LIONPOINT
An Alpha Group Company

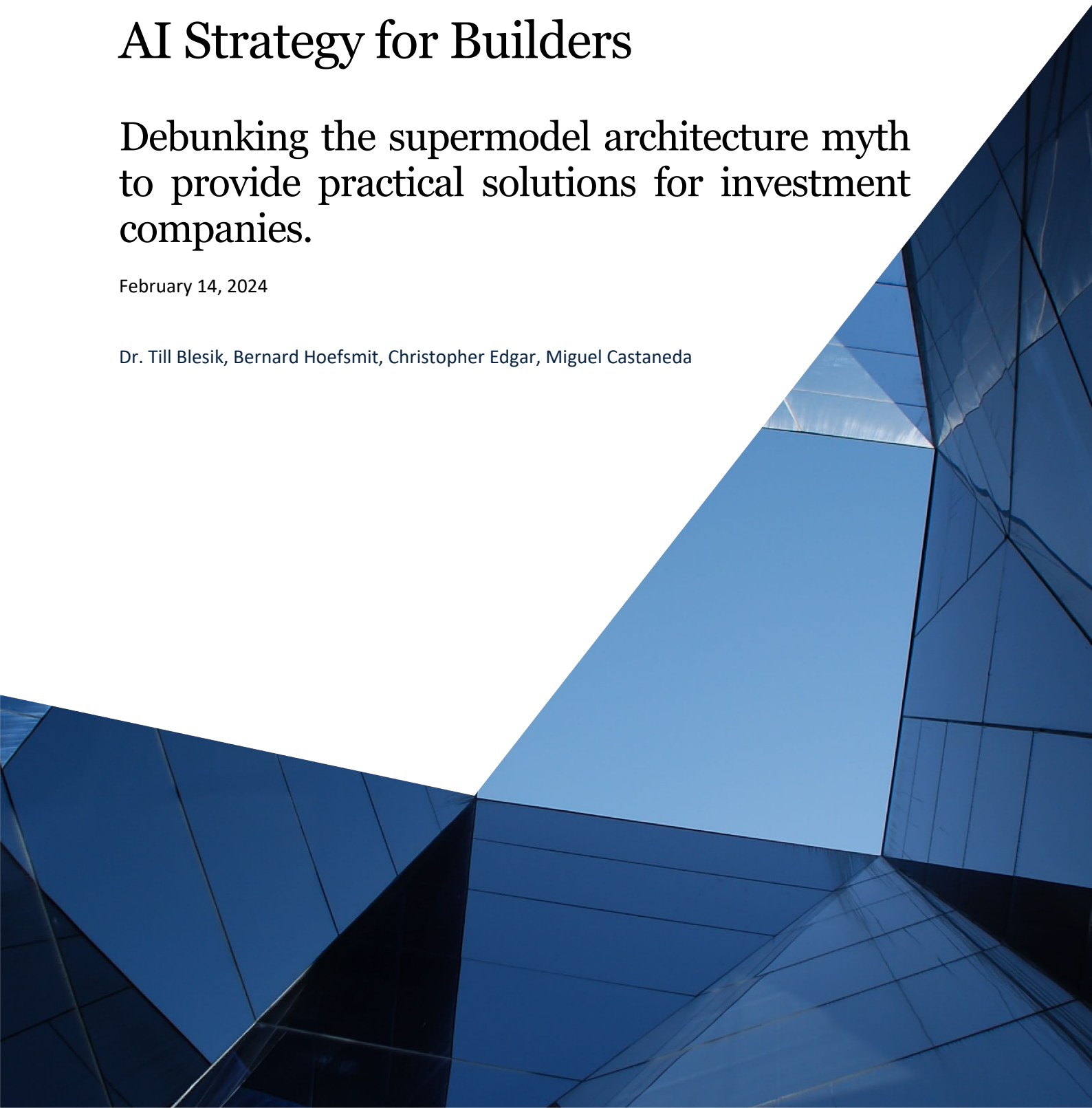
E info@lionpointgroup.com
W lionpointgroup.com
LI linkedin.com/company/lionpoint-group

AI Strategy for Builders

Debunking the supermodel architecture myth to provide practical solutions for investment companies.

February 14, 2024

Dr. Till Blesik, Bernard Hoefsmit, Christopher Edgar, Miguel Castaneda



Introduction

As the private markets landscape continues to evolve, there is a growing recognition of the transformative potential of artificial intelligence (AI). Some firms aspire to establish themselves as leaders in private equity by applying AI applications to create value for investors and portfolio companies. The ideal vision is to seamlessly integrate AI across many (or all) aspects of the investment process to enable more intelligent investment decisions, uncover hidden opportunities, and enhance operational efficiencies. This forward-looking approach positions private markets firms to navigate the complexities of the modern investment landscape and unlock new avenues for growth and success.

We will comprehensively examine decisions for a top-tier private equity firm's strategic employment of AI in its investment operations by offering insightful considerations about selecting and utilizing AI technologies and emphasizing choices beyond solutions like ChatGPT.

The (Generative) AI Perspective

Landscape

In the year following the release of ChatGPT, the AI landscape witnessed rapid acceleration. Almost daily, new models and updated versions emerge, various frameworks for model creation and manipulation are introduced, and platforms and services continually transition from general-purpose to highly specialized applications.

While OpenAI—and consequently its primary partner, Microsoft—currently appears to lead the space with ChatGPT as the most capable model, other tech giants are making substantial investments to close the gap, sometimes taking the lead in specific areas.

For instance, Meta's Llama has become a widely adopted foundational model upon which many high-performing open-source models are built. Google's PaLM formed the basis for the first fully AI-integrated office environment, encompassing mail, calendar, tasks, bookings, and other features, in addition to claiming to outperform OpenAI's GPT-4 with their new "Gemini" model. Amazon has entered the market with its Bedrock AI platform, offering fine-tuning, inference, hosting, and more, including open-source models and those from other companies. This comprehensive offering is now recognized as a Model as a Service (MaaS), commonly offered in the industry across competitors. Mistral, a company founded less than a year ago, already offers models that outperform ChatGPT and, in some instances, are on the level with GPT-4 while following a more open philosophy.

Various coding frameworks and patterns have emerged for the development of AI applications. While these frameworks share functionalities and center around common themes such as Retrieval-Augmented Generation (or RAG, which describes an application architecture that intelligently identifies and pulls relevant data from databases and other sources to augment the AI's internal knowledge), prebuilt tools, and function calling, they often specialize in specific areas, facilitating categorization across prompting, building agents, and engineering. Before discussing the frameworks in detail in the case study section, we will outline three functional categories:

1. **Chaining prompts:** LangChain takes the lead in this category, whereas Semantic Kernel serves as the lightweight Microsoft tool.
2. **Building and connecting agents:** AutoGen, TaskWeaver, and CrewAI place agents at their core, distinguishing themselves, while LangChain relies on prompts for its approach and recently introduced LangGraph as an agent framework.

3. **AI Engineering:** PromptFlow, with its agnostic and docker-based approach, stands in contrast to DialogFlow, Google's API builder. LangChain recently entered this domain with LangSmith.

Challenges

The capabilities of this new area of AI models are enormous, but there are some core challenges in operationalizing them reliably.

Getting the Prompt Right is Hard

A substantial factor contributing to the triumph of generative AI is its user-friendly chat interface. Conversing with a model yields impressive results and imparts a sense of interaction with a colleague due to the diversity of answers. However, the stochastic nature of Large Language Models (LLMs) introduces a significant challenge. Minor adjustments in a submitted question or task can lead to substantial and unexpected response differences. Even when submitting the exact text repeatedly, obtaining the same response is not guaranteed. Moreover, different models or versions of the same model exhibit variability in their responses to prompts. Beyond the inherent complexity of prompting, the output is heavily influenced by hyperparameters such as model temperature, which can best be described as a parameter to control the creativity of AI.

As a result, identifying a prompt that consistently produces desired outputs with a high likelihood becomes challenging, requiring structured testing and evaluation methods. This process needs to be continually repeated and monitored to identify quality issues. Coupled with finding prompts that generate satisfactory results based on their text and incorporate external data through RAG integrations, prompt engineering emerges as the most significant challenge, consuming the most time creating AI applications.

AI is Expensive

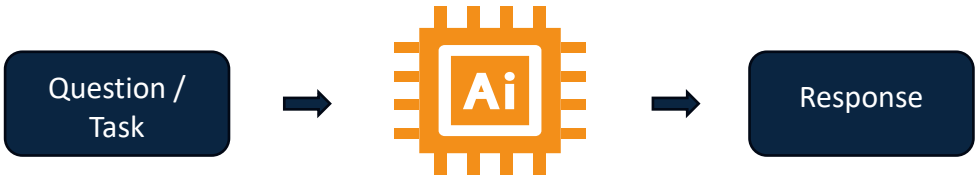
While one might expect Microsoft's market-leading position to be well-positioned in developing profit-generating AI applications, recent reports from the Wall Street Journal reveal a different reality. Microsoft reportedly faces financial challenges, losing an average of $20 per user subscription, with figures reaching up to $80 in the upper brackets.[1] This financial strain persists despite Microsoft hosting the underlying AI and having the ability to implement optimizations beyond what external API users could achieve.

Considering the current pricing structure of OpenAI for ChatGPT usage, generating an average page of text costs approximately 10 cents, and submitting the same amount costs about 5 cents with their most advanced model. At these rates, scaling an AI-based solution for production becomes expensive rapidly, especially when factoring in the augmentation of prompts with enterprise data or web search results.

[1] https://www.wsj.com/tech/ai/ais-costly-buildup-could-make-early-products-a-hard-sell-bdd29b9f

The "Supermodel Architecture Myth"

It is often assumed that an AI application architecture looks like this:



The underlying assumption posits an exceptionally powerful AI at the core, adept at tasks such as deciphering intent, discerning relevant context from prompts, and generating fitting responses. However, this presupposition may overstate the current capabilities of AI models. This architectural approach presents several challenges, echoing the previously discussed issues.

1. **Testing and Optimization Complexity:** Handling multiple inputs complicates the comprehensive testing and optimization of prompts. This complexity is amplified with methods like RAG, where consolidating inputs into a single node creates interdependencies, impeding the isolation and subsequent testing and optimization of individual steps.
2. **Dependency on Supermodel Capability:** The reliance on a core supermodel for every task necessitates its ability to handle intricate tasks. Consequently, meeting the upper limits of capability requirements demands a costly model, influencing overall pricing.
3. **High Dependency on Model Provider:** Building the entire application around a single supermodel results in a significant dependency on the model provider. Any change in the model requires thorough retesting of the whole system.
4. **Limited Adaptability to New Developments:** The monolithic nature of this approach hinders adaptation to new developments and excludes the integration of potentially superior models continually being published.
5. **Simplicity Facilitating Copying:** Although less critical for internal applications, this architecture's simplicity makes it susceptible to replication. Competitors could swiftly recreate the solution by copying the painstakingly optimized prompt.

To address these challenges, a solution combines multiple AI models, each specializing in different areas. These models are augmented with traditional programming and APIs to create robust and reliable AI applications.

Private Equity Innovator Case Study

Next, we'll delve into the strategies of a hypothetical premier private equity firm as it integrates AI into its investment process. Our focus will be to understand how the aforementioned challenges and frameworks related to an AI implementation can be applied in a practical setting. We will consider the firm's technology and operational processes along with their overall vision and internal culture to wholly understand how an organization can navigate the intricacies of an AI implementation in private markets.

Firm Profile

The hypothetical private equity firm we will study is noted for its ambitious goal to harness AI fully across all facets of its investment process. Their objective extends beyond merely enhancing investment strategies; it is a holistic approach to creating value for investors and portfolio companies. This firm seeks to control and customize its own in-house analytical model, willing and able to develop or customize

within rather than only employing a no-code, out-of-the-box solution. We will approach this study by focusing on the builders who will need to consider the tooling and platform.

Vision and Strategy

The firm we will explore is on its way to establishing itself as a leading entity in the private equity sector, driven by a future where AI is a core component of every investment decision. Their strategy is anchored in the belief that AI can uncover new opportunities, drive smarter investment choices, and streamline operational processes, elevating their market position.

Workforce and Culture

The firm's workforce comprises adept and quick learners who excel in grasping complex AI concepts. They follow scientifically based methodologies for investment analysis, ensuring well-informed decision-making. The dedicated technology team possesses the willingness and proficiency for hands-on engagement with new AI technologies, allowing them to implement or customize solutions in code and remain at the forefront of AI innovation.

Technological Infrastructure

The firm's technology stack is built on major cloud platforms like AWS, Azure, and GCP, supplemented with specialized web services, robust knowledge management, and operational systems. Our focus within this paper is on their predominant use of Microsoft Azure, reflecting their commitment to leveraging leading cloud technologies for business advancement.

Operational Processes

The company's workforce is highly specialized, following specific processes aligned with their respective areas of expertise. Generally, there is a consensus that the workforce is burdened with numerous non-core business-related tasks. Adopting new technologies, especially AI-based solutions, is contingent upon their ability to simplify processes or deliver a significant positive impact, justifying the additional effort required for implementation.

A Builder's Roadmap

Tooling & Platform Selection

When embarking on a journey of AI excellence, some pivotal decisions must be made regarding the platform and framework to apply.

1. **Platform choice:** A core consideration is to either go with a hyper-scaler that incorporates LLM hosting in their broader offering (e.g., Azure, AWS, GCP) or to use a two-fold approach hosting the application with a generic infrastructure provider or hyper-scaler and building against the endpoints provided by either a generic reselling and model hosting platform (e.g., together.ai) or directly against the offerings of model developers (e.g., OpenAI, Anthropic, Mistral).
2. **Framework choice:** A choice between using a platform-specific framework (e.g., PromptFlow) versus using an agnostic framework and potentially code components from scratch or customize framework components accordingly (e.g., LangChain). The reasons behind this choice are twofold:

- **LangChain's Dominance:** LangChain, the first and most widely used framework, offers a rich ecosystem with numerous examples and guides, but platform-specific frameworks potentially synergize better with the underlying platform.
- **Technology Team's Preference:** The firm's technology team might prefer a specific language (e.g., JavaScript/TypeScript versus Python), and not every framework supports every language.

Recommendation & Reasoning

The recommendation is to use a hyper-scaler framework for developments and deployments closely intertwined with the LLM endpoint (e.g., Azure PromptFlow with Azure AI Studio deployments) and an agnostic framework to develop application components focused on the firm's established processes and business logic (e.g., LangChain).

PromptFlow, tailored for AI engineering, supports prompt testing, optimization, deployment, and content safety. While it primarily provides an open and clean slate for development, it also offers prebuilt components that excel in interaction with other Azure Services. For example, it seamlessly integrates with AI Search for RAG implementations and provides features to test, optimize, and monitor AI solutions in development and deployed endpoints. Configuration is flexible through YAML files, with no concealment of proprietary components. The platform-independence of its core functionality, achieved through standard Docker containers, facilitates integration with other frameworks like LangChain. This flexibility enables the incorporation of LangChain components.

Furthermore, LangChain and Microsoft recently partnered up[2], and Microsoft is a primary hosting partner for LangSmith. LangSmith is LangChain's LLMOps platform and is like Azure's PromptFlow, even down to the design of the Web Interface. This integration allows the amalgamation of testing, deployment, safety, and management capabilities from PromptFlow with the rich ecosystem of LangChain or any other external service and framework. Considering the strengthened cooperation, an increasing junction can be expected for LangChan and LangSmith with the Azure ecosystem and PromptFlow framework. The open and agnostic design of PromptFlow suggests the likelihood of increased support and plugins in development tools beyond VSCode.

At the forefront of the AI evolution, Azure is one of the most powerful AI platforms globally[3]. Ensuring access to computing power is strategically essential, especially in the current surge in demand for AI compute-optimized environments. The partnership between Microsoft, OpenAI, and Nvidia further solidifies Azure's standing, working with the company possessing the most capable model and the primary producer of AI-optimized hardware. Azure is currently the exclusive platform for ChatGPT, even though there may be a few months delay for new models to become available[4]. Additionally, Microsoft's MaaS offering introduces two options with cost implications:

- Lower costs for inference endpoints for open-source models, especially when opting for smaller, fit-for-purpose models.
- Cost stability through dedicated VM deployments instead of pay-as-you-go endpoints.

Combining the strengths of Microsoft's PromptFlow and Azure offers anticipated benefits:

[2] https://blog.langchain.dev/langchain-expands-collaboration-with-microsoft/
[3] https://news.microsoft.com/wp-content/uploads/prod/2023/11/Microsoft-Ignite-Opening.pdf
[4] https://techcommunity.microsoft.com/t5/ai-machine-learning-blog/welcoming-mistral-phi-jais-code-llama-nvidia-nemotron-and-more/ba-p/3982699

- Seamless integration of various open-source models and ChatGPT within the same AI application.
- A PromptFlow backend can seamlessly integrate into the Azure MaaS service, enabling continuous fine-tuning of open-source and ChatGPT models.
- The challenge of model release cycles lagging behind OpenAI can be mitigated using a potentially LangChain-based node in PromptFlow that runs against an OpenAI, Google Gemini, or Mistral subscription if necessary.

Developing in their favorite language is highly important for a development team, and the discussed frameworks support the dominant languages in the respective fields. While frontend development in TypeScript is feasible, the backend in PromptFlow currently favors Python. The typical AI application stack, such as for Copilot, exhibits a clear separation between frontend, orchestration, and AI models. The frontend can be built as a TypeScript application using standard web frontend frameworks like React. However, when using PromptFlow, there is no recommended solution to use TypeScript instead of Python on the backend. Although interface libraries exist to interact between Python and TypeScript, their applicability in this case seems questionable. It is also possible that JavaScript support will be added to the PromptFlow framework. This challenge reflects a common architectural consideration, where AI code and the design of related API backends are predominantly executed in Python, while powerful web applications often leverage JS/TS frameworks.

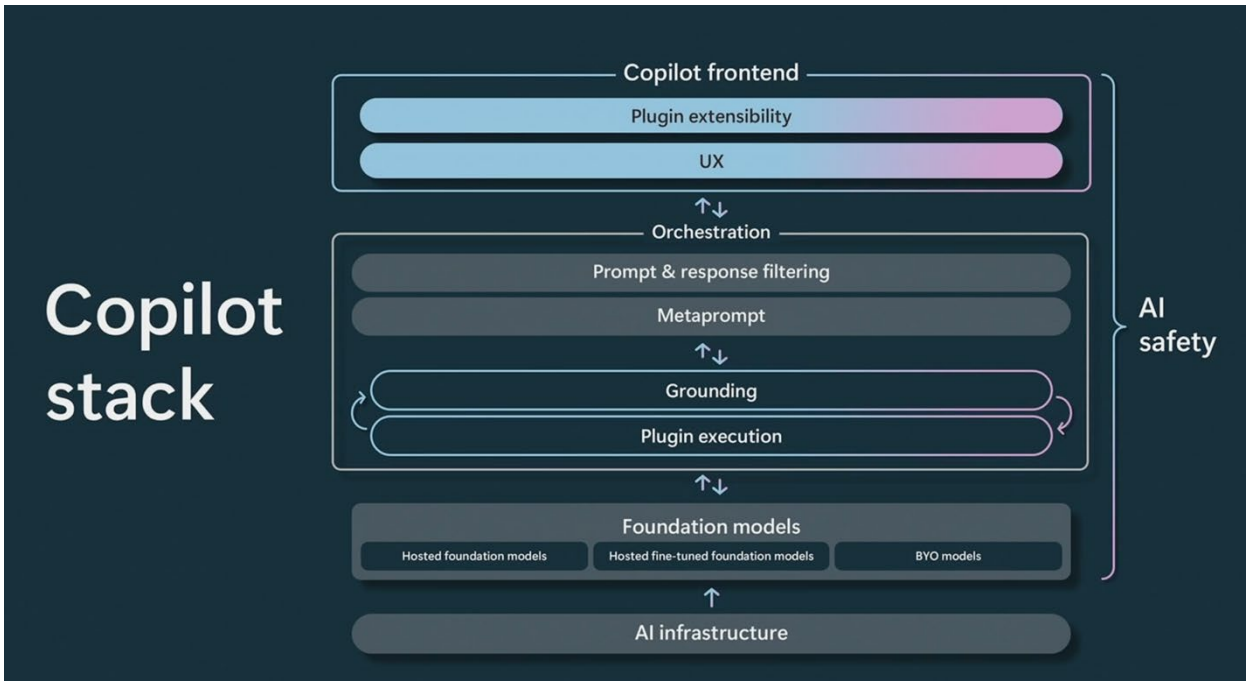


Figure 1: Copilot Stack

Cost vs. Performance Calculation: AI-Supported Document Analysis

In this section, we'll dive into the intricacies of cost and performance calculations for AI-supported document analysis. Specifically, we'll consider how resources are allocated across various tasks and application areas. The objective is to demystify the challenges of comparing different AI tools and provide a sample of an actionable framework for optimizing the balance between cost and performance in AI implementation. Note that the performance comparisons are based on purpose-fit, open-source models. We will discuss the models design and benchmarks in more detail in the chapter regarding open model ecosystems later on.

Consider the use case of AI-supported document analysis, augmentation, and reasoning. The application areas and their respective share in the overall solution are delineated as follows:

1. User query / intent analysis (5%)
2. Function calling for augmentation (10%)
3. Extraction from retrieved documents (35%)
4. Reasoning (40%)
5. Summarization/formulating (10%)

To build a basis for a monthly cost calculation, the following assumptions are made:

- 10 users
- 10 interactions per day
- 10 messages per interaction
- 5 relevant document chunks per message
- 200 words per chunk (Microsoft default)
- An average response length of half a page (300 words)

Following are a couple of example calculations and assumptions to provide context for the subsequent comparison table.

Calculations:

- Message count: 10 users x 10 interactions x 10 messages x 20 days = 20,000
- Direct Input tokens: (300 word system prompt + 100 word user query + 150 words web search augmentation) x 1.5 (token / word) = 825
- RAG token: 5 document chunks x 200 words x 1.5 (token / word) = 1,500
- Response token: 300 words x 1.5 (token / word) = 450
- Total Input Token: 20,000 messages x (825 direct + 1,500 RAG) 2,325 token = 46,500,000 token
- Total response token: 20,000 messages x 450 tokens = 9,000,000 token

Assumptions:

- Using Azure compute prices for hosting cost. Compatible compute instances cost between 0.6 € and 45 € per hour, depending on the required power. Considering that we use mostly 7B and smaller open models, an average cost of 6 € seemed reasonable.
- While Microsoft is still working on their MaaS offering, we used together.ai pricing as a guideline for pay-per-go pricing of open models and took a mid-level tier, similar to the compute.
- The calculation assumes one-off messages and does not factor in the context growth of multi-turn conversations that include chat history.

With those numbers in mind, and based on the Azure OpenAI pricing (01.02.2024), the following cost and performance comparison shows that a 90% performance can come at 0.4% of the cost:

Steps	Tokens		GPT-4		Open Models			Comparison
	In	Out	Cost	Perf.	Hosted	Maas	Perf	
Intent	400	100	0.03€	100%	n/a	0,000140€	90%	Orca 2/Zephyr
Function	150	100	0.02€	100%	n/a	0,000070€	91%	Orca 2/Zephyr
Extraction	1500	300	0.12€	100%	n/a	0,000502€	93%	Zephyr
Reasoning	400	300	0.06€	100%	n/a	0,000195€	85%	Orca 2
Summarize	600	400	0.08€	100%	n/a	0,000279€	90%	Orca 2/Zephyr /phi-2
Cost/Msg			**0.30€**	**100%**	**n/a**	**0,00012€**	**90%**	
Cost/Month			**5,995.00€**		**960.00€**	**23,72€**		

Table 1: Cost Comparison GPT-4 versus Open Models

A few limitations of the above table exist. Firstly, it concentrates on ongoing costs and thus excludes setup costs. Additionally, the performance comparisons rely on general benchmarks, potentially requiring more effort to obtain optimal results from each open model than simply leveraging GPT-4 comprehensively. This analysis does not account for the scenario where an average performance drop of 15% in a specific use case renders the result unusable. Moreover, it does not factor in downstream implications, such as making a wrong and expensive business decision.

The Open Model Ecosystem: Beyond ChatGPT, Gemini, and Claude

In this section, we compare a couple of open models in contrast to the closed models from OpenAI or Google. The term "open models" denotes models where both the training data and the model weights are published. These models can be executed locally, self-hosted, or accessed through various competitive platform offerings. Typically, open models employ parameters ranging from 1 billion to 70 billion. In comparison, OpenAI's GPT-3 utilized 175 billion parameters, and for GPT-4, the exact number remains undisclosed but is estimated to fall between 100 trillion and 170 trillion. Given that open models are generally approximately 10,000 times smaller than GPT-4 and can be hosted by numerous providers, their pricing proves highly competitive—roughly 100 times cheaper than OpenAI's flagship model, often exhibiting comparable or even identical performance in specific use cases.

New models or model versions are published weekly, and the benchmarks can change accordingly. The primary objective of this section is to discern how open models perform in comparison to proprietary alternatives, using a selection of models as examples. This understanding facilitates informed decision-making by evaluating potential performance gains against costs and the inherent uncertainty of using models where the training data cannot be audited. We will dive into models tailored for specific use cases, such as logic and reasoning, programming, extraction in document retrieval, function calling, and general purpose efficiency. Note that the following model descriptions and benchmarks are mostly derived from the provided sources and have been slightly reformatted or reworded.

For Logic and Reasoning: Orca 2

In Orca 2, the crafting of reasoning strategies is intricately tailored to the specific task, taking into account the capabilities of a student model to exhibit similar behavior. Generating nuanced data involves presenting the more capable LLM with intricate prompts strategically designed to elicit specific behaviors. This approach aims to yield more accurate results. Notably, during the training phase, the smaller model is exclusively exposed to the task and the resulting behavior, with no access to the original prompts that initiated such behavior. Prompt Erasure's training technique distinguishes Orca 2 as a Cautious Reasoner. It not only learns the execution of specific reasoning steps but also develops a higher-level strategic approach to tackling a given task. Rather than blindly mimicking powerful LLMs, Orca 2 treats them as a repository of behaviors, carefully selecting those most suitable for the task.[5,6]

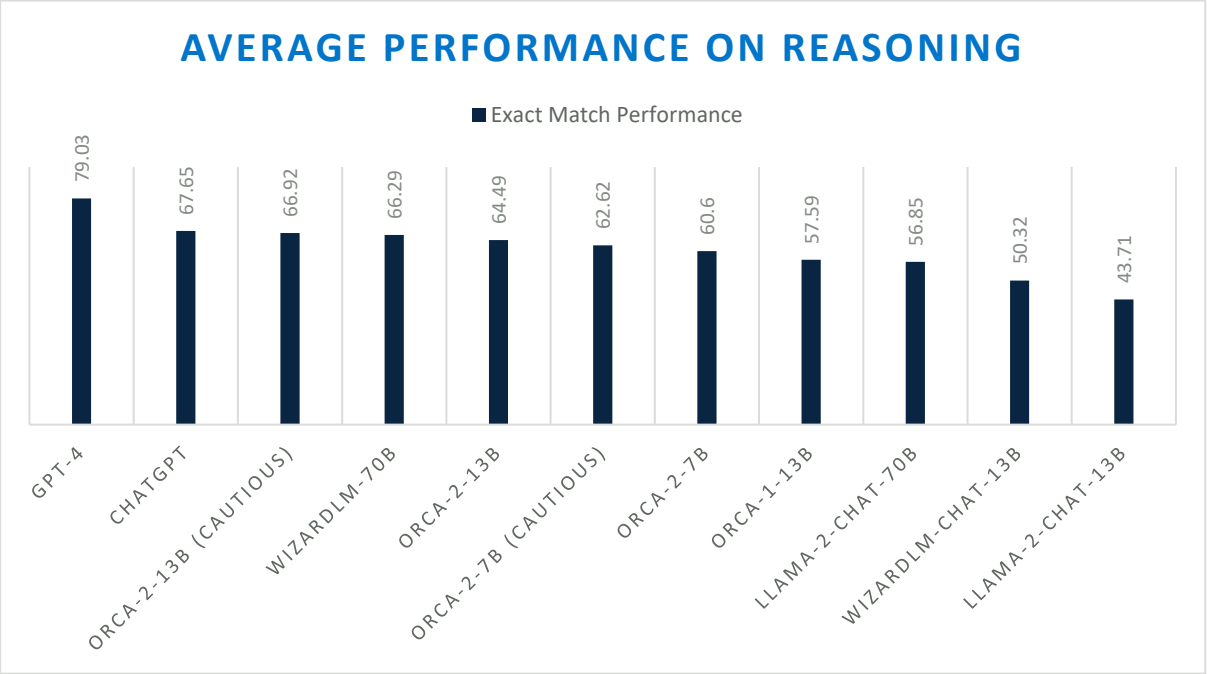


Table 2: Macro-Average Performance of Different Models on Reasoning Benchmarks

Model	AGI	BBH	DROP	CRASS	RACE	GSM8K
Orca 2-7B	45.10	45.93	60.26	84.31	80.79	47.23
w/ cautious	43.97	42.80	69.09	88.32	75.82	55.72
Orca 2-13B	49.93	50.18	57.97	86.86	82.87	59.14
w/ cautious sm	48.18	50.01	70.88	87.59	79.16	65.73
Orca-1-13B	45.69	47.84	53.63	90.15	81.76	26.46
LLaMa-2-Chat-13B	38.85	33.6	40.73	61.31	62.69	25.09
WizardLM-13B	38.25	38.47	45.97	67.88	62.77	48.60

[5] https://arxiv.org/pdf/2311.11045.pdf
[6] https://huggingface.co/microsoft/Orca-2-13b

LLaMA-2-Chat-70B	46.70	44.68	54.11	74.82	68.79	52.01
WizardLM-70B	48.73	51.08	59.62	86.13	78.96	73.24
ChatGPT	53.13	55.38	64.39	85.77	67.87	79.38
GPT-4	70.40	69.04	71.59	94.53	83.08	85.52

Table 3: Zero-Shot Performance Comparison of Different Models on Reasoning Benchmarks

The above benchmark examples show that Orca 2 – 13B reasoning is on par with ChatGPT and, in some cases, even with GPT-4.

For Programming: DeepSeek Coder

DeepSeek Coder comprises a series of code language models, each trained from scratch on 2T tokens, comprising 87% code and 13% natural language. Consequently, it is focused on and excels in performing coding-related tasks.

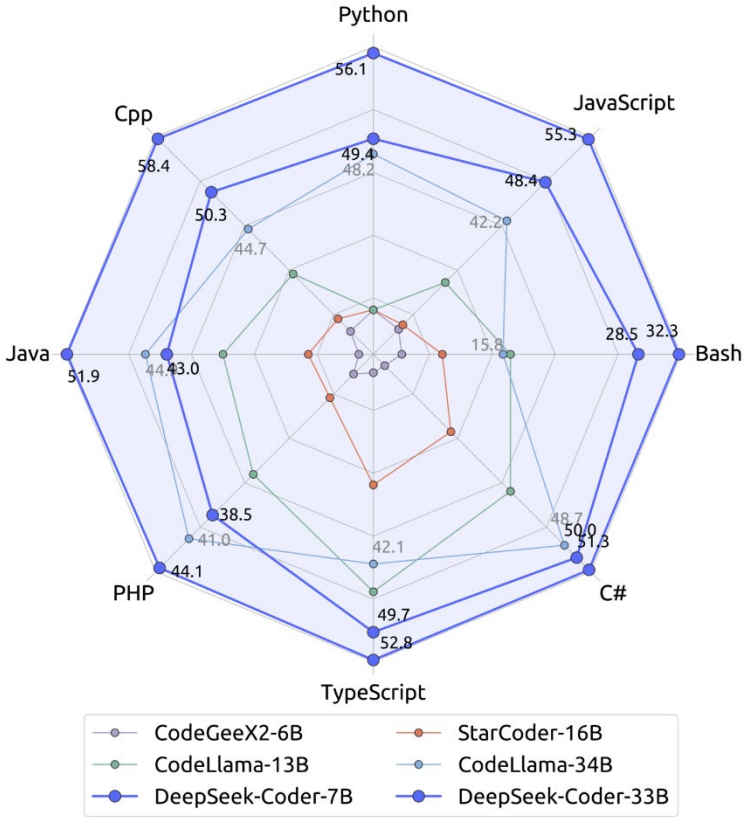


Figure 2: Coding Capabilities Performance among Open-source Code Models

		HumanEval			
Model	**Size**	**Python**	**Multilingual**	**MBPP**	**DS-1000**
Pre-Trained Models					
Codex-001	-	33.5%	26.1%	45.9%	20.2%
Codes-002	-	-	-	-	39.2%
StarCoder	16B	36.0%	28.7%	46.8%	27.2%
CodeGeeX2	6B	36.0%	24.5%	42.4%	22.9%
CodeLlama	7B	31.7%	29.2%	41.6%	22.1%
CodeLlama	13B	36.0%	35.4%	48.4%	26.8%
CodeLlama	34B	48.2%	41.0%	55.2%	34.3%
DeepSeek-Coder-Base	1.3B	34.8%	28.3%	46.2%	16.2%
DeepSeek-Coder-MQA-Base	5.7B	48.7%	41.3%	57.2%	27.7%
DeepSeek-Coder-Base	6.7B	49.4%	44.7%	60.6%	30.5%
DeepSeek-Coder-Base	33B	56.1%	50.3%	66.0%	40.2%
Instruction-Tuned Models					
GPT 3.5-Turbo	-	76.2%	64.9%	70.8%	-
GPT-4	-	84.1%	76.5%	80.0%	-
DeepSeek-Coder-Instruct	6.7B	78.6%	66.1%	65.4%	-
DeepSeek-Coder-Instruct	33B	79.3%	69.2%	70.0%	-

Table 4: pass@1 Results on HumanEval (Python and Multilingual), MBPP, and DS-1000

The results indicate a significant performance advantage for DeepSeek-Coder-Base-33B over existing open-source code LLMs. Compared to CodeLlama-34B, it exhibits a lead of 7.9%, 9.3%, 10.8%, and 5.9% on HumanEval Python, HumanEval Multilingual, MBPP, and DS-1000, respectively. Remarkably, our DeepSeek-Coder-Base-7B achieves performance equivalent to CodeLlama-34B. Furthermore, the DeepSeek-Coder-Instruct-33B model, after instruction tuning, outperforms GPT35-turbo on HumanEval and achieves results comparable to GPT35-turbo on MBPP.[7]

[7] https://github.com/deepseek-ai/deepseek-coder

For Extraction in Document Retrieval (RAG): Zephyr

Zephyr models are fine-tuned version of Mistral that was trained on on a mix of publicly available, synthetic datasets using Direct Preference Optimization (DPO). It showed that removing the in-built alignment of these datasets boosted performance on MT Bench and made the model more helpful. However, this means that model is likely to generate problematic text when prompted to do so.[8,9]

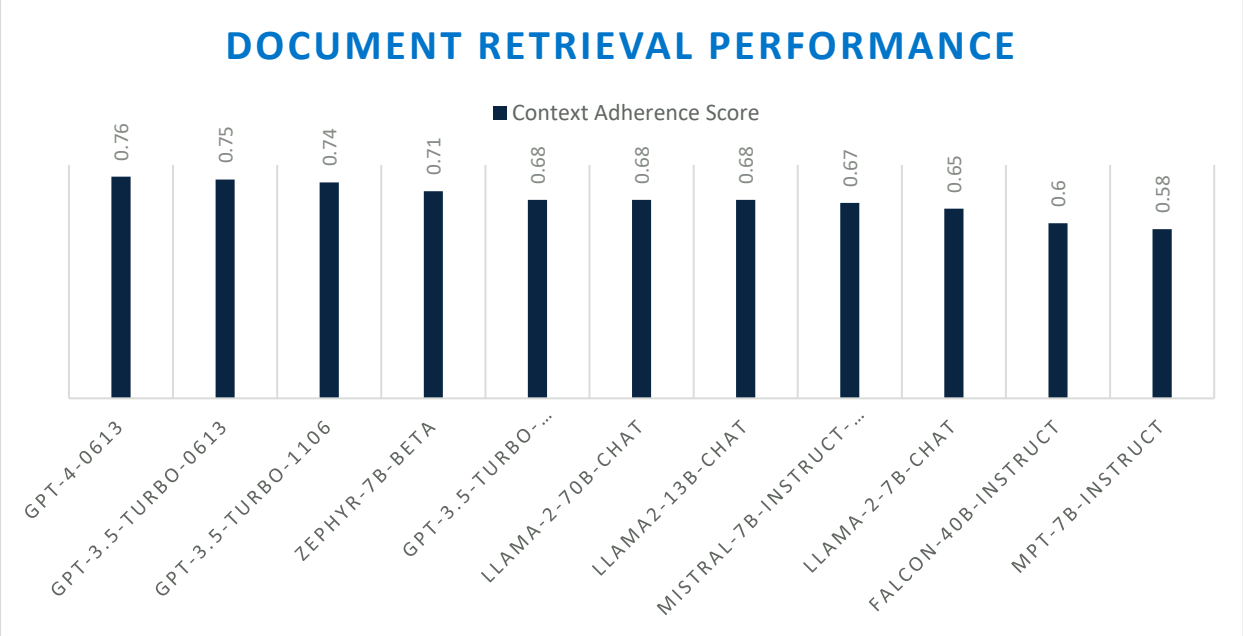


Table 5: Document Retrieval Performance[10]

For Function Calling: Gorilla

Gorilla enables LLMs to use tools by invoking APIs. Given a natural language query, Gorilla comes up with the semantically- and syntactically- correct API to invoke. Gorilla is the first model to demonstrate how to use LLMs to invoke 1,600+ (and growing) API calls accurately while reducing hallucination. The solution is based on the Gorilla recipe, and with a model with just 7B parameters, its accuracy is, surprisingly, comparable to GPT-4.[11]

[8] https://huggingface.co/HuggingFaceH4/zephyr-7b-alpha
[9] https://arxiv.org/abs/2310.16944
[10] https://www.rungalileo.io/hallucinationindex
[11] https://github.com/ShishirPatil/gorilla

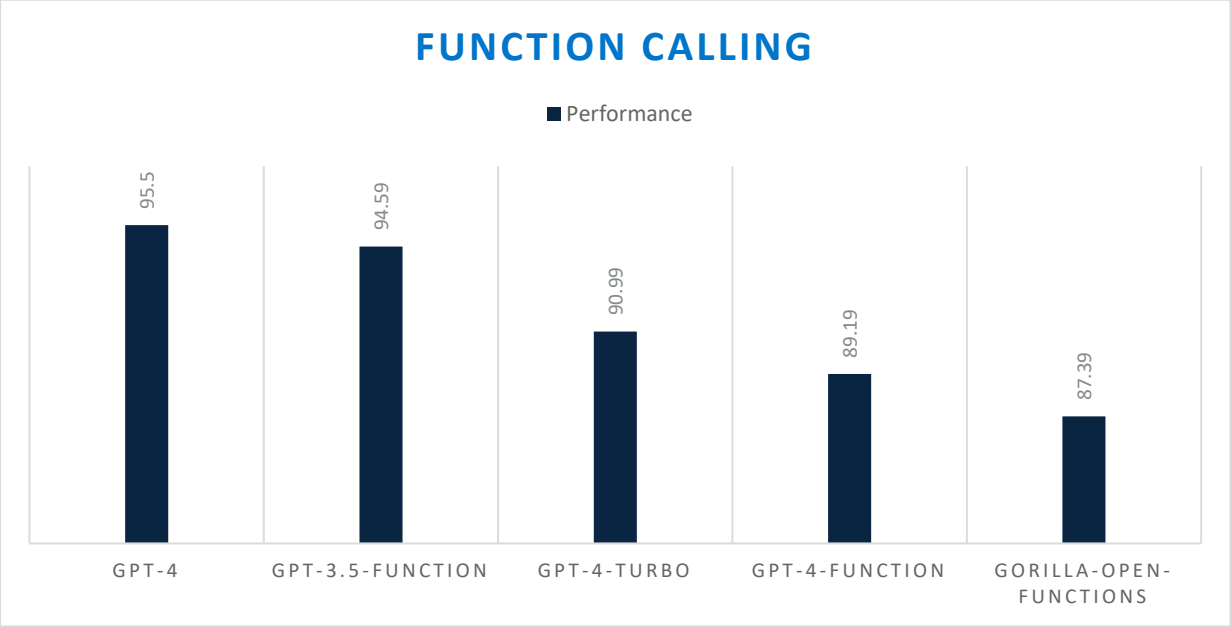


Table 6: Funtion Calling

"Supermodel" Alternative: Falcon 180B

Falcon 180B is the largest openly available language model, with 180 billion parameters, and was trained on a massive 3.5 trillion tokens using TII's RefinedWeb dataset. This represents the longest single-epoch pretraining for an open model.

In terms of capabilities, Falcon 180B achieves state-of-the-art results across natural language tasks. It tops the leaderboard for (pre-trained) open-access models and rivals proprietary models like PaLM-2. While difficult to rank definitively yet, it is considered on par with PaLM-2 Large, making Falcon 180B one of the most capable LLMs publicly known.[12]

Model	Size	Leaderboard score	Pretraining length
Falcon	180B	68.74	3,500B
Llama 2	70B	67.35	2,000B
LLaMA	65B	64.23	1,400B
Falcon	40B	61.48	1,000B
MPT	30B	56.15	1,000B

Table 7: Leaderboard Score Pre-trained LLM

General Purpose Efficiency Wonder: Phi 1.5 / 2

Economically, the cost of training, deploying, and maintaining such large models can be substantial. Scientifically, understanding whether similar capabilities can be achieved at a smaller scale could provide insights into the architectures and development of intelligent systems. From a responsible AI standpoint, the energy consumption of large-scale models is becoming an increasing concern, as is the question of how controllable or governable these large models can be. Finally, the ability to train compact models with cutting-edge capabilities would democratize advanced AI, enabling a broader range of individuals

[12] https://huggingface.co/blog/falcon-180b

and organizations to study and deploy them, instead of being an exclusive domain of a few with vast computational resources.[13,14]

	Train time (GPU hrs.)	MicroBatch (max)	Inf. Speed (per token)	Inf. Memory (at 2048 ctx.)	Data size (tokens)	Train tokens
Llama-7B	>80K	2	14ms	18G	1T	1T
Phi-1.5 (1.3B)	1.5K	8	<3ms	3.5G	30B	150B
Phi-1.5-web (1.3B)	3K	8	<3ms	3.5G	100B	300B

Table 8: Comparison of compute of different models using a single A100-80G with context length 2048 and fp16

In a nutshell, phi-1.5, a 1.3 billion parameter model was trained on a dataset of 30 billion tokens, which achieves common sense reasoning benchmark results comparable to models ten times its size that were trained on datasets more than ten times larger. Moreover, the dataset consists almost exclusively of synthetically generated data, which has important implications for the potential to control for the notoriously challenging issue of toxic and biased content generation with LLMs.

	WinoGrade	ARC-Easy	ARC-Challenge	BoolQ	SIQA
Vicuna-13B (v1.1)	0.708	0.754	0.432	0.835	0.437
Llama2-7B	0.691	0.763	0.434	0.779	0.480
Llama-7B	0.669	0.682	0.385	0.732	0.466
MPT-7B	0.680	0.749	0.405	0.739	0.451
Falcon-7B	0.662	0.719	0.363	0.685	0.452
Falcon-rw-1.3B	0.607	0.633	0.282	0.632	0.405
OPT-1.3B	0.610	0.570	0.232	0.596	-
GPT-Neo-2.7B	0.577	0.611	0.274	0.618	0.400
GPT2-XL-1.5B	0.583	0.583	0.250	0.618	0.394
Phi-1.5-web-only (1.3B)	0.604	0.666	0.329	0.632	0.414
Phi-1.5-web (1.3B)	0.740	0.761	0.449	0.728	0.530
Phi-1.5 (1.3B)	0.734	0.756	0.444	0.758	0.526

Table 9: Common Sense Reasoning Benchmarks

Efficient "Supermodel" and tiny model alternatives: Mixtral

Mistral 7B. The most cost-effective endpoint currently serves Mistral 7B Instruct v0.2, a new minor release of Mistral 7B Instruct. Mistral-tiny only works in English. It obtains 7.6 on MT-Bench.[15]

Mixtral 8x7B is a sparse mixture-of-experts network. It is a decoder-only model where the feedforward block picks from a set of 8 distinct groups of parameters. At every layer, for every token, a router network chooses two of these groups (the "experts") to process the token and combine their output additively.

[13] https://anakin.ai/blog/phi-2-microsoft/
[14] https://arxiv.org/pdf/2309.05463.pdf
[15] https://mistral.ai/news/mixtral-of-experts/

This technique increases the number of parameters of a model while controlling cost and latency, as the model only uses a fraction of the total set of parameters per token. Concretely, Mixtral has 46.7B total parameters but only uses 12.9B parameters per token. It, therefore, processes input and generates output at the same speed and for the same cost as a 12.9B model.

Mixtral is pre-trained on data extracted from the open Web – we train experts and routers simultaneously.

In the following figure, we measure the quality versus inference budget trade-off. Mistral 7B and Mixtral 8x7B belong to a family of highly efficient models compared to Llama 2 models.

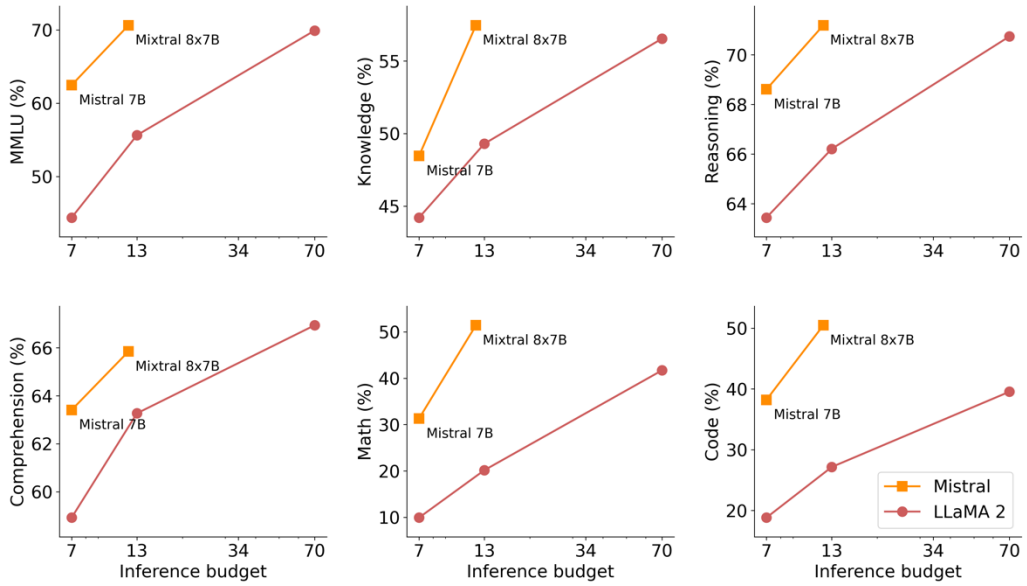


Figure 3: Quality versus Inference Budget Trade-off

Hallucination and biases. To identify possible flaws to be corrected by fine-tuning / preference modelling, we measure the base model performance on BBQ/BOLD.

	Llama 2 70B	Mixtral 8x7B
BBQ (higher is better)	51.50%	55.98%
BOLD (std) (lower is better)	0.094	0.084
Gender	0.073	0.045
Profession	0.073	0.045
Religious_ideology	0.133	0.089
Political_ideology	0.140	0.149
Race	0.049	0.052

Table 10: Hallucination and Biases Comparison

Conclusion

The AI landscape is undergoing rapid evolution, witnessing the release of numerous platforms, frameworks, and open-source and proprietary AI models in a relatively short time. Initially, OpenAI and Microsoft dominated the first 1.5 years of this new AI era, but other hyperscalers like Google and certain open-source models have caught up or, in some cases, even surpassed them. The increasing competitiveness of the landscape, coupled with new service providers entering the market, has sparked an API price war. Beyond just the raw capabilities of models and their utilization costs, introducing new functionalities within frameworks can bring significant benefits.

Given the substantial functional overlap among some frameworks, the swift development pace, and the trend towards consolidation and feature merging (as seen in the LangChain and Microsoft partnership), selecting a functionally fitting framework is crucial. However, it may be even more important for the organization to define the support model of the infrastructure providers, prioritizing their abilities both to execute the framework and to seamlessly integrate it into organization's platform. This emphasizes the importance of flexibility in realizing cost benefits, leveraging superior model capabilities, and expediting the incorporation of new functionalities.