Approach

This project aimed to extract meaningful information from PDF files, preprocess it, and enable efficient search and retrieval of relevant content for a company's need. The process was divided into multiple stages to ensure modularity and efficiency.

5 PDFs containing articles , QnA, and many more (totaled 100+ pages) were combined and processed into chunks.

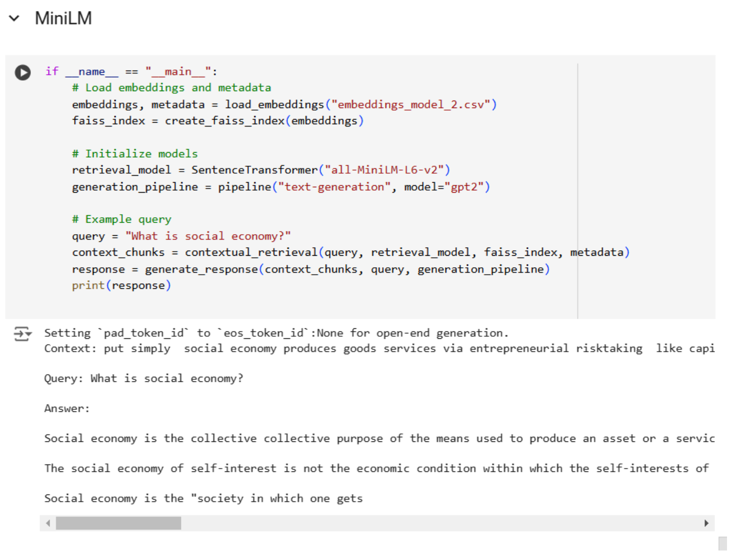
The initial step involved the installation of essential libraries and tools, including `poppler-utils`, `tesseract-ocr`, and `unstructured` for text extraction. Additional libraries, such as `sentence-transformers` for embedding and `faiss-cpu` for indexing, were also integrated. Text extraction was performed using the `Unstructured` library, leveraging its `partition_pdf()` method to handle PDFs of varying complexity. For PDFs with diverse structures, both the default and "fast" strategies were applied to ensure comprehensive text extraction.

Once the text was extracted, it underwent preprocessing, which involved cleaning, stopword removal, and tokenization into sentences using NLTK. This structured the data for downstream processes such as chunking and embedding. 248 text chunks were then created with specific token limits (`max_tokens`) and overlap to ensure compatibility with embedding models while retaining context.

Embedding creation was achieved using two pre-trained models, `distilbert-base-nli-stsb-mean-tokens` and `all-MiniLM-L6-v2`, which provided semantic representations of the text chunks. The embeddings, along with relevant metadata, were stored in CSV files to facilitate scalability and reuse. For indexing and retrieval, the FAISS library was utilized to create an optimized index of embeddings, enabling fast similarity searches. Additionally, runtime logging was implemented to monitor performance and identify bottlenecks in the workflow.

From seeing the result of each pre-trained models, `all-MiniLM-L6-v2` performs better because of more relevant answers compared to distilbert. Not just that, it requires less computational resources.

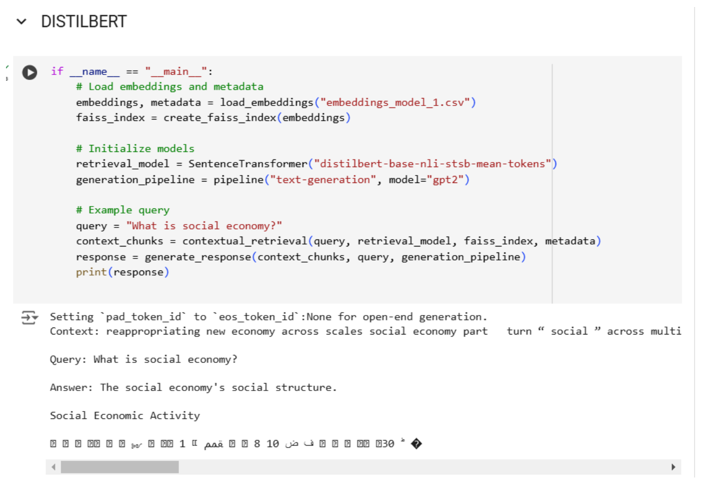
Here are the comparison of the results of each model :



MiniLM

```
if __name__ == "__main__":
    # Load embeddings and metadata
    embeddings, metadata = load_embeddings("embeddings_model_2.csv")
    faiss_index = create_faiss_index(embeddings)

    # Initialize models
    retrieval_model = SentenceTransformer("all-MiniLM-L6-v2")
    generation_pipeline = pipeline("text-generation", model="gpt2")

    # Example query
    query = "What is social economy?"
    context_chunks = contextual_retrieval(query, retrieval_model, faiss_index, metadata)
    response = generate_response(context_chunks, query, generation_pipeline)
    print(response)
```

```
Setting `pad_token_id` to `eos_token_id`:None for open-end generation.
Context: put simply  social economy produces goods services via entrepreneurial risktaking  like capi

Query: What is social economy?

Answer:

Social economy is the collective collective purpose of the means used to produce an asset or a servic

The social economy of self-interest is not the economic condition within which the self-interests of

Social economy is the "society in which one gets
```



DISTILBERT

```
if __name__ == "__main__":
    # Load embeddings and metadata
    embeddings, metadata = load_embeddings("embeddings_model_1.csv")
    faiss_index = create_faiss_index(embeddings)

    # Initialize models
    retrieval_model = SentenceTransformer("distilbert-base-nli-stsb-mean-tokens")
    generation_pipeline = pipeline("text-generation", model="gpt2")

    # Example query
    query = "What is social economy?"
    context_chunks = contextual_retrieval(query, retrieval_model, faiss_index, metadata)
    response = generate_response(context_chunks, query, generation_pipeline)
    print(response)
```

```
Setting `pad_token_id` to `eos_token_id`:None for open-end generation.
Context: reappropriating new economy across scales social economy part   turn “ social ” across multi

Query: What is social economy?

Answer: The social economy's social structure.

Social Economic Activity

□ □ □ □□ □ □ ﻴ□ □ □□ 1 ⌷ قمم □ □ 8 10 ض ف □ □ □ □□ □30 ˆ �
```

Problems and Solutions

Despite the systematic approach, several challenges arose. One significant challenge was handling complex PDFs, such as scanned documents or those with embedded images. This was addressed by integrating OCR tools like `tesseract-ocr` alongside the `Unstructured` library, ensuring effective text extraction. Another issue was managing large text data, where oversized chunks exceeded the token limits of the embedding models. To overcome this, a chunking mechanism with adjustable `max_tokens` and `overlap` parameters was implemented.