Final Model Report: Enhancing Chatbot with Hierarchical Indices for RAG and Summarization

1. Analytic Approach

- **Target Definition:**
  The primary goal of the project was to create a faithful financial but, by enhancing a baseline chatbot's ability to retrieve and generate relevant, accurate responses. This was achieved by integrating hierarchical indices into the RAG (Retrieval-Augmented Generation) framework and incorporating summarization capabilities.
- **Inputs (Description):**
  - User queries (natural language inputs)
  - Knowledge base documents (structured and unstructured text)
  - Hierarchical indices for improved document retrieval
- **Model Built:**
  The solution consists of a fine-tuned LLM (Large Language Model) integrated with a hierarchical indexing mechanism to optimize retrieval processes, coupled with summarization modules to distill information effectively.

2. Solution Description

- **Simple Solution Architecture:**
  Description and Implementation:
  The solution consists of changes and improvements to baseline functionality inside the streaming and inference pipeline:
  - **Streaming Pipeline:**
    After extracting and preprocessing the documents as in the baseline, and before inserting them into Qdrant, we added the following:
    - For each document, using the ChatGPT API, we classify it into:
      - **Sector:** The broad industry/field to which the document belongs (e.g., Healthcare, Technology).
      - **Subject:** The specific company/person/other subject mentioned in the document (e.g., Microsoft under the Technology sector).
      - **Event Type:** The type of event/activity described (e.g., for Microsoft under Technology, an event type might be a product launch).

- The classification is performed in separate steps:
    - Query to identify the **Sector**.
    - Query to identify the **Subject** within the sector.
    - Query to identify the **Event Type** within the subject and sector.
    - The resulting triplet (Sector-Subject-Event Type) is saved into the document's payload as the collection name.
    - The document is then inserted into Qdrant with its new collection name attribute.
    - While querying for documents, we maintain a tree-like structure inside a JSON file to store the hierarchy. For each document, we provide existing options for each level, asking GPT to choose the correct option if it exists or invent one otherwise.
    - Note: All documents are still saved inside the same Qdrant collection (like in the baseline). The collection name is only an attribute inside each document's payload.
- **Outcomes of Changes to the Streaming Pipeline:**
    - All documents are stored in Qdrant with a new field attached to their payload as the collection name.
    - The fully-created hierarchy is stored inside our JSON file.
- **Inference Pipeline:**
  Our changes focus on the context-retrieval part:
    - Similarly to the streaming pipeline, we classify the user query into Sector, Subject, and Event Type using GPT — this time ONLY allowing options that exist inside the JSON for each level. However, we allow several subjects and event types to capture broader context at most 3 subjects and at most 5 event types under each subject.
    - For all resulting hierarchy triplets, we fetch documents from Qdrant and select the top 10 most similar results to the query.
    - If a chosen document lacks the generated summary, we query GPT to create a concise summary.
    - Finally, we concatenate all summaries and send them to the Falcon 7B model as context for the user's query.

    - **Outcomes of Changes to the Inference Pipeline:**
        - The chatbot retrieve and generate relevant, accurate responses
        - Less hallucinations and answers are more grounded to truth

3. Data

- **Source:**
  Alpaca News API for batch and online learning datasets for fine-tuning, user query logs for evaluation.
- **Data Schema:**
- **In the streaming pipeline:**
    - **Alpaca news Document:**

```
{
    "type": "object",
    "properties": {
        "T": { "type": "string" },
        "id": { "type": "integer" },
        "headline": { "type": "string" },
        "summary": { "type": "string" },
        "author": { "type": "string" },
        "created_at": { "type": "string", "format": "date-time" },
        "updated_at": { "type": "string", "format": "date-time" },
        "url": { "type": "string", "format": "uri" },
        "content": { "type": "string" },
        "symbols": {
            "type": "array",
            "items": { "type": "string" }
        },
```

"source": { "type": "string" }

},

"required": ["T", "id", "headline", "created_at", "url", "content", "symbols", "source"]

}

- **Data in the training pipeline:**
    - Q&A object:

{

"type": "object",

"properties": {

"about_me": { "type": "string" },

"context": { "type": "string" },

"response": { "type": "string" }

},

"required": ["about_me", "context", "response"]

}

Data in the inference pipeline:

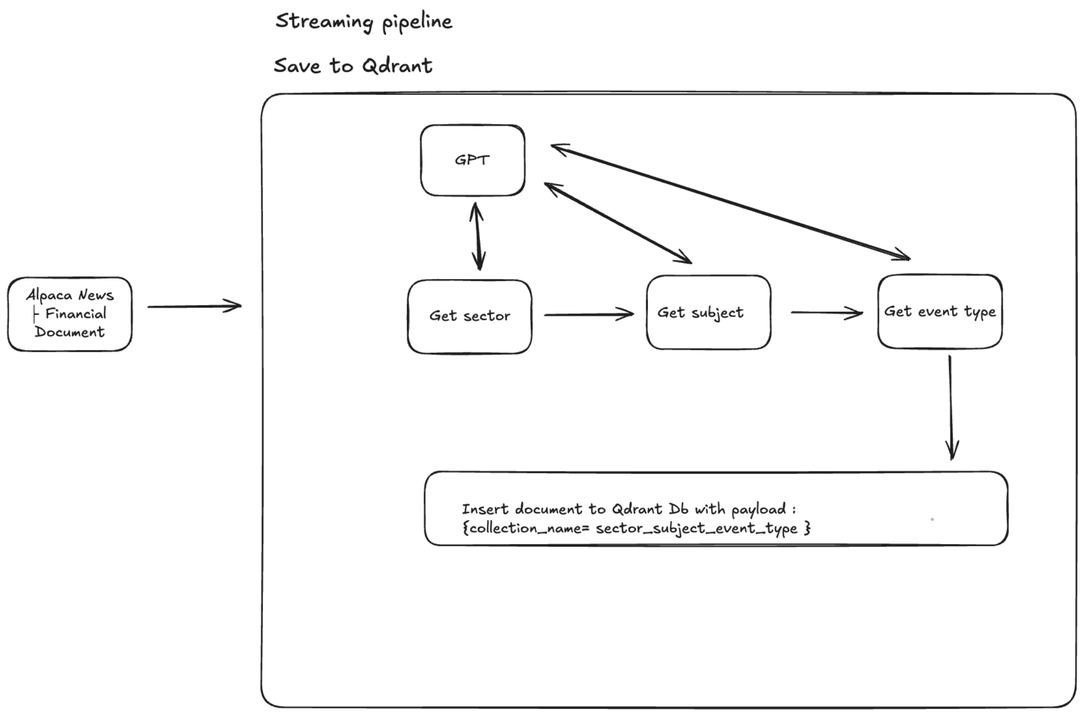
- User Query:
    - “ Query Text”:{ "type": "string" }

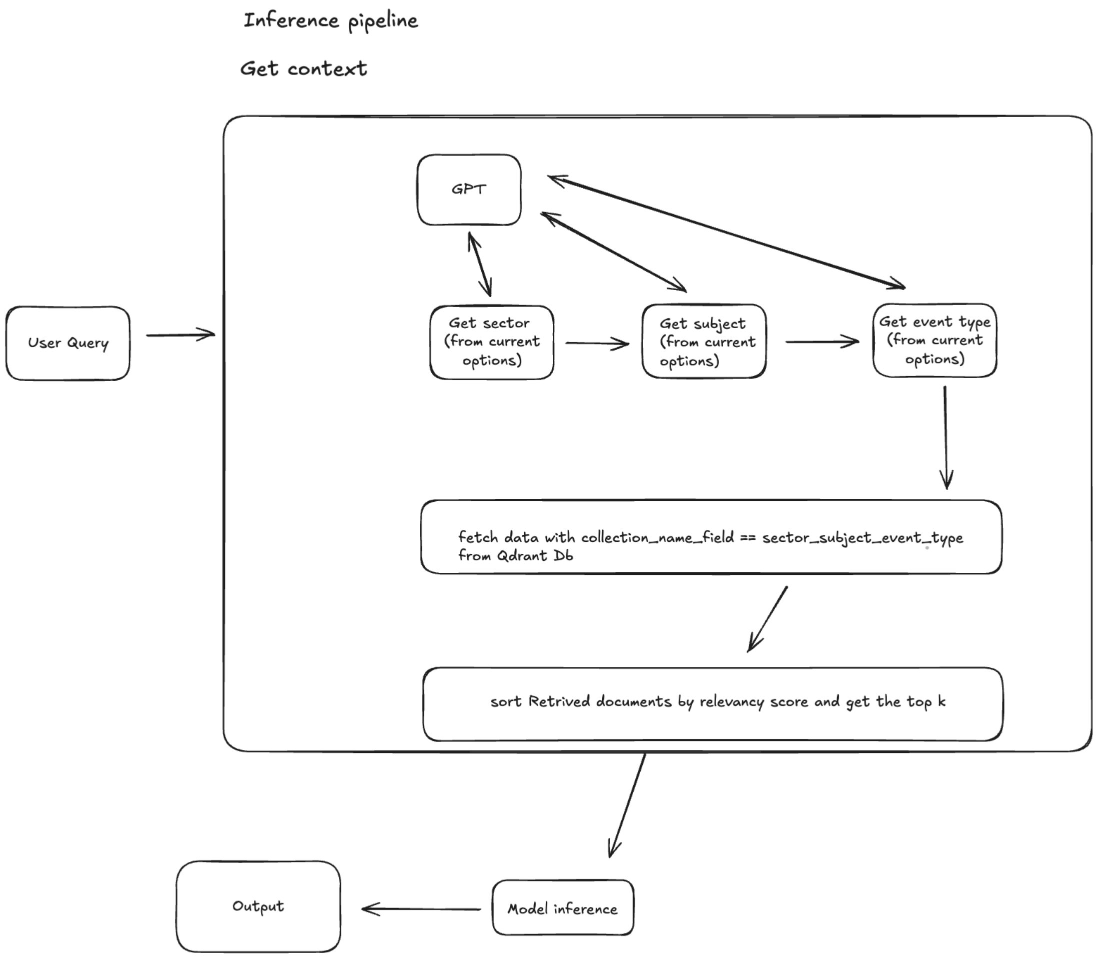
- **Selection (Dates, Segments):**
- **Streaming pipeline:**
  when running the streaming pipeline using command :”run batch” it takes data from alpaca from the past 8 days, and when working with “run batch dev” it takes from the last 2 days
- **Training pipeline:**
- The model was trained on data from 01.01.2023-05.01.2023 .

4. Features

- **List of Raw and Derived Features:**
  - Raw: Document text, document summary, query text, metadata
  - Derived: TF-IDF vectors, embeddings from pre-trained models, hierarchical index scores, sector-subject-event type classification
- **Importance Ranking:**
  - Hierarchical index scores
  - Embedding similarity
  - Query-document relevance scores
  - Sector-Subject-Event Type classification

5. Algorithm





- **Learners Used:**
    - Falcon 7B - Pre-trained LLM from hugging face, after finetuning on financial data questions.

6. Results

- **Performance Metrics:**
    - **Answer similarity:** 0.52 -> 0.64 (+0.12)
    - **Faithfulness :** 0.2932 -> 0.5392 (+0.2459)