Travel Guide Chatbot Report

Cady Baltz, Fanny Dolisy
cmb180010, fcd180001

Overview

This chatbot is designed to provide travel information for users who are interested in visiting different countries. As a travel agent, it offers invaluable assistance in understanding a country before traveling there. For instance, if a user is planning a trip to Canada and needs to know the languages spoken or the best mode of transportation, the chatbot can provide answers to their questions. To achieve this, the chatbot utilizes a range of natural language processing techniques, coupled with Google's Dialogflow framework. These techniques include named entity recognition, web scraping, part-of-speech tagging, and term frequency analysis. This report will delve deeper into each of these techniques. In addition, the chatbot also tracks user information, such as the countries the user has inquired about, the information they have been most interested in, and any dislikes they may have. With this combination of NLP, DialogFlow, and user tracking, the travel agent is fully equipped to answer all international inquiries in a conversational way.

Dialogflow

The main feature we used is called an “intent”. Intents are user or system-defined labels that essentially can explain the purpose of a user’s sentence, as well as extract parameters from the user input that are relevant to that intent. For example, if I defined an intent called “dislike” which implies the user dislikes something, the following sentence - “ I dislike cows' ' would be labeled with “dislike” and cows would be returned as the object that is disliked.

To create an intent, we were able to enter in various training phrases into the Dialogflow console, applying reinforcement learning until we felt that our agent was able to detect a sufficient variety of phrases. By leveraging the power of Google’s pre-built ML and NLP models, we were able to easily create an agent that could understand even vague and misspelled input from the user. During the intent creation, we also indicated exactly which parameters to extract and return, as well as what default responses we wanted to provide for each intent.

In our application, we defined a total of fifteen intents based on the information we extracted for our knowledge base (see the “Architecture” section for more details). These intents are: "Regions," "Cities," "Other Destinations," "Get In," "See," "Do," "Talk," "Buy," "Eat," "Drink," "Sleep," "Stay Healthy," "Stay Safe," "Connect," and "Respect". For example, the sentence “What should I do when I visit”, would trigger the “Do” intent and the sentence “ What’s the cell service like?” would trigger the “Get around” intent.

Now that we have our intentions, what do we do with them? Our very first step was writing basic “fulfillment text” that were handwritten sentences that simply filled in the blank with the country context. Our next step was to go into our knowledge base for the current country and return the document with the same name to be used for further processing. Dialogflow does have further features that allow it to "answer" on its own by extracting a sentence that in some cases works rather well. However, we decided to do our own NLP processing to further find the best answer for the question asked. This processing is done in separate Python callback functions that we tailored to each intent.

Dialog Logic

The logic for our dialog is primarily handled through the use of the Dialogflow intents described above, and is further refined through processing steps in our Python code.

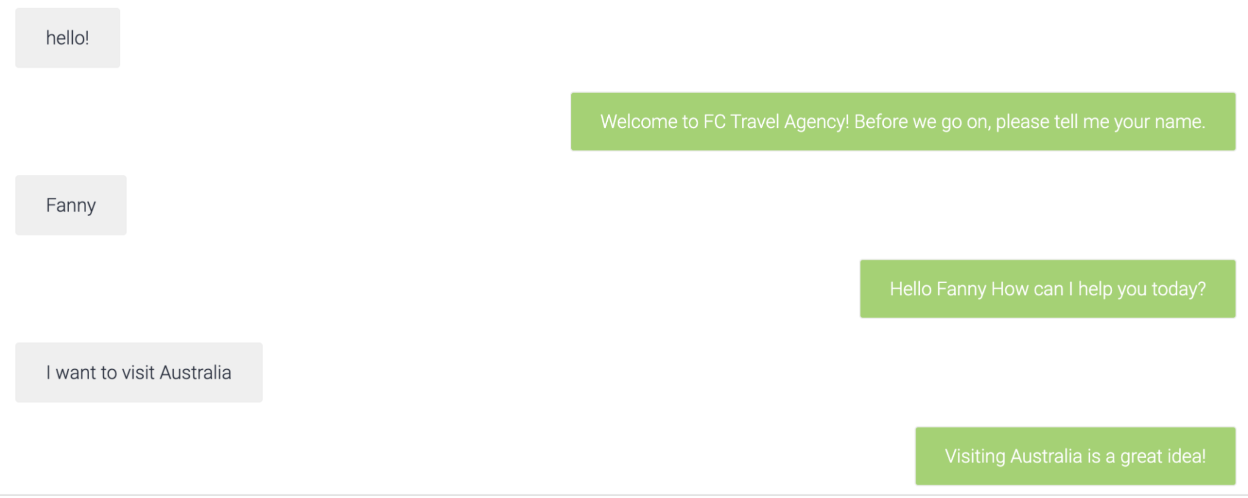
First, we designed a Welcome intent, which is triggered when the user first starts the program. The Welcome intent requires that a user inputs their name before continuing. Then, we set the country context that the user is interested in visiting. For a new user, we defined another DialogFlow intent called Country which can pick up any country names. If a user already exists, we call this intent ourselves in the code to set the context. At any point in the program execution, the Country intent can be triggered again, allowing the user to ask questions about a new country.

Additionally, we defined sixteen intents based on the documents stored in our knowledge base. These intents are: Regions, Cities, Other_destinations, Get_in, Get_around, See, Do, Talk, Buy, Eat, Drink, Stay_healthy, Stay_safe, Connect, and Respect. In our code, we have written a method for each of these intents defining how we respond to them. For example, in our "Eat" method, we parse the DialogFlow response for food terms to respond with.

Finally, we have a Dislike intent, allowing the user to indicate items they do not like and do not want to be recommended. To end the program execution, we use a Clsoe intent.

Sample Dialog Interactions

Welcome a new user



hello!

Welcome to FC Travel Agency! Before we go on, please tell me your name.

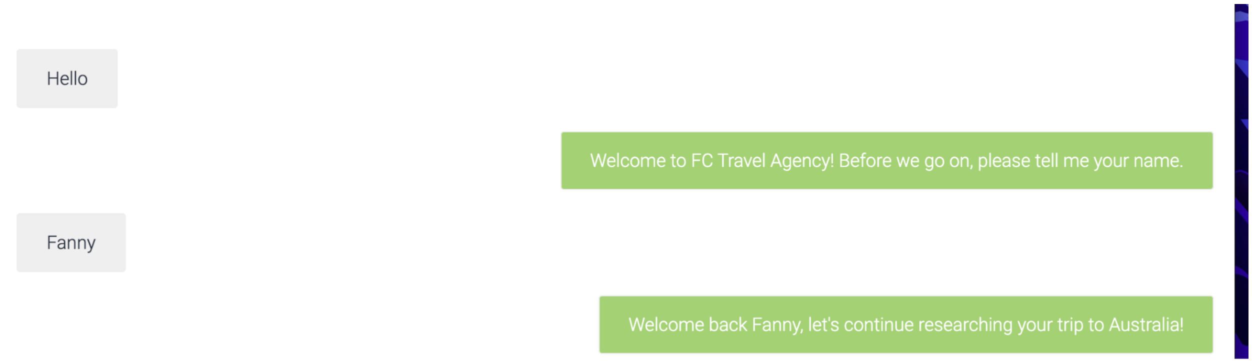
Fanny

Hello Fanny How can I help you today?

I want to visit Australia

Visiting Australia is a great idea!

Welcome an existing user



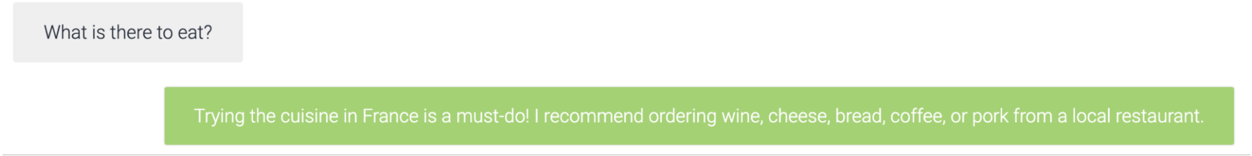
Hello

Welcome to FC Travel Agency! Before we go on, please tell me your name.

Fanny

Welcome back Fanny, let's continue researching your trip to Australia!

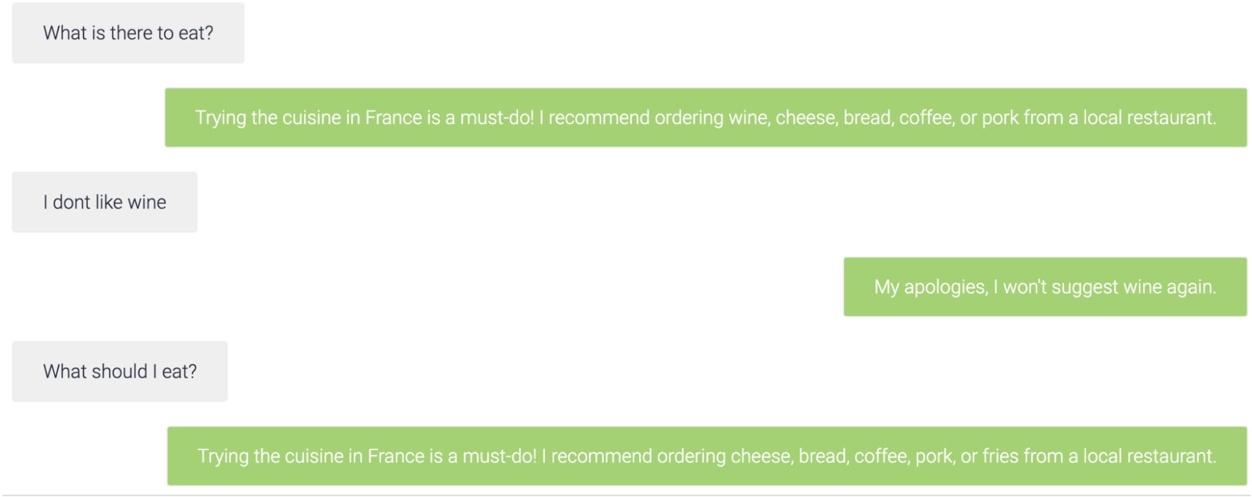
Eat - France



What is there to eat?

Trying the cuisine in France is a must-do! I recommend ordering wine, cheese, bread, coffee, or pork from a local restaurant.

Dislike



What is there to eat?

Trying the cuisine in France is a must-do! I recommend ordering wine, cheese, bread, coffee, or pork from a local restaurant.

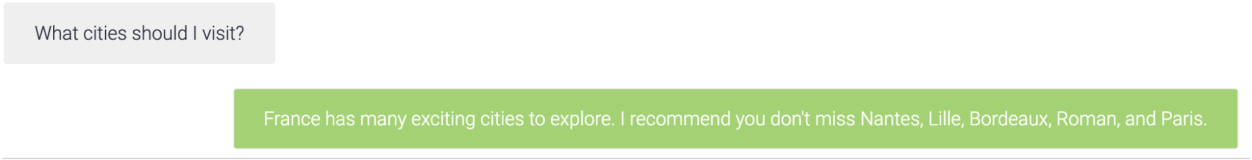
I dont like wine

My apologies, I won't suggest wine again.

What should I eat?

Trying the cuisine in France is a must-do! I recommend ordering cheese, bread, coffee, pork, or fries from a local restaurant.

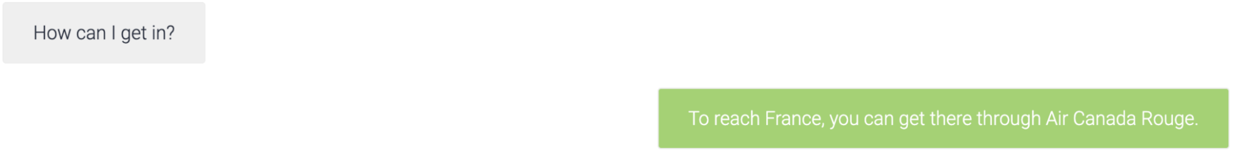
Cities - France



What cities should I visit?

France has many exciting cities to explore. I recommend you don't miss Nantes, Lille, Bordeaux, Roman, and Paris.

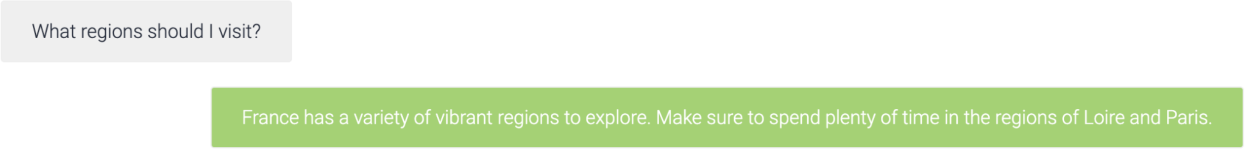
Get_In - France



How can I get in?

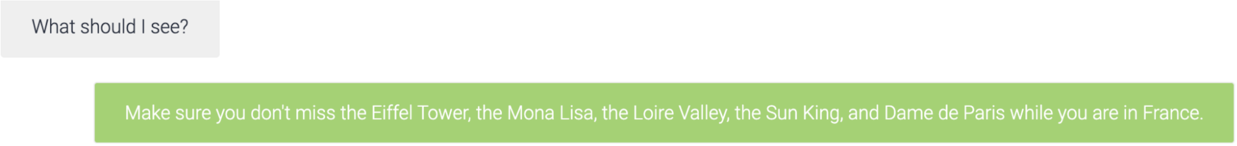
To reach France, you can get there through Air Canada Rouge.

Regions- France


What regions should I visit?

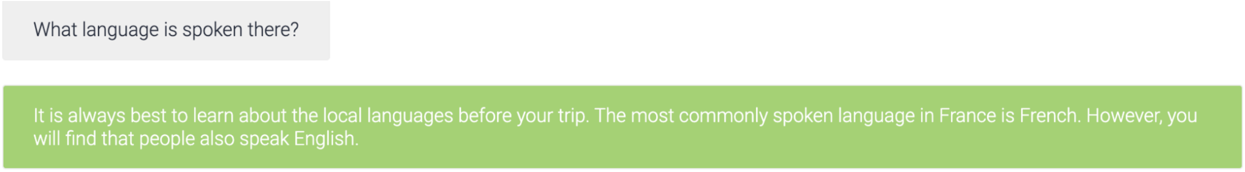
France has a variety of vibrant regions to explore. Make sure to spend plenty of time in the regions of Loire and Paris.

See - France


What should I see?

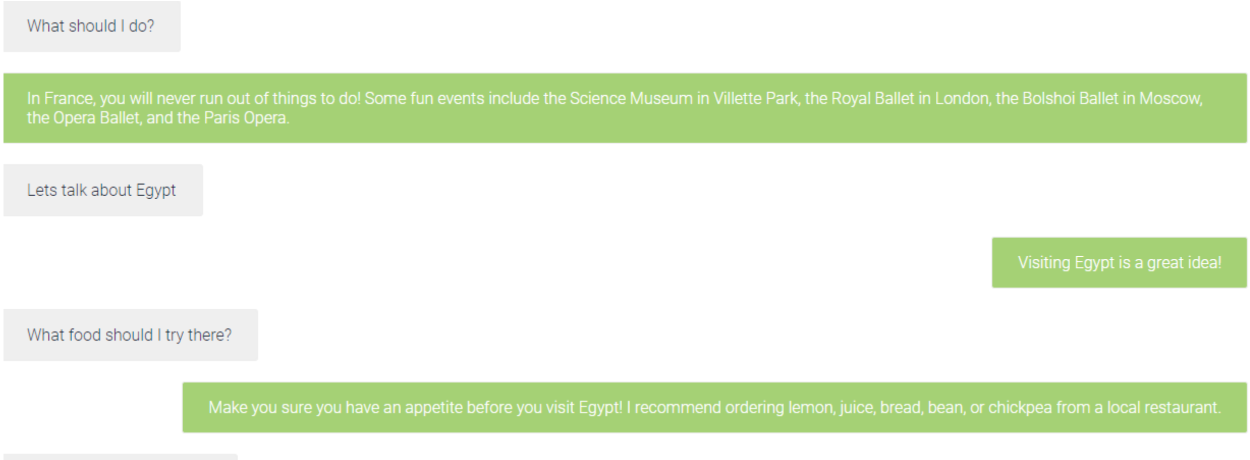
Make sure you don't miss the Eiffel Tower, the Mona Lisa, the Loire Valley, the Sun King, and Dame de Paris while you are in France.

Talk - France


What language is spoken there?

It is always best to learn about the local languages before your trip. The most commonly spoken language in France is French. However, you will find that people also speak English.

Switch countries - France to Egypt


What should I do?

In France, you will never run out of things to do! Some fun events include the Science Museum in Villette Park, the Royal Ballet in London, the Bolshoi Ballet in Moscow, the Opera Ballet, and the Paris Opera.

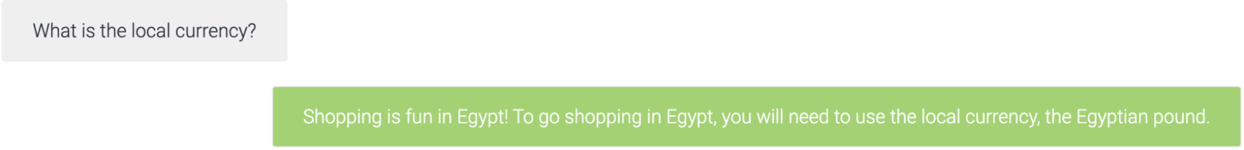
Lets talk about Egypt

Visiting Egypt is a great idea!

What food should I try there?

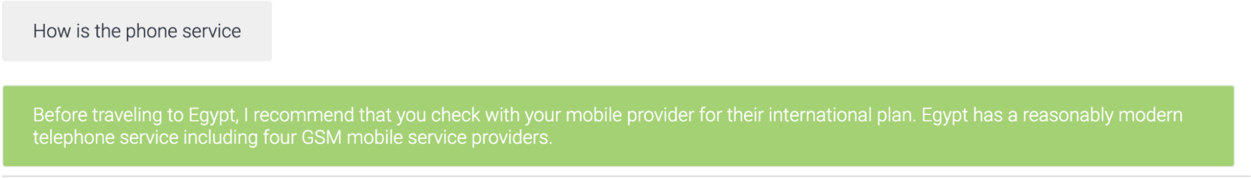
Make you sure you have an appetite before you visit Egypt! I recommend ordering lemon, juice, bread, bean, or chickpea from a local restaurant.

Buy - Egypt


What is the local currency?

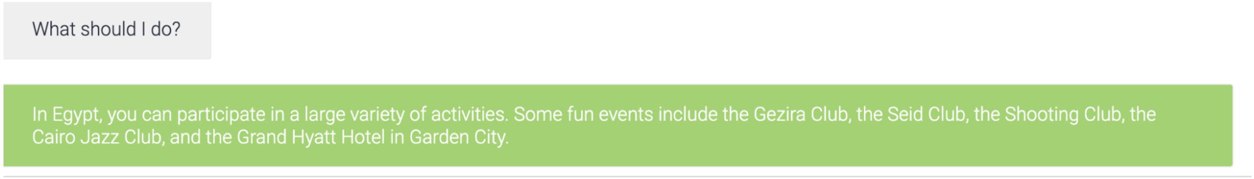
Shopping is fun in Egypt! To go shopping in Egypt, you will need to use the local currency, the Egyptian pound.

Connect - Egypt


How is the phone service

Before traveling to Egypt, I recommend that you check with your mobile provider for their international plan. Egypt has a reasonably modern telephone service including four GSM mobile service providers.

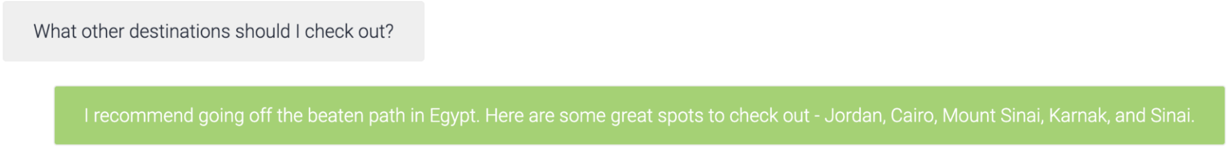
Do - Egypt



What should I do?

In Egypt, you can participate in a large variety of activities. Some fun events include the Gezira Club, the Seid Club, the Shooting Club, the Cairo Jazz Club, and the Grand Hyatt Hotel in Garden City.

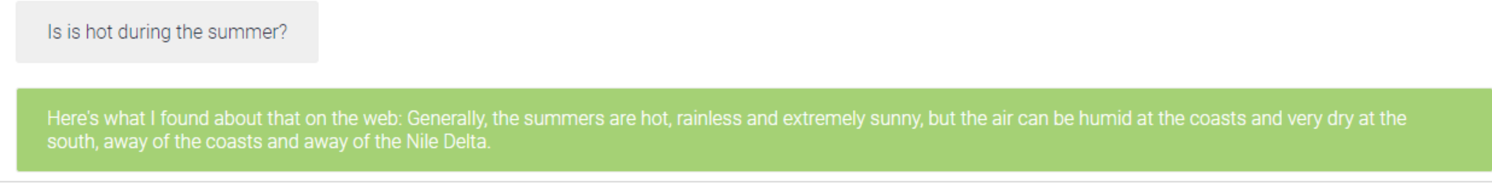
Other_destinations - Egypt



What other destinations should I check out?

I recommend going off the beaten path in Egypt. Here are some great spots to check out - Jordan, Cairo, Mount Sinai, Karnak, and Sinai.

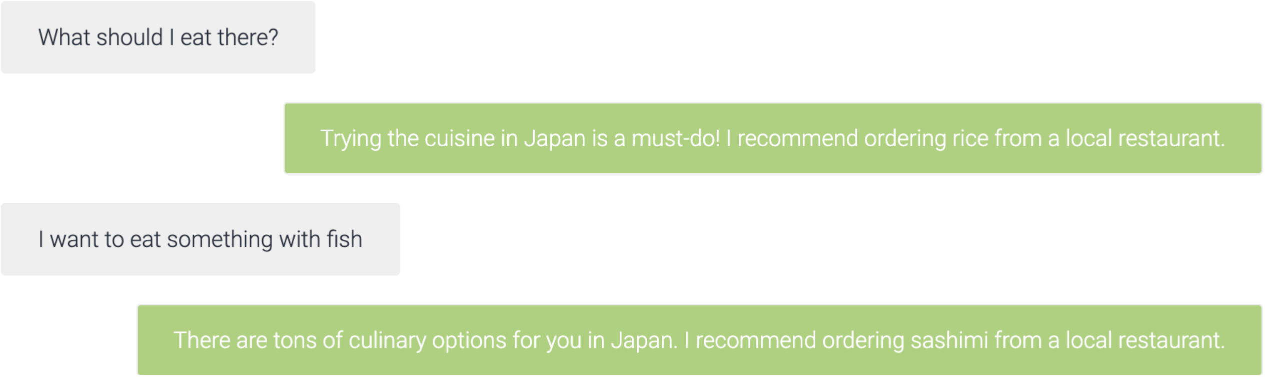
Default_fallback - Egypt



Is is hot during the summer?

Here's what I found about that on the web: Generally, the summers are hot, rainless and extremely sunny, but the air can be humid at the coasts and very dry at the south, away of the coasts and away of the Nile Delta.

Dynamic responses from the knowledge base - Japan



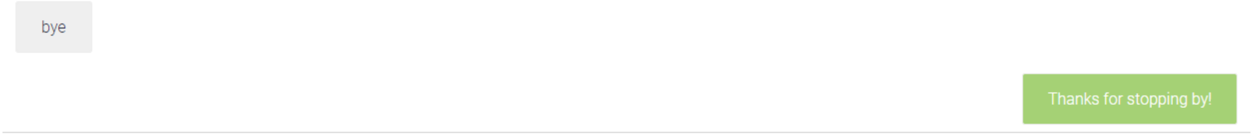
What should I eat there?

Trying the cuisine in Japan is a must-do! I recommend ordering rice from a local restaurant.

I want to eat something with fish

There are tons of culinary options for you in Japan. I recommend ordering sashimi from a local restaurant.

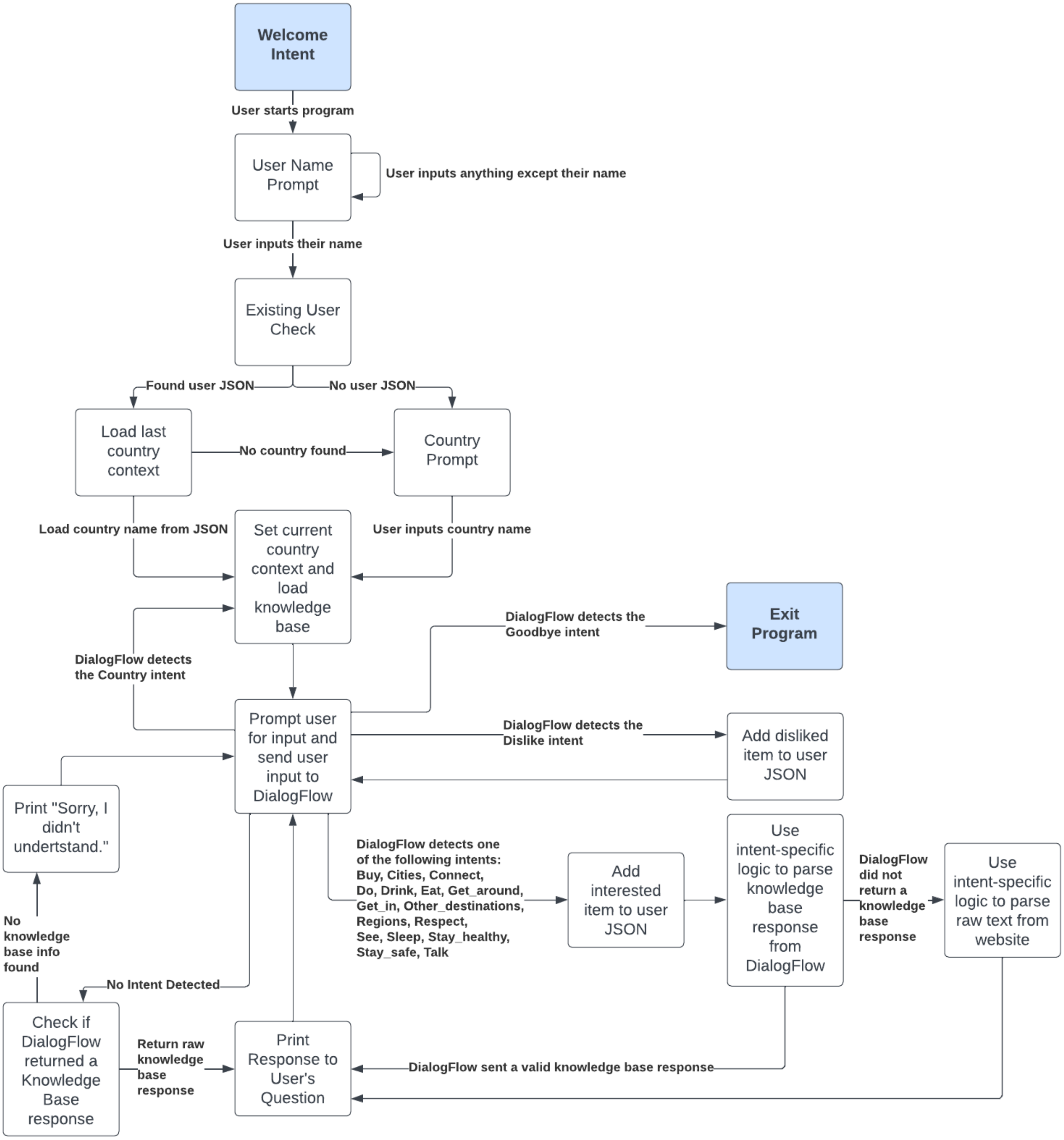
Exit



bye

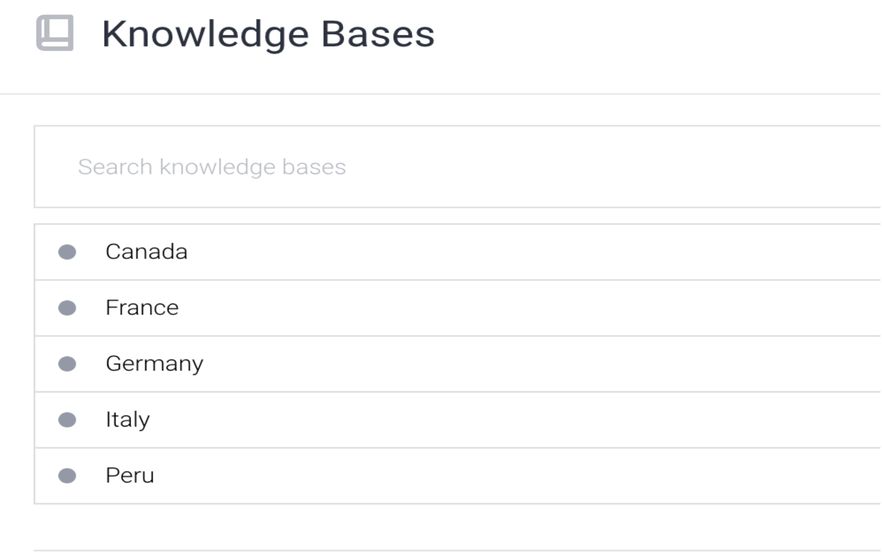
Thanks for stopping by!

Logic Dialog Diagram

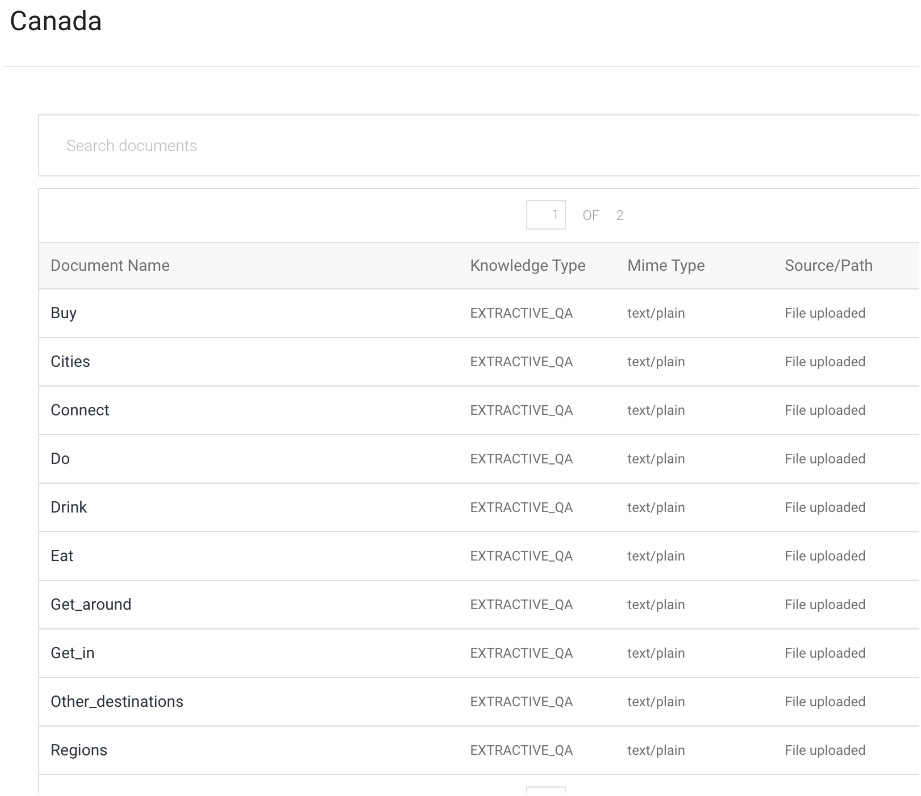

Welcome Intent
User starts program
User Name Prompt
User inputs anything except their name
User inputs their name
Existing User Check
Found user JSON
No user JSON
Load last country context
No country found
Country Prompt
Load country name from JSON
User inputs country name
Set current country context and load knowledge base
DialogFlow detects the Country intent
DialogFlow detects the Goodbye intent
Exit Program
Prompt user for input and send user input to DialogFlow
DialogFlow detects the Dislike intent
Add disliked item to user JSON
Print "Sorry, I didn't undertstand."
DialogFlow detects one of the following intents: Buy, Cities, Connect, Do, Drink, Eat, Get_around, Get_in, Other_destinations, Regions, Respect, See, Sleep, Stay_healthy, Stay_safe, Talk
Add interested item to user JSON
Use intent-specific logic to parse knowledge base response from DialogFlow
DialogFlow did not return a knowledge base response
Use intent-specific logic to parse raw text from website
No knowledge base info found
No Intent Detected
Check if DialogFlow returned a Knowledge Base response
Return raw knowledge base response
Print Response to User's Question
DialogFlow sent a valid knowledge base response

Architecture

Knowledge Base



- ● Canada
- ● France
- ● Germany
- ● Italy
- ● Peru



Document Name	Knowledge Type	Mime Type	Source/Path
Buy	EXTRACTIVE_QA	text/plain	File uploaded
Cities	EXTRACTIVE_QA	text/plain	File uploaded
Connect	EXTRACTIVE_QA	text/plain	File uploaded
Do	EXTRACTIVE_QA	text/plain	File uploaded
Drink	EXTRACTIVE_QA	text/plain	File uploaded
Eat	EXTRACTIVE_QA	text/plain	File uploaded
Get_around	EXTRACTIVE_QA	text/plain	File uploaded
Get_in	EXTRACTIVE_QA	text/plain	File uploaded
Other_destinations	EXTRACTIVE_QA	text/plain	File uploaded
Regions	EXTRACTIVE_QA	text/plain	File uploaded

To understand the structure of a chatbot's knowledge base, it's important to first understand the structure of knowledge bases in DialogFlow. In DialogFlow, you can create numerous knowledge bases, each containing its own set of documents. For example, there could be a knowledge base called "Books" with each document representing a book, and another called "Articles" with each document representing an article.

In this scenario, each knowledge base represents a country name, and each "document" contains a category of information about that country. The categories are the same for every country, which will become clear when we explain the process of scraping information for our knowledge base.

All of the knowledge bases are created through our Python code and uploaded to Dialogflow to be used in conjunction with our own natural language processing (NLP) techniques. When a user is prompted to enter a country to begin a discussion, one of two things will happen: either the chatbot will find an existing knowledge base for that country or it will create one.

The source of all of our data for this knowledge base comes from the WikiVoyage websites, where potentially travelers can look up any desired country for a WikiPage about it. This is an ideal source for the knowledge base because all countries have the same seventeen sections on their pages. This makes it easy to use web scraping techniques to extract information from each section for any desired country without having to change the code when users want information on a country that doesn't have a knowledge base yet. This allows for something called "dynamic scraping," in which the chatbot can scrape a country's information and create a knowledge base if that country is not known to it yet (though it is a time-consuming process). These headers are "Understand," "Regions," "Cities," "Other Destinations," "Get In," "See," "Do," "Talk," "Buy," "Eat," "Drink," "Stay Healthy," "Stay Safe," "Connect," and "Respect." Some of these headers are more self-explanatory than others, but when combined, they allow the user to ask many different things about a country. Each of these "headers" becomes its own document in that country's knowledge base.

In terms of scraping, minimal preprocessing was done in order to ensure the knowledge base was full of coherent sentences that can be further analyzed only when needed. The most that was done at this stage was ensuring all sentences meet a minimal length, as scraping data from Wiki can sometimes get unnecessary information.

Running the Program:

You can access our code https://github.com/fdolisy/TravelAgent here. In the repository's README, you will find instructions for running this program through your local python installation, or through the Flask web server we wrote.

NLP Techniques in Python

To connect our own python code to the knowledge base, we integrated the DialogFlow API into our client to access the knowledge base, as well as obtain some very useful features provided by DialogFlow to parse the user input. As briefly mentioned above, various NLP techniques were used to further manipulate the Dialogflow knowledge base answers. Upon detecting certain intents, our Python code will call specific methods written for each of our sixteen pre-defined intents.

WordNet Synset Detection

For the **Eat**, **Drink** and **Talk** intents, we first defined our own list of synsets that could be used to identify certain items related to these intents. For example for the Eat intent, we want to be able to detect food items, so our list of WordNet synsets includes synset('food.n.01'), synset('fruit.n.01'), synset('snack.n.01') and so on. This increases the chances that we will pick up and return an item to the user. Similarly, this technique is used for the “Drink” intent where types of beverages are identified, and the “Talk” intent where spoken languages are identified.

In testing this detection, we noticed certain unhelpful words were being returned. For instance, the “Food” intent would tell a user that they should order “dinner”, rather than a specific type of food. To fix this problem, we also manually defined a list of banned words to avoid in our response for each intent.

Additionally, we wanted to ensure our responses were dynamic and did not just return the same list of food every time. To do this, we first parse the knowledge base response returned from Dialogflow for our desired synsets. This response will vary based on the user’s input (e.g. if the user specifically asked about eating meat, the knowledge base response will likely be a sentence geared towards this type of food). In the case where the knowledge base parsing is not successful, we then default to parsing the raw article text from our knowledge base ourselves, which is less dynamic but more likely to return output.

Named Entity Recognition

The existing synsets that are part of NLTK do not include NER for locations, so we imported a separate Python library called “Location Tagger” which expands upon the NLTK synsets to further be able to identify cities, countries and other destinations. We use this library for the **Cities**, **Regions**, and **Other destinations** intents. Similarly to how we did with our synset detection, we first search for these named entities in the knowledge base response, and then parse the raw article text if needed. This allows for us to still have a diversity in the locations that are suggested to the user.

Term Frequency Analysis

Simple frequency analysis was used in conjunction with the techniques listed above. For example, in the Eat intent, we often parsed out over fifty food terms. To narrow down which terms to return, we initially used a simple word count over the identified terms, as well as a maximum number of terms to return. However, this often produced sub-optimal results, especially with the Talk intent. For instance, the Talk intent for France would return the language "Chinese" because it was mentioned once in the article. To fix this, we also incorporated a relative frequency value. This ensured the terms returned were actually significant. For example, if the language French is mentioned 50 times, the language English is mentioned 10 times, and the language Chinese is mentioned 1 time, then our relative frequency value of 0.2 would ensure we only return French and English, even though our max number of languages returned was set to 5.

Part-of-speech tagging

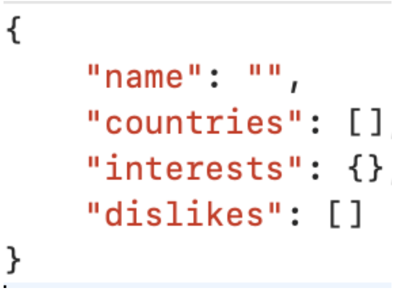
For the **Do** and **See** intents, we needed a way to identify landmarks to suggest to the user. Identifying landmarks is a relatively simple process that we accomplished through the use of part-of-speech tagging. By observing the data, we observed that landmarks tend to follow a common pattern within a sentence: there will generally be a group of proper nouns ("NNP") next to each other, often separated by a preposition ("IN"). For example, the landmark the "Statue of Liberty", consists of the part-of-speech tags: ["NNP", "IN", "NNP"]. To make the response to the user sound natural, we also checked whether or not there was a determiner ("DT") right before the landmark. This allowed us to tell the user to visit "the" Statue of Liberty, but not "the" Central Park, creating more realistic and conversational responses.

In certain contexts, this idea would not work because these could also be people's names and other entities, but since we only apply these techniques in the "See" and "Do" documents from our knowledge base, the chances that these back to back proper nouns are not landmarks or activities is minimal. Additionally, we used part-of-speech tagging in the same way for the **Get in** intent, but we filtered the results to specifically search for proper nouns containing the word "airport".

User Model

The python client keeps track of a relatively simple user model that is integrated into the chatbot dialog. When a user enters their name into the chatbot, the python client will look for a file within the format “name.json”. If one exists, the user will be welcomed back into the system and asked if they would like to continue discussing their most recently discussed country. If a file with their name does not exist, the client will create one.

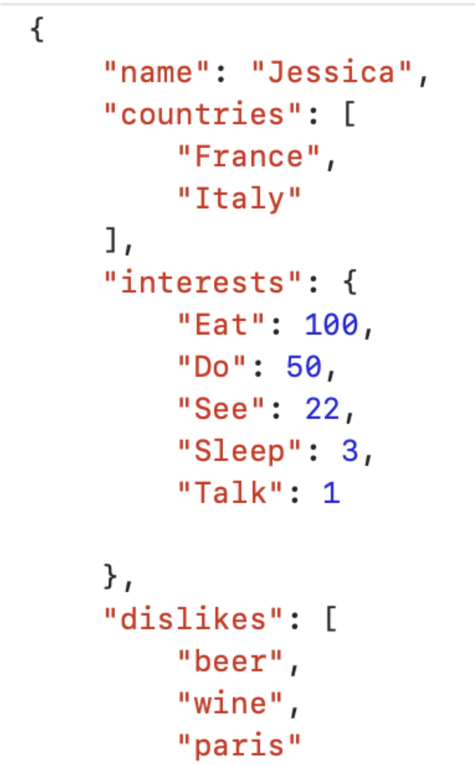
Each file follows the same JSON format -



```
{
    "name": "",
    "countries": []
    "interests": {}
    "dislikes": []
}
```

As seen on the right, the model keeps track of the user's name, countries, interests, and dislikes. The name is equivalent to the name entered by the user. The countries attribute keeps track of all the countries a user has inquired about. The interests attribute keeps track of all the intents the user has triggered, as well as a count so we know which intents the user is most interested in. For example, we can track whether a user is most interested in sightseeing or eating. Lastly, the dislikes attribute keeps track of nouns the user has explicitly said not to suggest so we can avoid suggesting them again.

A filled example would look like this. The example to the right indicates that the user “Jessica” has discussed France, Germany and Chile with the chatbot. It indicates that in those conversations she has questions about what to eat, do, what languages are spoken and common hotel information. Finally, we can see that Jessica has said a variant of “I don’t like beer” or “Stop suggesting wine”. These dislikes will never be brought up in any chatbot responses.



```
{
    "name": "Jessica",
    "countries": [
        "France",
        "Italy"
    ],
    "interests": {
        "Eat": 100,
        "Do": 50,
        "See": 22,
        "Sleep": 3,
        "Talk": 1

    },
    "dislikes": [
        "beer",
        "wine",
        "paris"
```

Evaluation and Analysis

To analyze our chatbot, we asked others to try it without giving them information about our backend logic in advance. This was to prevent them from being biased towards asking certain questions. After they tried the program, we then asked them five Likert-style questions.

Survey results:

Question (on a scale of 1 to 5)	Average score
Ease of Use - Was the program easy to use and understand?	3.33
Intent Detection - How happy were you with how the chatbot understood the question you were asking?	2
Response Coherence - How happy were you with the coherence of the chatbot's response? Was the response logical and easy to read/interpret?	3.33
Response Quality - How happy were you with the quality of the chatbot's response? Did the response include useful information, or was it too generic?	1.66
Conversation flow - How happy were you with the overall flow of the conversation? Did you find it confusing or frustrating to converse with the agent?	2

In analyzing our survey results, we can see that the strengths of our program were its usability and the coherence of the response. Based on this result, we feel confident that the output we are writing, whether through our manual parsing of intents or through Dialogflow's knowledge base responses, is written in such a way that the user can understand it.

We can see that our users were less happy with the overall conversation flow and intent detection. This is likely because our program can only pick up questions related to our predefined intents, so any queries outside of these may not receive helpful responses.

We can see that our users were least happy with the response quality. Again, this is likely another limitation of our pre-defined intents, which limit our ability to fully understand the semantics of a user's question in generating our response.

Strengths

Overall, one of the major strengths of our chatbot is that we were able to make it fairly good at detecting our fifteen defined intents thanks to our Dialogflow integration. By using a wide variety of training phrases, we are able to handle fairly ambiguous text from the user if it slightly relates to an intent.

Another major strength is that our chatbot is able to answer questions relatively quickly because we store static knowledge bases for each country. Additionally, we limited how much data we actually stored in the knowledge base, ensuring it does not take too long to search and form a response. Of course, in the case where we must dynamically scrape information about a new country, there will be a wait time. However, in the future when the user queries about that country again, there will be no wait time.

A strength of our chatbot is that our knowledge base contains rich and informative text, for any country. Thus, the result is very likely to contain accurate and coherent information, even if it does not fully answer the user's actual question.

Weaknesses

Of course, the major weakness of our chatbot is that its functionality is primarily limited to the fifteen defined intents we have. If the user asks a question outside of these intents (e.g. "How is the weather?"), we make a best attempt to reply based on Dialogflow's knowledge base response. While this works occasionally, it also has the potential to result in very confusing responses.

Another weakness of our chatbot is that our knowledge base currently only comes from one website (Wikivoyage). We chose to do this because this website uses consistent formatting for their data, and because it contains a wide variety of data. However, this has limited us from pulling in information from other websites. This made certain intents, such as the "Sleep" intent, especially difficult to make useful as we could not access data from hotel websites for instance.

Additionally, the chatbot is limited in how much context it can store about the user and locations. For instance, if the user were to say "What city should I visit?", and then the chatbot responded "You should visit Vancouver.", the chatbot will not remember its response. So, if the user then asked "Why?", they would not get a response. Instead, the user would have to say "Why should I visit Vancouver?" This limitation has negative impact on the overall conversation flow and usability, as indicated in our user testing.

Because of our external dependency on Dialogflow, the time to upload our knowledge base data is rather substantial. Though users will be able to access countries who are not on the list, it will take multiple minutes before the knowledge base is ready to the user, thus limiting users in the number of countries readily accessible.

With the lack of sentiment analysis in our manual parsing, this results in "bizarre" or unfit answers to simple questions. For example, when asked for fun events to do around Japan, the chatbot suggested "World War II" as a potentially fun thing to explore.