

Empowering Generative AI in Home Care Services – Developing a Chatbot by utilizing Prompt Engineering Techniques

User Guide

Prompt engineering is a methodology used in natural language processing (NLP) to design effective prompts or instructions for language models, such as GPT (Generative Pre-trained Transformer) models like ChatGPT. The goal of prompt engineering is to craft input queries or instructions in such a way that the model generates desired outputs or responses.

LLMs stands for "Large Language Models." These are advanced NLP models like GPT, BERT (Bidirectional Encoder Representations from Transformers), and others, which are pre-trained on large text corpora and can generate human-like text or perform various language-related tasks.

This user guide explains how to set up and interact with the Aged Care Policy Assistant Chatbot(an LLM), which is designed to provide information related to aged care policies and incidents. The chatbot utilizes OpenAI's platform to process queries and generate responses based on the data provided in a JSON file. For our Chatbot, we have used the Assistant API and gpt-4-Turbo model from Open AI.

This project follows the following 4 processes:

1. Data Extraction
2. LLM model creation
3. Integration Process with WhatsApp
4. LLM model evaluation

1. Data Extraction

To accomplish this, we employ three Python scripts:

- **combine_file.py** - Merges multiple JSON files into a single file.
- **ExtractData.py** - Retrieves text from PDF files, transforms Excel files to JSON format, and merges JSON files.
- **read_excel_file.py** - Transforms data from Excel sheets into JSON format.

System Requirements

- Python 3.11 or higher

Installation of Dependencies

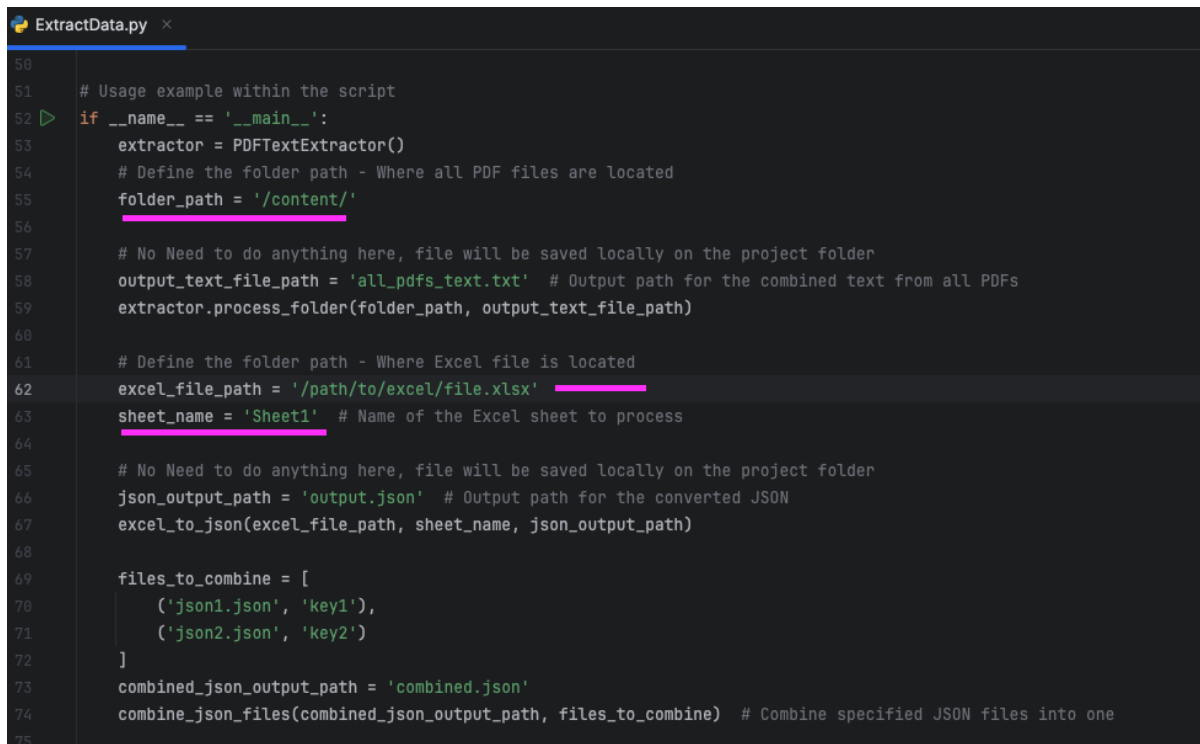
Install the necessary Python packages including **json**, **PyPDF2**, **pandas**, **os**, **openpyxl** and **re**. Execute the following command in your terminal:

```
pip install json PyPDF2 pandas
```

Setting Up Data Extraction

Focus primarily on the **ExtractData.py** script. You'll need to specify the paths to the folder containing the PDF files and the folder containing the Excel file. Note that all PDF files should be in one dedicated folder, and Excel files should be in a separate folder.

Ensure that the **ExtractData.py** script points to the correct folders by updating the paths highlighted in the below image. Additionally, confirm that all related scripts are saved in the same directory.



```
50
51 # Usage example within the script
52 if __name__ == '__main__':
53     extractor = PDFTxtExtractor()
54     # Define the folder path - Where all PDF files are located
55     folder_path = '/content/'
56
57     # No Need to do anything here, file will be saved locally on the project folder
58     output_text_file_path = 'all_pdfs_text.txt' # Output path for the combined text from all PDFs
59     extractor.process_folder(folder_path, output_text_file_path)
60
61     # Define the folder path - Where Excel file is located
62     excel_file_path = '/path/to/excel/file.xlsx'
63     sheet_name = 'Sheet1' # Name of the Excel sheet to process
64
65     # No Need to do anything here, file will be saved locally on the project folder
66     json_output_path = 'output.json' # Output path for the converted JSON
67     excel_to_json(excel_file_path, sheet_name, json_output_path)
68
69     files_to_combine = [
70         ('json1.json', 'key1'),
71         ('json2.json', 'key2')
72     ]
73     combined_json_output_path = 'combined.json'
74     combine_json_files(combined_json_output_path, files_to_combine) # Combine specified JSON files into one
75
```

2. LLM Model Creation

2.1 Prerequisites

Before you start, please ensure you have the following:

- Python installed on your machine (Python 3.8 or later is recommended).
- An API key from OpenAI. You must have an OpenAI account and access to their API services.
- The openai Python library installed.
- JSON files with the data you intend to use, stored in a location accessible by your Python script (e.g., policy.json, incident.json).

2.2 Setup Instructions

2.2.1 API Key Configuration:

- Replace "xyz" in the script with your actual OpenAI API key.

2.2.2 JSON Data File:

- Ensure your JSON file is correctly formatted and contains the necessary data for the chatbot.
- Place your JSON file in the /content/ directory or modify the file path in the script to match your file's location.

2.2.3 Running the Script:

- Open your Python IDE or command line.
- Navigate to the directory containing your script.
- Run the script

2.3 Operational Guide

2.3.1 Starting the Chatbot:

When you run the script, it first uploads your JSON data file to OpenAI and sets up the assistant with specified capabilities and instructions.

2.3.2 Interacting with the Chatbot:

- Once the chatbot is initialized, it will wait for user input. The default query in the script is "what to do incase I have covid infection?".
- You can change this query in the script to whatever you need by modifying the content parameter in the messages.create method.

2.3.3 Receiving Responses:

- The chatbot processes the input and retrieves information from the uploaded JSON data.
- It will print out the response in the console once the processing is completed.

2.4 Customizing the Chatbot

2.4.1 Modifying Instructions:

You can change the instructions provided to the assistant to alter how it handles queries. Modify the instructions parameter in the `beta.assistants.create` method.

2.4.2 Adding More Data:

To allow the chatbot to handle more topics, you can add more data to your JSON file or upload additional files. Ensure you adjust the file upload and reference sections in the script accordingly.

2.5 Troubleshooting

- **API Key Issues:** Ensure that your API key is correct and active. Errors related to authentication typically indicate an issue with the API key.
- **File Not Found:** If you get a file not found error, check the path specified in your script and ensure the file exists at that location.
- **Dependency Errors:** Make sure all required libraries are installed. If you encounter any missing library errors, install them using `pip`.

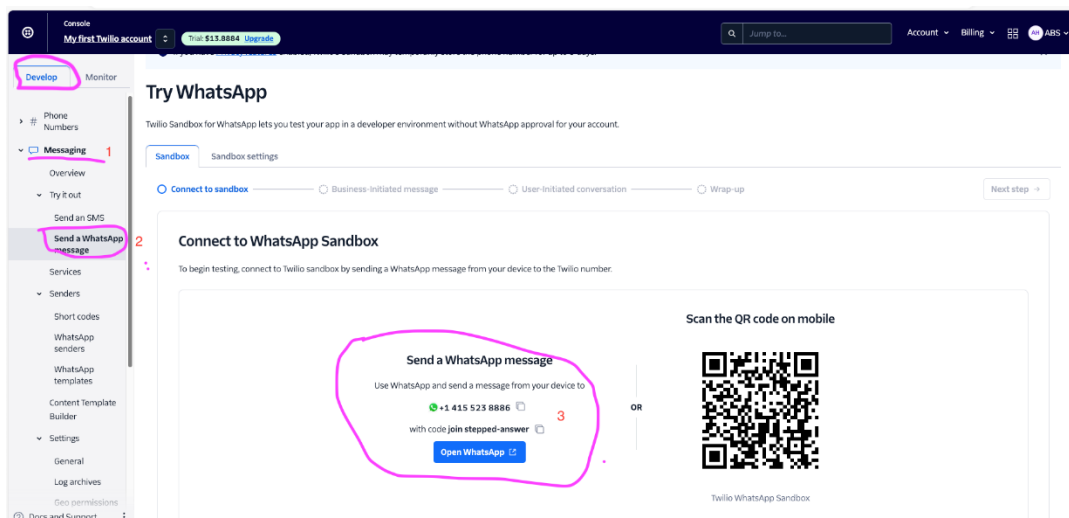
3. Integration Process with WhatsApp

Step 1: Twilio Account Setup – Phase 1

To send WhatsApp messages via Twilio:

1. Log in to your Twilio account at [Twilio's website](#).
2. Go to the 'Develop' section and select 'Messaging' then 'Try it Out'.
3. Choose 'Send a WhatsApp message' and access the 'WhatsApp Sandbox'.

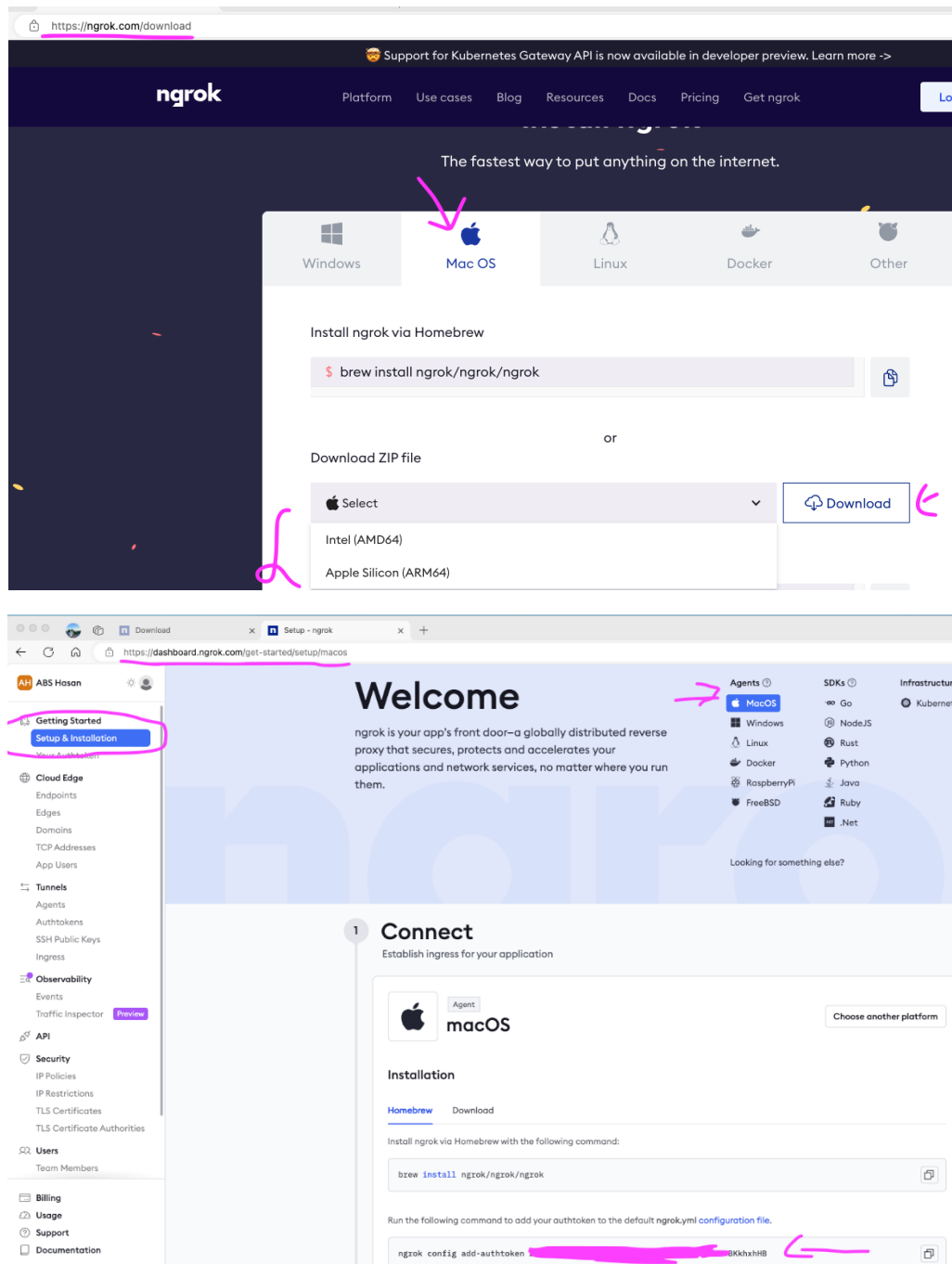
Follow the instructions provided in step 3 carefully, ensuring you set up the sandbox environment correctly.



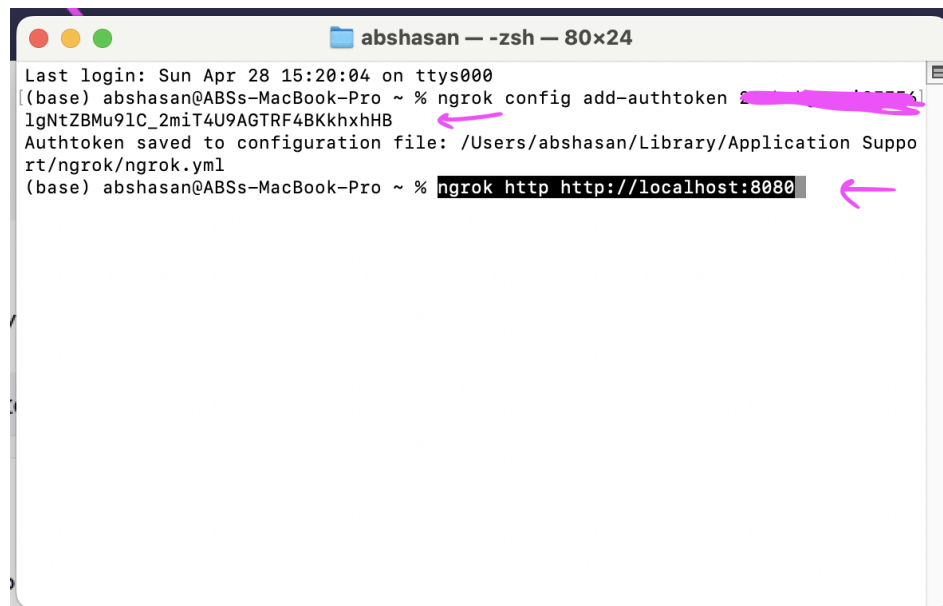
Additionally, it's important to configure the Sandbox Settings in Twilio. We will revisit the Twilio platform to finalize these settings after establishing our NGrok server and updating the script.

Step 2: NGrok Configuration

Download and set up NGrok for your operating system from [NGrok's official website](https://ngrok.com). If assistance is needed, refer to the provided screenshots.



Once the aforementioned steps are completed, it's time to run NGrok on our local machine. Simply open the terminal, input the Auth token, and provide the local host details to start NGrok. This will initialize NGrok, making it ready for use.



```
abshasan — -zsh — 80x24
Last login: Sun Apr 28 15:20:04 on ttys000
(base) abshasan@ABSs-MacBook-Pro ~ % ngrok config add-authtoken 2[REDACTED]
lgNtZBMu91C_2mit4U9AGTRF4BKkhxhHB
Authtoken saved to configuration file: /Users/abshasan/Library/Application Support/ngrok/ngrok.yml
(base) abshasan@ABSs-MacBook-Pro ~ % ngrok http http://localhost:8080
```

After above command this below screen will appear



```
abshasan — ngrok http http://localhost:8080 — 139x23
ngrok
Full request capture now available in your browser: https://ngrok.com/r/ti

Session Status      online
Account             ABS Hasan (Plan: Free)
Update              update available (version 3.9.0, Ctrl-U to update)
Version             3.9.0
Region              Australia (au)
Latency             8ms
Web Interface       http://127.0.0.1:4040
Forwarding           https://ce52-125-253-56-0.ngrok-free.app -> http://localhost:8080

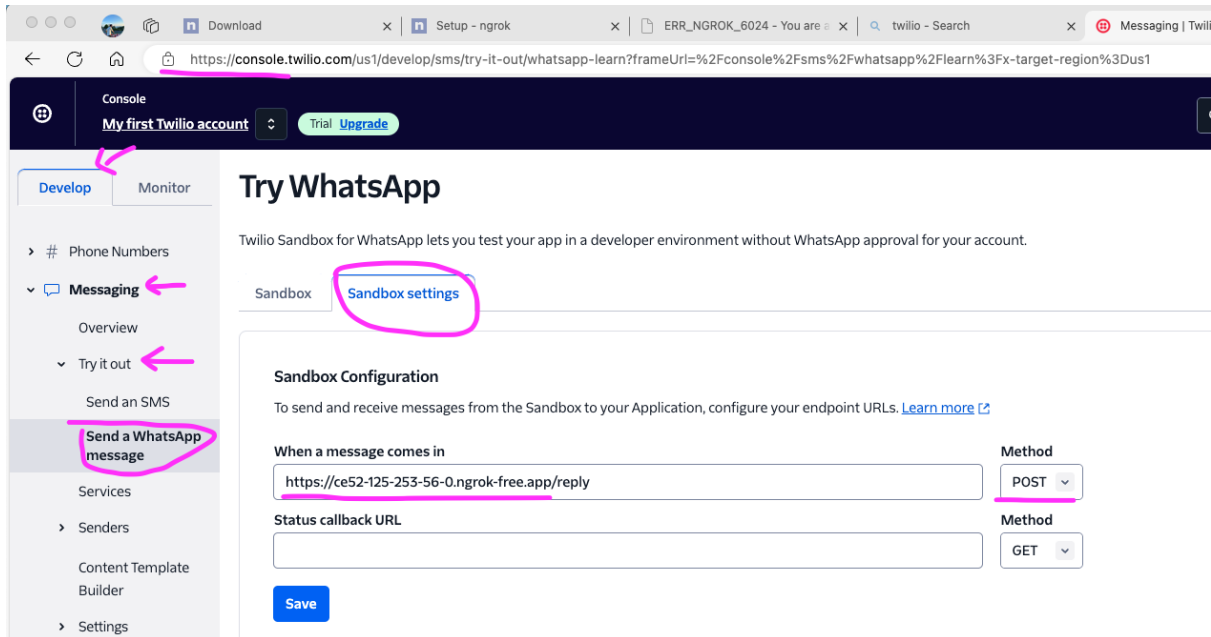
Connections
  ttl  opn  rt1  rt5  p50  p90
   0    0    0.00 0.00 0.00 0.00
```

After installation, retrieve your auth token from the NGrok dashboard and initialize NGrok with it on your local machine using the terminal, copy the forwarding address (e.g., <https://ce52-125-253-56-0.ngrok-free.app>) provided by NGrok.

Step 3: Twilio Account Setup – Phase 2

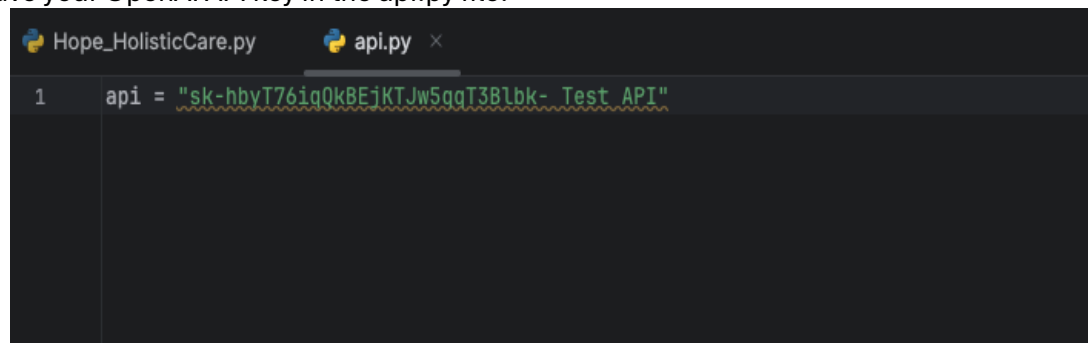
Return to your Twilio account and update the sandbox webhook with the NGrok forwarding address (<https://ce52-125-253-56-0.ngrok-free.app>) followed by /reply to allow the Flask app to connect.

Ensure the method is set to 'POST'.



Step 4: Finalize the Script

1. Install the required libraries using the command `pip install flask twilio openai`.
2. Save your OpenAI API key in the `api.py` file.



3. Run the `Hope_HolisticCare.py` script to connect with WhatsApp, ensuring NGrok and Twilio are configured as outlined above.

After completing these steps, the file is ready for use. Execute the script, and it will establish a connection with WhatsApp. Ensure that NGrok and Twilio are configured correctly and that the instructions in the Chatbot Creation section have been followed meticulously.

Additionally, it would be preferable to run this program in the PyCharm IDE, as opposed to VSCode or other IDEs. PyCharm is among the most user-friendly tools, offering an easier management and operation experience.

4. LLM Model Evaluation

4.1 Platform Used

The built chatbot was evaluated using UpTrain, an open-source platform for evaluating different LLM models, including OpenAI, Azure, Claude, Mistral, Together AI, Anyscale, and Ollama. UpTrain provides scores for more than 20 pre-configured evals, together with the analysis of failure cases by utilizing GPT 3.5 Turbo. In this project, we evaluated our chatbot using Response Quality Evals and Context Awareness Evals.

a) Response Quality Evals

The Response Quality Evals was evaluated in terms of *Response Completeness*, which measures whether the response generated from chatbot has adequately answered all aspects to the question that is given by users. A higher response completeness score means the generated response has fulfill answered the user's questions. A dataset of questions and corresponding responses generated from chatbot is required for measuring Response Completeness scores.

b) Context Awareness Evals

Context Awareness Evals were evaluated in terms of Context Relevance and Factual Accuracy.

Context Relevance measures whether the retrieved context has enough information to answer the question that is given by users. A low context relevance score indicates that the context provided was irrelevant to the question. A dataset of questions and corresponding contexts is required for measuring Context Relevance scores.

Factual Accuracy measures the degree of the information provided in the response is true according to the context provided. A higher factual accuracy score indicates that the generated response is factually correct. A dataset of questions, corresponding response, and relevant context is required for measuring Factual Accuracy score.

4.2 Dataset Preparation

The context data were generated by extracting the text from each document to a .json file. Among 20 documents, Grievance Management Policy and Grievance Management Procedure were merged into one, resulting in 19 contexts (See Table 1).

Table 1: Policy documents and corresponding contexts

No.	Policy Document	Context Names
1	Abuse and Neglect Policy_POL_V01_2209	abuse
2	BYOD_POL_V01_2209	byod
3	Client Money Management Policy_PRO_V01_2209	client_money
4	Client Rights_POL_V01_2208	client_right
5	Continence Management_POL_V01_2203	continence_mana
6	Continuous Improvement_POL_V01_2203	continuous_improve
7	Data Breach Response Plan_Pro_V01_2305	data_breach

8	Decision Making, Choice and Dignity of Risk_POL_V01_2208	decision
9	Deterioration Policy_POL_V02_2303	deterioration
10	Diversity and Cultural Inclusion_POL_V01_2208	diversity
11	Falls Prevention Management_POL_V01_2203	fall
12	Feedback and Complaint Management Policy_POL_V02_2210	feedback
13	Grievance Management Policy_POL_V01_2201	grievance
14	Grievance Management Procedure_PRO_V01_2201	
15	Guidelines for Salary Packaging GDL_V02_2110	salary
16	HHC Staff Handbook 2312	handbook
17	High Impact High Prevalence Risk Policy_POL_V01_2303	risk
18	Incident Management_POL_V02_2211	incident
19	Infection Prevention and Control_POL_V01_2209	infection
20	Privacy Policy_POL_V03_2312	privacy

For each context, five questions were generated in the form of frequently asked questions by staff to the managers, including incidents and information requests. Each question is then sent to the chatbot Playground to generate corresponding response. The responses were collected and stored in a .xlsx file, together with the question and the relevant context names (See Table 2). As a result, the evaluation dataset contains 95 questions, 95 corresponding responses generated by the chatbot, and their relevant contexts.

Table 2. A view of the first 10 rows of the evaluation dataset consisting of questions, contexts, and responses for Abuse and Neglect Policy (abuse), and BYOD policy (byod).

question	context	response
What happens if a staff member fails to report abuse or tries to cover it up?	abuse	If a staff member fails to report abuse or attempts to cover it up, under the Abuse and Neglect Policy [Policy_Abuse_and_Neglect_Policy_POL_V01_2209], they will face serious consequences. [...]
I'm unsure about my responsibilities in abuse cases. What should Case Managers do under this policy?	abuse	Hello! Under the Abuse and Neglect Policy [Policy_Abuse_and_Neglect_Policy_POL_V01_2209], Case Managers have specific responsibilities when it comes to abuse cases. [...]
If a client wants to report abuse, what options do they have? I want to make sure they feel supported.	abuse	Hello, Clients at Hope Holistic Care have several options for reporting abuse to ensure they feel supported and safe. [...]

What's the first thing I should do if I receive a report of abuse?	abuse	Hello, Clients at Hope Holistic Care have several options for reporting abuse to ensure they feel supported and safe. [...]
A client is hesitant to involve the police in reporting abuse. How should I handle this?	abuse	In situations where a client is hesitant to involve the police in reporting abuse, it's important to handle the matter sensitively while also fulfilling your legal and organizational responsibilities. [...]
I'm using my own device for work, but I'm not sure if it meets the security requirements outlined in our BYOD Policy. How can I check if my device is compliant?	byod	To check if your device is compliant with the BYOD (Bring Your Own Device) Policy [BYOD_Policy_POL_V01_2209] at Hope Holistic Care, you should ensure the following [...]
I've been using a personal cloud backup service. Can I keep using it, or should I switch to a different service?	byod	According to the BYOD Policy [BYOD_Policy_POL_V01_2209] at Hope Holistic Care, you should not be using any personal cloud [...]
My device is synced with other devices in my home. Is this allowed, or do I need to make changes?	byod	Under the BYOD Policy [BYOD_Policy_POL_V01_2209], syncing your work device with other home devices is not permitted due to security concerns [...]
I need to take some photos for work. What steps should I take to handle them according to our policy?	byod	When handling photos for work purposes, you should adhere to the guidelines set forth by the BYOD Policy [BYOD_Policy_POL_V01_2209]. [...]
I'm confused about restrictions on using the camera and recording functions of my device on company property. Can you clarify?	byod	According to the BYOD Policy [BYOD_Policy_POL_V01_2209], there are specific restrictions regarding the use of camera, video, and recording functions on personal devices when you are on company property [...]

4.3 Evaluating the chatbot using UpTrain

Firstly, you need to install uptrain and openai python libraries. You also need access to the OpenAI account and an active API key.

a) *Install Uptrain and OpenAI*

```
pip install uptrain
```

```
pip install openai
```

b) *Import library*

```
import pandas as pd
from uptrain import EvalLLM, Evals
import json
```

c) *Create data*

Create a dictionary to store context data using their Context Name

```
context_dict = {}

with open(json_file_path, 'r') as json_file:
    context_dict = json.load(json_file)
```

Read the evaluation dataset from .xlms file and transform it to a data frame using *pandas*

```
excel_file = "eval_dataset.xlsx"
df = pd.read_excel(excel_file)
```

d) *Run Evaluation*

```
import openai
import os
from dotenv import load_dotenv
load_dotenv()
openai.api_key = os.environ.get("OPENAI_API_KEY")
eval_llm = EvalLLM(openai_api_key=openai.api_key)
```

or

```
OPENAI_API_KEY = "sk-*****" # Insert your OpenAI key here
eval_llm = EvalLLM(openai_api_key=OPENAI_API_KEY)
```

Run evaluation for each context, by looping through the evaluation dataset, then run a batch of five rows at a time. Due to the limitations of requests for gpt-3.5-turbo and the length of input texts including the context data that contain extracted text from each policy documents, we can only run evaluation for five records at a time. More information, please see [Rate limits - OpenAI API](#).

e) Evaluate data

```
result = eval_llm.evaluate(  
    data=batch_data,  
    checks=[Evals.CONTEXT_RELEVANCE,Evals.FACTUAL_ACCURACY,  
    Evals.RESPONSE_COMPLETENESS])
```

f) Save results

```
with open(result_variable, 'w') as f:  
    json.dump(result, f, indent=4)
```

The results will be saved as 19 separated .json files alternative to 19 contexts. These files will be combined into one .json file for further statistical analysis and visualization. The evaluation results are shown in Table 3.

Table 3. Evaluation scores analysis

No.	File	Context Relevance (mean±standard deviation)	Factual Accuracy (mean±standard deviation-)	Response Completeness (mean±standard deviation)
1	abuse.json	1±0	1±0	0.9±0.2
2	byod.json	0.8±0.4	0.7±0.2	1±0
3	client_money.json	0.1±0.2	0.54±0.344093	1±0
4	client_right.json	0.9±0.2	0.6±0.309839	1±0
5	continenace_manajson	0.5±0	0.96±0.08	1±0
6	continenace_improve.json	0.2±0.244949	0.77±0.302655	1±0
7	data_breach.json	1±0	0.9±0.109545	1±0
8	decision.json	0.7±0.4	0.9±0.2	1±0
9	deterioration.json	0.2±0.244949	0.96±0.08	1±0
10	diversity.json	0.8±0.4	0.88±0.16	1±0
11	fall.json	0.8±0.244949	0.8±0.252982	0.9±0.2
12	feedback.json	0.6±0.374166	0.92±0.09798	1±0
13	grievance.json	0±0	0.96±0.08	1±0
14	handbook.json	0.2±0.244949	0.56±0.149666	0.9±0.2
15	incident.json	1±0	0.9±0.2	1±0
16	salary.json	1±0	0.92±0.9798	1±0
17	risk.json	0.1±0.2	0.875±0.146629	1±0
18	infection.json	0.4±0.489898	0.96±0.08	0.8±0.244949
19	privacy.json	0.4±0.374166	0.826667±0.258543	1±0