Vamshi Jandhyala

When the customer is an LLM

✠

September 2023

An essay from 2023 arguing that LLMs would become primary consumers of APIs, that API documentation would become in-demand training data, and that API product managers would soon be designing for a non-human customer. Republished with a 2026 note.

Editor's note, April 2026

I wrote this in mid-2023, before the Model Context Protocol existed, before agent frameworks were running in production at scale, and before anyone was publicly tracking what fraction of frontier-lab API traffic came from non-human callers. Three claims in the original piece have since landed:

1. *LLMs become primary consumers of APIs.* They have. The median caller of a frontier-lab API is now software written by another LLM, not a human pressing a button. MCP makes this explicit at the protocol level.
2. *API documentation becomes the new in-demand training data.* It is. Labs ingest docs continuously; teams with inconsistent or sloppy docs are silently losing share inside agents that route around them.
3. *API product managers serve a new customer.* They do. "Will an agent succeed on the first call?" is now a first-class API quality metric alongside the human developer experience.

The original essay follows, with citations linked through to source. The argument is preserved as written.

Ubiquity of APIs

The dynamic nature of the digital landscape in financial services forces businesses to innovate and ship new products and services quickly. One significant factor enabling this acceleration is the efficient use of APIs. APIs are the backbone of software development: protocols and tools for building applications, methods and data formats that programs use to perform specific tasks, interact with other software, and interoperate.

Rapid product development

APIs let businesses leverage existing services rather than build from scratch. The reuse saves time and resources and reduces the surface area for bugs. APIs also extend the functionality of existing products, letting companies offer value-added features without rewriting their core stack.

Developer productivity

Well-designed APIs provide pre-defined, reusable, modular units that developers can drop into their applications. Less time on plumbing means more time on the differentiated parts of the product. Well-documented APIs further compound the gain.

Faster experimentation

APIs let developers plug in and try out functionalities cheaply, enabling iterative and agile development. Agility is decisive in a digital landscape where the ability to evolve quickly is the difference between survival and decline.

Advent of LLMs

Large language models such as OpenAI's GPT-4 are increasingly shaping the trajectory of many industries, with financial services a notable example. Their capacity to generate human-like text, understand linguistic nuance, and produce contextually appropriate responses sits at the centre of natural language processing, a capability with significant applications in finance.

Applications in financial services

The sector stands to gain substantially from the integration of LLMs:

- **Sentiment analysis.** Analysing investor sentiment from social media, forums, and news. Negative sentiment in real time can signal trouble in a company's financial health or a change in market dynamics.
- **News classification.** Categorising news events by relevance to sectors, companies, or instruments: macroeconomic trends, specific industries, individual corporations.
- **Named entity recognition.** Locating and classifying entities (organisations, locations, monetary values, dates) from complex financial documents and news.
- **Question answering.** Sophisticated systems that respond to client queries on accounts, transactions, and products, and assist financial advisors with instant access to a vast information surface.
- **Customer service.** LLM-powered chatbots handling queries from balance enquiries to explanations of investment products around the clock.
- **Risk management.** Analysing textual data such as news, reports, and filings to identify potential risks, including negative sentiment that suggests financial stress.

- **Financial analysis.** Reading reports and earnings transcripts, surfacing trends, highlighting critical data points, summarising lengthy documents.

Domain-specific LLMs

General models cover many domains and perform well across a wide variety of tasks, which can obviate the need for specialisation at training time. But existing domain-specific models show that general models cannot fully replace them. In a financial services firm there is a large and diverse set of tasks well served by a general model, yet the majority of applications are within the financial domain, and a domain-specific model is often more effective. This is why Bloomberg developed \{}href{https://arxiv.org/abs/2303.17564}{BloombergGPT}, a model that achieves best-in-class results on financial benchmarks while remaining competitive on general-purpose ones. The key is the training data: a combination of general sources and high-quality curated domain-specific datasets.

The new challenge

With the plethora of APIs and the new capabilities afforded by general models and domain-specific LLMs, the developer faces a new question: **how do I combine all of these to create a compelling product or solve a business problem?**

The approaches that address this challenge involve eliciting new capabilities from LLMs. Today's models are fundamentally limited by what they can store in a fixed weight set and what they can compute through a static computation graph and limited context. As the world changes, models require retraining to update their knowledge and reasoning. By empowering LLMs to use tools, we grant access to vastly larger and changing knowledge bases and to complex computational capacity. Search and database access expands their dynamic knowledge surface; computational tools extend their reasoning reach.

This transition, from a small set of hand-coded tools to the ability to invoke a vast space of changing cloud APIs, could transform LLMs into the primary interface to computing infrastructure and the web. Application developers in the future might interact with an API provider or store using *prompts*. For example: *"Please give me a set of APIs to summarise the content of a financial news article and convert the summary to speech."*

Two early efforts pointed at this future.

TaskMatrix.AI

\{}href{https://arxiv.org/abs/2303.16434}{TaskMatrix.AI} is an AI ecosystem that connects foundation models with millions of APIs for task completion. Rather than improving a single model, it uses an existing foundation model as a brain-like central system and APIs of other AI models and systems as sub-task solvers. The advantages it claims:

- It can perform digital and physical tasks by using the foundation model to understand inputs (text, image, video, audio, code) and then generating code that calls APIs.

- Its API platform is a repository of task experts. **All APIs on the platform follow a consistent documentation format**, making them easy for the foundation model to use and for developers to add new ones.
- It supports lifelong learning by adding new APIs with specific functions to the platform.
- It offers interpretability: both the task-solving logic and the API outcomes are inspectable.

Gorilla

\{}href{https://arxiv.org/abs/2305.15334}{Gorilla} is a fine-tuned LLaMA-7B-based model that surpasses GPT-4 on writing API calls. It substantially mitigates the hallucination problem common when prompting LLMs directly, adapts to changes in API documentation, and reasons about constraints. It is trained on a large corpus of APIs from three major model hubs: TorchHub, TensorHub, and HuggingFace. An example prompt: *"Help me find an API to convert spoken language in a recorded audio to text using Torch Hub."*

Conclusion

The API ecosystem is going through a fundamental transformation with the advent of LLMs. Companies that offer APIs leveraging the new capabilities of domain-specific LLMs (such as BloombergGPT trained on proprietary data) and that let developers and non-developers alike compose them with other popular models and APIs through *prompts* will hold a significant competitive advantage.

API product managers now have to meet the requirements of a new set of customers: the LLMs. Consistent API design and documentation will be more important than ever for search, discovery, and use through models. **API documentation will be the new in-demand training data** as APIs are increasingly consumed by machines.

As LLMs become more fluent in using APIs, product managers need to become adept at prompt engineering to generate and test new ideas quickly. Domain-specific API and model aggregators of the future will compete on the quality of the LLM that, given a prompt, can identify the right set of domain-specific models and APIs, assemble and orchestrate them, and evaluate the results to surface the optimal combination for the prompt.

Long live the prompt.