Exit report- Indexify.AI

Customer: Anyone who's interested in investments and/or financial data- individuals, companies, financial advisors, etc.

Team members: Aviv Barel- Founder & CEO

Lihu Zur- CTO

Basel Amin- Senior Data Scientist

Ali Abu-Liel- Senior ML Engineer

Overview:

Indexify.AI is an innovative startup focused on tackling financial challenges through advanced Large Language Models (LLMs).

Recognizing the importance of high-quality context during inference, we developed an automated approach to optimize the context provided to LLMs, significantly improving accuracy and relevance.

Our solution fine-tunes a smaller LLM (Falcon 7B) using a more powerful model (GPT-4o-mini) to enhance context retrieval and generation. This ensures responses are context-rich, concise, and trustworthy while maintaining scalability across various domains.Our main goals are:

- **Context understanding-** Improving the query understanding, fetching and generating rich, highly related context.
- **Answer conciseness and relevancy**- Minimizing unnecessary details, aiming for a short, concise and relevant response. This does not contradict rich context but means that only the most relevant context will be used.
- **Scalability**- Our solution is fully automatic and generic- it is not limited to financial or any other type of data and can easily be applied and modified to fit each case.

Integrating our solution as a part of a RAG-based approach with an optimized context offers a unique, trust-worthy, and scalable financial guidance to our customers.

Business Domain:

Industry: Finance advising, Wealth Management, Investments**.**

Domain Characteristics:

- **Data intensity-** The finance industry deals with vast amounts of diverse data, including market trends, regulatory updates, and client financials. Managing and interpreting this data effectively is crucial for informed decision-making.
- **Trust and compliance-** The industry operates under strict regulatory frameworks, requiring accuracy, transparency, and adherence to compliance standards to maintain credibility and client confidence.
- **Client personalization-** Financial strategies must be customized based on individual and corporate client needs, factoring in goals, risk tolerance, and market conditions to provide optimal recommendations.

Business Problem:

Problems:

- **Unstructured and Overwhelming Data-** As mentioned above, the finance industry generates massive volumes of unorganized data. This fact makes it difficult to extract relevant insights, even for straightforward questions. Without proper structuring and contextualization, financial tools and advisors may struggle to deliver accurate and actionable guidance.
  Why it matters- Enhancing data organization and retrieval methods is essential for improving decision-making and financial recommendations.
  Relevant KPI- This problem is particularly relevant for **customer satisfaction** and **operational efficiency**, as disorganized data leads to inaccurate responses, frustrating customers and increasing the resources needed for manual intervention.

- **Limitations of Lightweight AI Models-** Smaller AI models, while more cost-efficient and easier to deploy, often fail to provide the depth of analysis needed for complex financial scenarios. Their inability to capture intricate market patterns and contextual nuances can lead to less precise recommendations, potentially resulting in financial losses or lower client satisfaction.
  Why it matters- In order to use a smaller model in practice, we must recognize and handle its limits beforehand.
  Relevant KPI- This issue directly impacts **customer satisfaction** and **conversion rates**, as inaccurate or insufficient financial advice can decrease user trust and lead to dissatisfaction, ultimately hindering users from acting on recommendations or using the service.

Data Processing:

Original Dataset Schema:

An internal set of specific Q&A examples was used to train and test the original model. Each training example included the following:

- **about_me-** General data about the user (age, financial status, areas of interest, etc.)
- **context-** background information which is relevant to the query.
- **Question-** The user's financial query (question, instruction, etc.).
- **Answer-** The system's response to the user's financial query.

Data Processing Workflow:

- **Finetuning-** Using the internal Q&A example shown above to finetune the original Falcon 7B parameters model, to better prepare it for answering various financial queries.
- **RAG Data collection-** Gather internal financial documents and additional data from Alpaca, and preprocess it into Document objects with specific attributes.
- **Document preprocessing-** Using ChatGPT API: for each document, classify it into sector, subject and event type, and add this classification to the document's payload.
- **Hierarchy generation-** While processing, create an accumulated hierarchy of sectors, subjects and event types.
- **Vector Indexing-** Insert the resulted documents into the Qdrant vector DB.
- **Documents retreival-** At inference, classify the query into sector, related subjects and relevant event type(s), infer the collection name(s). Choose the most similar documents only from the inferred collection names.
- **Context Summarization-** Summarize any resulted document which has no summary.
- **Final context creation-** Concatenate the summary attribute of all chosen documents.

Final input data Schema for Model:

- The user's original query (without any changes)
- The enhanced context.

Modelling, Validation

Modelling Techniques used:

- **Hierarchical Indexing-** Classifying each document and the user queries into 3 levels of hierarchy to better capture more concise and accurate context from the existing documents.
- **Summarization-** Creating a brief summarization for a text which includes its key ideas.

- **Query Optimization-** Develop sophisticated queries to LLM's to ensure that the responses are in the expected format and match our goal in a scalable way (works well for almost every query).

Validation approach:

We evaluated our improved model in 2 ways:

- **Manually-** Inventing queries of our own and sending each query to both the baseline model and our model. We determined which answer is better using common-sense, and without using any specific metric.
- **Evaluation on a test set-** We evaluated our model on a test set which covers a diversity of financial questions, focusing the following Ragas metrics:
  1. **Answer Similarity-** measures how closely the generated response matches a reference answer.
  2. **Faithfulness-** measures how well the response aligns with the provided source information, ensuring accuracy and minimizing hallucinations (fabricated or unsupported details).

Validation Results:

- **Manually-** On almost every query we tried, the response generated by our model seemed to be much better than the response generated by the baseline model. Our model's responses tended to be much more accurate and concise.
- **Evaluation on a test set-** We received the following scores:

Metric	Baseline Bot	Improved Bot	Difference
Answer Similarity	**0.52**	**0.64**	**0.12 (+23%)**
Faithfulness	**0.293204**	**0.5392**	**0.245 (+83%)**

We can see that in both manual and test-set evaluations, our model performs significantly better than the baseline model.

The largest improvement was in the **Faithfulness**, and results from the concise context, which helps reducing hallucinations and prioritizing high-quality summaries and data sources.

There was also a significant improvement in **Answer Similarity**, which results from the noise reduction caused by the hierarchical approach, narrowing down to relevant subsets of documents and finally resulting in a better answer (which is more like the reference answer).

Solution Architecture:

Description and implementation:

The solution consists of changes and improvements of baseline functionality inside the streaming and inference pipeline:

Streaming Pipeline:

Flow: for each document: classify it into sector → classify it into a subject under the sector → classify it to event type under the subject → add the resulted

collection name to the document payloads → insert the document to Qdrant.

Additionally, maintain a JSON file containing the total resulted hierarchy. Each

query contains the existing options and instructs GPT to invent new options only if none of the existing options matches the document.

Purpose: classify each document into 3 levels of hierarchy to improve context at inference. Additionally, save the final hierarchy in an organized, structured way.

Inference Pipeline:

Flow: User sends a query → classify the query into sector, subject(s) and event type(s) to find collection names → fetch top 10 matching documents from Qdrant according to the found collection names → if a fetched document has no summary, summarize it → concatenate all summaries and send them as context → Falcon 7B receives the query and the context and generates a response.

Purpose: Significantly improve the context sent to the Falcon 7B model by:

- Picking the most similar documents to the query only from the chosen collection names.
- Ensuring that each chosen document contributes to the context by summarizing it if no summary is originally present.

Architecture Diagram: Appears in the model report (for both streaming and inference)

Why we chose this architecture over other options:

Before choosing this architecture, we checked options for other hierarchical indexing implementations:

- Different amounts of levels and different level names.
- Different amounts of subjects and event types chosen for each query, and of documents chosen for from them.

After lots of trial and error, we came to conclusion that we should stick to a 3-level hierarchy (sector, subject, event type), choose 1 sector, at most 3 subjects and 5 event types from each subject and then pick at most 10 most similar documents from the resulted collection names. Additionally, we saw that the fetched documents were usually good, so we made sure to fill up missing summaries, to ensure that all the fetched documents add to the final context.

Benefits

Company Benefits (internal only):

- **Revenue Potential:** Our solution significantly improves the responses of the financial bot, leading to increased popularity and sales, improving the company's revenue.
- **Scalability:** Our solution is not specific for financial bot LLM architectures, but can be applied to any LLM that uses RAG, allowing us to increase our impact and expand our horizons to a variety of topics, users and companies as customers.
- **Simplicity:** Our solution is very simple and can easily be finetuned to match different use cases. This means that we won't need to invest a lot of time and resources to fit our solution to new customers.

Customer Benefits:

- **Faithfulness:** Our fine-tuned LLM significantly enhances the faithfulness of financial insights by minimizing hallucinations, ensuring users can confidently rely on the bot's recommendations. This reduces the risk of errors in decision-making, leading to more reliable financial analysis.
- **Scalability and cost efficiency:** Customers can easily adapt our model to similar tasks with minimal customization. Even for complex scenarios, only minor fine-tuning is required, reducing development time and operational costs, thus improving ROI.

Learnings:

Project Execution:

- **Expectation Alignment-** Early engagement with a variety of customers (individuals, companies, financial advisors) is crucial to capture all kinds of customer needs and demands from the product.
- **Continuous Feedback-** Keeping in touch with the customers as we progressed through different phases of the project. Each time, aligning the temporary results with customer expectations, catching misunderstandings and performing quick fixes on need. This helped to make sure the project is stays on track.

Data Science/Engineering:

- **Different LLM utilizations-** We learned that using an advanced LLM's API to improve a smaller, lightweight LLM can result in a model that is both simple and cheap to train and improve. This is because we invest resources into training only a smaller model but it performs well like a bigger model.
- **RAG optimization-** Using the Hierarchical Indexing and Summarization methods, we learned about the huge impact of the RAG utilization and the context it provides to the LLM on the performance.

Domain:

- **Context Sensitivity-** Minor details, such as regulatory disclaimers, local market dynamics, and product-specific limitations, must be incorporated into the AI workflow to ensure credibility and compliance.
- **Domain Complexity-** The financial domain covers a wide range of topics, regulations, and client needs, from investment strategies to tax compliance. AI systems must handle diverse and nuanced queries while ensuring accuracy and adherence to legal standards.

Product:

- **Hierarchical Indexing Capabilities-** We learned that this approach is simple and powerful and can be easily finetuned and adapted to many different use cases, significantly improving the model.
- **Summarization Capabilities-** We learned that a concise, straight-to-the-point summarization of our documents can enrich our context while reducing redundancies in the responses.

What's unique/Specific challenges:

- **Large and diverse document source-** Our approach assumes that our context database contains a big diversity of data, in order to find documents which match every possible query. Luckily, there is a huge amount of relevant data for almost every financial query a user may be interested in.

Links

GitHub repository link: https://github.com/AvivBarel23/hands-on-llms

Next Steps

- **Batch API calls (Basel, Q2 2025)-** Our model makes a lot of API calls to ChatGPT (3 for each document at steaming + more calls at inference). The task is to batch the API calls and apply parallelism (each document stands for itself, which makes this task "embarrassingly parallel"). This would reduce the streaming time and required resources.
- **Improve Hierarchical Storing (Ali, Q2 2025)-** The current hierarchy storage is within a simple JSON file. While JSON provides the required capabilities, storing the hierarchy inside a DB which efficiently deals with hierarchy would be much cleaner, faster and scalable.
- **Expanding client network (Lihu, Q3 2025)-** After completion and verification of the Q2 goals above, we can expand our cooperations to new and more demanding clients. Our improved model would be good enough to be used in bigger, harder tasks.

- **New ideas- (Aviv, Q4 2025)-** Our improved financial bot is working well, but not yet good enough for every possible task. Think of additional ways to improve our financial bot, for example prompt engineering tactics, etc. After weighing our options, start implementing the new ideas around the beginning of 2026.

Appendix

- **A Change from the midterm approach:**
  Before starting our work, we did not recognize the need of document summarization inside our solution. Only when we started working, we figured that in the baseline code, many documents are not having any summary and thus do not contribute anything to the final context.
- **The key metrics improvement (recap):**

Metric	Baseline Bot	Improved Bot	Difference
Answer Similarity	**0.52**	**0.64**	**0.12 (+23%)**
Faithfulness	**0.293204**	**0.5392**	**0.245 (+83%)**

- **An example response from our improved financial bot:**