LLM Agent for buying airplane tickets

Alexey Anosov

July 29, 2024

Abstract

This paper presents a report on the creation of an agent for purchasing airline tickets based on an open-source language model. Mistral7B-Instruct-v0.3 was chosen as the model. English is used as the language of communication. The project currently uses synthetic data and mock functions, but in the future it is possible to connect real APIs. It was shown that using a ReAct-like framework and additional training with LoRA can achieve a significant improvement in quality compared to the baseline. However, more data is needed to draw conclusions about the maximum capabilities of this approach.

Keywords: LLM agent, Mistral, LoRA, ReAct

1 Introduction

Large language models have proven themselves well for a large number of tasks in recent years. One of the most promising areas of practical application of such models is the creation of agents. An agent is able to receive tasks in human language and perform them by accessing various tools as needed. Knowledge bases, APIs of various services, etc. can act as tools.

This paper studies the possibilities of using small language models (up to 7B params) as a basis for agents. Mistral7B-Instruct-v0.3 was chosen as a model, but in the future we would like to conduct similar experiments with other models of this size.

The task was to book airline tickets. This project uses synthetic data and mock functions, but in the future it is possible to connect real APIs. English is used as the language of communication.

2 Related work

There are various ways to improve the behavior of agents based on large language models.

One way is to retrain the model, as for any other task. In practice, the LoRA method is most often used for this. The LoRA approach to updating the model weights is to add the product of two trained matrices of lower rank to the original (frozen) weights. This allows training much fewer parameters than with classical retraining of a pre-trained model, which makes this type of retraining preferable.[2]

However, there are other approaches aimed at changing the model's behavior using prompts. It has previously been shown that large language models work better in the few-shot mode. [1] Few-shot means that the model is given one or more examples of correct model behavior in the prompt. This can significantly improve the quality of answers for a number of tasks.

There are also a number of ways to improve the model's behavior for complex tasks.

Chain of Thoughts is a simple way to improve the model's behavior. It has been noted that if the model solves a problem by describing each step of the reasoning, the quality of the answers increases significantly. [5]

Tree of Thoughts is an extension of the Chain of Thoughts approach that allows the model to generate multiple hypotheses and evaluate them independently to select the best one. [3] [6]

ReAct is a method that allows agents to more naturally use external sources or other tools, increasing accuracy and reducing errors. This method combines reasoning and action generation at each step. Unlike traditional approaches that treat reasoning and action as separate entities, ReAct combines them. [7]

Reflexion is a method designed to improve language agents. This approach allows the model to reflect on task feedback and then store its conclusions from previous steps as text in an episodic memory buffer to make future steps more efficiently. [4]

There are also more complex approaches, such as ExpeL. During the training phase, the agent interacts with the environment, collecting experience through trial and error. This experience is stored in an experience pool. During the inference phase, the agent performs tasks, supplementing the received prompts with knowledge and successful trajectories from its pool of experience gained during the training phase. [8] However, such methods are aimed at working with the API of large language models, where there is a large context window and no possibility of retraining the model. In the case of working with local models, it is more convenient to retrain the model, but reduce the prompt than to speed up the performance and consumed resources.

3 Dataset

3.1 Aiplane flights dataset

The airplane flights dataset was generated independently, since it was not possible to find suitable data on the Internet. Data for individual countries (for example, the USA) was found, but I wanted to consider flight directions that were more understandable for me and those who would check the task.

Using Gpt-4o, 100 cities popular among Russians were generated, 20 of which are in Russia, the rest are abroad. After removing duplicates in the model response, 87 cities remained. For all ordered pairs of cities, using Gpt-4o, the following information was obtained:

1) how many direct flights per day on average between cities
2) how much flights between cities cost on average
3) how long flights between cities last on average

The information may not be completely correct, but this is not critical for this project.

Then, a schedule of all direct flights for one week was compiled. For each ordered pair of cities, for each day of the week, N random departure times were generated, where N is the number of direct flights obtained from Gpt-4o model in the previous step. The price suggested by the model became the cost of an economy class ticket. For each flight, the cost was multiplied by a random coefficient to create variability. This made queries like 'Buy the cheapest ticket from A to B for tomorrow' meaningful.

Also, for each flight, the cost of a business class ticket and extra baggage was added. These costs were obtained by multiplying the cost of economy class by random coefficients.

Next, flights with 1 or 2 stops were generated (3 of each type for all ordered pairs). For city pairs with direct flights, the data was obtained by multiplying the time and cost of a direct flight by random coefficients, such that:

1) a flight with transfers can be cheaper than a direct flight
2) a flight with transfers is always longer than a direct flight
3) a flight with 2 transfers can be faster and/or cheaper than one with 1 transfer

This method of data generation allows us to model those trade-offs that travelers often face. For city pairs with no direct flights, data on flights with transfers was generated randomly.

This resulted in a schedule for 1 week. It is assumed that it does not change from week to week, and when requesting flights on a specific date, the answer will be given based on the day of the week.

3.2 Evaluation dataset

A synthetic dataset was used to validate the model. The query bases were obtained using Gpt-4o and manually refined to accurately match the requirements.

The validation dataset currently includes 10 queries for 2 categories:

1) The initial query contains all the information needed to clearly select the most suitable ticket.
2) The initial query contains only part of the required information, the agent must make clarifying queries.

4 Experiments

4.1 Metric

To evaluate the quality of the model's work, it is not enough to simply check the correctness of the received answer. We also want to be sure that the model does not ask the user unnecessary questions and does not make unnecessary/incorrect tool calls. In this regard, a system for evaluating model responses was developed (Table 1).

Criterion	Number of points	Comment
The correct answer was received	+10	The ticket that was supposed to be selected, the answer is displayed in the correct format The selected ticket matches the filters, but does not match the user's priorities (the ticket is selected for the required date, but not the cheapest one, although the user asked for the cheapest one)
A valid response was received	+5	The selected ticket matches the filters, but does not match the user's priorities (the ticket is selected for the required date, but not the cheapest, although the user requested the cheapest)
A request was made to the user that is not required but is not a serious error.	-1	For example, if a customer requested 'cheapest ticket for the morning', asking them to specify specific hours would be an unnecessary request.
An unnecessary request was made to the user	-3	For example, the same question is asked twice
The required tool was not used, but this should not have affected the result.	-1	For example, the agent did not make sure that the cities from the user's request were among the available ones, but immediately made a request to the data

Table 1: Criteria for evaluation

Thus, the points for the task are in the range from 0 to 10 points (if there is no correct answer, points are not subtracted).

The final score of the model is equal to the share of the points scored from the maximum.

At this stage, validation was carried out manually. However, in the future, it can be automated.

4.2 Finetuning dataset

During the process of creating the agent, as new tools were added, the agent's ability to manage them quickly decreased. In this regard, there was a need for additional training of the model. Data for additional training was collected using the Gpt-4o model. A total of 325 examples were collected (45 complete dialogues, 3 of which were selected as a test sample during training).

4.3 Finetuning

Experiments with additional training were conducted with the Mistral7B-Instruct-v0.3 model (4bit quantization). The LoRA method was used. Experiments were conducted with LoRA hyperparameters, learning rate, lr scheduler, and so on. The training results can be viewed here: https://wandb.ai/alexej-anosov/MTS-NLP-task-2024

5 Results

As a result of retraining the model, its behavior has improved significantly. If it was almost impossible to get the correct answer for the base model, it is often possible to do so with the retrained model. However, the model regularly performs unnecessary actions. In this regard, due to penalties for unnecessary actions, the model's score is still significantly lower than that of Gpt-4o (Table 2). However, as the amount of data in the training set increases, the quality can be greatly improved using the same training pipeline.

Model	Score
Mistral7B-Instruct-v0.3	0
Mistral7B-Instruct-v0.3 (4bit+LoRA)	19.5
Gpt-4o	75.5

Table 2: Result scores

6 Conclusion

Thus, it was shown that a small language model can act as a basis for an agent capable of operating 6 tools to perform an applied task, but this requires additional training on a significant amount of data.

As further steps, it is supposed to automate the experiments:

- Automate the evaluation of the model's performance using the API of a larger model. Since the evaluation criteria are quite clearly formulated, this solution should work well. For a more stable evaluation, you can average over several runs of the evaluated and evaluating models.

- Automate the collection of a training dataset using the API of a larger model. Perhaps, even without selecting unsuccessful samples, you can get a significant improvement in quality.

This will make it possible to conduct many experiments, for example, with the prompt structure, the agent activity framework (Reflecxion, ExpeL), various models as a basis.

References

[1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

[2] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

[3] Jieyi Long. Large language model guided tree-of-thought, 2023. URL https://arxiv.org/abs/2305.08291.

[4] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

[5] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

[6] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023. URL https://arxiv.org/abs/2305.10601.

[7] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

[8] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19632–19642, 2024.