Comparing ReAct and ReWOO using Commercial LLMs - Travel Planning

Harshith Belagur
MS CS, Columbia University
Email: hb2779@columbia.edu

Addrish Roy
MS CS, Columbia University
Email: ar4613@columbia.edu

***Abstract*—This project explores the development and comparison of a travel planning and recommendation system utilizing the ReAct[1] and ReWOO[2] frameworks. While travel planning applications are a common use case for large language models (LLMs), our approach introduces a novel methodology by employing agents to customize travel plans based on user preferences, delivering highly personalized itineraries. In addition, we compare the frameworks in different scenarios to see their performance. Traditional LLMs, despite their strengths, fall short for such applications due to limited integration with structured reasoning and external knowledge sources. To address these limitations, our system combines multi-stage reasoning with advanced database integration, including vector and traditional structured databases.**

I. Introduction

Travel planning is a complex task that requires balancing diverse user preferences, practical constraints, and dynamic contextual information. While large language models (LLMs) have demonstrated remarkable capabilities in natural language understanding and generation, their standalone use in travel recommendation systems is limited. LLMs often struggle with tasks requiring multi-step reasoning or integration of external structured knowledge, which are essential for creating personalized and reliable travel plans.

Existing solutions in travel planning tend to focus either on creating itineraries or on personalizing pre-defined plans for individual users. Our project aims to bridge this gap by developing a system that combines both aspects: intelligent planning and customization tailored to user preferences. Additionally, we benchmark and compare state-of-the-art techniques for building multi-step reasoning agents, specifically evaluating the ReAct[1] and ReWOO[2] frameworks in this context. Through this work, we provide insights into how these frameworks perform in a new application area, offering a fresh comparison of their strengths in complex, user-centered travel planning.

To overcome the inherent limitations of LLMs, we adopt a hybrid methodology that incorporates multi-stage reasoning with external database integrations. By leveraging both vector databases for semantic search and traditional structured databases for precise data retrieval, the system synthesizes contextual and real-world knowledge. This combination enhances the quality and relevance of the recommendations, making the system more effective and user-centric.

This dual focus on planning and personalization, alongside the systematic evaluation of cutting-edge reasoning frameworks, sets our approach apart. We aim to push the boundaries of how AI can assist in planning and decision-making, demonstrating the potential of combining LLMs with structured and contextual data integration for intelligent travel planning.

II. Literature Review

Travel planning has been a subject of significant research and development in the field of artificial intelligence and natural language processing. Traditional approaches to travel planning systems often focused on rule-based algorithms and database queries to generate itineraries and recommendations. However, these systems frequently struggled to balance comprehensive planning and personalization, typically handling itinerary creation or customization separately, leading to suboptimal user experiences.

The advent of large language models (LLMs) has revolutionized the approach to complex reasoning tasks, including travel planning. Current approaches in the literature, such as Chain-of-Thought (CoT) and action-only models, address reasoning and planning individually but fall short when it comes to combining them effectively for multi-step reasoning tasks. The CoT approach focuses on generating thought sequences but lacks interaction with external data, leading to issues like hallucination and error propagation. Conversely, action-only methods perform actions but do not leverage abstract reasoning, limiting adaptability and understanding of high-level goals. Both approaches lack integration, which limits their applicability in complex, real-world tasks where synergy between reasoning and acting is essential.

The ReAct framework emerged as an innovative approach to bridge the gap between reasoning and acting in language models. It combines reasoning traces and task-specific actions, allowing for a more dynamic and adaptive problem-solving process. This framework demonstrated improved performance on various tasks, including question answering and task completion, by leveraging both abstract reasoning and concrete actions. In the context of travel planning, ReAct showed promise in handling multi-step tasks that require both information retrieval and decision-making.

The ReWOO framework addresses multi-step reasoning challenges by separating reasoning processes from observational tool feedback, which ReAct interweaves. While ReAct

uses an observation-dependent structure, halting reasoning for each tool response, ReWOO decouples these elements, organizing tasks into distinct planning, working, and solving stages. This modular approach minimizes prompt redundancy and cuts token usage, shown to be up to 5× more efficient on benchmarks like HotpotQA. Traditional systems face high computation and cost inefficiencies due to repetitive prompt contexts, which ReWOO avoids, making it suitable for scalable, real-world applications. To improve upon existing limitations in travel planning systems [3], our approach leverages a unique integration of both planning and personalization to ensure that recommendations meet user-specific constraints. Unlike traditional systems, which either handle itinerary creation or customization separately, our model dynamically tailors plans based on user preferences within complex, real-world conditions.

III. Methodology

Our travel planning agent was designed to focus on several key tasks that we identified as critical for creating a comprehensive travel planning experience:

1) **Recommending a Destination:** Based on explicit and implicit user preferences, the agent determines the most suitable travel destination.
2) **Flight Booking:** The agent retrieves and recommends flight options that align with the user's preferences and travel dates.
3) **Hotel Booking:** Accommodation options are suggested based on the destination, budget, and user preferences.
4) **Selecting Local Attractions:** The agent curates a list of activities and attractions in the destination city, personalized to the user's interests.
5) **Trip Cost Estimation:** A built-in calculator aggregates the estimated costs of flights, accommodations, and attractions to provide a total trip cost.
6) **LLM:** A general LLM to help get the destination's IATA code, this could be a datastore as well but to keep it fairly complex, we used an LLM (this is needed only for the ReWOO agent as ReAct either way uses an LLM at every turn in the reasoning).

To enhance the user experience, we integrated a relational database (SQL) to store information about attractions. Furthermore, a retrieval-augmented generation (RAG)-based recommender system was implemented to dynamically suggest destinations and activities based on user preferences. This combination of structured data and RAG enables the agent to provide highly relevant and personalized recommendations, bridging the gap between static databases and adaptive planning systems.

Let us go into the workings of each tool -

A. Recommending a Destination

To identify the most suitable destination, we utilize an in-memory FAISS database containing 15 cities along with their descriptions. This database serves as the foundation of our retrieval-augmented generation (RAG) system. When a user provides their preferences, the system identifies the most relevant chunk of information and retrieves the associated city using similarity-based matching. The embedding model used here is the SentenceTransformer model for vectorizing the text to be stored in the vector DB.

B. Flight Booking

For flight booking, we leverage SerpAPI's Google Flights API. The tool accepts the IATA codes for the source and destination airports, along with the departure and return (if applicable) dates, as input. Based on Google Flights' internal ranking algorithm, which optimizes for factors such as cost, number of stops, and duration, the API returns the top three flight options.

C. Hotel Booking

Similarly, SerpAPI's Google API is used to fetch hotel recommendations. The tool requires the destination city and the duration of the stay as inputs. The API returns the top three hotel options, prioritized based on Google's internal ranking criteria.

D. Selecting Local Attractions

To recommend local attractions, we maintain a SQL database that includes information about ten attractions for each of the 15 cities in the RAG database. By inputting the name of a city, the system retrieves a curated list of attractions relevant to that destination, enhancing the personalization of the travel plan.

E. Trip Cost Calculator

Once the user selects their preferred flight and hotel, a Python-based cost calculator is employed. This simple yet effective function processes a string equation to sum up the costs of the selected flight and hotel, providing the user with the total trip cost (excluding additional activities and excursions).

F. LLM

The LLM gets as input the name of the destination city and returns only the three-character IATA code of the arrival airport that is sent to the next tool.

We used Langgraph to build both the ReAct and ReWOO agents in our code implementation.

ReAct is provided straight out of the box using the `create_react_agent` function. We use Langgraph's prebuilt graph to build the ReAct agent. Figure 1 provides a good way to visualize the prebuilt graph for ReAct.

But, for ReWOO we manually constructed the agent by building the graph in Langraph as explained below.

- **Planner:** Generates a plan using a designated LLM.
- **Worker/Executor:** Executes the plan by doing a string parsing of the plan and performing tool calls.
- **Solver:** Processes results from the Worker to provide the final output.

Figure 2 provides an intuitive way to visualize the graph built for ReWOO.

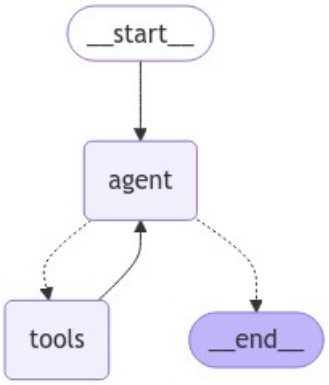

Figure 1. ReAct graph visualization

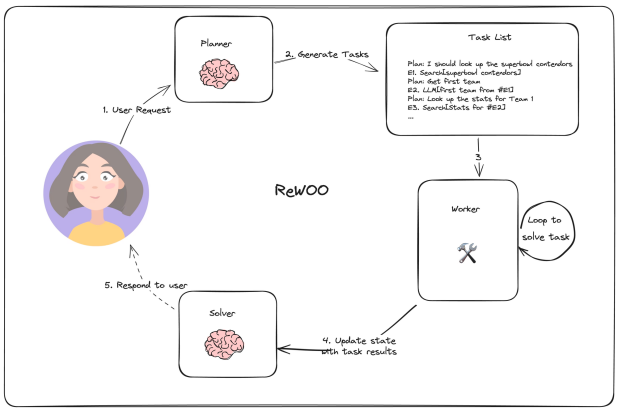

Figure 2. ReWOO graph visualization

Although the Planner and Solver nodes can utilize different LLMs, we used the same LLM for both due to time constraints. An important distinction to note is that ReWOO requires the entire context upfront before processing can begin—specifically before the Planner generates a plan. This necessitates collecting user information, such as destination preferences and travel dates, in advance.

In contrast, ReAct integrates this information seamlessly as part of the conversation. To ensure a fair comparison between the two tools, we introduced a separate component that interacts with the user to gather destination and travel dates before invoking the ReWOO component.

Therefore, to have a fair comparison, we take the user input for our system as a preference, along with the travel dates.

We use one-shot prompting for both techniques since ReWOO's planner does not know the different tools available and how it can create leverage those to create plans. It is important to note that we pass in the present date in the prompt as a lot of times the models have a cut-off date from a previous year. This causes them to input the wrong year when the user specifies just a month and year, for eg. December 30 would be construed as December 30, 2023, instead of December 30, 2024.

The prompt used for ReAct is:-

ReAct System Prompt

You are a helpful travel assistant named Fred.
Follow these steps in order:
1. CITY RECOMMENDATION: - Use the city_recommendation_tool and the user's given preference EXACTLY ONCE to suggest a city
2. FLIGHT SEARCH: - Use the user's given source as the departure_airport and the first recommended city as the arrival_airport - Ask for specific travel dates (departure and return, if return is not given, ask for it and get the duration of the trip) - Use flight_search_tool with IATA codes - Format: { 'departure_airport': '[IATA code]', 'arrival_airport': '[IATA code]', 'departure_date': 'YYYY-MM-DD', 'return_date': 'YYYY-MM-DD' Optional for one-way flights }
3. HOTEL SEARCH - Look for hotels in the recommended city for the dates of the flight (departure and return date) and pick the cheapest one
4. ATTRACTIONS: - Once the hotel and flight are selected, use the city_attractions_tool to get the top attractions in the recommended city. Make sure you match the correct formats for the tool
5. CALCULATOR: - Use the calculator_tool to sum up the selected flight and hotel costs and also provide the breakdown of the costs
6. END: - Once the user is done with the conversation, thank them and end the conversation
Remember that today's date is {datetime.now().strftime('%Y-%m-%d')}. Only search for future dates. Do not repeat city recommendations unless explicitly asked by the user.
For eg. If the user's name is John Lincoln, they like skiing and are flying from New York. First, recommend a city using the city_recommendation_tool. Ask the user where they are flying from and the dates of travel if they haven't given it. Use the flight_search_tool to get the flight details.
Then look for hotels in the destination for the dates of the flight (departure and return date) using the hotel_search_tool. Then look for attractions in the destination using the city_attractions_tool.
Then calculate the cost of the trip using the calculator_tool. Inform the user of their whole itinerary.

The prompt used for ReWOO is:-

ReWOO System Prompt

For the following task, make plans that can solve the problem step by step. For each plan, indicate which external tool together with tool input to retrieve evidence. You can store the evidence into a variable #E that can be called by later tools. (Plan, #E1, Plan, #E2, Plan, ...)
Tools can be one of the following:
(1) city_recommendation_tool[input]: Suggests cities for the user. Input should be user preferences or broad travel interests.
(2) flight_search_tool[input]: Finds flights based on given parameters. Input format: { 'departure_airport': '[IATA code]', 'arrival_airport': '[IATA code]', 'departure_date': 'YYYY-MM-DD', 'return_date': 'YYYY-MM-DD' # Optional for one-way flights }
(3) city_attractions_tool[input]: Fetches top attractions for a city. Input should be the city name.
(4) calculator_tool[input]: Computes costs or sums up expenses. Input should be a list of numbers to calculate the total without any text like the currency.
(5) hotel_tool[input]: Finds hotels in a city for a given date. Input format: { 'destination': name of the city, 'check_in_date': 'YYYY-MM-DD', 'check_out_date': 'YYYY-MM-DD' }
(6) LLM[input]: Get the IATA code of the city from #E1 and return only the 3 character IATA code and nothing else. For eg. if it is Dubai, return DXB and nothing else. Be concise.
Begin!
For example,
Task: My name is sam controlman and i want to plan a holiday from 14th to 15th december. I like skiing and am flying from barcelona
Plan: Since Sam likes skiing, pick a city. #E1 = city_recommendation_tool[recommend a city]
Plan: Get the IATA code of the city from #E1. #E2 = LLM[get the IATA code of the city from #E1 and return only the 3 character IATA code and nothing else. For eg. if it is Dubai, return DXB and nothing else. Be concise.]
Plan: Find flights from barcelona to #E2 on 14th december and return on 15th december. #E3 = flight_search_tool[departure_airport: barcelona, arrival_airport: '#E2', departure_date: 14th december, return_date: 15th december]
Plan: Find hotels in the destination. #E4 = hotel_tool[destination: '#E1', check_in_date: 14th december, check_out_date: 15th december] Plan: Find the top attractions in the city. #E5 = city_attractions_tool[#E1]
Plan: The total cost of the trip is the sum of the cost of the flights and the cost of the hotels. #E6 = calculator_tool[#E3 + #E4]
Do not forget that we are in the year 2024.
All inputs have to be in JSON format. # Task: {task}
Begin!
Describe your plans with rich details with what the described plan only. Each Plan should be followed by only one #E.

IV. Results and Analysis

This section presents a comprehensive analysis of our experimental results comparing the ReAct and ReWOO frameworks in the context of travel planning tasks. All the results have been generated by using **GPT-4o** as our main LLM. We examine various aspects of performance, including token consumption, cost using commercial APIs, and conversational capabilities across different scenarios. Our analysis provides insights into the efficiency, scalability, and robustness of both approaches, highlighting their strengths and limitations in real-world travel planning applications. The following subsections detail our findings, supported by quantitative data and qualitative observations, to offer a nuanced understanding of how these frameworks perform under various conditions.

A. Token Consumption Analysis

Our study compared the token consumption between ReAct and ReWOO across various scenarios. We observed significant differences in token usage patterns, which have implications for both performance and cost.

1) Complete Input Scenario With All Tools: With a complete input and 5 tools, the following observations were generated by averaging over 5 prompt inputs:

Metric	ReAct	ReWOO	Difference
Total Tokens	9094.8	8618.2	-5%
Prompt Tokens	8808.0	6829.6	-22%
Completion Tokens	486.8	1788.6	+267%

Table I
Token comparison with 5 tools

ReWOO showed a slight reduction in total tokens (-5%) but a significant increase in completion tokens (+267%).

We also analyzed the scenario where we used a dictionary instead of LLM in ReWOO for extracting the relevant input for a subsequent tool in the worker. The following observations were generated by averaging over 5 prompt inputs:

Metric	ReAct	ReWOO	Difference
Total Tokens	9094.8	7018.0	-22.8%
Prompt Tokens	8808.0	5429.6	-38.3%
Completion Tokens	486.8	1588.4	+227%

Table II
Token comparison with 5 tools and no LLM tool

As expected, this reduced the total token consumption in case of ReWOO from above. ReWOO here showed a more significant reduction in total tokens (-22.8%) with slightly less pronounced increase in completion tokens than above. The cost here is same for both.

2) Reduced Tool Scenarios: We also analyzed scenarios with fewer tools being used by the agent.

With a complete input and 4 tools (without the calculator tool), the following observations were generated by averaging over 5 prompt inputs:

Metric	ReAct	ReWOO	Difference
Total Tokens	6072.6	7004	+15.3%
Prompt Tokens	5454.2	5600.6	+2.7%
Completion Tokens	573.4	1403.4	+144.8%

Table III
Token comparison with 4 tools

With a complete input and 3 tools (without the calculator tool and attractions tool), the following observations were generated by averaging over 5 prompt inputs:

Metric	ReAct	ReWOO	Difference
Total Tokens	3640.4	6214.6	+70.7%
Prompt Tokens	3296.6	4684.2	+42.1%
Completion Tokens	345.8	1530.4	+343%

Table IV
Token comparison with 3 tools

As the number of tools decreased, ReWOO's total token consumption increased more relative to ReAct. This is mainly because as tools decrease, the fixed overhead in ReWOO remains constant. So if we have lesser tools, ReAct may be a better approach.

B. Error Handling and Robustness

We tested both approaches under various failure scenarios. First tool failure below refers to the failure of the City Recommendation tool and second tool failure refers to the failure of the flight search tool.

Metric	ReAct	ReWOO	Difference
Total Tokens	1246	2063	+65.5%
Prompt Tokens	1193	1662	+39.3%
Completion Tokens	53	401	+656%

Table V
Token comparison for first tool failure

1) First Tool Failure: When the first tool failed, ReWOO consumed significantly more tokens, particularly in completion, indicating a less efficient error recovery process.

Metric	ReAct	ReWOO	Difference
Total Tokens	2807	3111	+10.9%
Prompt Tokens	2687	2499	-7%
Completion Tokens	120	662	+451%

Table VI
Token comparison for second tool failure

2) Second Tool Failure: ReWOO again showed higher total token consumption although the difference is less this time,

suggesting that it may be less efficient in handling unexpected errors.

C. Conversational Capabilities

A key observation was that ReWOO cannot engage in conversations as it requires all data upfront. In contrast, ReAct can process information incrementally, making it more suitable for interactive scenarios. If we were to use ReWOO itself, we would have to add a separate agent prior to the ReWOO agent that would gather all the required information.

D. Cost Comparisons

The table below shows a comprehensive cost comparison for the various scenarios mentioned before across the 2 frameworks, ReAct & ReWOO.

Scenario	ReAct	ReWOO	Difference
5 tools	$0.10	$0.12	+20%
5 tools ReWOO modified	$0.102	$0.102	0%
4 tools	$0.08	$0.10	+25%
3 tools	$0.04	$0.09	+125%
1st tool failure	$0.01	$0.03	+200%
2nd tool failure	$0.03	$0.05	+67%

Table VII
COST COMPARISON FOR SCENARIOS

In commercial APIs, the cost for Completion Tokens is 2 times as much as Prompt tokens so ReWOO ends up being more expensive most of the times due to the much higher number of completion tokens being used as evident from the observations above.

Therefore, while ReWOO showed some efficiency gains in certain scenarios, it generally incurred higher costs due to increased completion token usage, given our tool choice.

Our finding is that one should not blindly pick one framework over the other but should rather perform exhaustive tests using the tools at hand. As we showed, using an LLM as a tool would cause the costs to go up significantly.

E. Discussion

Our results indicate that ReAct and ReWOO have distinct strengths and weaknesses:

1) **Efficiency**: ReWOO is more efficient in planning but less so in execution, especially for complex tasks.
2) **Scalability**: ReAct scales better with simpler tasks and fewer tools.
3) **Error Handling**: ReAct appears more robust in handling tool failures and unexpected errors.
4) **Interactivity**: ReAct's ability to process information incrementally makes it more suitable for conversational applications.
5) **Cost**: While ReWOO can be generally more efficient, it can also incur higher costs sometimes due to increased completion token usage.

These findings have important implications for the choice between ReAct and ReWOO in different application contexts, particularly in travel planning scenarios where task complexity and interactivity are key factors.

V. LIMITATIONS

Initially, our objective was to develop a conversational agent capable of engaging with users step by step, collecting their preferences incrementally through an interactive dialogue. This approach aligns well with the ReAct framework, which supports multi-step reasoning and dynamic interactions. However, the ReWOO framework operates differently. ReWOO requires the entire plan to be constructed in a single LLM call, making it suitable only when all relevant data is available beforehand. In scenarios where the complete dataset is accessible, ReWOO can efficiently generate comprehensive travel plans, as demonstrated in our experiments (add specific results when the entire plan is generated).

We did not fine-tune any models specifically for this study, unlike the original ReWOO authors who fine-tuned a Llama-7B model to create their Planner-7B. This limitation may have impacted the performance of the ReWOO framework in our experiments, as a fine-tuned model could potentially yield better results.

Our study focused specifically on travel planning tasks. While this provides valuable insights for this domain, the results may not be fully generalizable to other types of multi-step reasoning tasks. Further research would be needed to validate these findings across different domains and task types.

The performance of both frameworks was evaluated using a specific set of tools (city recommendation, flight search, hotel search, attractions database, and calculator). The relative efficiency of ReAct and ReWOO might vary with different tool combinations or more complex tool interactions.

While we tested scenarios with tool failures, our error simulations were limited to specific cases (first tool failure, second tool failure, and arbitrary tool failure). Real-world applications may encounter a wider variety of error types and combinations, which could affect the relative performance of the two frameworks differently.

Our cost calculations were based on the current pricing model of GPT-4o. Changes in pricing structures or the use of different language models could alter the cost-effectiveness comparisons between ReAct and ReWOO.

VI. CONCLUSIONS

In this project, we were able to develop a robust travel planning system, combining multi-step reasoning and personalization to deliver tailored itineraries. Leveraging the ReAct and ReWOO frameworks, we could not only integrate structured planning and customization but also address real-world constraints such as budget limits, location preferences, and specific user needs. By benchmarking ReAct and ReWOO, we showed that it is very important to test out the frameworks using the tools required for a particular task before deploying.

Another interesting observation that we had is that as the reasoning process gets longer, the costs associated with ReWOO start getting close to that of ReAct which indicates

that the fixed cost associated with the Planner gets amortized over longer workflows.

Further, this project explored the strengths and limitations of each framework, generating valuable insights into their utility for multi-step reasoning tasks that require contextual intelligence and practical adaptability. This project ultimately contributes to the broader AI field by demonstrating how advanced reasoning frameworks can be harnessed to create intelligent, user-centered applications for real-world planning scenarios.

The code for our work is publicly available at https://github.com/harshithbelagur/travel-planner-agent.

VII. Acknowledgements

We extend our gratitude to Prof. Parijat Dube and Prof. Chen Wang for offering an exceptional class and the opportunity to tackle a real-world problem using advanced techniques. The comprehensive course content provided valuable insights into the evolving trends in the industry and highlighted areas of active research that can be further explored.

References

[1] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. *arXiv:2210.03629*, 2023.

[2] Binfeng Xu, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu and Dongkuan Xu. ReWOO: Decoupling Reasoning from Observations for Efficient Augmented Language Models. *arXiv:2305.18323*, 2023.

[3] Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. Travelplanner: A benchmark for real-world planning with language agents. *arXiv:2402.01622*, 2024.

[4] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel and Douwe Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *arXiv:2005.11401*, 2021.

[5] https://langchain-ai.github.io/langgraph/tutorials/rewoo/rewoo