# Assignment on RAGs

**Deadline:**

**Course:** Advanced Customer Analytics

**Professor:** Theodoros Lappas

## Objective

You will implement and compare **three Retrieval-Augmented Generation (RAG)** approaches using:

* A corpus of 10 .txt files (docs.zip)
* An open source LLM model (e.g., tinyllama-1.1b.Q4\_K\_M.gguf)
* An open source sentence transformer for the embeddings (e.g., all-MiniLM-L6-v2)
* A vector store for the embeddings (e.g., FAISS) and similarity retrieval
* A Q&A reference document to validate your outputs (q\_a.json)

Your answers should be **very close** to the reference Q&A doc we provide. You can use whatever **framework**, **python library** or **model** you prefer.

## Docs Overview

The text files collectively cover key topics related to **climate change**, including (but not limited to):

* Definition and scientific basis of climate change
* Greenhouse gases and the greenhouse effect
* Human and natural drivers of climate change
* Observed and projected impacts (temperature, sea level, ecosystems)
* Mitigation strategies and adaptation measures
* Environmental, economic, and societal implications

### Constraints

* The data is **static** and must not be modified semantically.
* Do **not** manually rewrite or summarize the content.
* All answers generated by the model must be grounded **only in the retrieved text** from these files.

## Data Setup

1. Unzip and inspect the 10 .txt files
2. Preprocess the text:
   * Preprocess text if need
   * Chunk text into manageable pieces (e.g., paragraph level, sentence level etc) for embedding and retrieval.
3. Store the chunks in a vector database (e.g., FAISS or similar) after embedding them.
4. Embedding strategy: You could use a sentence transformer for the embeddings (e.g., “all-MiniLM-L6-v2”)

## Usage of the Suggested Models

### LLM model

pip install llama-cpp-python

wget -O tinyllama-1.1b.Q4\_K\_M.gguf https://huggingface.co/TheBloke/TinyLlama-1.1B-Chat-v1.0-GGUF/resolve/main/tinyllama-1.1b-chat-v1.0.Q4\_K\_M.gguf

llm = Llama(model\_path="tinyllama-1.1b.Q4\_K\_M.gguf", verbose=False, n\_ctx=2048)

output = llm("Q: What is the latest version of Python?\nA:")

print(output["choices"][0]["text"])

### Sentence Embedder

pip install faiss-cpu sentence-transformers

embedder = SentenceTransformer('all-MiniLM-L6-v2')

query\_emb = embedder.encode([query])

## 

## Approach 1 - Simple RAG

This is the baseline RAG pipeline.

### Steps

1. Embed the user query to a vector
2. Retrieve top-k chunks similar to the query from your vector store
3. Concatenate retrieved chunks with the query
4. Feed to the tinyllama model to generate an answer

## Approach 2 - Multi-Query Retriever

This technique generates multiple reformulations of a query and retrieves documents for each one.

### Steps

1. Use tinyllama to generate several query variants from the original question (e.g., paraphrases or focused sub-questions)
2. For each variant:
   * Embed the variant,
   * Retrieve top-k chunks from the vector store
3. Merge the retrieval results (union of retrieved chunks)
4. Use merged chunks plus the *original* query to generate the final output with tinyllama

## Approach 3 - HyDE (Hypothetical Document Embeddings)

This approach improves retrieval by generating a synthetic document that *represents the expected answer* before retrieval.

### Steps

1. Prompt tinyllama with the user question to generate a *hypothetical answer* document
2. Embed that hypothetical answer (the synthetic text) rather than the original query
3. Use this hypothetical embedding to retrieve similar chunks from your corpus  
   Generate the final answer using the retrieved context + original query

## 

## Evaluation and Submission

Your submission must include:

### A. Implementation Code (80%)

* A notebook or script implementing all three RAG methods
* Clear instructions on how to run it and any dependencies
* 20% RAG, 30% Multi-query RAG, 30% HyDE

### B. Report (20%)

* Description of implementation details
* For a set ofsample questions, show:
  + Retrieved chunks
  + Generated answer from each RAG approach
* Comparison with Reference Q&A
  + Compare your outputs to the given reference Q&A doc
  + Explain any differences and whether the answer is faithful to the reference
  + Aim for output quality that is **very close** to the reference doc
  + Compare the three approaches and highlight their pros and cons based on the retrieved documents and answers
* Screenshots that support the above

Good luck and Merry Christmas !