
##Instruction:
 - choose an app from the avaiblable apps: {'app': 'system', 'action': 'switch_app', 'target_app': [THE_APP_YOU_CHOOSE]}
##Command:

======================================================== Completion ========================================================
```json
{
  "app": "system",
  "action": "switch_app",
  "target_app": "calendar"
}
```
```

Figure 5: **Prompt for application switching** used in OFFICEBENCH

```
========================================================== System ==========================================================
Today is 2020-05-01 (Friday). The current time is 10:00 AM. You are an AI assistant for user Bob.
You can help solve the task step by step.
You can interact with an operation system and use apps to solve the task.
You must follow the instructions and use the given json format to call APIs.
You can only generate one action at a time.
You can find files for your task in `/testbed/data`.
You have following apps installed in the system:
 - calendar: an app to manage daily events on calendar.
 - excel: an app to manipulate excel files, including reading, writing, etc.
 - ocr: an app to recognize text from images.
 - pdf: an app to manipulate pdf files, including format conversion and file reading.
 - shell: an app to run shell commands in the system.
 - word: an app to manipulate word files, including reading, writing, converting, etc.
 - email: an app to manage emails, such as sending and reading emails.
 - llm: an app to interact with the large language model to answer questions, generate text, etc.

========================================================== Prompt ==========================================================
##Task: Add a meeting to Bob's calendar at 5/17/2024 10:30 a.m to 11:00 a.m
##History:
 - Step 0: {'app': 'system', 'action': 'switch_app', 'target_app': 'calendar'} -> [Successfully switched to app: calendar]
 - Step 1: {'app': 'calendar', 'action': 'create_event', 'user': 'Bob', 'summary': 'Meeting', 'time_start': '2024-05-17
 10:30:00', 'time_end': '2024-05-17 11:00:00'} -> [Successfully create a new event to Bob's calendar.]
##Current apps: calendar
##Instruction: Choose one action from the list as the next step.
 - create a new event to a user's calendar where the time format is '%Y-%m-%d %H:%M:%S':{'app': 'calendar', 'action': '
 create_event', 'user': [USER_NAME], 'summary': [EVENT_SUMMARY], 'time_start': [EVENT_START_TIME], 'time_end': [
 EVENT_END_TIME]}
 - delete an event from a user's calendar given the event summary:{'app': 'calendar', 'action': 'create_event', 'user': [
 USER_NAME], 'summary': [EVENT_SUMMARY]}
 - list all events from a user's calendar: {'app': 'calendar', 'action': 'list_events', 'username': [USER_NAME]}
 - switch to another app among ['excel', 'ocr', 'pdf', 'shell', 'word', 'email', 'llm']: {'app': 'system', 'action': '
 switch_app', 'target_app': [THE_APP_YOU_CHOOSE]}
 - finish the task with your answer as None if the task is not a question: {'app': 'system', 'action': 'finish_task', '
 answer': 'None'}
 - finish the task with your answer if the task is a question: {'app': 'system', 'action': 'finish_task', 'answer': [
 ANSWER]}
##Command:

======================================================== Completion ========================================================
{'app': 'system', 'action': 'finish_task', 'answer': 'None'}
```

Figure 6: **Prompt for planning next operation based on the trajectory** used in OFFICEBENCH

```
========================================================= System =========================================================
Today is 2020-05-01 (Friday). The current time is 10:00 AM. You are an AI assistant for user Bob.
You can help solve the task step by step.
You can interact with an operation system and use apps to solve the task.
You must follow the instructions and use the given json format to call APIs.
You can only generate one action at a time.
You can find files for your task in `/testbed/data`.
You have following apps installed in the system:
 - calendar: an app to manage daily events on calendar.
 - excel: an app to manipulate excel files, including reading, writing, etc.
 - ocr: an app to recognize text from images.
 - pdf: an app to manipulate pdf files, including format conversion and file reading.
 - shell: an app to run shell commands in the system.
 - word: an app to manipulate word files, including reading, writing, converting, etc.
 - email: an app to manage emails, such as sending and reading emails.
 - llm: an app to interact with the large language model to answer questions, generate text, etc.

========================================================= Prompt =========================================================
##Task: Add a meeting to Bob's calendar at 5/17/2024 10:30 a.m to 11:00 a.m
##History:
 - Step 0: {'app': 'calendar', 'action': 'create_event', 'user': 'Bob', 'summary': 'Meeting', 'time_start': '2024-05-17
 10:30:00', 'time_end': '2024-05-17 11:00:00'} -> [Successfully create a new event to Bob's calendar.]
##Instruction: Choose one action from the list as the next step.
 - create a new event to a user's calendar where the time format is '%Y-%m-%d %H:%M:%S':{'app': 'calendar', 'action': '
 create_event', 'user': [USER_NAME], 'summary': [EVENT_SUMMARY], 'time_start': [EVENT_START_TIME], 'time_end': [
 EVENT_END_TIME]}
 - delete an event from a user's calendar given the event summary:{'app': 'calendar', 'action': 'delete_event', 'user': [
 USER_NAME], 'summary': [EVENT_SUMMARY]}
 - list all events from a user's calendar: {'app': 'calendar', 'action': 'list_events', 'username': [USER_NAME]}
 - read the excel file to see the existing contents: {'app': 'excel', 'action': 'read_file', 'file_path': [
 THE_PATH_TO_THE_EXCEL_FILE]
 - write text to a cell in the excel file (index starts from 1): {'app': 'excel', 'action': 'set_cell', 'file_path': [
 THE_PATH_TO_THE_EXCEL_FILE], 'row_idx': [THE_ROW_INDEX], 'column_idx': [THE_COLUMN_INDEX], 'text': [THE_TEXT_TO_WRITE]}
 - delete text in a cell of the excel file (index starts from 1, delete means set empty): {'app': 'excel', 'action': '
 delete_cell', 'file_path': [THE_PATH_TO_THE_EXCEL_FILE], 'row_idx': [THE_ROW_INDEX], 'column_idx': [THE_COLUMN_INDEX]
 - create a new excel file: {'app': 'excel', 'action': 'create_new_file', 'file_path': [THE_PATH_TO_THE_NEW_EXCEL_FILE]}
 - convert an excel document to a pdf: {'app': 'excel', 'action': 'convert_to_pdf', 'excel_file_path': [
 THE_PATH_TO_THE_EXCEL_FILE], 'pdf_file_path': [THE_PATH_TO_THE_PDF_FILE]}
 - recognize the text from an image file: {'app': 'ocr', 'action': 'recognize_file', 'file_path': [
 THE_PATH_TO_THE_IMAGE_FILE]}
 - convert a pdf file to an image file: {'app': 'pdf', 'action': 'convert_to_image', 'pdf_file_path': [
 THE_PATH_TO_THE_PDF_FILE], 'image_file_path': [THE_PATH_TO_THE_IMAGE_FILE]}
 - convert a pdf file to a word file: {'app': 'pdf', 'action': 'convert_to_word', 'pdf_file_path': [
 THE_PATH_TO_THE_PDF_FILE], 'word_file_path': [THE_PATH_TO_THE_WORD_FILE]}
 - read a pdf file: {'app': 'pdf', 'action': 'read_file', 'pdf_file_path': [THE_PATH_TO_THE_PDF_FILE]}
 - Send an email to a recipient: {'app': 'email', 'action': 'send_email', 'sender': [SENDER], 'recipient': [RECIPIENT], '
 subject': [SUBJECT], 'content': [CONTENT]}
 - List emails for a given username: {'app': 'email', 'action': 'list_emails', 'username': [USER_NAME]}
 - Read a user's email by the given Email ID: {'app': 'email', 'action': 'read_email', 'username': [USERNAME], 'email_id':
  [EMAIL_ID]}
 - run a shell command: {'app': 'shell', 'action': 'command', 'command': [THE_COMMAND_YOU_WISH_TO_RUN]}
 - convert a word document to a pdf: {'app': 'word', 'action': 'convert_to_pdf', 'word_file_path': [
 THE_PATH_TO_THE_WORD_FILE], 'pdf_file_path': [THE_PATH_TO_THE_PDF_FILE]}
 - create a new word file: {'app': 'word', 'action': 'create_new_file', 'file_path': [THE_PATH_TO_THE_NEW_WORD_FILE]}
 - read the content of the word file: {'app': 'word', 'action': 'read_file', 'file_path': [THE_PATH_TO_THE_WORD_FILE]
 - write text to a word file: {'app': 'word', 'action': 'write_to_file', 'file_path': [THE_PATH_TO_THE_WORD_FILE], '
 contents': [THE_CONTENTS_YOU_WISH_TO_WRITE]}
 - Query an LLM model for an answer to a given prompt: {'app': 'llm', 'action': 'complete_text', 'prompt': [PROMPT]}
 - finish the task with your answer as None if the task is not a question: {'app': 'system', 'action': 'finish_task', '
 answer': 'None'}
 - finish the task with your answer if the task is a question: {'app': 'system', 'action': 'finish_task', 'answer': [
 ANSWER]}
##Command:

======================================================= Completion =======================================================
```json
{
  "app": "system",
  "action": "finish_task",
  "answer": "None"
}
```
```

Figure 7: **Prompt of *List All Operations* used in the ablation study**