Climate Disclosures Framework

# Objective

Automatically extract and synthesize relevant information from financial institutions’ disclosures to accurately respond to climate risk assessment questionnaires. This system improves interpretability, consistency, and auditability in climate risk reporting.

# Methodology

**1. Document Collection and Parsing**

* Sources: Annual reports, climate disclosures, investor presentations.
* File Format handled: .pdf, .docx and .txt files

**2. Preprocessing & Chunking**

* Chunking strategy: Documents are split into smaller units using sentence windowing and RecursiveCharacterTextSplitter

**3. Embedding Generation**

* Model: OpenAI’s ada embedding used to vectorize the chunks
* Granularity: Embeddings are created at the chunk level for fine-grained retrieval.

**4. Vector Store Indexing**

* Tech: FAISS used to store chunk embeddings.
* Hybrid approach: Combined with sparse indexing (e.g., BM25) for robustness against lexical gaps.

**5. Query Transformation & Decomposition**

* Parsing: Complex questionnaire are decomposed into simpler sub-queries and each question is rephrased into multiple variants
* Tools: LLM-based task-specific prompt and rule-based templates.

**6. Retrieval (Dense + Sparse)**

* Dual strategy:
* Dense retrieval matches semantically relevant chunks.
* Sparse retrieval captures keyword-based matches using traditional search (BM25).
* Results are merged and deduplicated.

**7. Reranking (Cross Encoder)**

* Model: Cross-encoder (bge-reranker-large) reranks top N retrieved chunks.
* Objective: Score chunks based on their true relevance to the question using full pairwise attention.

**8. Context Construction**

* Selection: Top reranked passages are assembled into a context window within LLM token limits.
* Structuring: Context is optionally grouped by sub-question or document source to enhance traceability.

**9. Response Generation (LLM)**

* Model: GPT4 omni generates a final answer, grounded in the provided context.
* Features:

- Citations with traceable document references.

- Structured output formatting to match questionnaire templates.

# Proposed User Interface

* User uploads all relevant reports/documents etc. for an organization from which the response needs to be extracted.
* User uploads the questionnaire in .csv/.txt file and clicks on Submit button
* Once the Submit button is clicked, the end-to-end python script gets executed in the background.
* UI displays original question along with the generated response on screen one at a time.
* Option to enter a list of question(s) which were answered incorrectly by the model and request to re-run the script to get the response until all answers were generated correctly.
* A download link/button to download the results in excel format with the following fields: Questions, Response, Citation and Explanation.