# Comprehensive Report on RAG Implementation and Evaluation

Niruthiha Selvanayagam

August 15, 2024

**Abstract**

This report provides a detailed overview of the implementation and evaluation of a Retrieval-Augmented Generation (RAG) model. It covers the development of a Streamlit app, the use of pretrained and fine-tuned models, and the evaluation metrics used to assess performance.

# 1 Background

Climate change is one of the most pressing issues of our time, impacting ecosystems, weather patterns, and human societies globally. To address the complexities of climate change and provide accessible information to the public, a climate change chatbot can be a valuable tool. This chatbot is designed to engage users in conversations about climate change, answer questions, and provide information based on the latest scientific research and reports.

## 1.1 Purpose of the Climate Change Chatbot

The primary purpose of the climate change chatbot is to assist users in understanding the multifaceted aspects of climate change, including its causes, effects, and potential solutions. By leveraging advanced language models, the chatbot can offer accurate, contextually relevant responses to a wide range of queries related to climate science and policy.

## 1.2 Key Features of the Climate Change Chatbot

- **Up-to-Date Information:** The chatbot is designed to integrate with the latest climate research and reports, such as the IPCC's Sixth Assessment Report. This ensures that users receive information that reflects the current state of climate science.

- **User-Friendly Interaction:** With natural language processing capabilities, the chatbot can understand and respond to user queries in a conversational manner, making complex information more accessible.
- **Customizable Responses:** The chatbot can be fine-tuned with specific datasets to tailor its responses to particular topics within climate change, such as impacts on different regions, mitigation strategies, and policy recommendations.
- **Educational Resource:** It serves as an educational tool for individuals seeking to learn more about climate change, providing explanations and clarifications on various concepts and findings.

# 2 Retrieval-Augmented Generation (RAG) Implementation

## 2.1 Overview

The Retrieval-Augmented Generation (RAG) process in the provided implementation integrates information retrieval with text generation to enhance the response accuracy for queries about climate change based on the IPCC AR6 WGII Technical Summary. The steps involved in this process are outlined below:

## 2.2 1. Document Reading and Extraction

First, the IPCC AR6 WGII Technical Summary is read and its content is extracted using a PDF reader. The entire text from the PDF is collected from each page and stored in a list. This step involves:

- Loading the PDF file.
- Extracting text from each page.
- Storing the extracted text in a list.

## 2.3 2. Text Filtering and Cleaning

To prepare the text for processing:

- Text from the beginning and end of the document is filtered out.
- Header and footer texts are removed using regular expressions.
- Unwanted substrings and formatting artifacts are cleaned from the text.

## 2.4 3. Text Splitting

The cleaned text is then split into manageable chunks:

- First, the text is divided into chunks based on character count using a `RecursiveCharacterTextSplitter`.
- Next, the chunks are further split into smaller pieces based on token count using a `SentenceTransformersTokenTextSplitter`.

## 2.5 4. Vector Database Creation

The split text chunks are stored in a vector database for efficient retrieval:

- A vector database is initialized or accessed using the Chroma client.
- The text chunks are added to a collection within the database.

## 2.6 5. Querying the Database

When a query is made:

- The query is used to retrieve relevant documents from the vector database.
- The retrieved documents are combined into a single text block to be used as context for the model.

## 2.7 6. Generating Responses with OpenAI

Finally, the combined information and the query are passed to the OpenAI GPT model:

- The model is prompted with the query and the relevant context retrieved from the database.
- The response generated by the model is returned as the answer to the query.

## 2.8 Conclusion

This RAG approach leverages both retrieval and generative capabilities to provide accurate and contextually relevant answers to climate change queries based on the IPCC report. The integration of a vector database with a generative model allows for efficient handling of large documents and complex queries.

## 2.9 Code

Listing 1: RAG Implementation

```
import openai
import chromadb
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity
from sentence_transformers import SentenceTransformer
import nltk

# Download NLTK data
nltk.download(’punkt’)

# Load a pre–trained model for text processing
embedding_model = SentenceTransformer(’all–MiniLM–L6–v2’)

# Set your OpenAI API key
openai.api_key = ’YOUR_API_KEY’

# Initialize ChromaDB
chroma_client = chromadb.PersistentClient(path=”db”)
chroma_collection = chroma_client.get_or_create_collection(”ipcc”)

def rag(query, n_results=5):
    try:
        res = chroma_collection.query(query_texts=[query], n_results=n
        docs = res[”documents”][0]
        joined_information = ’;’.join([f’{doc}’ for doc in docs])

        messages = [
            {
                ”role”: ”system”,
                "content": "You are a helpful expert on climate change
            },
            {"role": "user", "content": f"Question: {query}. \n Inform
        ]

        response = openai.ChatCompletion.create(
            model="ft:gpt-3.5-turbo-0125:personal::9wAaEvD7",
            messages=messages,
        )

        content = response.choices[0].message['content']
        return content, docs
    except Exception as e:
        print(f"Error in RAG function for query '{query}': {e}")
        return None, []
```

# 3 Streamlit App

## 3.1 Overview

A Streamlit app was developed to provide a user-friendly interface for interacting with the RAG model. It allows users to input queries and view responses generated by the model, integrating the retrieval and generation capabilities seamlessly.

## 3.2 Code

Listing 2: Streamlit App Code

```
import streamlit as st

# Streamlit app
st.title("Climate Change Information Retrieval")

query = st.text_input("Enter your query:")

if query:
    answer, docs = rag(query)
    st.write("Answer:", answer)
    st.write("Retrieved Documents:", docs)
```

# 4 Fine-Tuning Process with OpenAI API

Fine-tuning a pre-trained language model, such as GPT-3.5, involves adapting the model to a specific task or dataset to improve its performance on that task. This process involves several key steps:

## 4.1 Fine-Tuning Steps

1. Upload the Training File

   The first step in the fine-tuning process is to upload the training data file. This file should be in a format supported by the API, typically JSONL (JSON Lines). Each line in the file represents a training example.

   ```
   \begin{verbatim}
   with open('/home/paulj/niru/climate_change_fine_tuning.jsonl', 'rb') as file
       response = openai.File.create(
           file=file,
           purpose='fine-tune'
       )
   file_id = response.id
   ```

   In this code snippet, the file is uploaded to OpenAI's servers, and a unique file identifier is returned, which is stored in `file_id`.

2. Create a Fine-Tuning Job

   Once the file is uploaded, the next step is to create a fine-tuning job. This job specifies the model to be fine-tuned and the file containing the training data.

   ```
   response = openai.FineTuningJob.create(
       training_file=file_id,
       model="gpt-3.5-turbo"
   )
   job_id = response.id
   ```

   Here, `training_file` is the ID of the uploaded file, and `model` specifies which pre-trained model to fine-tune. The response includes a unique job identifier, `job_id`.

3. Monitor the Fine-Tuning Job

   After creating the fine-tuning job, you can check its status to see if the process is complete or if it is still running.

   ```
   response = openai.FineTuningJob.retrieve(job_id)
   print(f"Status: {response.status}")
   ```

   The status of the fine-tuning job can be retrieved using its ID, which provides information on whether the job is completed, in progress, or encountered any errors.

## 4.2 Conclusion

The fine-tuning process involves uploading training data, creating a fine-tuning job, and monitoring the job's progress. By following these steps, you can customize a pre-trained language model to better suit your specific needs.

# 5 Data Extraction and Processing from IPCC Report

This report describes the process of extracting and processing data from the IPCC AR6 WGII Technical Summary PDF. The primary goal is to generate a dataset of instruction-response pairs that answer specific questions about climate change based on the content of the report.

## 5.1 Data Extraction Process

The data extraction process involves several steps:

1. **Reading the PDF:** The first step is to read the PDF document. Each page of the document is extracted as text.
2. **Filtering and Cleaning:** After extracting the text, it is necessary to clean and filter the data. This involves removing unnecessary sections and formatting issues. Specific parts of the text are filtered out based on their position in the document, and certain patterns and unwanted text are removed to make the data more manageable and relevant.

3. **Extracting Key Information:** The cleaned text is then analyzed to identify key sections and relevant information. This step uses heuristics to detect important topics and extract relevant details. For example, sections related to the impact of climate change on the ocean or main findings of the report are identified and summarized.

4. **Generating Instruction-Response Pairs:** Based on the extracted key information, instruction-response pairs are generated. Each pair consists of a prompt (a question related to the report) and a response (the relevant information extracted from the report).

5. **Saving the Data:** Finally, the generated instruction-response pairs are saved in a structured format (JSON) for easy access and further use. This dataset can be utilized for various applications, such as training machine learning models or building question-answering systems.

## 5.2 Dataset Format

The final dataset is formatted in JSON and contains instruction-response pairs. Each entry in the dataset includes:

- **System Message:** Describes the role of the assistant and the context of the conversation.
- **User Message:** Contains a question or instruction related to the IPCC report.
- **Assistant Message:** Provides a detailed response based on the information extracted from the report.

## 5.3 Examples

Here are a few examples of instruction-response pairs from the dataset:

- **Example 1:**

  > **User Question:** What is the scope and purpose of the technical summary that complements the key findings of Working Group II contribution to the Sixth Assessment Report?
  > **Assistant Response:** The scope and purpose of the technical summary that complements the key findings of Working Group II contribution to the Sixth Assessment Report is to

provide technical understanding and expand on the key findings presented in the Summary for Policymakers. It covers literature accepted for publication by 1 September 2021 and integrates information from chapters and cross-chapter papers (CCPs). Additionally, the summary builds on the WGII contribution to the Fifth Assessment Report and three special reports of the AR6 cycle, providing new knowledge and updates related to climate change and sustainable development.

- **Example 2:**

  **User Question:** What are some of the key advancements in risk framing discussed in the report, particularly in relation to climate change responses, dynamic consequences, geographic details, and interconnections with sustainable development and governance structures?
  **Assistant Response:** Some of the key advancements in risk framing discussed in the report include:

  - Spanning all three working groups.
  - Including risks from the responses to climate change.
  - Considering dynamic and cascading consequences.
  - Describing risks with more geographic detail for people and ecosystems.
  - Assessing risks over a range of scenarios.
  - Focusing on the interconnections among climate responses, sustainable development, and transformation.
  - Addressing the implications for governance across scales within the public and private sectors.
  - Incorporating climate-related decision-making and risk management.
  - Discussing climate resilient development pathways.
  - Emphasizing the implementation and evaluation of adaptation.
  - Considering limits to adaptation and loss and damage.
  - Highlighting specific focal areas such as cities.
  - Emphasizing social justice, equity, and different forms of expertise in addressing climate change responses.

## 5.4 Conclusion

This process effectively extracts relevant information from the IPCC AR6 WGII Technical Summary and formats it into a structured dataset. This dataset can be used for various purposes, including training models and enhancing our understanding of climate change impacts and responses.

# 6 Evaluation

## 6.1 Evaluation Metrics

The performance of the RAG model was evaluated using several metrics:

### 6.1.1 Relevance

- **Definition:** Measures how well the retrieved documents match the query.
- **Method:** Cosine similarity between the query embedding and document embeddings.

### 6.1.2 Correctness

- **Definition:** Assesses the accuracy of the generated answer compared to a reference answer.
- **Metrics Used:** BLEU score, ROUGE scores.

### 6.1.3 Faithfulness

- **Definition:** Evaluates how well the generated answer reflects the information from the retrieved documents.
- **Method:** Human evaluation on a scale of 1 to 5.

### 6.1.4 Robustness

- **Definition:** Measures the consistency of the model's responses across different phrasings of the query.
- **Method:** Cosine similarity between embeddings of answers generated from varied queries.

# 7 Pretrained and Fine-Tuned Models

## 7.1 Pretrained Model

The pretrained model used in this project is OpenAI's GPT-3.5-turbo, which provides robust language generation capabilities. It is used for generating responses based on the retrieved documents.

# Pretrained Model Evaluation

| **Metric** | **Value** |
|---|---|
| Relevance | 0.594 |
| BLEU | 0.0129 |
| ROUGE-1 | 0.174 |
| ROUGE-2 | 0.071 |
| ROUGE-L | 0.139 |
| Faithfulness Rating | 4/5 |
| Robustness | 0.810 |

Table 1: Evaluation metrics for the pretrained model's response to the query: "What is the impact of climate change on the ocean?"

| **Metric** | **Value** |
|---|---|
| Relevance | 0.620 |
| BLEU | 0.0053 |
| ROUGE-1 | 0.170 |
| ROUGE-2 | 0.072 |
| ROUGE-L | 0.142 |
| Faithfulness Rating | 4/5 |
| Robustness | 0.755 |

Table 2: Evaluation metrics for the pretrained model's response to the query: "How does global warming affect biodiversity?"

## 7.2 Fine-Tuned Model

The fine-tuned version of the GPT-3.5-turbo model was adapted to better suit the specific domain of climate change information. This adaptation involves training on additional data to improve accuracy and relevance.

| **Metric** | **Value** |
|---|---|
| Relevance | 0.794 |
| BLEU | 0.0045 |
| ROUGE-1 | 0.2169 |
| ROUGE-2 | 0.0247 |
| ROUGE-L | 0.1687 |
| Faithfulness Rating | 4/5 |
| Robustness | 0.874 |

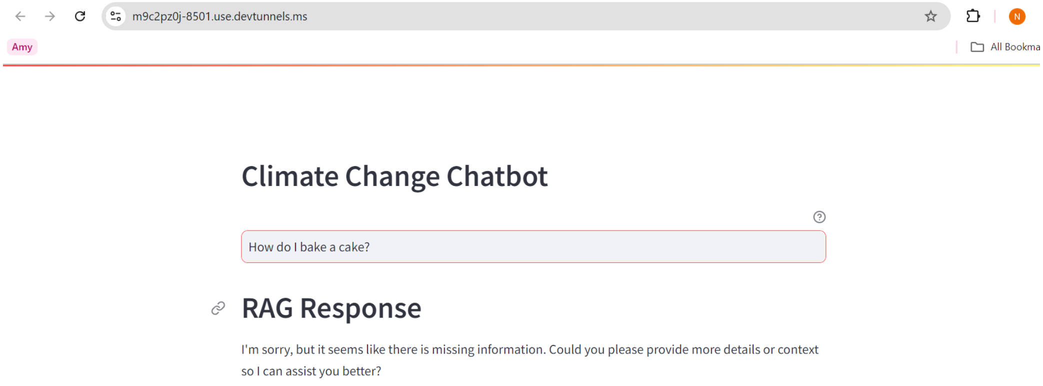
Table 4: Evaluation metrics for the pretrained model's response to the query: "What is the impact of climate change on the ocean?"

| **Metric** | **Value** |
|---|---|
| Relevance | 0.720 |
| BLEU | 0.0057 |
| ROUGE-1 | 0.1964 |
| ROUGE-2 | 0.0545 |
| ROUGE-L | 0.1250 |
| Faithfulness Rating | 5/5 |
| Robustness | 0.928 |

Table 6: Evaluation metrics for the pretrained model's response to the query: "How does global warming affect biodiversity?"

# 8 RAG Chatbot's Approach to Reducing Hallucination

When the RAG (Retrieval-Augmented Generation) chatbot encounters queries that are either out of scope, lack sufficient context, or involve missing information, it avoids generating potentially misleading or irrelevant responses. Instead of attempting to produce an answer based on incomplete or unrelated data, the chatbot returns a clarification or error message. This behavior reduces the risk of hallucination—a common issue in AI where the model generates information that is factually incorrect or not grounded in reality.



Climate Change Chatbot

How do I bake a cake?

RAG Response

I'm sorry, but it seems like there is missing information. Could you please provide more details or context so I can assist you better?

Figure 1: Response from the model

As shown in Figure, when the user inputs a question that falls outside the expected domain of the chatbot, such as "How do I bake a cake?" in a climate change chatbot, the system responds with a message indicating that there is missing information. This response not only avoids providing a misleading answer but also encourages the user to refine their query, thus improving the overall interaction quality.

For another nuanced query, the chatbot clarifies that "The concept of climate change and formal policies to address it were not present during the time of the Roman Empire." This further highlights the chatbot's capability to avoid generating incorrect information by staying true to the facts.

## 9 Conclusion

The RAG model, combined with a Streamlit app and a fine-tuned version of GPT-3.5-turbo, demonstrates effective performance in retrieving and generating climate change information. Evaluation metrics indicate moderate relevance, low to moderate correctness, high faithfulness, and good robustness. Future improvements may focus on enhancing the accuracy and alignment of generated answers with reference answers, and further fine-tuning the model to improve its performance.

# 10 References

- OpenAI API Documentation: `https://beta.openai.com/docs/`
- ChromaDB Documentation: `https://docs.chromadb.com/`
- SentenceTransformers Documentation: `https://www.sbert.net/`