Mark – Your Personal Al Investment Assistant

Ciro Francesco Amabile, Vincenzo Carbone, Gerardo D'Arco

Introduction

Our goal is to create a virtual financial assistant capable of answering questions about publicly traded companies.

To ensure that the data is always up to date, we plan to use a **VPS** (**Virtual Private Server**), where the pipeline will run periodically using crontab.

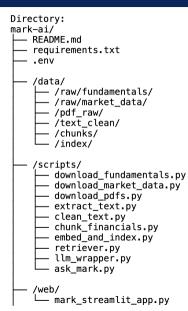
Mark is powered by a RAG (Retrieval-Augmented Generation) pipeline that combines:

- **Retrieval:** the system searches the financial data for the most relevant content related to the user's question
- Augmented Generation: the retrieved content is used as context to generate an
 accurate answer with an advanced language model (GPT)

In summary, Mark combines:

- Collected and cleaned financial data
- NLP techniques for semantic understanding
- GPT models for clear and professional answers

Project Structure



Project Structure - In Depth

Main directory: mark-ai/

- data/
 - Contains all raw and processed data. Subdivided into:
 - raw/fundamentals, raw/market_data: data downloaded via API
 - pdf_raw: company PDFs obtained via web scraping
 - text_clean: cleaned and formatted text extracted from PDFs
 - chunks: information blocks ready for embedding
 - index: FAISS vector database and mapping
- scripts/
 - Contains all Python scripts for the pipeline. Each file performs a specific function: scraping, cleaning, embedding, retrieval, and response generation.
- web/
 Future upgrade for a web-based user interface.
- README.md, requirements.txt, .env
 Support files: project documentation, required libraries, and environment variables (e.g., API keys).

Data: Collection and Cleaning

1. Automated data collection:

- We use yfinance to obtain:
 - Fundamental data: revenue, net income, P/E ratio, operating margin, ROE, etc.
 - Market data: current price, historical prices, beta, trading volumes
- Outputs are saved as .json or .csv files in the data/raw/ directory
- In parallel, we can download PDFs (quarterly reports, ESG documents, press releases) via web scraping from official websites (Investor Relations, SEC, CONSOB)

2. Cleaning and pre-processing:

- Conversion of numbers and percentages into standardized formats
- Removal of noise from text (page numbers, repeated headers, unwanted symbols)
- Normalization of content to ensure consistency across different sources
- Production of "clean" text ready for chunk segmentation

Scripts - Detailed Overview

1. Data Collection

- download_fundamentals.py: downloads key variables (revenue, income, P/E, etc.)
- download_market_data.py: retrieves stock prices, trading volumes, beta, etc

2. PDFs and Text

- download_pdfs.py: scrapes PDFs from official company websites
- extract_text.py: extracts raw text from the downloaded PDFs
- clean_text.py: cleans and normalizes the extracted content

3. Chunk Preparation

• chunk_financials.py: segments the text into semantic blocks ("chunks") with metadata

4. Embedding and Indexing

• embed_and_index.py: generates numerical embeddings and creates a FAISS index

5. User Interaction

- retriever.py: retrieves the most relevant chunks for the given question
- 11m_wrapper.py: builds the prompt and calls the ChatGPT API
- ask_mark.py: CLI script that allows direct interaction with Mark

Embedding and FAISS

1. Semantic Embedding

- Each **text chunk** (300–500 words) is transformed into a high-dimensional **numerical vector** that captures its meaning.
- We use OpenAl's text-embedding-ada-002 model, one of the most efficient and semantically accurate models available.
- Result: each information block becomes a set of numbers that reflects its semantic essence.

2. FAISS (Facebook AI Similarity Search)

- The vectors are stored and indexed in a vector store using the FAISS library, designed for fast similarity search across millions of vectors.
- When a user submits a question, it is also embedded into a vector.
- The system compares this vector with all chunk vectors and retrieves the most semantically similar ones.

Final expected Output

Example user question:

"What is Apple's P/E ratio and how does it compare to Microsoft's?"

1. Retrieval of relevant chunks

- The system converts the question into a vector embedding.
- Using FAISS, it retrieves the most semantically similar chunks (e.g., data on Apple and Microsoft).

2. Prompt construction

- The selected chunks are formatted as "context" for GPT.
- A structured prompt is built, including both the relevant data and the user's question.

3. Answer generation

- The GPT-3.5/4 model analyzes the information and generates a clear, professional answer.
- Output: a comparison of the P/E ratios, with possible comments on high/low values and market implications.

8 / 10

Scalability: Future Extensions

The project is designed to be easily scalable. Potential extensions include:

1. Integration with new data sources

- Expanding data sources via APIs (e.g., Bloomberg)
- Automated web scraping to collect data from financial portals, Investor Relations pages, and open-access databases

2. Market sentiment analysis

 Scraping of news and articles to perform sentiment analysis, useful for anticipating market reactions to events or announcements

3. Regulatory analysis and legal transparency

• Extraction of content from public institutional sources to generate responses that also reflect the **rights and obligations** of a fully informed investor

4. New user interfaces

- Integration with a Telegram bot for quick queries
- Development of a web dashboard using Streamlit or FastAPI

Thank you for your attention!

Project developed by:

- Ciro Francesco Amabile
- Vincenzo Carbone
- Gerardo D'Arco

"Investing is no longer just about numbers — it's about automated intelligence."