Aligning LLM Tool-Use with Human Insight: Project Milestone

Will Li [*1] **Durga Malladi** [*1] **Alfred Wechselberger** [*1]

Abstract

This project explores the alignment of large language models (LLMs) with human feedback to enhance tool-use capabilities through APIs. We investigate the integration of reinforcement learning techniques with LLMs to perform complex tasks by leveraging external tools such as scheduling, travel planning, and home automation. Building on previous work like Toolformer, our approach aims to improve sampling efficiency and performance through human-in-the-loop feedback, using both Reinforcement Learning from Human Feedback (RLHF) and Direct Preference Optimization (DPO). This milestone report details our methodologies, implemented APIs, task generation, and preliminary baselines, setting the stage for subsequent robust evaluations and refinements.

1. Introduction

1.1. Motivation

The power of large language models (LLMs) to transform human workloads extends beyond language itself with recent research in augmenting LLM agents with external tooling through APIs (eg. code, the Internet, WolframAlpha). These tool integrations can expand the functionality and versatility of language-based agents to not only generate text but to interact dynamically with digital and non-digital environments (Sumers et al., 2024). Familiar examples include knowledge retrieval (Lewis et al., 2021), OpenAI GPT4's code interpreter (OpenAI, 2024a), and LLM powered calendar schedulers (Gordley, 2024).

As tool-use becomes increasingly complex, language agents will gain more agency but also encounter downstream consequences in the real world. For example, GPT4's Technical Report demonstrated that the language model when augmented with tools could purchase potentially adversarial chemicals online and even trick humans into completing tasks (OpenAI, 2024b). Active tool use requires not only more subtle agent world understanding but also agent guidance to avoid harmful outputs and align with human values. Reinforcement learning (RL) techniques have successfully aligned LLMs with human feedback, but their applications for LLM agents and tooling have yet to be fully explored.

1.2. Related Work

Previous work for language model tooling for Toolformer used labeled demonstration and semi-supervised learning to instruct an AI on how to interact with external APIs (Schick et al., 2023). Additionally, language agent papers have demonstrated substantial performance gains through self critique (Aksitov et al., 2023). These techniques may not work as well for more complicated tool tasks without labeled demonstrations. For example, Toolformer can only use APIs to improve future token prediction on downstream tasks with automatic evaluation and cannot handle tasks without defined objective functions. Moreover, their API sampling is incredibly inefficient and cannot learn performance scores (Schick et al., 2023). This project explores if complex tool use can be learned more efficiently and on a wider range of tasks through human feedback on the outputs.

RL techniques such as Reinforcement learning from human feedback (RLHF) (Ouyang et al., 2022) and Direct preference optimzation (DPO) (Rafailov et al., 2023) have greatly improved the alignment of pre-trained LLMs with desired tasks. We aim to combine preference alignment and language tooling work to explore the performance improvements realized from fine-tuning LLMs when using external tools through APIs calls.

2. Approach

2.1. Tools

We present below each of the APIs that the language model will have access to. We will have a simulated home and calendar and track their states.

`getCalendar(day)`: Returns all the events, their ids, their times, and their descriptions on the calendar for day.

`setEvent(day, start_time, end_time, description)`: Sets an event on the calendar at the provided time and with

[*]Equal contribution [1]Department of Computer Science, Stanford University, CA, USA.

d$escription$. Returns an id for the event.

`delEvent(id)`: Deletes the event on the calendar with id

`getTemperature(city, time)`: Returns the forecast temperature for city at time. Uses the National Weather Service's real RESTful API.

`getForecast(city, time)`: Returns the weather forecast for city at time. Uses the National Weather Service's real RESTful API.

`setHomeTemperature(temperature, time)`: Sets the virtual home's thermostat to t$emperature$ at time.

`waterPlant(duration, time)`: Waters the virtual home's outdoor plant for d$uration$ at time.

`askWolframAlpha(question)`: Returns the WolframAlpha NLU System's response to q$uestion$.

All the above APIs have been implemented.

2.2. Tasks

We will train and evaluate our model on the following tasks relevant our API tools. We have generated hundreds of instances of each task through GPT4. We are considering expanding the task list's diversity.

Task	APIs
Scheduling	calendar, weather, WolframAlpha
Travel Planning	calendar, weather, WolframAlpha
Outfit Planning	calendar, weather
Home Automation	watering, thermostat

Table 1. Tasks

The task data has been partially generated.

2.3. Baselines

For all our base models we will use LLaMA 2 7B.

2.3.1. Zero-Shot Prompting

Our first baseline will be providing the LLM with the API documentation and have it execute the task. We will experiment with providing different levels of detail and tool explanation.

This baseline has been implemented.

2.3.2. Few-Shot Prompting

Our second baseline will be similar to the first, but we will use human knowledge to set up few-shot / in-context learning. This is the most basic form of human alignment.

This baseline has been implemented.

2.3.3. Semi-Supervised Fine-Tuning

Our third baseline will be similar to Toolformer (Schick et al., 2023). During generation for a prompt response, we will record the top k candidate positions for including the API token, $p_M(< API > |P(x), x_{1:i-1})$. We will then execute these k API calls per generation. Unlike Toolformer, we will use human evaluation to score the final output and filter out instances where the usage of the tool was not preferred to . We will then finish by finetuning the language model on the effective tool use sequences.

This baseline has not been fully implemented– it is just missing the human scoring interface

2.4. Methodology

2.4.1. Sampling

We will sample in the same way as the semi-supervised fine-tuning baseline, but we will shrink k to 1. In particular, if a sequence $\{x_n\}$ makes API calls in indices I, our pair sequence $\{x'_n\}$ will match $\{x_n\}$ until $\max I$ but continue generation without the last API call. If a sequence $\{x_n\}$ does not contain any API calls, we will construct $\{x'_n\}$ by setting $x_{argmax_i p_M(<API>|P(x),x_{1:i-1})} =< API >$ and continuing generation.

This step has been completed but not tested

2.4.2. Scoring

We will include human feedback in the same way as the semi-supervised fine-tuning baseline. We, however, will not filter any results and store all our human comparisons.

2.4.3. Learning

We will then use existing RLHF and DPO techniques for tool use learning by the LLM. We will take the human scores and either train a reward model or directly optimize to the human preferences with a Kullback–Leibler divergence penalty. Specifically, we will update the language model's token generation policy to iteratively improve the LLM through feedback on the following loss function for RLHF and DPO, respectively:

$$L_{RLHF}(r_\phi, D) = -\mathbb{E}_{x,y_w,y_l\sim D}\left[\log \sigma(r_\phi(x, y_w) - r_\phi(x, y_l))\right]$$

$$L_{DPO}(\pi_\theta; \pi_{ref}) = -\mathbb{E}_{x,y_w,y_l\sim D}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{ref}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{ref}(y_l|x)}\right)\right]$$

The Toolformer paper mentioned that the prompt-only sampling efficiency was very poor. We could counteract this by bootstrapping with the increasingly human aligned generation policies.

2.5. Evaluation

We will evaluate the performance of tool usage on mostly qualitatively human benchmarks. We may also use quantitative task specific metrics like penalizing schedule conflicts or evaluating travel plans with GPT4 as a judge. The generated API calls should be valid as a baseline level of performance.

Because API calls can often be expensive in the real world, we will also track the number of API calls required during the training process.

2.5.1. Ablation

We will compare our main method to the Toolformer performance to compare human feedback tuning against semi-supervised fine tuning on labeled demonstration. Within our main method, there we can try zero-shot and few-shot prompting to generate our tool-use samples for human preference and evaluate prompt quality in generating effective samples. If the sampling and human scoring is still ineffective, we can try training a baseline model with semi-supervised fine-tuning to generate samples instead of the base model. We will also plot performance against rounds of human feedback.

3. Next Steps

1. Finalize all the APIs, tools, and tasks. Choose domain sepcific metrics from there. Acquire baseline metrics for all tasks.

2. Integrate and test all APIs with existing Toolformer pipeline. Ensure that RESTful APIs are working properly and parsing results.

3. Add a human scoring interface to the supervised fine-tuning and evaluate that benchmark.

4. Set up an RLHF and DPO module in the generation code. Test the pipeline of sampling then scoring then learning.

5. Conduct robust evaluations with high quality human feedback/volunteers.

References

Aksitov, R., Miryoosefi, S., Li, Z., Li, D., Babayan, S., Kopparapu, K., Fisher, Z., Guo, R., Prakash, S., Srinivasan, P., Zaheer, M., Yu, F., and Kumar, S. Rest meets react: Self-improvement for multi-step reasoning llm agent, 2023.

Gordley, J. Google calendar assistant, 2024. URL https://github.com/jgordley/GoogleCalendarAssistant?tab=readme-ov-file. Accessed: 2024-05-21.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., tau Yih, W., Rocktäschel, T., Riedel, S., and Kiela, D. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021.

OpenAI. Code interpreter, 2024a. URL https://platform.openai.com/docs/assistants/tools/code-interpreter.

OpenAI, e. a. Gpt-4 technical report, 2024b.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback, 2022.

Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C. Direct preference optimization: Your language model is secretly a reward model, 2023.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools, 2023.

Sumers, T. R., Yao, S., Narasimhan, K., and Griffiths, T. L. Cognitive architectures for language agents, 2024.