Earnalyst-Earnings Call Q&A Bot

🌐 Overview

This Earnings Call Q&A Bot is a Retrieval-Augmented Generation (RAG) system built using LangChain, OpenAI, and Pinecone. It allows users to query a company's earnings call transcript in natural language and receive grounded, context-aware answers directly from the document, complete with speaker roles, page references, and citation-aware explanations.

🔍 Key Features

✉️ Context-Aware Q&A

- Uses OpenAI's text-embedding-3-small model to embed both the question and transcript chunks.
- Retrieves the top relevant chunks using Pinecone and formulates answers using GPT-4o-mini.
- Ensures answers are derived strictly from the document context, avoiding hallucinations.

📄 Speaker & Role Attribution

- Integrates speaker metadata (name and role) into every chunk.
- Uses a separate speaker.py module to:
    - Extract the introductory section from the PDF.
    - Automatically map speaker names to their respective roles using GPT.
- Provides role-tagged answers, e.g., "[Page 3 | John Smith, CFO]"

📈 Financial-Focused Instruction

- System prompt tuned specifically for financial document Q&A.
- Instructs the model to cite pages and reject queries if the answer isn't found in the transcript.

🌐 Fully Automated Pipeline

- Loads the transcript.
- Extracts speaker roles.
- Chunks the document.
- Adds metadata.
- Embeds content.
- Uploads to Pinecone.
- Handles user queries using an agent with the RAG tool.

✨ What Makes This Bot Stand Out

✅ Role-Based Grounding

Unlike generic RAG bots, this tool grounds each answer in speaker identity and position (e.g., CEO, CFO), enhancing trust and traceability.

✅ Real-Time Answering with Guardrails

The bot refuses to answer if the confidence is low or the information isn't in the source, reducing hallucination risk.

✅ Seamless Pinecone Integration

Automatically creates and populates a Pinecone index if it doesn't exist. Stores chunk metadata including speaker, role, and page for easy referencing.

✅ Modular Design

The logic is broken into separate components:

- main_pipeline.py: Ingestion, embedding, retrieval, and QA agent
- speaker.py: Role extraction logic (GPT-powered)

📖 How It Works

1. **PDF Loading**: Loads the full transcript using LangChain's PyPDFLoader.
2. **Intro Extraction**: Extracts the speaker introduction section for role mapping.
3. **Speaker Mapping**: Uses GPT to infer roles (CEO, Director, etc.) from intro text.
4. **Chunking**: Splits transcript into 1000-character overlapping chunks.
5. **Metadata Assignment**: Adds speaker, role, page, and date (if applicable) to each chunk.
6. **Embedding**: Generates vector embeddings using OpenAI Embeddings.
7. **Pinecone Upsert**: Uploads chunk vectors with metadata to Pinecone.
8. **Query Handling**: Accepts natural language queries, performs similarity search, returns a page-cited answer.

🤖 Sample Query

How much orders have been booked in H1?

Response:

- The bot returns a direct answer from the transcript, mentioning the page number and the speaker who provided the info.

📊 Tech Stack

- **LangChain**: Document loading, splitting, agents, tools.
- **OpenAI**: GPT-4o-mini for Q&A, Embedding-3-Small for dense retrieval.
- **Pinecone**: Vector storage and retrieval.
- **PyMuPDF**: PDF introspection.
- **Python**: Modular orchestration.

🔹 Future Enhancements

- Add hybrid search support (sparse + dense retrieval).
- Integrate diagram/image captioning from earnings decks.
- Add multi-query evaluation or chart-based answers.
- Plug in LangChain evaluation for automated QA scoring.

📁 File Structure

.

```
├── main_pipeline.py      # Main driver pipeline for ingestion + Q&A
├── speaker.py            # Helper script for extracting speaker roles
├── bel_transcript.pdf    # Input transcript file
├── .env                  # API keys and Pinecone config
```

🎉 Author

Created by Amrith Sebastin — a practical, modular tool designed for **enterprise-level retrieval of financial insights** from unstructured transcripts.