Machine Learning Engineer Take-Home Project

Project Overview

In this challenge, you will work with a corpus of **SEC filings** from eight public companies. Your task is to build a **semantic search pipeline** over these filings and expose that functionality via an **MCP (Model Context Protocol) server**. The goal is to assess your ability to design a useful end-to-end ML workflow, balancing NLP, systems thinking, and interface design.

This challenge is deliberately under-specified to evaluate how you approach ambiguity, make architectural decisions, and balance tradeoffs.

Objectives

You are expected to:

1. **Preprocess and chunk** the raw SEC filing documents into meaningful units suitable for embedding.
2. Generate **semantic embeddings** using OpenAI's `text-embedding-3-small` model.
3. **Upload these embeddings** to a pre-configured **Pinecone vector database**, including any metadata you believe will improve search utility.
4. Build an **MCP server** that can query your semantic index and return useful results based on natural language prompts.

Project Scope

Please aim to spend no more than **5 hours** on this exercise. Your focus should be building a clean and useful baseline implementation. If you don't have time to implement certain enhancements, briefly describe in your README how you would improve or scale your solution in the future.

Deliverables

- A complete codebase with:
    - Embedding pipeline (including preprocessing, chunking, metadata, and upload logic)
    - A working MCP server implementation (with semantic search and retrieval functionality)
    - At least 2–3 illustrative test cases added to `test_mcp.py`

    - A requirements.txt for any necessary packages
- A short README outlining:
    - Any installation instructions
    - Your architecture and design decisions
    - Your chunking/metadata strategy
    - What your MCP server does and how to use/test it
    - Ideas for next steps or future improvements

Provided Materials

- `processed_filings/`:
    - Contains folders for each company with 10-K and 10-Q raw text filings (5 years). Filenames follow: `TICKER_FORMTYPE_DATE` (e.g., `AAPL_10K_2020-10-30`)
    - Tables are denoted via `[TABLE_START]` and `[TABLE_END]` markers, `[PAGE BREAK]` markers are present to denote new pages
- `embed_skeleton.py`: Provides a `process_filings()` method that calls a `process()` function on each document. This is your entry point for preprocessing/chunking and embedding
- `clients.py`: Initializes OpenAI and Pinecone clients. Use this to avoid duplicate setup. Also initializes the preconfigured Pinecone index called `sec-embeddings`
- `.env`: Contains:
    - `OPENAI_API_KEY`
    - `PINECONE_API_KEY`
- `test_mcp.py`: Basic MCP test script that leverages the OpenAI Agents SDK to interface with your MCP server. You will need to add more test cases and complete MCP functionality
- **Note**: TODOs for preprocessing/chunking, embedding/uploading, and mcp testing are marked in the code. You will need to create the actual MCP server code/structure from scratch.

Flexibility & Expectations

- The provided skeleton code and structure is for utility and convenience only — feel free to restructure, rename, or create new functions and modules as needed.
- You may store raw document text directly in Pinecone metadata for this exercise, even though this is not best practice in production.
- If you are unfamiliar with MCP, we recommend referencing this FastMCP Quickstart Guide.

Evaluation Criteria

Your submission will be evaluated on:

- The soundness of your preprocessing and chunking strategy
- Effective use of metadata in your vector index
- Correctness and clarity of your embedding pipeline
- Usefulness and responsiveness of your MCP server
- Code quality, structure, and documentation