Configuration Manual

MSc Research Project
Data Analytics

Reinaldo Zanello Klostermann
Student ID: x21133018

School of Computing
National College of Ireland

Supervisor: Vladimir Milosavljevic

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Reinaldo Zanello Klostermann

Student ID: x21133018

Programme: Masters in Data Analytics **Year:** 2023

Module: MSc Research Project

Lecturer: Vladimir Milosavljevic

Submission Due Date: 14/08/2023

Project Title: Large Language Model Powered Chatbot for Comprehensive Citizens Information Services

Word Count: 1600..... **Page Count:** 17.....

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: .....

Date: .....

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Reinaldo Zanello Klostermann
Student ID: x21133018

1 Introduction

The goal of this project was to develop a sophisticated chatbot that retrieves, processes, and interacts with data from a Citizens Information website using a Large Language Model (LLM). WhatsApp users can engage with this chatbot through the platform. Visual Studio Integrated Development Environment (IDE) and Python 3.11 were used for this project. The language model used is GPT-4. Additionally, Pinecone is employed as a vector database management system. To interact with GPT-4 and Pinecone, API calls are used. This project uses the LangChain framework. Twilio API is also used to integrate with WhatsApp. The Twilio API and the chatbot server run on localhost are communicated using ngrok, a tool that exposes local servers through public URLs. The references for coding this project are in README.md file in the ICT Solution Artefact.

2 Account Requirements and API Keys for Development

It is necessary to have access to specific platforms and API keys to replicate this implementation. Table 1 provides URLs to create the necessary accounts. Following the account creation, the respective API keys must be generated.

Table 1: Required Accounts and Corresponding URLs

Account	URL
OpenAI	https://openai.com/
Pinecone	https://www.pinecone.io/
Twilio	https://www.twilio.com/en-us
ngrok	https://ngrok.com/
Inspired Critique	https://docs.inspiredco.ai/critique/

Except for OpenAI, account creation is free, and the free APIs and services are sufficient to implement this solution. The embeddings and calls to the LLM are minimal since this is not a production environment. This project cost approximately fifteen euros.

3 Python Libraries

To reproduce this system implementation successfully, certain Python libraries are required. These libraries provide several functions that are essential to the operation of the application. Table 2 lists these required Python libraries in addition to links to their documentation. Prior to running the system, these libraries must be installed in your Python environment. Typically, pip or conda are used for installation. Pipenv was used for the creation of a virtual environment for installing the libraries.

Table 2: Required Python Libraries and their Documentation Links

Python Library	Documentation
openai	https://pypi.org/project/openai/
flask	https://pypi.org/project/Flask/
twilio	https://pypi.org/project/twilio/
pinecone-client	https://pypi.org/project/pinecone-client/
langchain	https://pypi.org/project/langchain/
tiktoken	https://pypi.org/project/tiktoken/
beautifulsoup4	https://pypi.org/project/beautifulsoup4/https://pypi.org/project/beautifulsoup4/
lxml	https://pypi.org/project/lxml/
nltk	https://pypi.org/project/nltk/

4 Implementation

Downloading the ICT Solution Artefact file is the first step in the implementation process. This file will be made publicly available on GitHub[1]. As shown in Figure 1, the project is currently set to private. Once the file has been downloaded, it should be unzipped and loaded into an IDE. A detailed explanation of the system's functionality can be found in the following sections.

[1] GitHub Repository: https://github.com/rklostermann/ICT-Solution-Artefact

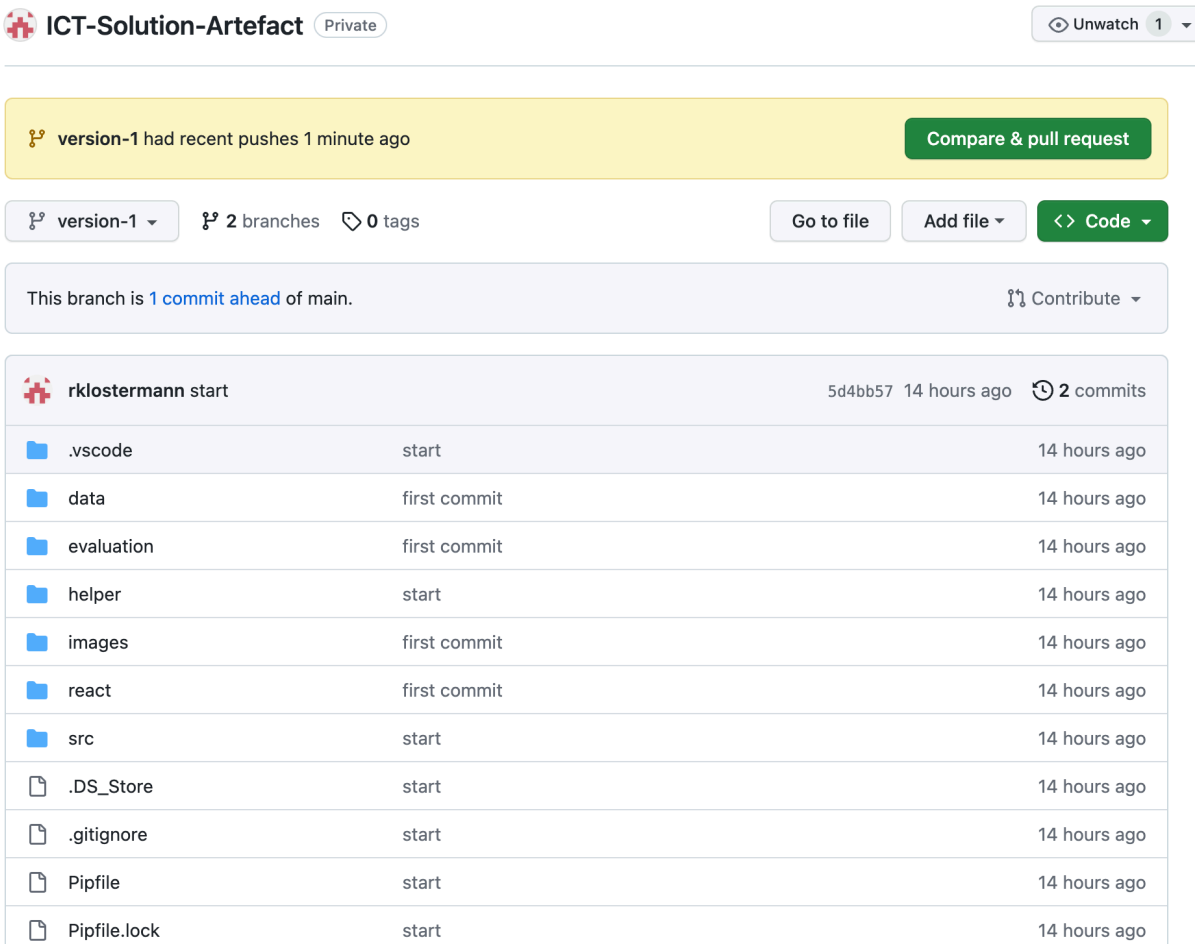


Figure 1: Project Repository in GitHub

4.1 Solution Structure

This solution's structure allows some parts to be reproduced and debugged independently. It is designed to facilitate the understanding and replication. Table 3 shows this structure.

Table 3: Solution Structure

Folder	File	Description
data	data_load.ipynb	This notebook includes code for: • Mapping Citizens Information website • Data Loading • Embeddings • Loading to the vector database
	dataset.csv	Question-Answer Dataset .csv format
	dataset.txt	Question-Answer Dataset .txt format (easy to read)
evaluation	evaluation_1_questions.ipynb	This notebook includes code for: • Random Selection of Citizen Information Pages • Exclusion of Link-Only Pages • Query-Answer Dataset Generation
	evaluation_2_metrics.ipynb	This notebook includes code for: - Model Evaluation • Answer Generation • Comparison of Real vs. Predicted Answers • Visualization with Evaluation Charts
	metrics_summary.csv	Evaluation results, including the mean and standard deviation of the metrics.
helper	conversation.py	This file includes code for: • Initialize vector database • Conversational agent • API call to the LLM
	twilio.py	This file includes code for: • Twilio setup
react	react.ipynb	This notebook includes code for: • Income Tax Simulation Toolkit • Demonstration of Chain of Thought/ReAct
	tax_template.txt	Step by step template for income tax calculation
src	app.py	This file includes code for: • Connect all the functions
main	run.py	This file includes code for: • Run the application
main	README.md	References for the code development

4.2 Collecting and Loading the Data

Implementation begins with data collection. All pages of the citizens' information website can be mapped by accessing the sitemap. Sitemaps contain all 1567 URLs of the website. The file 'data_load.ipynb' contains the Python code for this process presented in Figure 2.

```python
import requests
from bs4 import BeautifulSoup

response = requests.get('https://www.citizensinformation.ie/sitemap.xml')
assert response.status_code == 200

soup = BeautifulSoup(response.text, 'html.parser')
urls = [element.text for element in soup.find_all('loc')]
for url in urls:
    print(url)
```

Python

```
http://www.citizensinformation.ie/en/
http://www.citizensinformation.ie/en/health/
http://www.citizensinformation.ie/en/health/covid19/
http://www.citizensinformation.ie/en/health/covid19/living-with-covid19-plan/
http://www.citizensinformation.ie/en/health/covid19/public-health-measures-for-covid19/
http://www.citizensinformation.ie/en/health/covid19/covid19-isolation-and-restricting/
http://www.citizensinformation.ie/en/health/covid19/testing-for-covid19/
http://www.citizensinformation.ie/en/health/covid19/contact-tracing/
http://www.citizensinformation.ie/en/health/covid19/face-coverings-during-covid19/
http://www.citizensinformation.ie/en/health/health-system/
http://www.citizensinformation.ie/en/health/health-system/entitlement-to-public-health-services/
http://www.citizensinformation.ie/en/health/health-system/registration-of-health-and-social-care-professionals/
http://www.citizensinformation.ie/en/health/health-system/department-of-health-and-children/
http://www.citizensinformation.ie/en/health/health-system/health-service-executive/
http://www.citizensinformation.ie/en/health/health-system/general-medical-services-payments-board/
http://www.citizensinformation.ie/en/health/health-system/health-services-and-visitors-to-ireland/
http://www.citizensinformation.ie/en/health/health-system/emergency-health-services-in-ireland/
http://www.citizensinformation.ie/en/health/health-system/making-a-complaint-about-the-health-service-executive/
http://www.citizensinformation.ie/en/health/health-system/patient-advocacy-service/
http://www.citizensinformation.ie/en/health/health-system/private-health-insurance/
http://www.citizensinformation.ie/en/health/health-system/health-service-agencies/
http://www.citizensinformation.ie/en/health/medical-cards-and-gp-visit-cards/
http://www.citizensinformation.ie/en/health/medical-cards-and-gp-visit-cards/gp-visit-cards-for-under-6s/
http://www.citizensinformation.ie/en/health/medical-cards-and-gp-visit-cards/gp-visit-cards/
http://www.citizensinformation.ie/en/health/medical-cards-and-gp-visit-cards/medical-card/
...
http://www.citizensinformation.ie/en/my-situation/mysituation/
http://www.citizensinformation.ie/en/my-situation/categorytree/
http://www.citizensinformation.ie/en/my-situation/lifeevents/
http://www.citizensinformation.ie/en/my-situation/overview/
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
/opt/homebrew/lib/python3.11/site-packages/bs4/builder/__init__.py:545: XMLParsedAsHTMLWarning: It looks like you're parsing an X
  warnings.warn(
```

```python
num_pages = len(urls)
print(f'The number of pages is: {num_pages}')
```

Python

```
The number of pages is: 1567
```

This script makes use of the UnstructuredURLLoader from the langchain library to load content from a list of URLs

```python
from langchain.document_loaders import UnstructuredURLLoader
loaders = UnstructuredURLLoader(urls=urls)
document = loaders.load()
```

Python

Figure 2: Python Code for Website Mapping and Data Collection

4.3 Embeddings and Vector Database

A vector database requires splitting and embedding data before it can be integrated. A Pinecone index is created first, as illustrated in Figure 3. The name of the index and the API key are both important to record when creating this index. Pinecone index should be configured using the same parameters as the OpenAI embedding model since it generates a 1536 dimensional output and uses cosine similarity for its distance metric.

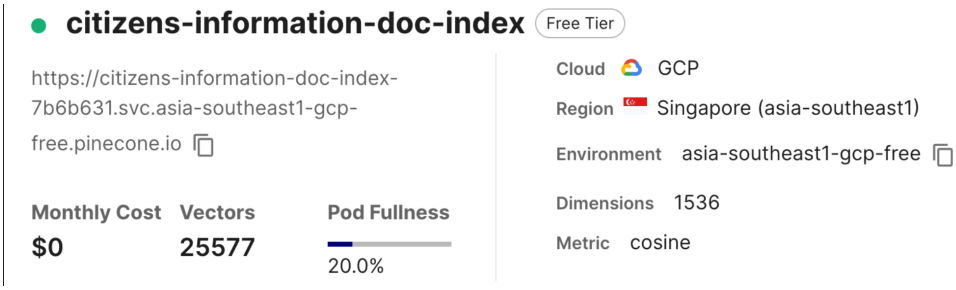


Figure 3: Pinecone Index

The process of splitting, embedding, and loading the data into the vector database is illustrated in Figure 4. The number of vectors in Pinecone also matches the quantity of vectors in Pinecone.

```
text_splitter = RecursiveCharacterTextSplitter(
        chunk_size=500, chunk_overlap=50
    )
texts = text_splitter.split_documents(document)
print(f"Split into {len(texts)} chunks")
```

Python

```
Split into 25577 chunks
```

Enter the API keys

```
from getpass import getpass
OPENAI_API_KEY = getpass("OpenAI API Key:")
YOUR_API_KEY = getpass("Pinecone API Key: ")
YOUR_ENV = input("Pinecone environment: ")
```

Python

```
index_name = 'citizens-information-doc-index'
pinecone.init(
    api_key=YOUR_API_KEY,
    environment=YOUR_ENV
)
```

Python

Embeddings and load the vectors to the vector database

```
embeddings = OpenAIEmbeddings(openai_api_key=OPENAI_API_KEY)
print(f"Going to add {len(texts)} to Pinecone")
Pinecone.from_documents(texts, embeddings, index_name=INDEX_NAME)
print("*** Loading to vectorestore done ***")
```

Python

```
Going to add 25577 to Pinecone
*** Loading to vectorestore done ***
```

Figure 4: Split, Embeddings and Vector Database

4.4 Large Language Model

After storing data in the vector database, retrieving the data in response to user inquiries is the next step. Creating a conversational model is the first step in this process. Figure 5 shows how 'conversation.py' uses the GPT-4 model. In addition, the vector database must be initialized, and this information must be incorporated into the retrieval process.

```python
# Initialize Pinecone
pinecone.init(
    api_key=os.environ["PINECONE_API_KEY"],
    environment=os.environ["PINECONE_ENVIRONMENT_REGION"],
)

INDEX_NAME = "citizens-information-doc-index"

embeddings = OpenAIEmbeddings(openai_api_key=os.environ["OPENAI_API_KEY"])
docsearch = Pinecone.from_existing_index(
    embedding=embeddings,
    index_name=INDEX_NAME,
)


# Conversation agent
def create_conversation() -> ConversationalRetrievalChain:
    llm = ChatOpenAI(model_name="gpt-4-0613", temperature=0.0)
    memory = ConversationBufferMemory(
        memory_key="chat_history",
        return_messages=True,
        output_key="answer",
        verbose=True,
    )
    qa = ConversationalRetrievalChain.from_llm(
        llm=llm,
        chain_type="stuff",
        memory=memory,
        retriever=docsearch.as_retriever(),
        return_source_documents=True,
        verbose=True,
    )

    return qa
```

Figure 5: Conversation Agent

4.5 Application

By using the Twilio API, a Flask application is set up to handle incoming user queries from WhatsApp. In conversation.py, the create_conversation function creates a conversation model that responds to user queries. The send_message function in twilio.py makes it possible to send a message to a user through WhatsApp. Flask has two routes:
The root route (/) confirms that the application is running and capable of handling requests. /twilio provides a specific route for Twilio API POST requests. This route extracts the user's message and sender ID (WhatsApp number) from the request. By using the conversation model, an appropriate response is generated. The response and any relevant source URLs are returned to the user by using the send_message function. For debugging purposes, print statements also provide visibility into incoming queries and generated responses. This is illustrated in Figure 7.

```python
from flask import Flask, request
from helper.conversation import create_conversation
from helper.twilio_api import send_message

qa = create_conversation()

app = Flask(__name__)


@app.route("/", methods=["GET", "POST"])
def home():
    return "OK", 200


@app.route("/twilio", methods=["POST"])
def twilio():
    query = request.form["Body"]
    sender_id = request.form["From"]
    print(sender_id, query)
    res = qa({"question": query, "chat_history": {}})
    output = f"Answer: {res['answer']}\nURL Source:\n"
    unique_urls = set(
        document.metadata["source"] for document in res["source_documents"]
    )
    for i, url in enumerate(unique_urls, start=1):
        output += f"{i} - {url}\n"

    print(res)

    send_message(sender_id, output)
    return (
        "OK",
        200,
    )
```

Figure 7 – Application ‘app.py’

4.6 Running the Application

For successful system integration and application launch, one must follow the subsequent steps:

1. Run the file ‘run.py’
2. In the ‘Terminal’ click on the URL as in Figure 8. The web browser opens, and the OK message confirms it works.
3. Open another terminal window and type ‘ngrok http 5002’. Figure 9 shows connection with ngrok.
4. Click on the Forwarding URL, in this case it is https://b3b1-2a02-8084-6aa4-3500-78de-dcbe-1a4d-30e1.ngrok-free.app. Again, a confirmation message with ‘OK’ will be presented in the web browser.

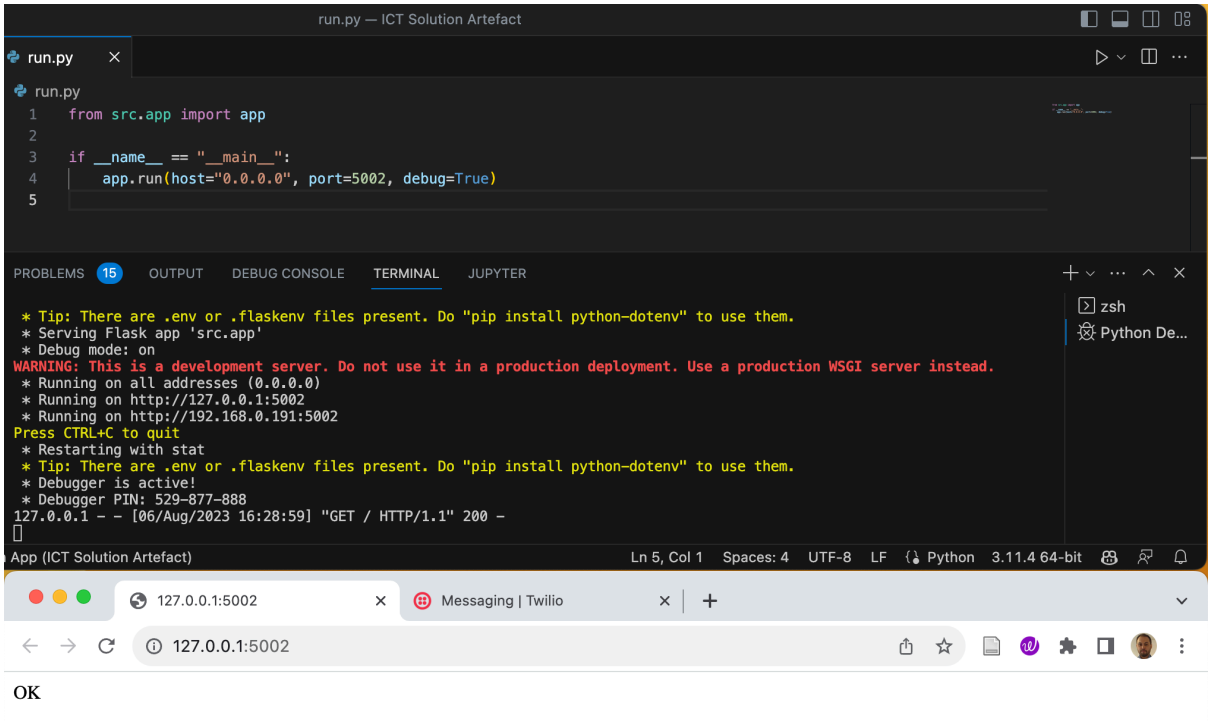


Figure 8: Connection

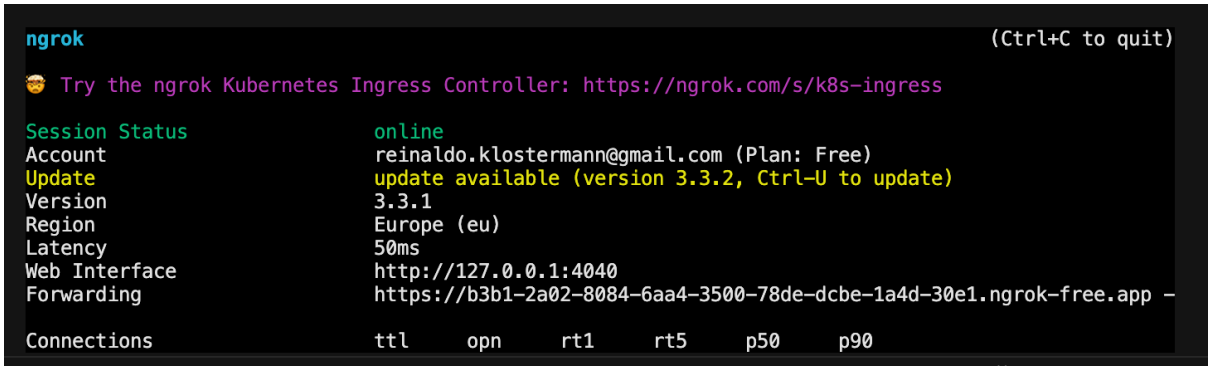


Figure 9: ngrok connection

5. In Twilio web site, access the tab ‘Send a WhatsApp Message’ follow the instructions scanning the QR code and connecting with the Sandbox as presented in Figure 10.
6. Then in ‘Sandbox settings’ paste the URL from step 4. Illustrated in Figure 11. In the end add ‘/twilio’ and hit save.

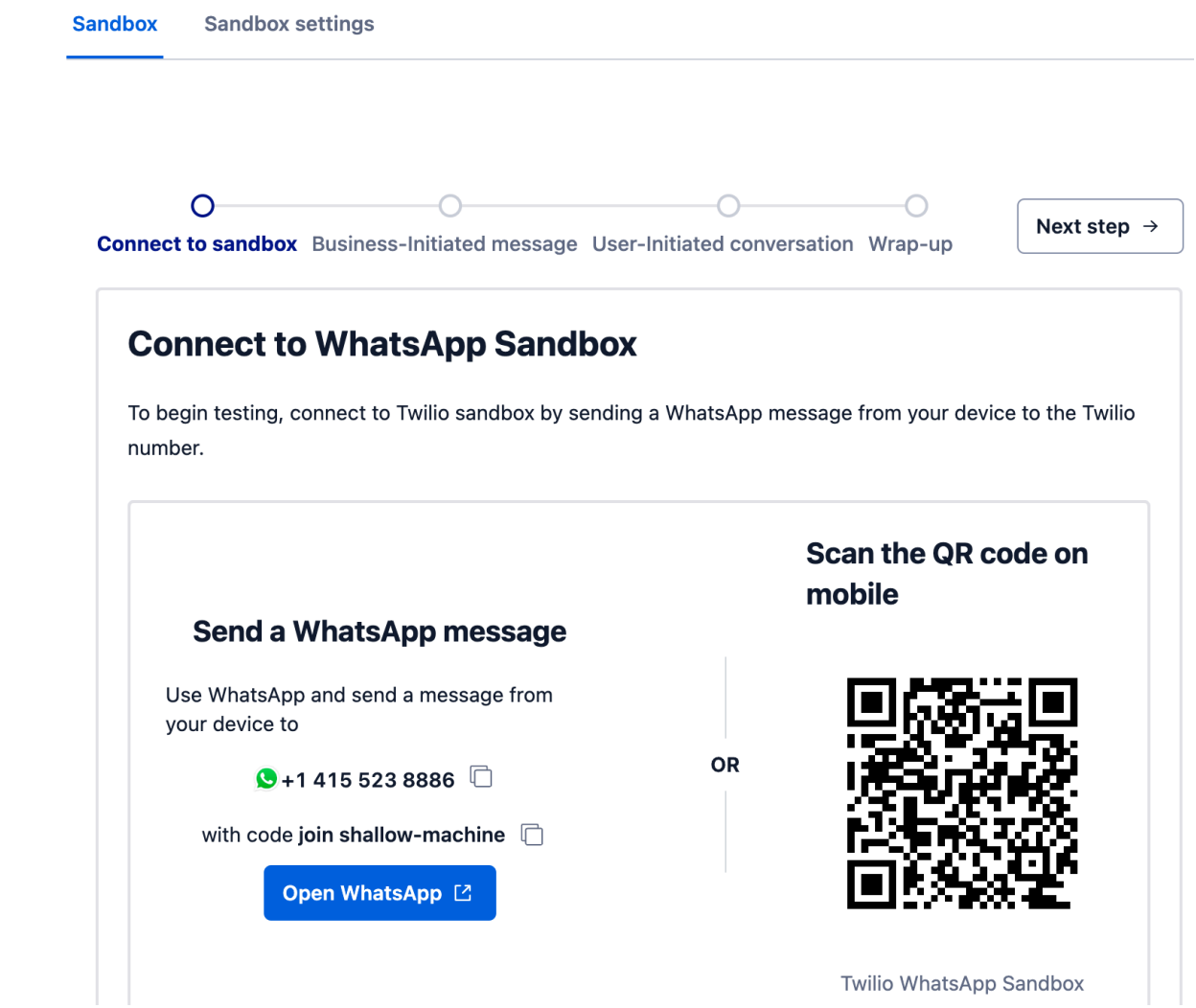


Figure 10: Twilio Sandbox WhatsApp connection

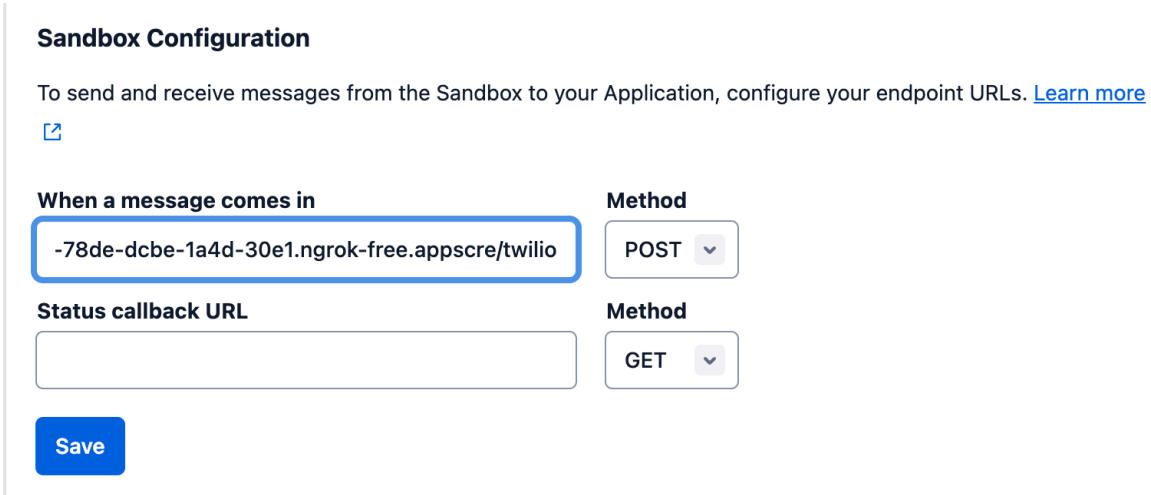


Figure 11: Twilio Sandbox Configuration

Now, the LLM is connected to WhatsApp. When a message is sent, it should appear in the 'Terminal'. As shown in Figure 12, a question sent from WhatsApp is displayed in the terminal. On the left, the LLM chain of thought is visible (Wei *et al.*, 2023), while the connection is shown on the right. Notably, the mention of 'POST/Twilio 200 OK' in the requests signifies the system's operational status.

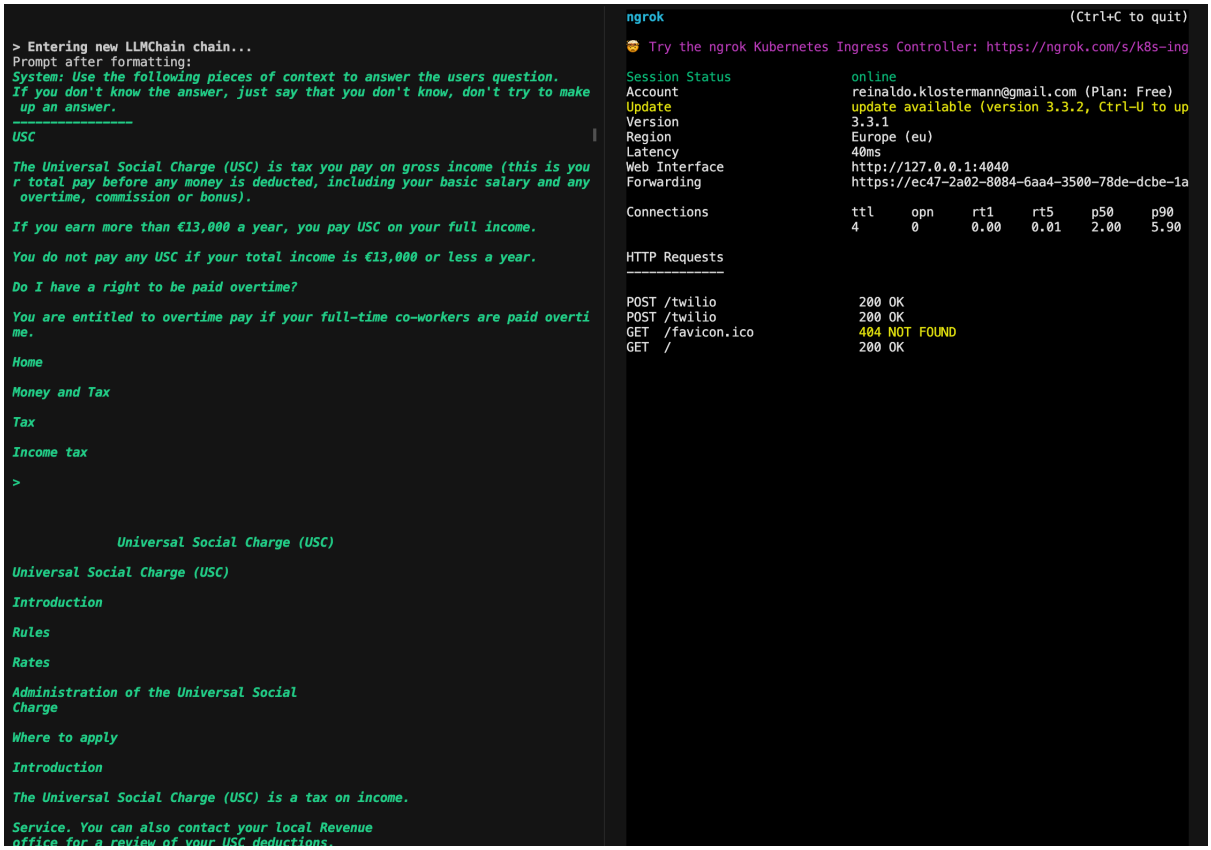


Figure 12: Terminal when the user asks a question.

4.7 Reasoning and Acting

The 'react.py' file contains the template for computing income taxes. This template, designated tax_template.txt, is intended to assist the LLM in providing accurate answers. It accomplishes this by guiding the LLM through a series of methodical steps. On the 'Terminal', Figure 13 demonstrates how this template is prompted to the LLM, underscoring its utility in determining income taxes accurately. The prompt is based on ReAct (Yao *et al.*, 2023).

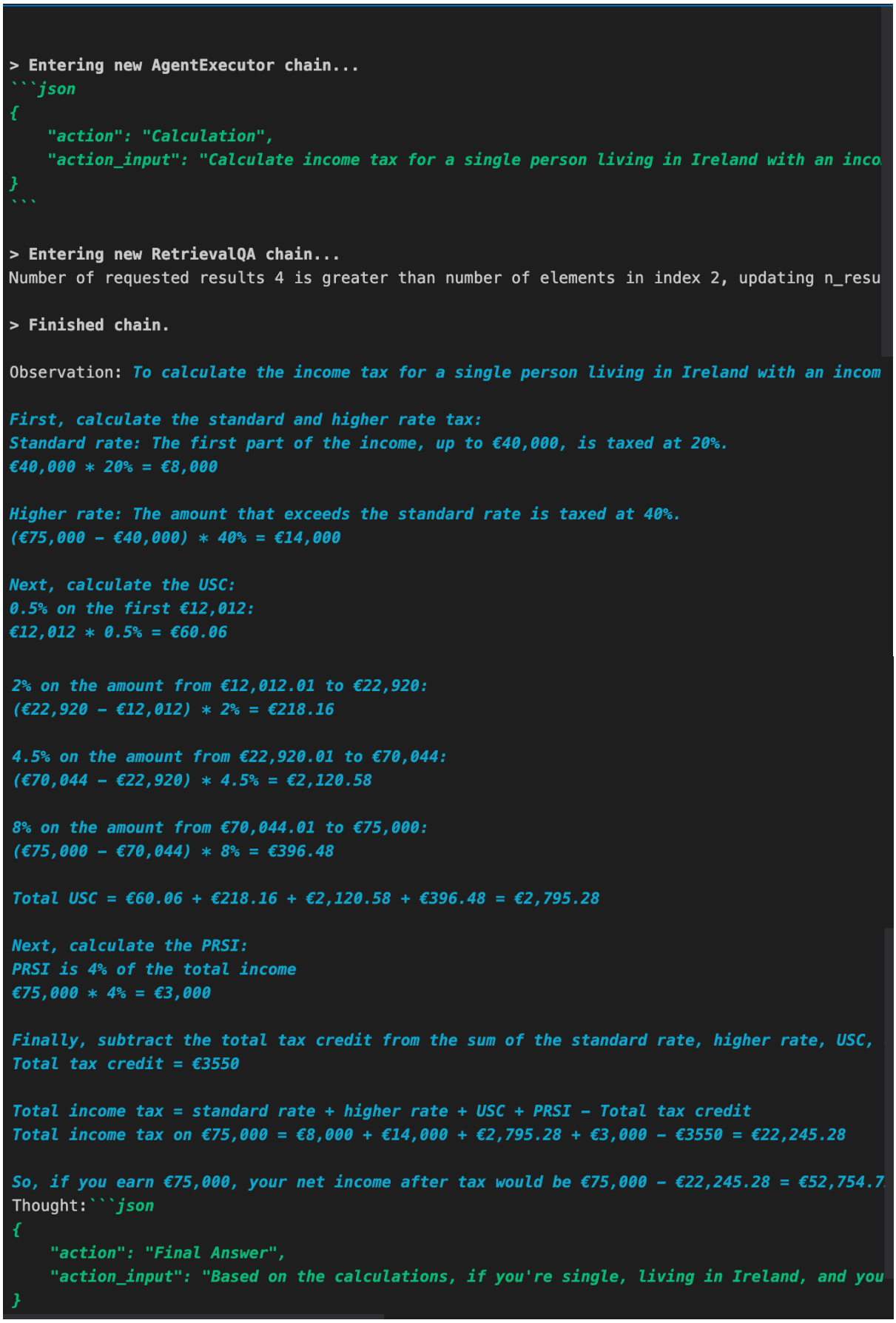


Figure 13: Terminal Prompt of Tax Calculation.

4.8 Evaluation

There are two files in the 'evaluation' folder. The ‘evaluation_1_questions.ipynb’, is responsible for question generation and dataset creation. The content from previous years deemed less relevant is omitted from the initial 1567 pages. A random selection of 15 pages is then made. The pages that consist solely of links have been removed, leaving 13 pages. Splitting, embedding, and storing them in a vector database are standardized procedures. The LangChain function is used to construct questions from various segments of the stored vector. Upon completion of this process, a dataset consisting of questions derived from various pages of the original data is presented in Figure 14.

```python
from langchain.evaluation.qa import QAGenerateChain

example_gen_chain = QAGenerateChain.from_llm(OpenAI())

examples = example_gen_chain.apply_and_parse([{"doc": t} for t in texts])

examples
```

Python

```
[{'query': "What are the requirements for qualifying for Jobseeker's Benefit?",
  'answer': "To qualify for Jobseeker's Benefit (JB) you must be aged under 66 and have social insurance (PRSI) cont
 {'query': "What must an individual be aged under in order to qualify for Jobseeker's Benefit (JB)?",
  'answer': '66'},
 {'query': "How much of the normal rate of Jobseeker's Benefit is deducted for each day of part-time work?",
  'answer': "1/5th of the normal rate of Jobseeker's Benefit is deducted."},
 {'query': "What must have occurred for a person to be eligible for Jobseeker's Benefit?",
  'answer': "They must have suffered a substantial loss of employment in any period of 7 consecutive days, have lost
 {'query': "How many weeks of Class A, H or P PRSI paid contributions are required to qualify for Jobseeker's Benefi
  'answer': "At least 104 weeks of Class A, H or P PRSI paid contributions are required to qualify for Jobseeker's B
 {'query': "How many Class A, H, or P contributions must be paid in order to qualify for Jobseeker's Benefit?",
  'answer': '104 Class A, H, or P contributions.'},
 {'query': "What must an individual have done to be disqualified from receiving Jobseeker's Benefit?",
  'answer': "An individual may be disqualified from getting Jobseeker's Benefit for 9 weeks if they left work volunt
 {'query': "How long is a period of disqualification for Jobseeker's Benefit if an individual receives a redundancy
  'answer': '8 weeks'},
 {'query': "How many PRSI contributions must be made to re-qualify for Jobseeker's Benefit?",
  'answer': "13 PRSI contributions for at least 13 weeks must be made to re-qualify for Jobseeker's Benefit."},
 {'query': "What must a part-time or systematic short-time worker have suffered in order to re-qualify for Jobseeker
  'answer': 'They must have suffered a substantial loss of employment.'},
 {'query': "What is the maximum amount of time a claimant can sign off of their Jobseeker's Benefit claim before it
  'answer': '26 weeks.'},
 {'query': 'How many days must a claimant wait before receiving a payment if they are making a new Jobseekers Benefi
  'answer': '3 days'},
 {'query': "What is the maximum personal rate of Jobseeker's Benefit in 2023?",
...
  'answer': 'The first step is usually to speak to the class teacher.'},
 {'query': "What should be done if a complaint is made about a teacher's fitness to teach? ",
  'answer': "The complainant should first use the school's complaints procedure. If the outcome is unsatisfactory, a
 {'query': 'What is the phone number for the Citizens Information Phone Service?',
  'answer': '0818 07 4000'}]
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Figure 14: LangChain QA generation Code

he subsequent file, evaluation_2_metrics.py, contains the code for comparing actual responses (originating from dataset creation) with predicted responses (generated by the chatbot). Four distinct metrics are used in this comparison. The methodology is based on the Inspired

Critique[2]. Figure 15 illustrates the code associated with these metrics visually. Figure 16 shows a data frame containing metrics results created and presented in chart format.

This configuration manual provides information on the essential software tools and settings required to replicate an experimental setup successfully. From the LLM's connection with WhatsApp to the details contained in the 'evaluation' folder, this guide guides the reader through the core aspects of the program. To focus on the specific aspects of this project, the manual deliberately avoids discussing standard software installations. The purpose of implementing this approach is to provide individuals seeking to replicate the setup with a clear path.

[2] Inspired Critique: https://docs.inspiredco.ai/critique/

```python
import inspiredco.critique
CRITIQUE_API_KEY = getpass("Critique API Key:")
critique = inspiredco.critique.Critique(api_key=CRITIQUE_API_KEY)
```

Python

```python
metrics = {
    "rouge": {
        "metric": "rouge",
        "config": {"variety": "rouge_l"},
    },
    "bert_score": {
        "metric": "bert_score",
        "config": {"model": "bert-base-uncased"},
    },
    "uni_eval": {
        "metric": "uni_eval",
        "config": {"task": "summarization", "evaluation_aspect": "relevance"},
    },
    "bleu": {
        "metric": "bleu",
        "config": {"variety": "bleu"}}
}

critique_data = [
    {"target": pred["result"], "references": [pred["answer"]]} for pred in predictions
]
eval_results = {
    k: critique.evaluate(dataset=critique_data, metric=v["metric"], config=v["config"])
    for k, v in metrics.items()
}
```

Python

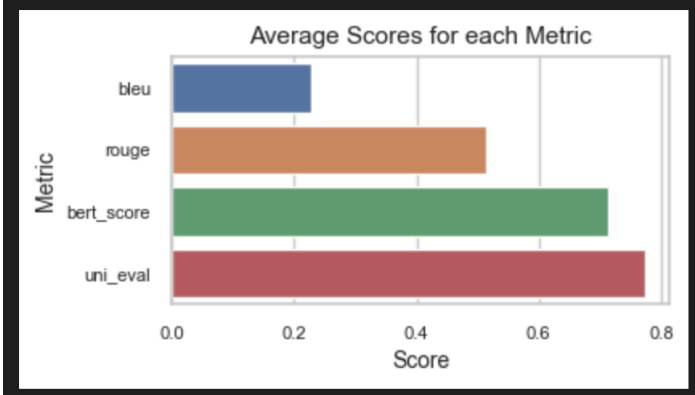
```python
for i, eg in enumerate(examples):
    score_string = ", ".join(
        [f"{k}={v['examples'][i]['value']:.4f}" for k, v in eval_results.items()]
    )
    print(f"Example {i}:")
    print("Question: " + predictions[i]["query"])
    print("Real Answer: " + predictions[i]["answer"])
    print("Predicted Answer: " + predictions[i]["result"])
    print("Predicted Scores: " + score_string)
    print()
```

Python

```
Example 0:
Question: What are the requirements for qualifying for Jobseeker's Benefit?
Real Answer: To qualify for Jobseeker's Benefit (JB) you must be aged under 66 and have social insurance (PRSI) cont
Predicted Answer:  To qualify for Jobseeker's Benefit (JB), you must be aged under 66, unemployed (fully unemployed
Predicted Scores: rouge=0.6129, bert_score=0.7913, uni_eval=0.9140, bleu=0.2312

Example 1:
Question: What must an individual be aged under in order to qualify for Jobseeker's Benefit (JB)?
Real Answer: 66
Predicted Answer:  66
Predicted Scores: rouge=1.0000, bert_score=1.0000, uni_eval=0.9413, bleu=0.0000
```

Figure 15: Evaluation Metrics Code

```python
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

sns.set(style="whitegrid")

metrics_summary = pd.read_csv('metrics_summary.csv')

metrics_summary = metrics_summary.sort_values(by='mean')

plt.figure(figsize=(4, 2))
sns.barplot(x='mean', y='metric', data=metrics_summary)
plt.title('Average Scores for each Metric', fontsize=11)
plt.xlabel('Score', fontsize=10)
plt.ylabel('Metric', fontsize=10)
plt.xticks(fontsize=8)
plt.yticks(fontsize=8)
plt.show()

plt.figure(figsize=(4, 2))
sns.barplot(x='std_dev', y='metric', data=metrics_summary, order=metrics_summary['metric'])
plt.title('Std of Scores for each Metric', fontsize=11)
plt.xlabel('Standard Deviation', fontsize=10)
plt.ylabel('Metric', fontsize=10)
plt.xticks(fontsize=8)
plt.yticks(fontsize=8)
plt.show()
```



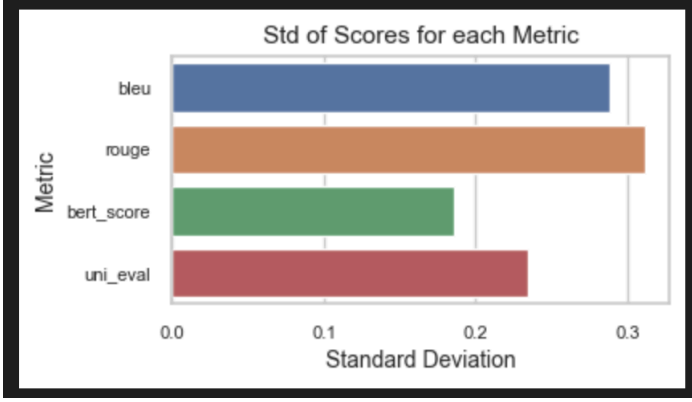


Figure 16: Evaluation Charts

References

Wei, J. *et al.* (2023) ‘Chain-of-Thought Prompting Elicits Reasoning in Large Language Models’. arXiv. Available at: http://arxiv.org/abs/2201.11903 (Accessed: 3 June 2023).

Yao, S. *et al.* (2023) ‘ReAct: Synergizing Reasoning and Acting in Language Models’. arXiv. Available at: http://arxiv.org/abs/2210.03629 (Accessed: 3 June 2023).