**LLM-Based SEC 8-K Filing Extraction Report**

**Methodology**

The project was approached in several iterations using three main Python scripts:

**LLM Document Analysis.py**

This script attempted to scrape 8-K filings using custom logic to parse the Atom feed from the SEC’s EDGAR system and extract filings directly. XML parsing, URL construction, and entity extraction were handled manually. However, it failed to generate any useful results due to limitations in locating and downloading complete submission text files and dealing with SEC rate limiting. The implementation of XML namespaces and retrieval logic for documents was inconsistent, leading to the error message: No text found in filing. Additionally, the reliance on live scraping without appropriate back-off and retry strategies caused it to hit access limits frequently.

**LLM Mock.py**

This script was created as a placeholder to simulate the expected output of the final working pipeline. It used fake or hardcoded JSON responses to mock the behavior of an LLM. While it demonstrated what the expected output should look like, it did not involve any actual text processing or real SEC data and thus did not fulfill the assignment requirements. However, it played a crucial role in shaping the correct CSV output format and helped visualize how LLM responses should be structured. Its strength was in helping shape the eventual structure of LLM output, giving a reference for both JSON parsing and CSV formatting.

**LLM 2.py**

This version focused on retrieving 8-K filings from a pre-generated browse-edgar.txt index and parsed them using XML tools. While technically more stable than the first attempt, it suffered from the same critical issues of not being able to locate and parse complete text documents correctly. Additionally, attempts to extract embedded XML from the content blocks failed due to malformed or incomplete data. This version introduced more verbose logging to debug namespace issues, but it still didn’t successfully extract any new product information.

The final version of the script switched strategies by using the sec-edgar-downloader package to reliably download 8-K filings for multiple tickers. It parsed the downloaded .txt files and sent them to a locally running LLM using Ollama. This approach introduced delays and retries to avoid overwhelming the local server. It attempted to chunk the text into 2000-character blocks and only query the first chunk. A JSON output was expected in return, with proper schema validation. LLM responses were then parsed and written to a CSV file using a pipe-separated structure.

Despite these improvements, the script did not generate 100 valid entries. Many filings either did not announce new products, resulted in timeouts, or failed to produce valid JSON responses. The local LLM struggled with long or ambiguous filings, and in many cases, responded with irrelevant or empty outputs. Additionally, because each LLM query was synchronous and slow (due to processing time), and some failed due to local server or parsing issues, the result was a CSV file with significantly fewer than 100 usable entries.

**CSV Files Analysis**

Three CSV files were generated during this project:

1. **extracted\_product\_launches.csv** — This file was generated from early versions of the pipeline and contained mostly placeholder entries or failed to populate real data. The issue was with incomplete LLM parsing and JSON extraction.
2. **product\_announcements.csv** — Generated from the final working script, this file attempted to output real results using the Ollama+Mistral stack. However, despite pulling data from up to 300 filings, fewer than 10 entries were successfully parsed with confirmed product announcements due to model timeout, irrelevant responses, or empty JSON output.
3. **extracted\_8k\_data.csv** — This CSV was generated during the development of the LLM 2.py and LLM Mock.py experiments. It suffered from similar parsing problems and contained very limited usable rows.

The CSVs also highlight one of the major bottlenecks of the project: the fact that most 8-K filings are not product announcements, and the LLM still needs to read and interpret them to make that determination. With longer inference times, even with retries and chunking, it became nearly impossible to scale to 100 cleanly extracted rows within a reasonable time.

In all cases, failure to reach 100 valid entries was primarily due to three reasons:

* Most 8-K filings do not announce new products.
* LLM inference was slow or returned incorrect outputs (e.g., boolean values instead of structured JSON).
* Several timeouts or local server connection issues with Ollama prevented full batch processing.

**Challenges Faced**

Several challenges were encountered throughout the project:

* **Data Source Complexity**: EDGAR’s 8-K filings come in various formats and levels of structure. The inconsistency in where and how data is stored made automation difficult.
* **LLM Latency and Output Reliability**: Local LLMs like Mistral via Ollama were prone to timeouts, inconsistent outputs, and misinterpretation of the prompt. These models are not fine-tuned on SEC filings, which led to many irrelevant results.
* **Timeouts and API Issues**: The Ollama server, running locally, timed out frequently, especially under batch loads. This necessitated implementation of retry logic, but it still led to high failure rates.
* **Insufficient Filings With Product Data**: Even with hundreds of 8-K filings downloaded, only a small portion contained new product announcements. The rest were earnings, personnel changes, or routine disclosures.
* **Tooling**: The sec-edgar-downloader library was the most stable part of the project, allowing consistent retrieval of documents. However, all subsequent processing was brittle due to dependency on LLM quality.

**Conclusion**

While the final script represents a functional pipeline capable of parsing SEC 8-K filings and extracting product-related information using LLMs, it is limited by inference speed, LLM output quality, and the nature of the filings themselves. Only a small subset of 8-K filings report new products, and parsing them reliably requires either fine-tuned models or more advanced heuristics. Despite iterating across three major versions of the code, none succeeded in producing a complete CSV with 100 usable product entries.

Nonetheless, the final version demonstrates end-to-end capability: reliable file download, chunked text extraction, robust prompt construction, retry-on-failure logic, and clean CSV generation. Future improvements could include:

* Switching to an API-based LLM like OpenAI for faster and more accurate outputs.
* Using embeddings or classification models to pre-filter which 8-K filings likely contain product announcements.
* Scaling with multi-threaded or asynchronous processing to reduce total runtime.
* Caching successful results and only retrying failed ones.

This project provided valuable insights into real-world application of LLMs for regulatory document analysis and highlighted the gap between theoretical capability and practical performance when working with unstructured financial disclosures.