A study on techniques for improving LLMs as general purpose agents

Harsh Raj,[1] **Allan Niemerg,**[1]

[1] Yield Inc

Introduction

AgentBench is a multi-dimensional benchmark that includes eight distinct environments for evaluating the reasoning and decision-making abilities of Language Model-as-Agent (LLM-as-Agent) in a multi-turn, open-ended generation context. This benchmark focuses on the practical evaluation of LLMs through Chain-of-Thought (CoT) prompting, incorporating code-grounded, game-grounded, and web-grounded scenarios. It identifies promising directions for LLM applications in autonomous mission completion and ensures a variety of tasks to prevent overperformance by task-specific models. The agent-environment interactions in language models can be broadly categorized as follows:

1. **Interaction Format**: This category is based on the format of agent-environment interaction, further divided into:
   (a) **Language Commands**: Environments where interaction is through language commands, such as `move straight`, `turn right` in Alfworld. These environments are typically natural language adaptations of embodied AI environments like MatterPort3D, Habitat, Chess, Solitaire, etc. We experimented with creating language-based environments using GPT-4 for such interactions, including household[1] and city[2] environments. The conversations between the agent and the synthetic language-based environment are documented here[3]. Notably, these datasets show high redundancy in the agent's actions, with the agent often using a single command `move_forward` for movement, indicating the simplicity of the environment construction.
   (b) **Function Calling**: Environments where interactions are through calling from a set of predefined functions, with arguments supplied by the environment. Numerous datasets for such interactions are openly available, such as the Function Calling Dataset Collection[4] on HuggingFace.
   (c) **Selenium/Playwright/Bash/Python/SQL**: Environments where the interaction format involves some programming language. Although the data may not be well-structured, a significant amount is available.
2. **Environment State**: This category is based on the state of the environment, further divided into:
   (a) **Static Environment**: Environments that do not react to the agent's actions but follow a set of predefined rules, such as Alfworld, Webshop, OS-Interaction, DBBench, Solitaire.
   (b) **Dynamic Environment**: Environments that react to the agent's interactions by taking actions in return, functioning like a two-player game. Examples include Chess, Card Games, Go.

The primary **goal** was to explore the improvement of models on AgentBench by creating synthetic data. This approach is based on the hypothesis that simulating environments could be a cost-effective way to scale data, compared to actual interactions within these environments. Collecting agent data, especially in multi-turn conversations, is increasingly challenging as it cannot be directly found on the web and is in high demand following the generative AI revolution.

Related Works

Discussions on related works like Genie: Generative Interactive Environments, and others that have influenced the development and evaluation of language models in interactive settings.

Overview of AgentBench

AgentBench is designed to challenge language models through a variety of agent-environment interaction formats, including language commands and function calling, among others. Its objectives are to evaluate and enhance the interaction capabilities of language models in dynamic and complex environments.

Experimentation and Results

For our experiments, we consistently used Llama-2-7B as the base model, except where explicitly stated otherwise.

[1] environment creation prompt 1
[2] environment creation prompt 2
[3] agent-environment conversation 1
[4] dataset 1

The models were finetuned using the LoRA technique, leveraging the axolotl repository, with the aim of enhancing the base model's performance on AgentBench tasks. Table 1 displays the performance metrics of some models before finetuning.

Different Dataset Combinations

Through tuning the model with a variety of diverse datasets, we observed that the specificity of a task inversely correlates with the benefits derived from general datasets. In essence, as tasks become more specialized, the utility of general datasets diminishes, necessitating datasets that are closely aligned with the specific task at hand. This trend became apparent as we transitioned from utilizing datasets with general capabilities (such as teknium/OpenHermes-2.5, LDJnr/Puffin) to those more closely related to AgentBench (like THUDM/AgentInstruct), resulting in improved model scores. Despite experimenting with numerous dataset combinations, none matched the performance of AgentLM-7B, which is trained on the AgentInstruct dataset. The AgentInstruct dataset is specifically designed for AgentBench by running GPT-4 against environments on tasks that closely mirror the AgentBench test split. Our training of the LLama-2 7B-chat model on various popular but general datasets, as detailed in Table **??**, generally led to a degradation in chat model performance on AgentBench tasks. It is evident from the table that none of the models enhanced the base model's performance but instead resulted in its deterioration.

Our results further demonstrate the high sensitivity of finetuning to the combination of datasets. As illustrated in Table 3, the impact of incorporating various datasets on model performance in tasks such as DBBench varies significantly. For instance, combining the model with the alfworld dataset detrimentally affects performance, whereas integration with OS-Interaction data yields a positive enhancement. In the context of multi-turn agent conversations, datasets focused on multi-turn dialogues enhance performance more effectively than those based on single-turn conversations, even if the latter has more potential in teaching reasoning or agentive capabilities. Notably, despite the kaist-ai/CoT-Collection dataset's greater similarity to the evaluation suite, the LDJnr/Puffin dataset demonstrates superior performance upon evaluation because of being multi-turn in structure. Furthermore, it is intriguing that the medalpaca/medical_meadow_wikidoc and wiki_movies outperform other models, despite their domains being vastly different from the evaluation suite. It is crucial to acknowledge that each model diminishes the base model's performance, suggesting that direct comparison solely based on results may be misleading. This outcome suggests that enhancing model performance on a benchmark through finetuning requires data of a similar nature. Such a result concludes more towards the fact that to improve a model on a benchmark using finetuning specifically, one needs a similar kind of data. This observation implies the existence of a similarity threshold between the evaluation suite and the dataset, beyond which finetuning might positively influence performance.

How to obtain task specific data?

Through extensive experimentation with various datasets, we have concluded that fine-tuning is highly sensitive to the combination of data used. To enhance model performance on a benchmark that demands specificity, it is crucial to emulate the data originating from the same distribution as the benchmark. For AgentBench, which requires proficiency in instruction following and multi-step reasoning, merely presenting data that encompasses these aspects is insufficient for accuracy improvement. It is imperative to expose the model to samples that mirror the benchmark's characteristics, thereby improving its reasoning and instruction-following skills within that domain. The feasible method for acquiring such data is through meticulous curation. The subsequent sections will delve into the methodologies we explored for generating synthetic data. Synthetic data generation involves utilizing a high-performance teacher model, such as GPT-4, to create data samples (input-output pairs) from which a student model (e.g., Llama-2-7B-chat) can learn via techniques like fine-tuning. These data samples are crafted by imposing certain constraints on the teacher model to produce outputs closely aligned with a specific distribution. A prevalent strategy for constraining the teacher model is to prime it with original data samples through few-shot learning.

Naive Prompting and Fine-Tuning

1. **Instructing the Environment** Initially, we attempted to instruct the GPT-4 model by naively prompting it to simulate an environment. These prompts began with phrases like "Pretend you are a .." and were both lengthy and detailed to sketch out the environment's overview. The issue with this naive prompting approach is that the model lacks grounding, leading to the environment affirmatively responding to even incorrect queries. For instance, the environment in DBBench reacts to the SQL query `DELETE FROM orders WHERE status = 'pending' AND order_date < '2021-01-01';` with `Query OK, 2 rows affected (0.14 sec)`, despite the query being incorrect because the operators < and > cannot be used with strings. It's crucial to highlight that a common failure mode is the model's production of outputs that align with the environment but are incorrect responses to the agent, such as indicating that a malformed query executed successfully.

   Additionally, such an environment lacks consistency with its previous outputs, and in this type of freeform generation, the model tends to repeat outputs similar to what it has most often encountered during training. For example, in tasks involving OS interactions, when the agent generates a bash command like `find example_directory -name "*.txt" -exec wc -c ;`, the model frequently responds with `example_directory not found`. This suggests that GPT-4 might be biased towards such errors, possibly because it encountered them frequently dur-

Model	Description	OS	DB	KG	M2W	LTP	HH
ft:gpt-3.5-turbo-0613;yields-inc::8LIEaIUZ	GPT-3.5-turbo finetuned on AgentInstruct data	53.8	54.9			55	
GPT-4-0613		50	33.3	49.5	26.1	0.26	75
THUDM/agentlm-7b		26.9	33.3	26.7	11.3	0.01	15
YieldInc/agentlm_repo_jan21	our effort to replicate agentlm-7b	33.3	20.4	10			10
mistraIai/Mistral-7B-Instruct-v0.2		26.9	5				0
meta-llama/Llama-2-7b		26.9	6.6	0	13	0.05	0

Table 1: Scores of some top performing models on AgentBench.

Model	Dataset	OS	DB	KG	M2W	LTP	HH
GPT-4-0613		50	33.3	49.5	26.1	0.26	75
THUDM/agentlm-7b		26.9	33.3	26.7	11.3	0.01	15
YieldInc/agentlm_repo_jan21		33.3	20.4	10			10

Table 2: Scores comparison of model finetuned using synthetic data generated via Naive Prompting

ing training. This leads to another failure mode: The environments default to common errors, even if these are not consistent with the specified environment. Such problematic data samples lead to poor performance of the model, as highlighted in Table 2.

2. **Few Shot Prompting:** This method involves providing the model with examples of agent-environment conversations as context, prompting it to generate a similar task and the corresponding conversation. This approach, a form of free-form generation, occasionally produces accurate agent responses. However, the environment side of the conversation may contain errors or exhibit biases towards certain outputs, as previously mentioned.

   Additionally, it is noteworthy that in this setting, the model frequently employs the `find` command when solving tasks in the OS-Interaction split of AgentBench. This tendency may stem from biases introduced by the training samples it has encountered or the biasses induced by the few-sot examples. During this type of free-form generation, the model completes the conversation in a way that reflects its biases, as there is no ground truth for the responses. Consequently, the main issue with this approach is the lack of data diversity, leading to biased or repetitive outputs. The results presented in Table 3 illustrate a similar trend of poor performance, attributable to data samples that skew the model's behavior. This underscores the model's sensitivity to unfavorable data samples created via this approach, highlighting the significant impact a single poor data point can have on the finetuned model.
3. **Finetuning** Finetuning GPT-3.5-turbo to function as an environment can lead to a stricter model, addressing issues arising from incorrect response formats. An improper format can severely disrupt the agent-environment interaction, as both, being chat models, are inclined to follow the conversation flow set by the erroneous format. An incorrect format not only signifies that the environment responds with an unexpected syntax but also indicates that the environment tends to utilize its general chat model capabilities instead of acting within a constrained setting.

   Table needed

   We used the OpenAI finetuning API to refine the model, aiming for a more resilient environment model. Although the model indeed becomes more robust, it also develops a significant bias towards the most common outputs seen during training. For instance, it frequently generates empty `'[]'` outputs when required to select specific column values from a table. As shown in the agent-environment conversation below, the environment behaves as though the database is devoid of values, yielding no data points. Language systems are inherently susceptible to biases and necessitate a diverse and balanced dataset for improvement. Despite ongoing research in this area, we observed that the primary challenge lies not in adhering to specific output formats but in the lack of a reference for the environment model to act accordingly or a basis for grounding their responses.

Intervening in the trajectories

Our manual review of trained models highlights the numerous subtle distinctions a larger model grasps that a smaller model might overlook, and the significant impact these can have on performance. Small errors can accumulate, derailing the model from the correct path. This underscores the advantages of employing larger models (for superior language modeling) and incorporating more extensive data sets. And yet, small datasets like AgentBench do seem to have an impact, so there must be ways to identify common issues and correct them. One innovative approach for generating data could involve an online variant of our synthetic environments, with interventions designed to correct the smaller model's trajectories. The strategy is to adhere as closely as possible to Llama-2's existing answer distribution, interven-

Model	Dataset Description	DB
GPT-4-0613		33.3
meta-llama/llama -2-7b-chat-hf	Base Model	3.3
agentinstruct-sh	Complete AgentInstruct and mixed	35
aregpt-lora	ShareGPT in the ratio 2:8	
YieldInc/agentinstruct_all_orig_and_db_syn-5	Complete AgentInstruct + DB 446 new tasks and conversations generated by the few-shot approach.	6.6

Table 3: Scores comparison of model finetuned using synthetic data generated via Few Shot Prompting

ing only where errors occur. This could generate training examples that are more impactful when used for training, as opposed to merely prompting Llama-2 to adjust its output style, akin to GPT-4's approach.

1. **GPT-4 Intervention Have to check this section** Have GPT-4 play the environment and LLama2 play the agent. Have Llama2 work through the task. GPT-4 then identifies where LLama-2 went wrong and takes over to try to correct the problem. We did by having GPT-4 step in at the place where Llama2 makes a mistake and then operate from there.

   What if we tried this with just synthetic OS interactions of DBBench interactions? We take the synthetic environments creator and replace GPT-4 with LLama2 (accessed via FastChat). We then identify where Llama2 goes off the rails using GPT-4 (compared with the GPT-4 solution). We then re-simulate from that point using GPT-4. However, since that may result in a lot of re-simulations from the beginning, maybe it is better to try to coax the correct performance out of the model. Maybe we have GPT-4 identify where Llama2 went wrong and then write instructions to help it the next time (without "giving away" the answer). Then re-run. If it gets the answer, create a datapoint using the chat plus the original prompt.

2. **Human Intervention** We manually annotated 100 examples from DBBench to ensure correct output formats. We introduced prompts at each step of the environment side of the conversation to ensure strict instruction following to direct the agent towards the desired output. The data thus created was used to finetune Llama-2-7B-chat. The table 5 demonstrates that models trained on the manually corrected Llama-2 trajectories and those trained on the original DBBench gold data exhibit similar performance. Both models incorporate a mix of 100 DBBench dataset examples with an 80% ShareGPT composition. The accuracy calculated on the DBBench dataset stands at 6.6% for both models, while Llama-2, without fine-tuning, also achieves a 6.6% score. This indicates neither improvement nor deterioration.

   One notable aspect to recall here is that fine-tuning is highly sensitive to erroneous examples. Had the manually annotated Llama-2 data not matched the quality of the original 100 DBBench samples, the accuracy would have undoubtedly decreased. The sensitivity of fine-tuning to erroneous examples is further discussed in section **add the Key takeaways section here**, emphasizing that while generating data with GPT-4 isn't sufficient, filtering out gold samples from the generated data completes the synthesis process, thereby creating a dataset capable of significantly improving the model. Achieving better accuracy for both models could have been possible if all AgentInstruct data, along with 80% ShareGPT, had been combined with all the annotated DBBench samples. However, it is presumed that both models would have achieved the same score. Due to lack of human labellers we were not able to conduct this experiment.

Grounding the Environment

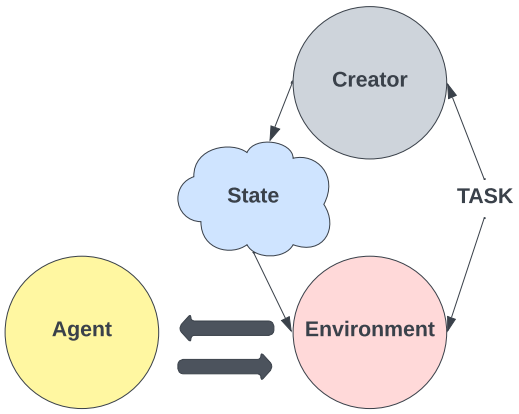
The objective of environment grounding is to prevent the model from generating unfounded details about the environment by establishing a well-defined environment beforehand. To minimize hallucinations where GPT-4 behaves as the environment, we initialize the environment before the agent begins interacting. Initialization involves creating the environment through structured environment creation processes. For instance, in an OS-Interaction task, we first instruct GPT-4 to generate codes to create the necessary files, directories, etc., and providing these codes to the instance of GPT-4 acting as the environment. This provides GPT-4 with context about the state of the environment, allowing it to base its responses on this predefined setup. Although this technique reduces hallucinations, it still necessitates further filtering. The prompts for grounding the environment require high precision, and we employ the synchaev library to observe and augment the agent-environment interaction. For every task in AgentBench, we apply a different technique for grounding, but the essence remains consistent: to initialize the environment prior to agent interaction. We explore environment initialization in two ways: firstly, by step-by-step prompting to create items in the environment, organize the environment, identify available actions, and then define tasks. Secondly, the environment side of the conversation uses chain-of-thought prompting based on the environment and the agent's actions. However, this method does not yield

Model	Dataset Description	DB
GPT-4-0613		33.3
meta-llama/llama-2-7b-chat-hf	Base Model	3.3
agentinstruct-sharegpt-lora	Complete AgentInstruct and mixed ShareGPT in the ratio 2:8	35
agentinstruct_db_gold_100-sharegpt_400	AgentInstruct DB 100 gold samples + 400 ShareGPT data	6.6
YieldInc/agentinstruct_db_llama2_manual_reviewed_100-sharegpt_400	AgentInstruct DB 100 llama2 produced and manually reviewed samples + 400 ShareGPT data	6.6

Table 4: Scores comparison of model finetuned using synthetic data generated via Human Intervention

interactions that closely resemble real ones, as it involves synthetically creating a wide range of variables, from tasks to the environment and agent side of the conversation. After analysing the data produced through this setup we realised that in synthetic data creation pipeline there needs to be a significant portion of real data which can constrain the synthesis in the right direction. We have tuned some models using these method. A more detailed explanation is provided below.

1. **Generating synthetic tasks and environment**
   (a) **Environment creation and Task creation** An overview of the environment is given and with CoT style prompt GPT-4 is asked to generate a task description or structure and subsequently generate some tests for the environment which are considered as tasks.
   (b) **Action format/ defining the available actions** Given the environment description and the tasks created in the previous step, GPT-4 is prompted to design an action format. For example, action format could be a set of functions the agent can use to interact with the environment.
   (c) **Environment simulation** The environment is simulated using an instance of GPT-4. The model is given the environment overview, task from step (a) and the interaction schema produced from step (c) and the history of the agent-environment conversation. The model is prompted to produce the environment side output. To make the environment side conversation to be more realistic a Thought-Environment-Critic loop is used where the 'Thought' is to think before responding to the agent's action and the Environment is the environment output. The critic further criticises if the environment output has any flaws refraining it be seen as real.
2. **Generating synthetic environment with AgentInstruct tasks** The previous method did not create interactions close to the real ones because there were a lot of variables to create synthetically ranging from tasks to the environment side of the conversation. So next we decided to reduce the number of variables to create, and now took the already present tasks from AgentInstruct and only simulated the environment synthetically. Now the environment was initialised based on the tasks. We used an instance of GPT-4 to act like an initializer. Initialization in this case means to set the state of the environment such that the task can be solved in some way. It is still difficult to control the GPT-4 model from acting differently than a real environment when the agent does not produce its output in the right format or thinks the environment is a living entity and starts chatting with it. For instance, at times the agent asks questions to the environment which is out of scope of the environment and to answer such questions the environment loses its hardness to produce outputs very different from what a real environment would have produced. But as long as the agent outputs are in the right format and as long as the agent interacts by keeping in mind the limitations of the environment things go fine. There have been instances where the final answer is not in the right format and which fails the stopping criteria for the conversation. We filtered such samples where the environment flow loses its hardness.



Filtering is done by observing the data and pointing out common problems in the samples. The filtered sam-

ples are replaced by its AgentInstruct counterpart. We use a streamlit based application to view the agent-environment conversations. Now before filtering we were getting quite unexpected and poor results but after filtering we were able to come closer to the model tuned on real AgentInstruct. It is to note that the synthetic data created has some errors which are the errors because of the inaccurately of the GPT-4 agent which isn't completely perfect so we didn't expect our model's results to exactly match the model tuned on real AgentInstruct but would be dependent on the percentage of correctness of GPT-4. For example if on a task GPT-4 is correct 60% of the time then the synthetic data should achieve 60% of the accuracy of the model tuned on real AgentInstruct data and that is what we were seeing. Our goal isn't to optimise the GPT-4 but to set up a pipeline for synthetic agent-environment interaction where the only factor to improve will be to improve the accuracy of the agent and that's how the model tuned on synthetic data will improve. Below are specifics about how we grounded the environments of different tasks in AgentBench.

DBBench Below is the steps through which the agent-environment interaction takes place in such a setting:

(a) **Initialization** Based on the tasks which are drawn from AgentInstruct dataset GPT-4 model is prompted to generate a database to perform the task in. Specifically, the database is generated as a list of tables which are necessary for the task to be completed. The creation of the database is made through one shot prompting technique giving examples of a similar database (tables) constructed based on the task. The "Generation Prompt" is the prompt we used for generating tables.

(b) **Interaction** The tables created are given to an instance of GPT-4 as context with instructions to pretend as a real database and use the table to respond to an agent's action. Again a one-shot example is given to stabilise the environment responses. The agent is set up in a similar way as in the original AgentBench paper with the prompts exactly the same. We used "Environment Prompt" for simulating the GPT-4 environment.

(c) **Filtering** Finetuning the base model directly on the synthetic data generated through this interaction does not result in satisfactory results. Upon further going through this data we found that the format of the data doesn't match the real ones. Upon filtering the data by replacing the samples with the original AgentInstruct samples we were able to improve our results. Below are the ways on which filtering was done:

i. **Action Format** The agent instruction prompts the model to use 3 different kinds of actions: "Operation", "Answer". If the agent conversations don't have these actions in the correct syntax we filter the sample out. The environment expects the final answer to be produced after "Final Answer: " which is also used as a signal to stop the agent-environment conversation. If the sample misses the final answer format we filter it out.

ii. **Match output format** The environment expects the final answer to be produced after "Final Answer: " which is also used as a signal to stop the agent-environment conversation. If the sample misses the final answer format we filter it out.

iii. **Standardise the final answer for database update commands** The agent is instructed to produce anything in answer fields when the query is about updating the database. But the AgentInstruct data instead of anything fills the answer field with "none" to make the outputs consistent. We also edited the data samples similarly.

OS Interaction Below is the steps through which the agent-environment interaction takes place in such a setting:

(a) **Initialization** Before interacting with the environment we use an instance of GPT-4 to generate a set of bash commands which initialises the environment to a necessary state. For example, if the task is to find the number of files with extension .txt in the home directory, the initialization command should create some dummy files with .txt extensions in the home directory. We use "Generation Prompt" for generating the bash commands.

(b) **Interaction** To make the environment GPT-4 aware of the state of the OS environment we fill the generated bash commands in its context. Using the "Environment Prompt" the model is instructed to respond to the agent's query based on the initialised state.

(c) **Filtering** Finetuning the base model directly on the synthetic data generated through this interaction does not result in satisfactory results. Upon further going through this data we found that the format of the data doesn't match the real ones. Upon filtering the data and replacing them with the original AgentInstruct samples we were able to improve our results. Below are the ways on which filtering was done:

i. **Action Format** The agent model is instructed to produce either of the actions: `bash`, "finish" or "answer". We filter the samples which didn't consist of the specified actions. Surprisingly many of the samples were filtered based on this criteria. So we just resynthesized the data with the resampling he output till the agent output contains the action in the right format. The resampling limit was set to 5.

ii. **Match output format** To constrain the environment outputs we used a different style in conversation. For example, the terminal output from the environment GPT-4 was output like `Output [terminal_response]` to enforce code outputs only. We edited the data samples with the format as in the AgentInstruct data and the evaluation suite.

iii. **Add terminal signatures** One interesting result we found is that the AgentInstruct data and the real OS environment produces a signature after its output. A signature something like

Model	Dataset Description	DB
GPT-4-0613	Base Model	33.3
meta-llama/llama-2-7b-chat-hf	Base Model	3.3
agentinstruct-sh	Complete AgentInstruct and mixed ShareGPT in the ratio 2:8	35
YieldInc/agentinstruct_db_svm_v3–sharegpt	Complete AgentInstruct with DB split replaced by synthetic DB (without filtering) and ShareGPT with a mixing ratio of 2:8.	3.3
YieldInc/agentinstruct_db_svm_v4–filtered_v1–sha–recegpt	Complete AgentInstruct with DB split replaced by synthetic DB (without a, b filtering) and ShareGPT with a mixing ratio of 2:8.	6.6
YieldInc/agentinstruct_db_svm_v4–filtered_v2–sha–recegpt	Complete AgentInstruct with DB split replaced by synthetic DB (without a, b, c filtering) and ShareGPT with a mixing ratio of 2:8.	10.2

Table 5: **DBBench** evaluation scores comparison of model finetuned using synthetic agent-environment conversation data generated via the grounding mechanism

```
1                    \r\n\x1b]0;
                         root@4d37337ef2d6
                         : /\
                         x07root@4d37337ef2d6
                         :/# \x1b[K
```

We tuned the model with and without these signatures in our synthetic data and found that the performance significantly improves when we remove such noise.

Alfworld Below is the steps through which the agent-environment interaction takes place in such a setting:

(a) **Initialization** For initialising a household environment with a task an agent has to perform in a household environment. We instruct an instance of GPT-4 to initialise the state of the house before the agent starts interacting. The initialization consists of mapping each item mentioned in the task to the items present in its vicinity. For example, when the agent opens drawer 1 it sees pen2; so pen2 should be mapped to drawer 1. The mapping is created as a JSON format and the model is given few-shot examples to generate such consistent maps. "Grounding Prompt" gives the complete prompt.

(b) **Interaction** The created JSON map is given as context to the environment instance of GPT-4. The environment then responds to the agent's action in accordance to the state of the environment arranged in a JSON. If the agent interacts with the environment by changing the location of an item the environment subsequently modifies the JSON map reflecting the change. The complete prompt with a few-shot example is given in "Environment Prompt".

KnowledgeGraph Below is the steps through which the agent-environment interaction takes place in such a setting:

(a) **Initialization** To initialise a freebase knowledge base we used an instance of GPT-4 prompting it with the question and a given set of entities to think of a possible solution in 1, 2 or 3 queries to the knowledgebase. While thinking about the possible solution the model is prompted to think of the relations which need to be associated with a given entity. Then the model is asked the generate the relations to the given entities including the relations which helps it answer the question. The complete set of prompts to generate relations for the entities is in "Grounding Prompt".

(b) **Interaction** Upon constructing the entities with their relations the environment model is given the entity with the relations. Via the few-shot prompting the model taught to use entities mapped with their relations for answering the queries agent puts on. "Environment Prompt" gives a full view of the prompt used to harden the environment through the created dummy knowledge graph.

(c) **Filtering** Finetuning the base model directly on the synthetic data generated through this interaction does not result in satisfactory results. Upon further going through this data we found that the format of the data doesn't match the real ones. Upon filtering the data by replacing the samples with the original AgentInstruct samples we were able to improve our results. Below are the ways on which filtering was done:

Model	Dataset Description	DB
GPT-4-0613		50
meta-llama/llama-2-7b-chat-hf	Base Model	10.4
agentinstruct-sharegpt-lora	Complete AgentInstruct and mixed ShareGPT in the ratio 2:8	14.5
YieldInc/agentinstruct_os_syn-sharegpt	Complete AgentInstruct with OS split replaced by synthetic OS (without filtering) and ShareGPT with a mixing ratio of 2:8. Complete AgentInstruct with OS split replaced by synthetic OS (without a, b, c filtering) and ShareGPT with a mixing ratio of 2:8.	10.4

Table 6: **OS-Interaction** evaluation scores comparison of model finetuned using synthetic agent-environment conversation data generated via the grounding mechanism

Model	Dataset Description	DB
GPT-4-0613		75
meta-llama/llama-2-7b-chat-hf	Base Model	0
YieldInc/metallama2_agentinstruct-axolotl	Complete AgentInstruct and mixed ShareGPT in the ratio 2:8	30
	Complete AgentInstruct with HH split replaced by synthetic HH (without filtering) and ShareGPT with a mixing ratio of 2:8.	
	Complete AgentInstruct with HH split replaced by synthetic HH (without a, b, c filtering) and ShareGPT with a mixing ratio of 2:8.	

Table 7: **Alfworld** evaluation scores comparison of model finetuned using synthetic agent-environment conversation data generated via the grounding mechanism

i. **Final Answer format** In this case the agent is instructed to produce the answer as "Final Answer #". The final answer format for this task does not impart strictness because it is close to freeform generation. Instead a better final answer format like "Final Answer:" should have been used. Anyways we strictly filtered out all the data samples which did not obey this syntax.

Mind2Web Below is the steps through which the agent-environment interaction takes place in such a setting:

(a) **Initialization** Initialising the Mind2Web environment was relatively simpler with just prompting a GPT-4 instance with the agent's input namely the question, the options and the previous actions to generate a webpage with HTML codes. To generate the HTML code to be aligned with the question we used few-shot prompting. The prompts used to generate the HTML codes are in "Grounding Prompt".

(b) **Interaction** The environment GPT-4 model is given the generated HTML codes along with the agent's input containing the question, the options and the previous actions. The model is few-shot prompted to produce responses aligned with the style of the original samples. The prefix-prompt for the environment can be viewed in "Environment Prompt".

(c) **Filtering** Finetuning the base model directly on the synthetic data generated through this interaction does not result in satisfactory results. Upon further going through this data we found that the format of the data doesn't match the real ones. Upon filtering the data by replacing the samples with the original AgentInstruct samples we were able to improve our results. Below are the ways on which filtering was done:

i. **Resampling to include Thought and Answer pairs.** The agent GPT-4 even though prompted via few-shots to produce answer after "Thought-Action" pairs interestingly didn't obey every time in this setting. So we resampled the agent's output until the model produces "Thought-Action" pairs in the right format. The maximum number of resampling was set to 5.

Key Takeaways

1. **Finetuning is Highly Sensitive to Data Combinations** We conducted an ablation study to see the effect of finetuning on data combinations. Observing the Table 10 reveals the interdependence of tasks in AgentBench, where a model's performance on DBBench is influenced by the inclusion of datasets from other tasks in the finetunes. Specifically, tuning a model solely with DBBench data does not achieve the highest score (by Llama-2-chat finetune) of 26.6%, which is attained only when the model is trained on a combination of tasks. This highlights the significant impact of data mix on the efficacy of finetuning. It also underscores the challenge of identifying which specific data combinations enhance model performance.

   The interconnected performance between tasks such as DB and OS Interaction is evident in Table 10, showcasing the similarities in structure of conversation. For instance, both tasks require the agent to choose from a predefined set of actions during a conversation, with action types like `Operation` and `Answer` in OS Interaction closely corresponding to `bash` and `answer` actions in DBBench. This shared structure provides a rationale for the improved task performance due to certain data combinations.

   In contrast, the inclusion of Alfworld data negatively impacts DBBench scores. This can be attributed to Alfworld's open-ended interaction style, where actions are not as strictly defined compared to the other tasks. For example, an agent in Alfworld might generate actions such as `go to drawer 1` or `open drawer 1`, differing from the more structured action types in DBBench and OS Interaction. This distinction highlights the sensitivity of model performance to the specific characteristics of training data, indicating that not all data combinations are beneficial and the impact of data diversity on model tuning is complex and nuanced.

2. **Finetuning's Sensitivity to Erroneous Samples** From our most effective technique for grounding the environment described in the previous section, we find that the model is highly sensitive to erroneous examples during finetuning. In this case, we collect the data similarly to AgentInstruct by running a high-performing model like GPT-4 on the same set of tasks against a synthetic environment. We collect and manually review the data samples for each task. We observe that the environment prompting in the grounding case works in most cases, except a few that are filtered out and replaced with gold samples. However, we do not achieve performance comparable to the original AgentInstruct. It should be noted that the GPT-4 agent also makes some errors, and we do not filter out the samples where GPT-4 is incorrect, unlike the creation of the original AgentInstruct dataset. The presence of erroneous agent responses in our dataset, compared to the gold samples of the AgentInstruct dataset, leads to a difference in performance in the finetuned models. This observation leads to the insight that generating data from a high-performing model like GPT-4 is not sufficient; filtering out gold samples in the generated data is essential to complete the synthesis process. The synthetic data we generate for DBBench, using the grounding trick with results in Table **??**, shows that the model tuned on it achieves 33% of the accuracy of the model tuned on the gold AgentInstruct dataset. Surprisingly, the accuracy of GPT-4 on the DBBench task is also 33%. Though this relation does not hold true for other tasks, there is undoubtedly a performance degradation due to erroneous samples.

3. **Environment Responsiveness to Agent Queries** When the agent deviates from predefined actions, seeking new information or clarification, the environment, designed constraining a general chatbot, often responds by breaking its intended constraint of acting solely as an environ-

Model	Dataset Description	DB
GPT-4-0613		49.5
meta-llama/llama-2-7b-chat-hf	Base Model	0
YieldInc/metallama2_agentinstruct-axolotl	Complete AgentInstruct and mixed ShareGPT in the ratio 2:8	10
YieldInc/agentinstruct_kg_env-sharegpt	Complete AgentInstruct with KG split replaced by synthetic KG (without filtering) and ShareGPT with a mixing ratio of 2:8.	0
YieldInc/agentinstruct_kg_env-sharegpt-v3	Complete AgentInstruct with KG split replaced by synthetic KG (without i filtering) and ShareGPT with a mixing ratio of 2:8.	5

Table 8: **KnowledgeGraph** evaluation scores comparison of model finetuned using synthetic agent-environment conversation data generated via the grounding mechanism

Model	Dataset Description	DB
GPT-4-0613		26.1
meta-llama/llama-2-7b-chat-hf	Base Model	13
YieldInc/metallama2_agentinstruct-axolotl	Complete AgentInstruct and mixed ShareGPT in the ratio 2:8	13
YieldInc/agentinstruct_kg_env-sharegpt	Complete AgentInstruct with KG split replaced by synthetic KG (without filtering) and ShareGPT with a mixing ratio of 2:8.	
YieldInc/agentinstruct_kg_env-sharegpt-v3	Complete AgentInstruct with KG split replaced by synthetic KG (without i filtering) and ShareGPT with a mixing ratio of 2:8.	10

Table 9: **Mind2Web** evaluation scores comparison of model finetuned using synthetic agent-environment conversation data generated via the grounding mechanism

Model	Dataset Description	DB
YieldInc/db_orig_and_syn-5	DB	18.3
db_os_orig_and_db_os_syn	DB + OS	21.6
YieldInc/kg_db_orig_and_db_syn-5	DB + KG	15
YieldInc/db_webshop_orig_and_db_syn-5	DB + WS	16.6
YieldInc/agentinstruct_db_os_hh_orig_and_syn	DB + HH	6.6

Table 10: Scores comparison of model finetuned using synthetic data generated via Human Intervention

ment. It thus starts to utilize its capabilities as a versatile chatbot. For instance, in the agent-environment dialogue in Appendix **??**, the rigidity of the environment as defined in GPT-4 is compromised when the agent requests the creation of a table. The environment accommodates this request, demonstrating its ability to go beyond the limitations of a mere environmental entity.

4. **Models finetuned on datasets that achieve the same score as the base model should also be considered as effective finetuning datasets.** Through extensive experiments, we observe that most of the finetuned models score lower than the base LLaMA-2-7B-chat model (without finetuning), indicating that the performance of a model post-finetuning can range from 0 to 100%. A dataset that is highly irrelevant may significantly deteriorate the model's existing performance, while a dataset that does not decrease the model's performance could be considered to reinforce what the model already knows. Conversely, a dataset that enhances the model's score may introduce both reinforcement of existing knowledge and new information that improves performance. Thus, it can be intuitively concluded that if a dataset does not reduce a model's score, suggesting it reinforces existing knowledge, then it could indeed be beneficial for any other model performing less efficiently than the initial model.

5. **Lack of Fine-Grained Evaluation** The evaluation of agents in AgentBench or any existing agent benchmarks is not sufficiently detailed. It predominantly relies on comparing the final outputs with a predefined ground truth through string matching. Occasionally, an agent does not adhere precisely to the instructions, leading to a failure in parsing the output. Moreover, the system does not terminate the agent-environment interaction when the final answer is incorrectly formatted, resulting in endless unnecessary conversation exchanges. For instance, after the agent completes the task but fails to format the final answer correctly, this leads to endless exchanges of thanks between the agent and the environment until the conversation loop is terminated by the maximum loop limit. Relying solely on regular expressions for parsing proves inadequate; a more generalized parser that incorporates elements from language models is essential for interpreting the model's outputs. Regular expressions are too rigid and outdated for these scenarios.

   The scoring system for these evaluations is binary and does not encourage detailed assessments of language model agents. Multi-turn conversation evaluations represent a significant challenge in the industry, and this issue persists here as well. For instance, a model might initially take correct steps toward accomplishing the task but then veer off course due to an error at a specific step. It is crucial to evaluate the severity of this mistake and to acknowledge the model for its initial correct actions. Consider the example agent-environment interaction here for Mistral-7B-v0.1 and AgentLM-7B, involving an OS-Interaction task where the agent is asked, `Tell me the max number of threads in my computer`. Although neither model delivers the correct answer, leading to a score of 0, a closer examination of their action trajectories through the provided PDFs shows that Mistral significantly underperforms in following instructions compared to AgentLM. Despite its inaccuracies, AgentLM adheres to the instructions and demonstrates an understanding of the task, albeit with flaws in its reasoning. In contrast, Mistral fails to grasp the instruction and produces everything simultaneously, acting as both the agent and the environment. This behavior likely originates from its design as a completion model rather than a chatbot model. Scoring AgentLM, a more adept chatbot model, equally with Mistral would be unfair.