BABEŞ BOLYAI UNIVERSITY, CLUJ NAPOCA, ROMÂNIA FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

TravelPlannerAI

- ITSG -

Laurentiu Stancioiu Nicolae Morosan

Data Science for Industry and Society laurentiu.stancioiu@stud.ubbcluj.ro nicolae.morosan@stud.ubbcluj.ro

Contents

1	1. Introduction	1
	1.1 What? Why? How?	1
2	Problem to be Solved	4
	2.1 Problem definition	4
	2.2 Related work	4
3	Application Development	8
	3.1 Versioning tools	8
	3.2 Application Design	9
	3.3 Application Pipeline	12
	3.4 Application Code	14
	3.5 Model Deployment	14
4	Philosophical Aspects	16
5	Conclusions and future improvements	17

List of Figures

1.1	Kanban Board of our Application
2.1	Trip Advisor AI solution results
2.2	AI Trip Planner Chatbot UI
2.3	Wanderlust Demo Presentation at OpenAI devDay
3.1	Tools interactivity in the map module
3.2	App knowing references from questions asked before
3.3	Application Pipeline
3.4	Application monitoring tool
3.5	Grafana Dashboard containing metrics of the app

1. Introduction

1.1 What? Why? How?

Since the infamous paper Attention Is All You Need[1] from 2017 the Transformer architecture took the Natural Language Processing world by storm. This culminated with the fastest growing application that ever existed, ChatGPT[2], in 2022. After that, the field experienced one of the biggest booms that ever existed, introducing image processing and generation, open-sourced models led by Meta[3] and Mistral[4] and the introduction of multimodality[5], meaning that a transformer can take as input both text and images and furthermore, with a bit of engineering, you can input voice with text-to-speech models and receive voice as an output with a speech-to-text model.

With all this advances within just a few years, a lot of apps have emerged in the space, most of them using ChatGPT or an open-sourced model and Retrieval Augmented Generation (RAG) [6] through an embedding database in order to add some context that the Large Language Model (LLM) could use so that it could give a good enough answer for the user.

In June 2023, OpenAI announced Function Calling [7], which at the time did not bring so much fuss as it should have. For short with function calling you could describe some functions in a json format that have a name, a description and some parameters which also have some properties and a type. If we make a parallel to a Python function the name would be the function name the description would be the documentation of the function, the parameters would be the arguments of the function and their type can be an object a number an integer and so on. People at the time did not know what to do with them, as they were not extensively presented and documented, and furthermore, they were probably focusing to deliver something using ChatGPT alone.

So what is so great about functions? Well, if we continue the parallel, in python you don't always need to call all the functions you make within an app, and sometimes you just need a few of them.

How many you need depends on the way you designed your app. The trick with functions in OpenAI is the following: the LLM decides what functions to use based on the input you give it to them. So, for example if I have a GetCurrentWeather function that has the arguments longitude and latitude and in my input I say I want to see what is the weather like in Berlin then the LLM will call this function and will magically give me the weather in Berlin. How does this happen. Probably the LLM was trained with data from Berlin and "knows" its longitude and latitude, so it can auto-complete the arguments. You can even say which parameters are required and which of them are optional, exactly like in Python. This transforms the model from an unsupervised one, that can spew out answers with what was pretrained with into a semi-supervised one, where we give factual real-time information to the model so it could answer with it. For example, if you would ask what's the weather like in Cluj-Napoca to a model it would either say I don't know or make hallucinations or respond with some general monthly or seasonal weather conditions in Cluj-Napoca. By using these functions, we give the model the ability to answer with verifiable data.

This functionality, although very useful, was not getting the traction it should have deserved. What changed? In November, OpenAI announced on their DevDay the Assistants API[8]. Within Assistants you can easily add functions instructions and files that act as a Knowledge Base when the retrieval is called. So you have the RAG functionality mixed together with the function one, which are now called Tools. Also, you can add instructions and furthermore, you also get Code Interpreter functionality, which means that the assistant can process data such as a .csv file, can write code and also run it, if you ask for example to make a plot or something. That is now something which a lot of developers were amazed by, including ourselves.

Finally, we tried, inspired by this announcement, to create a travel assistant that can search for destinations, hotels, airports, attraction points, current day weather, to make reservations based on what we tell it to do, to scan airport tickets from pdf's and so on. The functionalities enabled in the assistant are only limited by the API's which we incorporate our functions with. In figure 1.1 we can see the kanban board with features we successfuly managed to get done in the app, and with features that we will do in the future, or ones that we are currently developing.



Figure 1.1: Kanban Board of our Application

Problem to be Solved

2.1 Problem definition

Planning a holiday has a lot of variables. The search of accommodation, weather, places of interest to visit such as museums, historical sites, places to eat, booking your flights and hotels at the best prices and so on. Furthermore, we would like something that would recommend us things without having to do a lot of research. And we know such a trip, even if it only is for a few days requires quite a lot of research beforehand and accessing a few websites or apps, where you even have to settle for something that kind of matches your expectations, after this time consuming experience. Having this in mind, and all the recent advancements of Large Language Models, we decided to try to create an AI Assistant within an app that has all these functionalities in mind. A chat interface where you talk or give text information to your assistant, which will also be personalized for you. Given the history of the chats it will know all your preferences without you having to be restrained to place it like in other apps.

2.2 Related work

There have been some apps that resemble parts of what we want to achieve. We will enumerate a few examples the first one being Trip Planner AI. Here, from our experience it seems more like an app that uses some API's following some parameters inputed by the user in order to generate some useful findings. On the plus side, it has a map where you can find some local interests during your trip and also scrapes prices for museums.

Another app that uses the same characteristics and and some extra benefits is TripAdvisor. Here you also have some constraints, for example the location, the date when you would like to go, and the

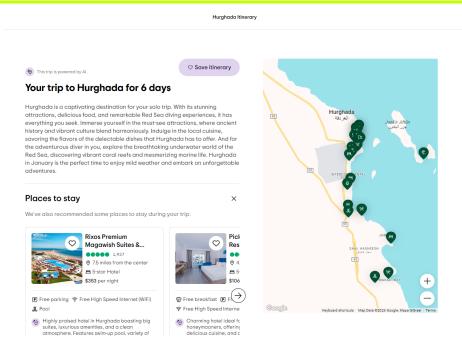


Figure 2.1: Trip Advisor AI solution results

number of people that are travelling. For pluses, we have some kind of communication, from what which we know that this solution at least uses an AI and their own API for searching for points of interest or hotels. The user experience though, is really intuitive and pleasing for the eye as you can see in 2.1.

The third app is AI Trip Planner by Booking.com. This is a chatbot such as ChatGPT, as we can see in 2.2. Although we can get information about weather, planning a trip or hotels, we don't actually get a map or flight information or point of interest prices and so on. So our guess is that this a wrapper around ChatGPT that also uses information given from Booking.com Databases. Also this is not available in Europe yet, but the feeling from their examples is that it won't answer difficult questions. Also it has limited interactibility and user experience, which is okay for a all around interface chatbot, but lacks a lot for one used for a single purpose, like travelling.

The final work which we will cover, that was also our main inspiration when developing the app, was Wanderlust, a demo application made by OpenAI at their devDay held in November. Although we decided and also had one of our first iterations of our app in the middle of November, the practicability and design of the app took us by surprise. Also, this demo app was a statement of the usefulness of the Assistants API, which we also used in our final implementation 2.3. From there we discovered an UI replica of it coming from Solara, but without implementing the actual functionality of it, so we decided to do it.

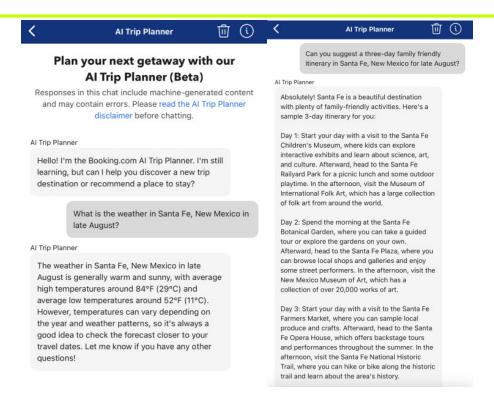


Figure 2.2: AI Trip Planner Chatbot UI

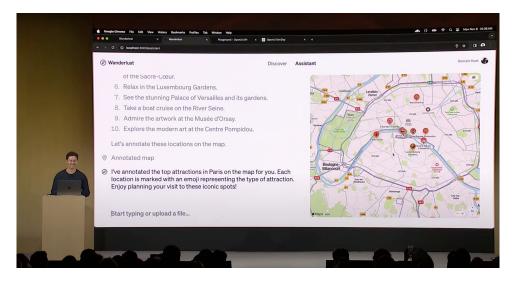


Figure 2.3: Wanderlust Demo Presentation at OpenAI devDay

Our proposed application consists of a combination of all the other proposed one: the simplicity of making frontend development in Python with Solara, an interactive map because it is said that an image is equal to 1000 words, and for the user to make an idea of how far away are places to one another, and a chat interface so the user can ask questions and receive answers based on them. Rather than hardcoding constraints such as budget, number of people or the duration of our visit, we let the user chat in a natural way and receive answers based only on information that was given. Finally, by

enabling that history, the LLM will make an idea of what the user likes or dislikes and give answers accordingly.

Application Development

3.1 Versioning tools

For versioning we primarily used git. Also, each time we created a new version, we containerized the application, deployed a docker image through a Dockerfile and used docker tags in order to track the version of our app. An example of how this works would be the following:

```
# Set up a new user named "user" with user ID 1000 for permission

# RUN useradd -m -u 1000 user

# Switch to the "user" user

USER user

# Set home to the user's home directory

ENV HOME=/home/user \
PATH=/home/user/.local/bin:$PATH

# Upgreade pip

RUN pip install --no-cache-dir --upgrade pip

COPY --chown=user requirements.txt .

# Install requirements

RUN pip install --no-cache-dir --upgrade -r requirements.txt

# COPY --chown=user *.py *.css /
```

Laurentiu Stancioiu Nicolae Morosan

```
21 ENTRYPOINT ["solara", "run", "app.py", "--host=0.0.0.0", "--port", "8080"]
```

Listing 3.1: Dockerfile used for containerizing the application

Then for making a new version we would have to run the following commands in a command line or powershell:

```
docker build -t TravelPlannerAI:testing-1.0
```

3.2 Application Design

HolidayPlannerAI application leverages the Solara framework. Solara lets you build web apps from pure Python using ipywidgets or a React-like API on top of ipywidgets. These apps work both inside the Jupyter Notebook and as standalone web apps or with frameworks like FastAPI. With Solara, you benefit from a paradigm that promotes component-based code and simplifies state management, making your development process more efficient and your applications more maintainable. Using the same API as React, but ported to Python, Solara lets you build apps with the same trustworthiness as React. With a decade of experience, React is battle-tested and proven to be a reliable and robust framework to build large scale apps. This enabled us to build and iterate our app instantly and to test it for each functionality that was added. The app consists of tools (functions) where we inform the LLM what functions we have available, the Assistant API from OpenAI, where we make our calls to the LLM and our frontend, where the user can see the results.

For Module design we can see automated and manual map interactivity. When the API is searching for point of interests or restaurants for example it zooms in automatically being integrated in the tools stack. For example, the map interactivity tools looks like in the json listing below:

```
{
1
     "type": "function",
2
     "function": {
3
       "name": "update_map",
4
       "description": "Update map to center on a particular location",
5
       "parameters": {
6
         "type": "object",
7
         "properties": {
8
           "longitude": {
9
             "type": "number",
10
```

```
"description": "Longitude of the location to center the map
11
      on"
            },
12
            "latitude": {
13
              "type": "number",
14
              "description": "Latitude of the location to center the map on
15
            },
16
            "zoom": {
17
              "type": "integer",
18
              "description": "Zoom level of the map"
19
            }
20
         },
21
         "required": [
22
            "longitude",
23
            "latitude",
24
            "zoom"
25
         26
       }
27
     }
28
29
```

Listing 3.2: Tool for updating the map

We can see in 3.1 that if this tool is activated we will get the longitude and latitude of our mapped components and zooms in on them. It also calls the add marker tool so it would put on the latitude and longitude a marker that depends on what type of point of interest, restaurant or airport we have. The same functionality is used when searching for all the other tools.

Finally, we can say that we have built-in reactivity through the Solara framework, as it updates the UI automatically ensuring a smooth user experience. State management is provided by Solara state hooks, allowing for shared state across components. Also, we have components that are loosely coupled via well-defined interfaces and callbacks, promoting reusability. For example the assistant always knows the conversation's history, so it would know any references of this, there, that and others. We can see an example in 3.2

ITSG 2023-2024

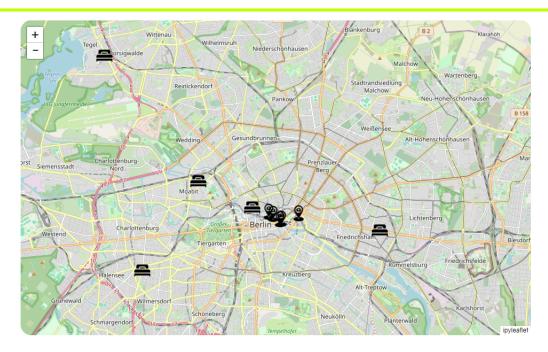


Figure 3.1: Tools interactivity in the map module



Berlin's heritage includes three World Heritage Sites: Museum Island, the Palaces and Parks of Potsdam and Berlin, and the Berlin Modernism Housing Estates. Other well-known landmarks include the Brandenburg Gate, Reichstag building, Potsdamer Platz, the Memorial to the Murdered Jews of Europe, and the Berlin Wall Memorial. The city is also rich in museums, galleries, and libraries.

What is the weather like there?

The current weather in Berlin is quite chilly with a temperature of 0.4°C. The humidity is high at 82%, and the wind is blowing at a brisk pace of 18.5 meters per second. Make sure to dress warmly if you plan to visit soon!

Yes I am planning to do that. What are the main point of interest?

- Berlin offers a variety of points of interest, reflecting its rich cultural and historical heritage. Some of the main attractions include:
 - 1. Berliner Dom (Berlin Cathedral) A grand historic church known for its impressive architecture.
 - 2. Pergamon Museum A world-famous museum that houses a vast collection of ancient artifacts.
 - 3. DDR Museum A museum dedicated to life in East Germany during the DDR era.
 - Alexanderplatz A large public square and transportation hub, also known for the iconic Fernsehturm (TV Tower).

What else would you want to know?

Figure 3.2: App knowing references from questions asked before

ITSG 2023-2024

3.3 Application Pipeline

The application pipeline can be seen in 3.3. It is composed of the app written in python and communicating with the solara framework in order to create the frontend components. After that there is an authentication step which was implemented using Autho. This pipeline was containerized as a Docker image and then deployed as a fly.io app. From there all the metrics go into a grafana board and all the logs are put in a monitoring tool incorporated into fly.io as it can be seen in figure 3.4. We are receiving logs for each response and each tool used. The Grafana dashboard receives metrics such as Memory and CPU utilization, app concurrency, latency, network i/o and the response counts and times from a Prometheus as a data source. We can see the grafana dashboard at figure 3.5

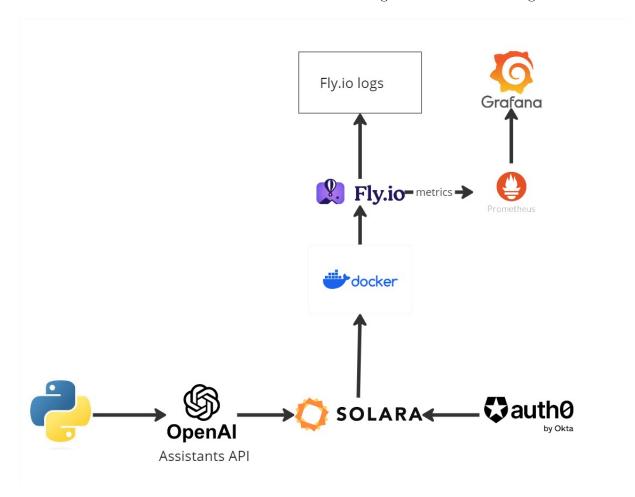


Figure 3.3: Application Pipeline

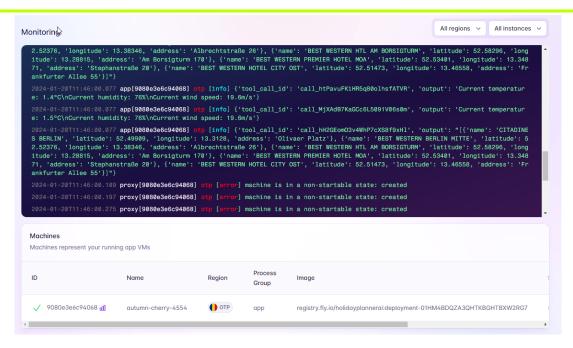


Figure 3.4: Application monitoring tool.

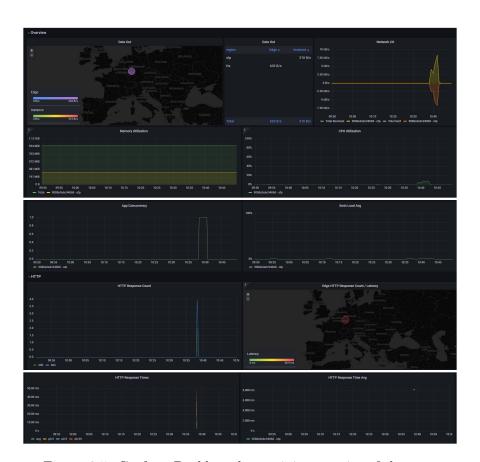


Figure 3.5: Grafana Dashboard containing metrics of the app.

ITSG 2023-2024

3.4 Application Code

We have not used any Machine Learning libraries for this application. The only one that could have Machine Learning affiliations is the OpenAI library. Other than that, we have used the Wikipedia API for information about the places to visit, the Open-Meteo API for near real-time weather information and we used the Amadeus API for hotel, airport and point-of-interest information. One Machine Learning library that will be developed is the LangSmith library which is a library developed by Langchain and that has the purpose of storing data coming from LLM's. Debugging was at first done using the VS Code debugger, but for the app it is done using sentry errors, which is a product from Sentry.io that is integrated within fly.io.

3.5 Model Deployment

We deployed our model as a docker container using fly.io. Fly is a cloud computing platform that provides hosting for applications with a focus on simplicity and speed. In order to deploy an app, all you have to do is to create a Dockerfile, and then run the following commands:

```
flyctl auth login
 fly launch
3 Scanning source code
 Detected a Dockerfile app
  Creating app in /Users/name/test/my-app-name
 We re about to launch your app on Fly.io. Here's what you're getting:
 Organization: MyOrg
                                         (fly launch defaults to the
     personal org)
 Name:
                                         (derived from your directory name)
                my-app-name
                                         (this is the fastest region for
10 Region:
                Secaucus, NJ (US)
     you)
n App Machines: shared-cpu-1x, 1GB RAM (most apps need about 1GB of RAM)
12 Postgres:
                <none>
                                         (not requested)
13 Redis:
                 <none>
                                         (not requested)
   Do you want to tweak these settings before proceeding? Yes
  // also we can create some secrets from the <code>CLI</code>
```

Laurentiu Stancioiu Nicolae Morosan

```
flyctl secrets set MY_SECRET=romance

Secrets are staged for the first deployment

fly deploy
```

Listing 3.3: Command line example of how to deploy a fly.io application

Our application is hosted on a shared CPU of 1GB of RAM. Another advantage of this product is the fact that you only pay for what you use and small apps are freely hosted. It also has features such as load balancers to keep our app up and running even if we have a lot of requests. The virtual machine that we use is located in Bucharest, Romania. That means that the app will run smoothly here, but if we get farther away, let's say Japan, it will have a small latency. This would happen for every cloud hosted application. Of course, after some time that our app will enter the market, we will know, based on the grafana dashboard, where the majority of our users come from and we can scale the app to have low latency in those areas accordingly.

Philosophical Aspects

The social impact on this application is expected to be high, when ready to ship into production and when it will have all the features we proposed to do. First of all, this will automate the difficult task of finding a holiday that will be suitable for each individual. We developed the app with a human centered approach in mind, to be as minimalistic and easy to use as possible. Furthermore, when we will develop the voice component, the accessibility part will also be enabled, especially for the blind people.

When the application scales, we will also make use of our data and use algorithms such as Direct Preference Optimization (DPO) [9] to fine-tune an open-sourced model in order to make a model that would be really good at function calling and answering queries about travelling. In this way we will also have a sense of control for the model used. All data is saved either in the grafana dashboard or in the fly.io logs, and we will monitor it and maybe also change some API's if we see that it gives unsatisfactory answers. Furthermore, we plan to make a like/dislike button, so the user can give us feedback if the answer was helpful or not.

Conclusions and future improvements

In conclusion, we can say that we are slowly moving towards a Travel Generalized Assistant (TGA) that can make your dream holiday come true or make a nice experience for a business one. We will continue to work towards realizing other features that would make our app shine even more in contrast to others such as accessibility for blind people, feedback from users, multimodal interface, so the user can also send images from a location and the application would give information about it if its a historical place or reviews if its a restaurant. Because the LLM space is moving at an extremely high pace, we can say for now, that for personal assistants, sky is the limit.

Bibliography

- [1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017.
- [2] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.
- [3] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timo-thée Lacroix, Baptiste RoziÃ"re, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.
- [4] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.
- [5] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023.
- [6] 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, 2021.
- [7] Function calling and other api updates. https://openai.com/blog/function-calling-and-other-api-updates.

- [8] New models and developer products announced at devday. https://openai.com/blog/new-models-and-developer-products-announced-at-devday.
- [9] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2023.