Boğaziçi University : Software Engineering Spring'23

SWE599 – Cloud Application

Hotel Recommendation System with Machine Learning and Large Language Models & OpenAI

Instructor: Atay Özgövde

Student: Damla Alkan
damlaalkan@gmail.com

Table of Contents:

1. Introduction
2. System and Software
3. Software Architecture
4. Summary

1. Introduction

In today's era of technology, Artificial Intelligence , Machine Learning and Large Language models have become buzzwords in every industry. The hospitality industry has not been left behind, as it seeks to leverage these technologies to provide personalized and seamless experiences for customers. In line with this, the “Hotel Recommendation System with Machine Learning and OpenAI” project aims to develop an application that will revolutionize how travelers plan their trips by recommending the best hotels and travel plans based on their preferences.

Planning a trip can be a really challenge, especially when you are unfamiliar with a particular location. It involves a lot of research to find the best hotels, activities, and travel plans that match your preferences and budget. This project aims to solve this problem by developing a recommendation system that leverages with artificial intelligence to suggest the best hotels and travel plans based on the user's input. This will save travelers time and effort, while also providing a personalized experience that meets their needs. Additionally, this project can benefit the hospitality industry by providing insights into customer preferences, which can be used to improve their services and offers.

In summary, the future of the Hotel Recommendation System with Machine Learning and Large Language Models such as OpenAI project is promising. As technology continues to evolve, the possibilities for enhancing the application are endless. With continued improvements and advancements, the application can become a game-changer in the hospitality industry.

2. System and Software

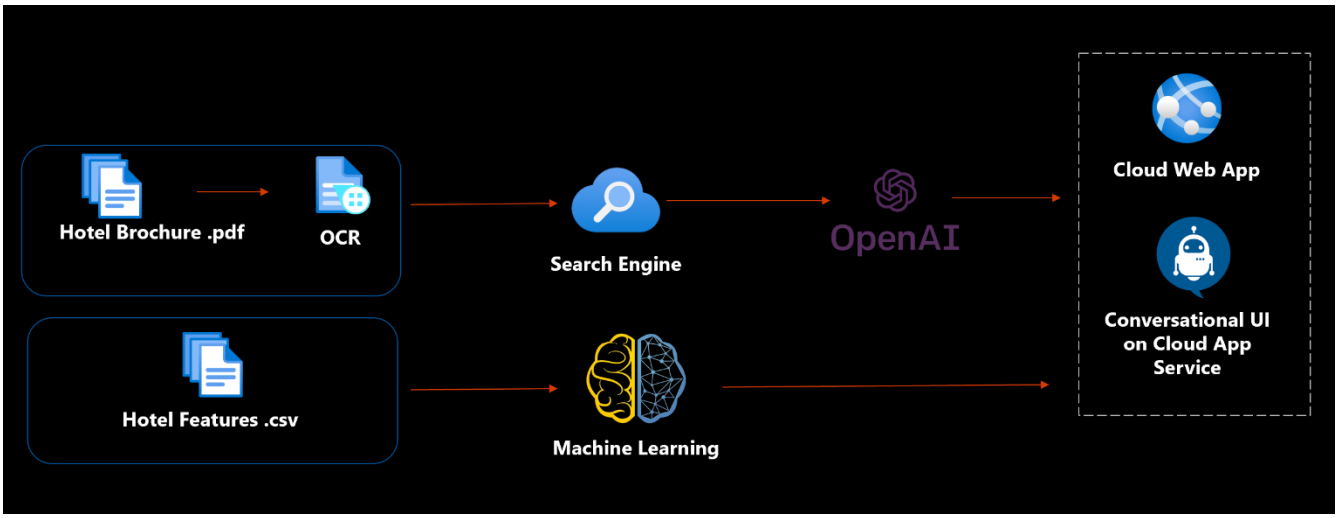
This project is an innovative application that combines cutting-edge technologies to provide users with personalized recommendations for their travel plans. The system consists of several components, including the front-end website interface with Conversational UI, back-end servers on cloud, Machine Learning models and OpenAI Rest APIs.

The front-end interface is developed using .Net and Bot Frameworks which provides users with an easy-to-use platform for inputting their travel preferences. The interface is designed to be intuitive and user-friendly, with clear and concise instructions that guide users through the process of providing their information including search and filter options.

The back-end servers on cloud are responsible for processing the user input and generating personalized recommendations and hosting search engine. These cloud applications run Machine Learning models serverless and Search Engine hosted as a platform service on cloud.

To take the Hotel Recommendation System to the next level, it has integrated OpenAI's large language models. OpenAI's GPT-3 language model is an advanced tool for natural language processing that can generate human-like text based on the input provided to it. This enables it to extract keywords and generate personalized descriptions of the recommended hotels and travel plans. This feature enhances the user’s experience by providing detailed and engaging descriptions of the recommended options.

3. High Level Architecture



The designed system for the “Hotel Recommendation” project includes a hotel database with a search engine and a prompt engineering module. The hotel database with hotel brochures as pdf files is a collection of hotels with various attributes such as location, activities, restaurants, accessibility, pet availability and several needed information. The public hotel features data includes several booking.com hotels with locations and reviews. The search engine allows users to search for hotels based on specific attributes, such as the location they want to stay in or the amenities they desire.

The prompt engineering on OpenAI text-davinci-003 model is designed to extract key information from users input about their dream travel plan. This module utilizes natural language processing techniques to analyze user input and identify relevant entities such as location, activity, and travel duration. The module then generates a prompt that helps guide the user to provide more specific details about their travel plan.

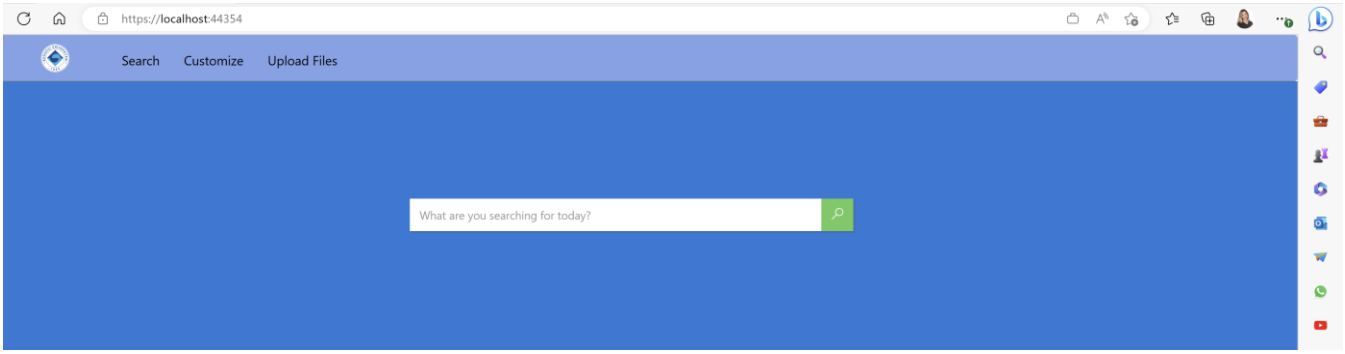
Together, these components provide a solid foundation for the Hotel Recommendation System. The hotel database and search engine enable the system to provide users with a wide range of hotel options that match their preferences, while the prompt engineering module helps to ensure that the system receives accurate and detailed input from the user. By leveraging these components on cloud web app ,the system can generate personalized recommendations that meet the user's needs and preferences, making it an effective tool for planning a dream trip in a user friendly interface.

A. Search Engine

For test purpose, hotel brochures data is selected from a Turkish Travel agency Setur, which provides public brochures for Bodrum location. The data may find in the link https://cdn2.setur.com.tr/content/e-kataloglar/Pdf/setur_ege_2022.pdf .

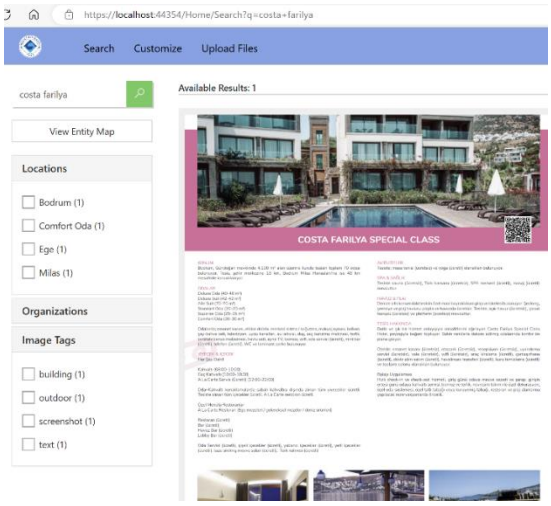
Firstly, the hotel brochures would need to be uploaded and stored in a data source, such as Azure Blob Storage which is a cloud-based file system. Then, Azure Cognitive Search needs to be configured to index the data source, which involves defining the schema for the data and specifying which fields should be searchable and filterable.

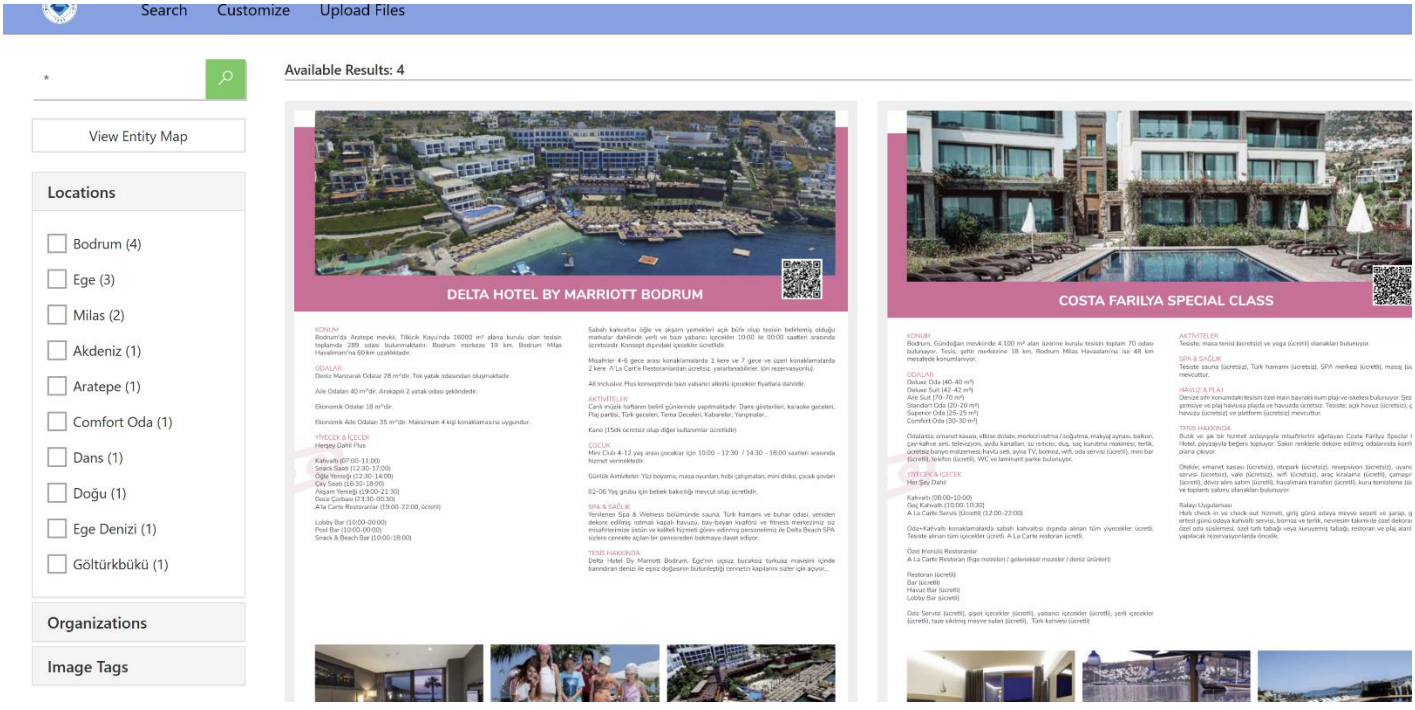
Once the index is set up, the search engine website can be built using Azure Cognitive Search's .NET SDK. The website would include a search box where users can enter keywords related to the hotels, they are interested in. The search engine would then use the index to retrieve a list of hotels that match the search query.



To enable users to filter the results by attributes, such as location, the website would need to include a filtering interface. This could be achieved by exposing filterable attributes as facets, which would enable users to select one or more values from a list of options. When a user selects a faceable value, the search engine refines the search results to only show hotels that match the selected attribute.

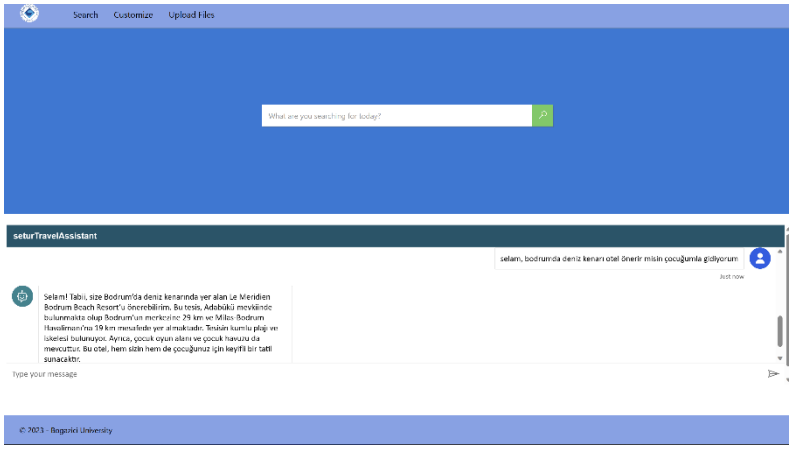
To enable users to learn more about a hotel, the search engine website could display a brochure or summary of the hotel's amenities, location, and other relevant information. This information could be pulled from the data source and displayed alongside the search results.



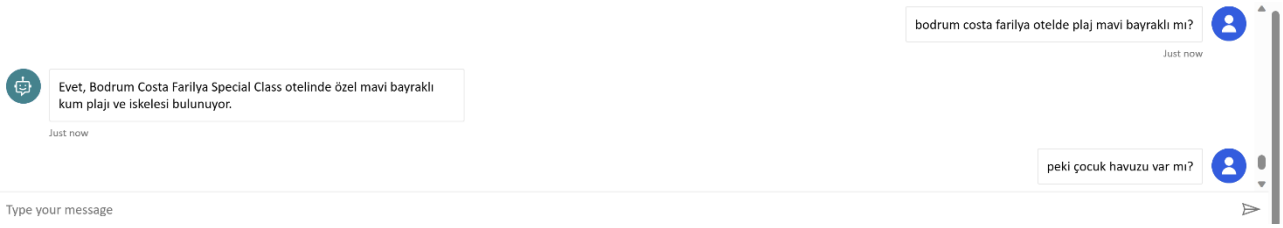


B. Conversational UI with OpenAI API

In this sample website app, the site includes a search engine and a conversational UI agent as a chatbot powered by OpenAI. This conversational UI hosted also in cloud, Azure Bot App Services and provides an interface to chat with assistant. An Assistant is designed also with Search Engine, so that when a question comes from user, the input requests to Search Engine first with REST API , then the output from search engine sends a request to OpenAI with a correct prompt , and it returns an answer to user.

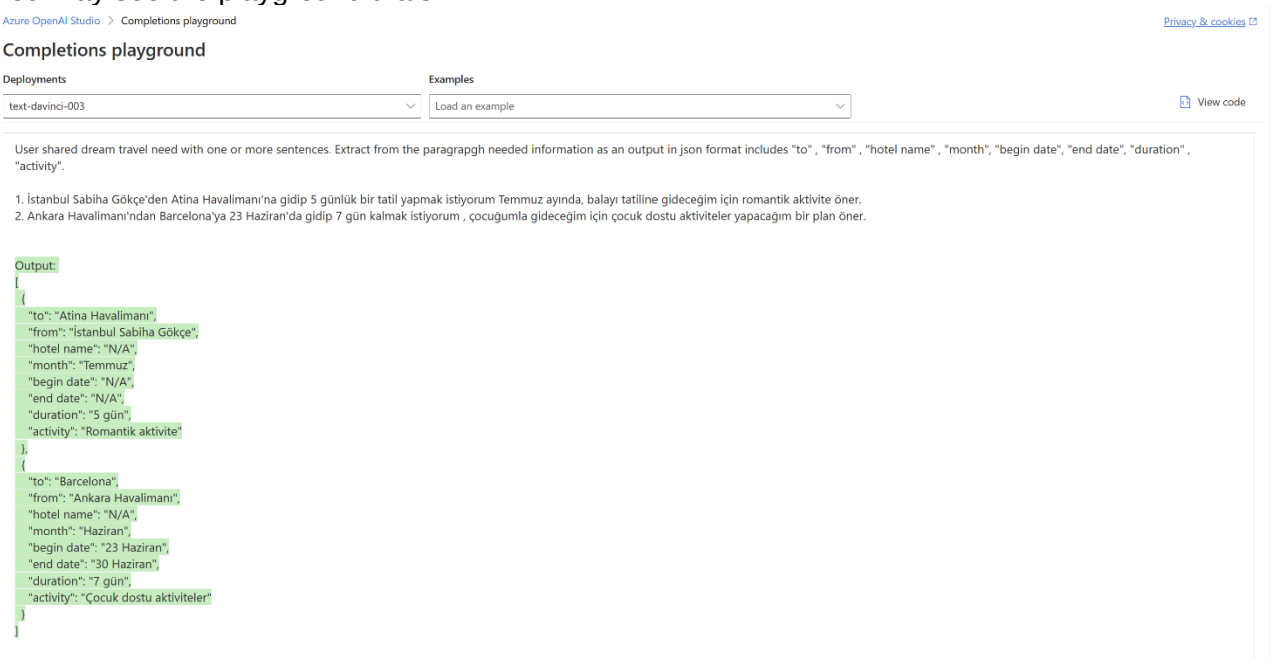


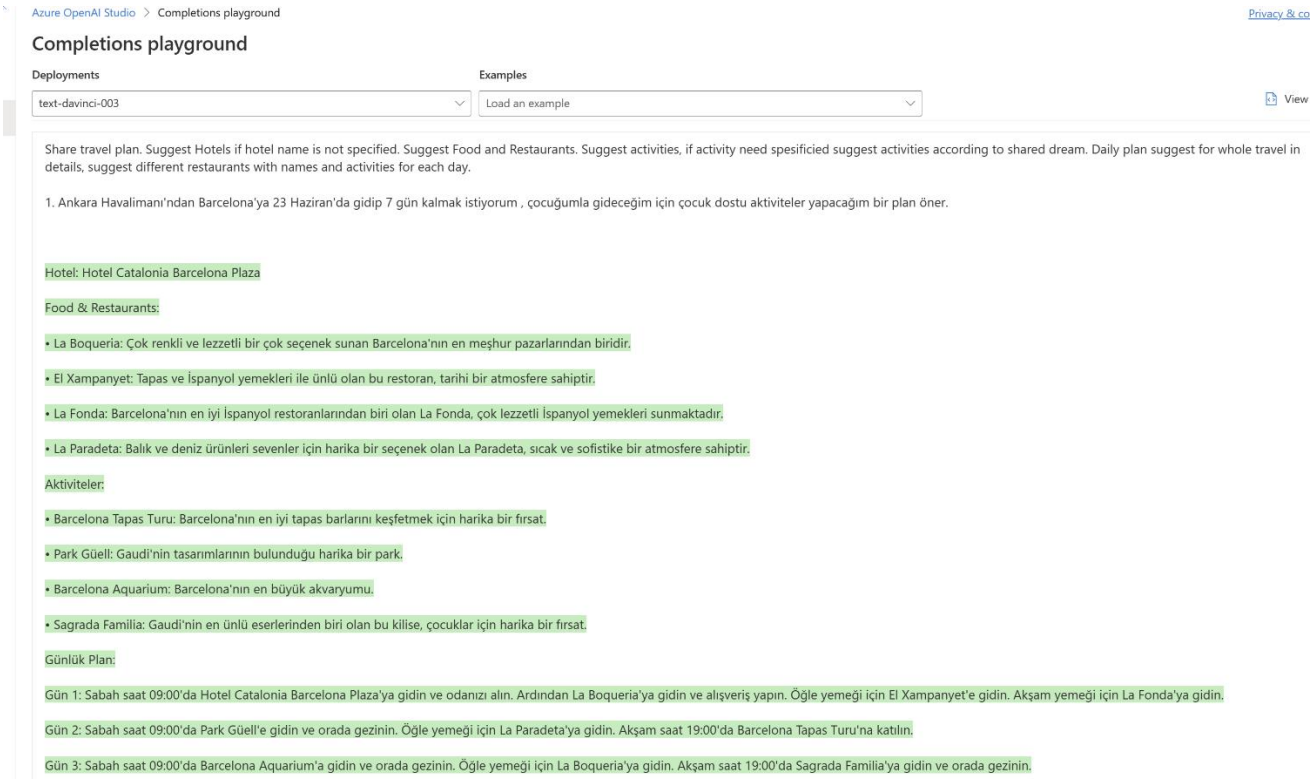
Since the total database has currently includes only 4 hotel, the user may ask about this 4 hotels, compare hotel, recommendation about these hotels and asking about detail question such as “Is there a SPA?” or “Is the hotel accessible?” or “Does hotel allow pet?”.



C. OpenAI Prompt Engineering

For OpenAI API, Azure OpenAI has used since it is same API with OpenAI but it invoices from Azure credits and it allows to secure the endpoint. The text-davinci-003 selected since this model provides more strong generative ai outputs for generic topics such as hospitality. The prompt has designed for 2 different aim , one of them is to extract needed key-values and the other is suggesting hotel, restaurant and daily activity plan. You may see the playground trials:





Sample Script to request OpenAI text-davinci-003 model:

```
#Note: The openai-python library support for Azure OpenAI is in preview.
import os
import openai
openai.api_type = "azure"
openai.api_base = "https://demo-open-ai.openai.azure.com/"
openai.api_version = "2022-12-01"
openai.api_key = os.getenv("OPENAI_API_KEY")

response = openai.Completion.create(
  engine="text-davinci-003",
  prompt="User shared dream travel need with one or more sentences. Extract from the paragrapgh needed information as an output in json format includes \"to\" , \"from\" , \"hotel name\" , \"month\", \"begin date\", \"end date\", \"duration\" , \"activity\".\n\n1. İstanbul Sabiha Gökçe'den Atina Havalimanı'na gidip 5 günlük bir tatil yapmak istiyorum Temmuz ayında, balayı tatiline gideceğim için romantik aktivite öner.\n2. Ankara Havalimanı'ndan Barcelona'ya 23 Haziran'da gidip 7 gün kalmak istiyorum , çocuğumla gideceğim için çocuk dostu aktiviteler yapacağım bir plan öner.\n\n",
  temperature=1,
  max_tokens=2000,
  top_p=0.5,
  frequency_penalty=0,
  presence_penalty=0,
  best_of=1,
  stop=None)
```

Sample key-value extraction prompt:
User shared dream travel need with one or more sentences. Extract from the paragrapgh needed information as an output in json format includes "to" , "from" , "hotel name" , "month", "begin date", "end date", "duration" , "activity".

Sample suggestion prompt:
Share travel plan. Suggest Hotels if hotel name is not specified. Suggest Food and Restaurants. Suggest activities, if activity need spesificied suggest activities according to shared dream. Daily plan suggest for whole travel in details, suggest different restaurants with names and activities for each day.

Sample user dream travel prompt:
Ankara Havalimanı'ndan Barcelona'ya 23 Haziran'da gidip 7 gün kalmak istiyorum , çocuğumla gideceğim için çocuk dostu aktiviteler yapacağım bir plan öner.

Result of davinci-003 model in Jupiter Notebook:

In [6]:
```
#Note: The openai-python library support for Azure OpenAI is in preview.
import os
import openai
openai.api_type = "azure"
openai.api_base = "https://demo-open-ai.openai.azure.com/"
openai.api_version = "2022-12-01"
openai.api_key = [redacted]

response = openai.Completion.create(
  engine="text-davinci-003",
  prompt="User shared dream travel need with one or more sentences. Extract from the paragrapgh needed information as an outp
  temperature=1,
  max_tokens=2000,
  top_p=0.5,
  frequency_penalty=0,
  presence_penalty=0,
  best_of=1,
  stop=None)
```

In [7]:
```
print(response)
```


```
{
  "choices": [
    {
      "finish_reason": "stop",
      "index": 0,
      "logprobs": null,
      "text": "\nOutput: \n[\n  {\n    \"to\": \"Atina Havaliman\u0131\",\n    \"from\": \"\u0130stanbul Sabiha G\u00f6k\u00
e7e\",\n    \"hotel name\": \"\",\n    \"month\": \"Temmuz\",\n    \"begin date\": \"\",\n    \"end date\": \"\",\n    \"dur
ation\": \"5 g\u00fcn\",\n    \"activity\": \"Romantik aktivite\"\n  },\n  {\n    \"to\": \"Barcelona\",\n    \"from\": \"An
kara Havaliman\u0131\",\n    \"hotel name\": \"\",\n    \"month\": \"\",\n    \"begin date\": \"23 Haziran\",\n    \"end dat
e\": \"\",\n    \"duration\": \"7 g\u00fcn\",\n    \"activity\": \"\u00c7ocuk dostu aktiviteler\"\n  }\n]"
    }
  ],
  "created": 1681671457,
  "id": "cmpl-761r7n1m4AoY1vQsExM92sE1MvSNn",
  "model": "text-davinci-003",
  "object": "text_completion",
  "usage": {
    "completion_tokens": 200,
    "prompt_tokens": 213,
    "total_tokens": 413
  }
}
```

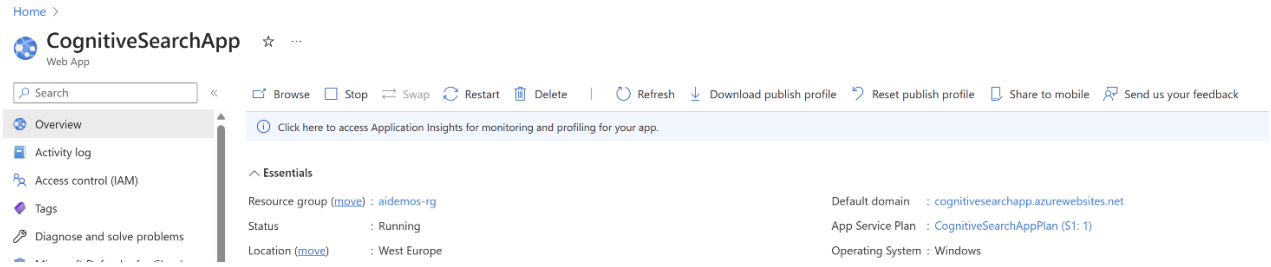

D. Deploy App to Cloud

You may reach the website in the link : Home (cognitivesearchapp.azurewebsites.net)

The website deployed to Azure App Service as a containerized application in .Net. There are many advantages of deploying an application to cloud such as Azure App Service can be explained in the below:

1. Scalability: Azure App Services can easily scale up or down to accommodate changes in traffic and user demand, allowing the application to be highly available and responsive.
2. Reliability: Azure App Services provides high availability and disaster recovery capabilities, ensuring that the application is always accessible, and that data is protected.
3. Easy deployment: Azure App Services allows for easy deployment of .NET applications from a variety of sources, including source control, Docker containers, and other tools.

4. Integration with other Azure services: Azure App Services can easily integrate with other Azure services, such as Azure SQL Database, Azure Cosmos DB, and Azure Blob Storage, providing a seamless and integrated solution.
5. Security: Azure App Services provides robust security features, including identity and access management, network security, and data protection, ensuring that the application and its data are secure.



E. Custom Knowledge Base With OpenAI: Embeddings

Embeddings are numerical representations of words or phrases that capture the semantic meaning and relationships between them. In the context of NLP, embeddings are generated by training deep neural networks on large text corpora. These networks learn to map words or phrases into a continuous vector space, where similar words are placed closer together, and dissimilar words are farther apart. This ability to encode semantic relationships allows the model to capture context and meaning, enabling more nuanced analysis and query understanding.

Ada, a variant of OpenAI's GPT-3.5, is a powerful language model that incorporates the use of embeddings. It is trained on an extensive dataset comprising a wide range of texts from the internet. Ada has been fine-tuned to perform specific tasks and can be utilized for various applications, including question-answering systems.

Once the hotel data is transformed into embeddings, we can perform various operations, such as similarity search and question answering, with respect to the encoded information. For example, we can compare the embeddings of different hotels to find the most similar ones based on their descriptions. This allows users to discover hotels that share similar attributes or appeal to similar preferences. Furthermore, by combining embeddings with a custom database, we can build a question-answering system that understands and responds to natural language queries about hotels. Users can ask questions like, "What are the top-rated hotels with swimming pools in Istanbul?" or "Which hotels near Taksim have complimentary breakfast?" The Ada model, with its fine-tuned embeddings, can interpret these queries, match them with the appropriate hotel embeddings, and provide relevant answers. Utilizing embeddings with OpenAI's Ada model offers several advantages. Firstly, it allows for semantic understanding and comparison of hotel data, enabling more sophisticated querying capabilities. Secondly, the embeddings capture nuanced relationships between hotels, providing valuable insights for recommendation systems. Finally, the Ada model's flexibility and adaptability allow developers to fine-tune it further for specific tasks or domains.

In this project, we have used a hotel data which includes bad reviews and good reviews for more than 50.000 hotels to ask custom questions about hotels such as “Which hotel has best breakfast and good location?”.

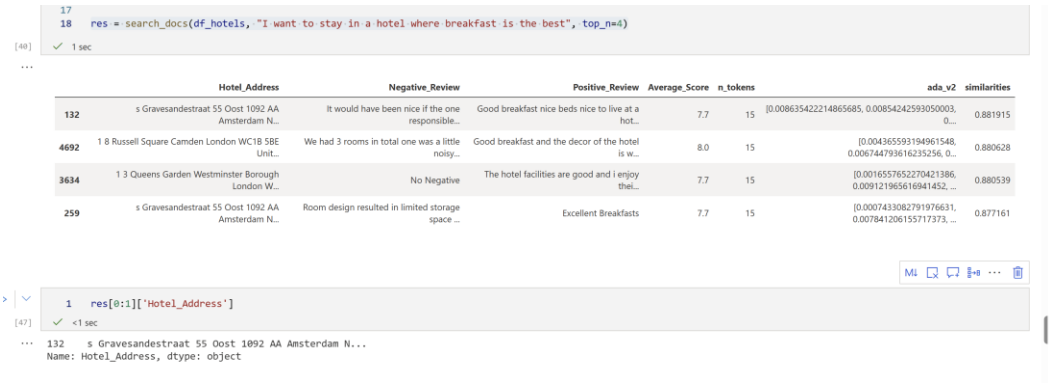
However, it is important to consider potential limitations. Embeddings rely heavily on training data, so if the hotel dataset used for training is limited or biased, the embeddings may not accurately represent all aspects of hotels. Additionally, embeddings may not capture extremely nuanced information or domain-specific knowledge that is not present in the training data. In this project, since we have limited access to OpenAI service with same request per second limitations, it gives error However, it is important to consider potential limitations. Embeddings rely heavily on the training data, so if the hotel dataset used for training is limited or biased, the embeddings may not accurately represent all aspects of hotels. Additionally, embeddings may not capture extremely nuanced information or domain-specific knowledge that is not present in the training data.

r for bigger databases. I completed 100 rows and it runs in 2 minutes and then I tried 500 and it lasts around

10 minutes

In this project , for 1000 random positive and negative review has used for this embedding and we may ask custom questions for hotels.

If we ask “I want to stay in a hotel where breakfast is the best”, the embedding scores is sorted with top 4 results and top score provides the Hotel Name and Address. The embedding model collects data from positive reviews and provides the output result for this question.



```
res = search_docs(df_hotels, "I want to stay in a hotel where breakfast is the best", top_n=4)
```

1 sec

	Hotel_Address	Negative_Review	Positive_Review	Average_Score	n_tokens	ada_v2	similarities
132	s Gravesandestraat 55 Oost 1092 AA Amsterdam N...	It would have been nice if the one responsible...	Good breakfast nice beds nice to live at a hot...	7.7	15	[0.008635422214865685, 0.0085424259305000 3, 0...	0.881915
4692	1 8 Russell Square Camden London WC1B 5BE Unit...	We had 3 rooms in total one was a little noisy...	Good breakfast and the decor of the hotel is w...	8.0	15	[0.004365593194961548, 0.006744793616235256, 0...	0.880628
3634	1 3 Queens Garden Westminster Borough London W...	No Negative	The hotel facilities are good and i enjoy thei...	7.7	15	[0.001655765227042 1386, 0.009121965616941452, ...	0.880539
259	s Gravesandestraat 55 Oost 1092 AA Amsterdam N...	Room design resulted in limited storage space ...	Excellent Breakfasts	7.7	15	[0.007433082791976631, 0.007841206155717373, ...	0.877161

```
res[0:1]['Hotel_Address']
```

<1 sec

```
132    s Gravesandestraat 55 Oost 1092 AA Amsterdam N...
Name: Hotel_Address, dtype: object
```

For breakfast and location question , the top 4 score had listed in below:

```
res = search_docs(df_hotels, "I want to stay in a hotel where breakfast is the best and the location is good", top_n=4)
```

2 sec

	Hotel_Address	Negative_Review	Positive_Review	Average_Score	n_tokens	ada_v2	similarities
4306	1 8 Russell Square Camden London WC1B 5BE Unit...	Overpriced bad nights sleep due to noise from ...	Nice breakfast good location	8.0	15	[0.003134180558845401, 0.005315182730555534, 0...	0.911084
2733	1 3 Queens Garden Westminster Borough London W...	got a basement room with no natural light Tiny...	good location good breakfast	7.7	15	[-0.005336749367415905, 0.007927946746349335, ...	0.910969
3665	1 3 Queens Garden Westminster Borough London W...	Bed matress was a little too hard	Good location great breakfast	7.7	15	[-0.003040354698896408, 0.01167367771267891, 0...	0.908914
477	1 15 Templeton Place Earl s Court Kensington a...	Two years in a row we checked in for 10 days O...	Location location location and breakfast	8.5	20	[-0.005637835245579481, -0.0015237834304571152...	0.908499

For children activity question you may see the result:

```
res = search_docs(df_hotels, "I will have a vacation with my 2 children and would like to have good activites for my children", top_n=4)
```

2 sec

	Hotel_Address	Negative_Review	Positive_Review	Average_Score	n_tokens	ada_v2	similarities
1453	1 2 Serjeant s Inn Fleet Street City of London...	nothing	Staff went out of their way to make my daughte...	9.2	22	[0.001751572941429913, 0.0017613766249269247, ...	0.841167
3652	1 3 Queens Garden Westminster Borough London W...	Nil	Good location Excellent staff Good fascilities...	7.7	15	[-0.005229237023741007, 0.033077966421842575, ...	0.834441
2370	1 3 Queens Garden Westminster Borough London W...	The mattress could have been better didn t hav...	I had a very good stay here with my young chil...	7.7	15	[0.00628528231754899, 0.03105131722986698, -0....	0.831846
1933	1 2 Serjeant s Inn Fleet Street City of London...	None	Perfect location perfect weekend away	9.2	22	[-0.005650183651596308, 0.018797416239976883, ...	0.817381

4. Summary

The term project involved multiple components focused on hotel data and analysis. Firstly, public data was gathered, including information on hotels, locations, reviews, and hotel details. This dataset served as the foundation for subsequent analysis.

Next, the hotel data was thoroughly analyzed to gain valuable insights. This analysis likely involved examining

various aspects such as customer reviews, ratings, amenities, and geographical factors. By delving into the data, meaningful patterns and trends could be identified, enabling a deeper understanding of the hotel industry. To enhance user experience and facilitate efficient searching, a search engine was developed. This search engine integrated multiple hotel datasets, allowing users to filter and sort hotels based on activity details and location preferences. This feature provided users with a tailored and personalized search experience, catering to their specific needs and preferences.

To further assist users, a chatbot was implemented. This chatbot was designed to handle user inquiries and answer questions related to hotels within the database. Users could engage in conversations with the chatbot, seeking information about various aspects of hotels, including amenities, availability, pricing, and more. Additionally, a prompt was built to detect entities from user inputs. The prompt was specifically designed to identify relevant information such as locations, travel dates (including start and end dates), and duration. By capturing these details, the system could generate personalized travel plans based on the user's preferences and requirements.

Lastly, the project involved utilizing OpenAI for a custom database with an embeddings model. This allowed for efficient querying of the custom database by employing embedding techniques to represent and index the data effectively. The system was trained to comprehend and respond to questions related to the custom database, providing users with accurate and relevant information.

In summary, the term project encompassed acquiring and analyzing public hotel data, building a search engine and chatbot for hotel inquiries, developing prompt for entity detection and travel recommendations, and leveraging OpenAI for a custom database with question-answering capabilities. These components collectively aim to enhance the understanding and accessibility of hotel information, ultimately providing users with an improved hotel search and recommendation experience.

Links

1. You may find all scripts in Github Repo : damlaalkan/599: Hotel Recommendation System with Machine Learning and Large Language Models & OpenAI (github.com)
2. Public Website includes Search and Chatbot : Home (cognitivesearchapp.azurewebsites.net)
3. Custom Database with OpenAI Script : https://github.com/damlaalkan/599/blob/main/OpenAIEmbedding-Final.ipynb