Mixture-of-AI
Efficience for AI models

Joel AGBOGLO[1], Hilario HOUMEY [1]

HUMIRIS AI

research@humiris.ai

Abstract

We present Mixture of AI, a model that integrates a mixture of AI models designed to deliver superior performance, optimize costs, ensure data privacy, and reduce the carbon footprint. Our gating mechanism currently surpasses that of RouterLLM and several other LLM routers.Unlike traditional routing approaches, which often rely solely on criteria such as cost and quality, Mixture of AI's dynamic gating offers a more holistic, adaptive, and multiparameter approach.In addition to cost and quality, Mixture of AI's gating also considers factors such as data privacy, speed, and carbon footprint, providing a more comprehensive and customized request management.Our gating enables real-time adaptation to changing routing conditions and user needs, unlike static approaches, which are less responsive to variations in demand.

1. Introduction

Large language models (LLMs) demonstrate impressive capabilities across many tasks, but choosing the right model often requires considering multiple factors such as performance, cost, privacy, speed, and more.More powerful models deliver high-quality results but tend to be more expensive, slower, and often closed-source, while less powerful models are more cost-effective, faster, and often open-source.Large models like OpenAI's GPT-4, Anthropic's Claude 3, or models developed by Google DeepMind or other major labs tend to offer high performance in terms of language understanding, text generation, and answering complex questions. These models can process rich data and produce high-quality results but often require substantial computational resources.
Powerful commercial models are generally expensive to use, especially for applications requiring high volumes of API calls or extended usage. This includes infrastructure costs for hosting and operational costs related to energy and server maintenance. On the other hand, smaller or open-source models can provide a more economical alternative, especially when hosted on private infrastructure or more affordable cloud solutions.Using closed models hosted on third-party servers can raise privacy concerns, especially if the data being processed is sensitive. Companies concerned with data security may prefer solutions where they have full control over the infrastructure.Open-source models can be deployed internally, offering better control over data privacy.Although large models deliver impressive performance, they can be slow to execute due to their complexity. Latency can become an issue for applications that require real-time responses. Smaller models, or those optimized for efficiency, can provide faster responses and may be more suitable for environments with strict latency constraints.
Open-source models are often more accessible, allowing the community to adapt and improve them. They offer great flexibility to customize solutions based on specific needs. Proprietary models may provide better technical support and stability but often limit customization or extension

possibilities.The ecological footprint, or "green footprint," of language models is an increasingly important factor in the selection and use of AI technologies, including LLMs. This refers to the environmental impact of training and using these models, particularly in terms of energy consumption and carbon emissions.

To address the trade-offs between performance, cost, privacy, speed, accessibility, and environmental impact of large language models (LLMs), we propose **Mixture of AI**, a technology that blends different language models to enable efficient use of LLMs.

Our **Gating** system analyzes queries based on the mix instructions provided to it and directs them to the appropriate models, which is a complex process.An effective gating model must be able to determine the intent, complexity, and domain of an incoming query, while assessing the capabilities of the available models to route the query to the most suitable one. Additionally, this gating model must be cost-effective, fast, and adaptable to a constantly evolving environment, characterized by the regular introduction of new models with enhanced capabilities.

2. Related Work

Several approaches have been developed to enhance and optimize the use of large language models (LLMs). Single-model enhancements, like fine-tuning (Rafailov et al., 2023), improve task-specific performance but require additional training and domain-specific data. Techniques such as Chain-of-Thought (CoT) prompting (Wei et al., 2022; Zhou et al., 2023; Wang et al., 2022) and Tree of Thoughts (ToT) reasoning (Yao et al., 2023) aim to boost LLM capabilities without the need for fine-tuning. Mixture-of-Experts (MoE) (Eigen et al., 2014; Shazeer et al., 2017) is another strategy that increases efficiency by routing inputs to specialized "experts" within the model, yet these approaches tend to be model-specific and struggle to take advantage of the growing diversity of LLMs. To move beyond single-model optimization, LLM synthesis combines the outputs of multiple models to produce enhanced results (Jiang et al., 2023b), with research showing that smaller models can sometimes outperform larger ones when used strategically (Lu et al., 2024). However, this method involves multiple steps, increasing both latency and costs, limiting its practical use. FrugalGPT (Chen et al., 2023) offers a more cost-effective approach by using a generation judger that sequentially evaluates responses from different LLMs, invoking models until a satisfactory answer is reached. In addition to these, other multi-LLM strategies have emerged. RouteLLM focuses on routing models based on user preferences, while ROUTERBENCH provides a benchmark to evaluate the efficiency of multi-LLM routing systems. A Multi-LLM Debiasing Framework tackles the issue of biases across various models, while another study, "Small LLMs Are Weak Tool Learners," explores the challenges smaller models face in tool learning. CollabStory leverages multiple LLMs for collaborative story generation and authorship analysis. Cost-Effective Online Multi-LLM Selection presents versatile reward models to optimize real-time model selection. Additionally, a Multi-LLM Orchestration Engine facilitates personalized, context-rich assistance by coordinating multiple models, and Martian Model Router focuses on optimizing costs in routing LLMs. Lastly, Arcee.Ai merges outputs from various models, while Lamini's MoME (Mixture of Models for Efficiency) aims to improve factual accuracy and reduce hallucinations in LLM outputs, offering further refinements in multi-model systems.

3. Mixture-of-AI Methodology

3.1. Overview

Mixture of AI is an innovative technology comprising a set of n large language models (LLMs) {M1, . . . , Mn}, coupled with a "gating" mechanism, G. Each LLM represents a particular area of expertise or specialization, offering a diverse range of skills to handle various types of queries. The crucial role of the gating mechanism, which is itself an LLM equipped with a softmax activation function, is to evaluate each incoming query and determine the most suitable model to respond. The gating model operates by analyzing the query's characteristics, such as its content, intent, complexity, and thematic domain. Based on this analysis, the gating model activates the foundational models that possess the most relevant capabilities to handle the query optimally, according to the provided mix-instructions.

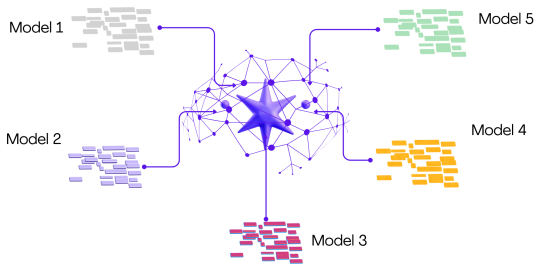


Fig1. MoAI overview representation

For example, for a query that requires deep natural language understanding and high precision, the gating mechanism might direct the query to a model specialized in these skills. Conversely, for a less complex query, the gating could select a lighter model to save resources while still providing an adequate response.
In this section, we present our proposed methodology for leveraging multiple models to achieve our objectives. We begin with the basic step of routing queries to the appropriate models. Next, we demonstrate that LLMs possess a collaborative capability, allowing them to improve their responses by utilizing outputs from other models. Finally, we explore the advanced aspect, which involves having multiple models (whether the same or different) interact to respond to a query. Afterward, we will introduce the concept of mix-tuning.

3.2. Methodology Mo-AI Basic

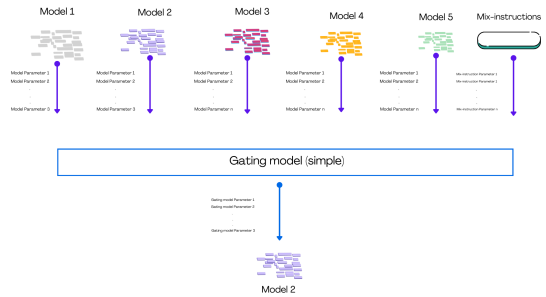


Fig2. MoAI-Basic architecture

3.2.1 Selection of LLMs for Query Routing

The first essential element of the Mo-AI methodology is the selection of large language models (LLMs) to which queries will be routed. The selection of LLMs is based on a thorough analysis of their specific capabilities and performance across various tasks. Selection criteria include the diversity of knowledge domains covered by each model, the quality of the generated responses, and performance in terms of latency and computational cost. This diversity maximizes the effectiveness of the Mixture of AI technology by ensuring comprehensive coverage of possible queries, ranging from simple information retrieval to complex text generation. In the first version of Mo-AI, we focus solely on text-generation models.

3.2.2 Selection of Inference Servers

Choosing the inference servers is crucial for ensuring smooth and efficient execution of the selected LLMs. This choice is guided by several factors, such as the processing capacity of the servers, their geographical location to minimize latency, and energy efficiency. High-performance cloud infrastructures are preferred for their flexibility and ability to handle varying workloads while ensuring scalability as needed. Simultaneously, robust security measures are implemented to protect sensitive data processed by the models. In our case, we will use Together AI and Groq servers for the inference of open-source models.

3.2.3 Creation of the Gating System: The Core of Mixture of AI

The central component of the Mo-AI methodology is the creation of the gating system, which plays a crucial role in managing queries. The gating mechanism is a sophisticated artificial intelligence model based on a neural network and equipped with a softmax activation function. This system is responsible for analyzing incoming queries and deciding how to route them to the most appropriate LLM. The decision-making process relies on the mix instructions provided to the gating mechanism, as well as an assessment of the query's nature, complexity, and the specific capabilities of the available models. The gating allows for optimization of both the quality of responses and the computational resources used, reserving the most powerful models for the most complex queries in cost optimization scenarios, or utilizing open-source models for queries involving sensitive data.

3.2.4 Creation of a Benchmark Dataset for Gating Training

To effectively train the gating system, a rigorously designed benchmark dataset is essential. This dataset consists of representative data for various types of queries that the system is likely to encounter. It includes samples of queries covering a broad spectrum of domains and complexities. The process of collecting and labeling this data is meticulously planned to ensure high quality and diversity. The dataset is then used to train the gating model, enabling it to learn how to evaluate queries and select the most appropriate model for each situation.

3.2.5 Creation of a Real-Time Data Collection System for Monitoring LLM Consumption and Usage

Finally, a data collection system is established to monitor various performance and usage dimensions

of the LLMs within the Mo-AI infrastructure. This system collects metrics such as the number of queries, the number of tokens per LLM, response time, energy consumption, inference costs, and user feedback. These data are crucial for the continuous improvement of the system, allowing for an understanding of the actual performance of each LLM in various scenarios and enabling adjustments to routing strategies accordingly. Furthermore, this monitoring helps identify opportunities for cost and energy efficiency optimization, thus contributing to the sustainability of the system.

3.2.6 Collaborativeness of LLMs

In Mixture of AI, collaboration among language models (LLMs) is facilitated through "mix instructions," which are directives provided by developers and orchestrated by a central component: the gating model. This gating model plays a crucial role in executing the instructions to guide interactions between the various LLMs, with the goal of optimizing final responses.

A key approach to maximizing the benefits of collaboration among multiple models is to characterize the specific skills of each model in different aspects of collaboration. In this process, we can categorize the models into two distinct roles:

Proposers:These models excel at generating useful reference responses for other models. A good proposer may not produce exceptional answers on its own, but it must provide context and diverse perspectives that enrich the final responses when used by an aggregator.

Aggregators:These models specialize in synthesizing responses provided by other models into a single high-quality answer. An effective aggregator should be able to maintain or enhance the quality of the output, even when integrating responses that are of lower quality than its own.

Managing Redundancy and Complementarity: The "mix instructions" include guidelines to avoid redundancy and maximize the complementarity of responses. For example, if multiple LLMs provide similar answers, the instructions may guide the gating model to prioritize complementary aspects or diversify the angles of the responses.

Execution Flexibility: Although the "mix instructions" are predefined, the gating model has some flexibility in their execution. This flexibility allows it to adapt in real-time to the specifics of each query and the observed performance of the involved LLMs, while still adhering to the provided instructions.

Integration of New Models:The "mix instructions" can be updated by developers to include new LLMs or to adjust roles and collaborations based on technological advancements or specific needs. The gating model then applies these new instructions, ensuring that Mixture of AI remains efficient and up-to-date.

Examples of Collaboration: Consider a question requiring both legal analysis and ethical reflection. Developers may define "mix instructions" where a law-specialized LLM acts as a proposer to provide a detailed legal analysis, while another LLM, focused on ethics, adds moral considerations. The gating model, following the "mix instructions," aggregates these contributions to deliver a final response that incorporates these different perspectives.

3.3. Methodology Mo-AI Advanced

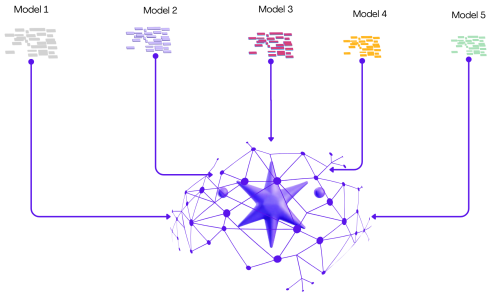


Fig3. MoAI Advanced Architecture

Mixture of AI Advanced takes the concept of collaboration among artificial intelligence models even further by enabling the simultaneous interaction of multiple models, whether identical or different, to respond to a given query. This approach leverages the diversity of capabilities and perspectives provided by various LLMs to enhance the quality and relevance of the generated responses. We delve into the underlying mechanisms, coordination strategies for interactions between the models, and the implications of this advanced methodology.By facilitating concurrent model interactions, Mixture of AI Advanced can harness the strengths of each model to create a more comprehensive and nuanced answer. This system encourages an ecosystem where models can contribute their unique expertise, leading to a richer synthesis of information. Moreover, effective coordination among models is essential for maintaining the integrity of responses, managing potential redundancies, and ensuring that the final output reflects a cohesive understanding of the query.The implications of this advanced methodology are significant, particularly in applications requiring multifaceted analysis, such as legal assessments, scientific research, or creative problem-solving. By integrating multiple

viewpoints, Mixture of AI Advanced not only improves the accuracy of responses but also promotes a more holistic approach to complex queries, ultimately enhancing user satisfaction and trust in AI-generated content.

3.3.1. Interactions Multi-LLMs

In the Mo-AI Advanced methodology, interactions among multiple LLMs are orchestrated to maximize synergy between the models. These interactions can occur at various levels and in different ways:

Sequential Interaction: In this configuration, one model handles part of the query, producing a partial or intermediate response that is then used as input by another model. This process can continue over several iterations, with each model adding an additional layer of analysis or detail. For example, one model might first generate a technical analysis, which is subsequently enriched by another model providing economic perspectives.

Concurrent Interaction: Models can also work in parallel, each processing the query independently or addressing different aspects of the query simultaneously. The responses produced by these models are then combined through an aggregation process, often guided by the mix instructions predefined by the developers. This approach is particularly beneficial for complex queries requiring a multifaceted solution.

Feedback Interaction**:** Another level of interaction may involve the revision of initial responses by additional models. For instance, after a set of models has produced a response, a supplementary model may be utilized to verify, correct, or refine that response, ensuring maximum coherence and quality.

3.3.2. Coordination of Interactions: The Role of the Gating Model

The gating model plays a central role in coordinating interactions between the various LLMs. The gating model, based on the mix instructions, determines which model should be solicited at each step and in what order. This determination relies on the specific strengths of the models for different tasks or aspects of the query.Once the different models have produced their respective responses, the gating model intervenes to orchestrate the aggregation process. It uses the mix instructions to guide the combination of responses, ensuring that important aspects are not overlooked and that the final response is coherent.The gating model is also responsible for managing any potential conflicts or inconsistencies between the responses provided by different models. It may decide to weight the contributions of certain models more heavily or eliminate contradictory elements based on the directives of the mix instructions.

3.3.3. Mix Instructions and Adaptive Flexibility

In the Mo-AI advanced methodology, mix instructions play a vital role by providing a framework for collaboration between models. However, they must also be flexible enough to adapt to changing contexts and next-generation models. Developers have the ability to update the mix instructions to incorporate new models or adjust collaboration strategies based on observed performance or the new capabilities of available models.The Mo-AI advanced methodology is particularly effective in scenarios where the complexity of the query and the expected performance exceed the capabilities of a single model.

For example:

Complex Medical Diagnosis: One model can provide an analysis based on symptoms, another can evaluate laboratory data, while a third can integrate genetic information. The gating model coordinates these interactions to deliver a comprehensive and accurate diagnosis.One model can analyze the overall sentiment of a text, another can assess specific cultural connotations, and a third can provide a comparative analysis with similar texts. This multi-model collaboration can offer a deep and nuanced understanding of content.

4. Mixture-of-AI Architecture

4.1 Analogy to Mixture-of-Experts

Mixture-of-Experts (MoE) is a well-established technique in machine learning that involves the use of multiple expert networks, each specialized in a particular skill or domain. A gating network directs inputs to the most appropriate experts, enabling efficient resolution of complex problems. This approach is particularly effective for handling diverse tasks, as it allows for the maximization of each expert's strengths.

Formally, for a given layer iii, the process can be represented by the following equation:

$$y_i = \sum_{j=1}^{n} G_{i,j}(x_i) E_{i,j}(x_i) + x_i$$

with :

- x_i is the input to the layer,
- $E_{i,j}$ is the j-th expert,
- $G_{i,j}$ is the gating weight assigned by the gating network for the j-th expert.

The Mixture of AI (MoAI) architecture draws inspiration from Mixture of Experts (MoE) but expands on this concept by integrating full language models (LLMs) as fundamental units of expertise. Instead of utilizing specialized sub-networks, MoAI leverages the full capacity of LLMs to address queries in a varied and nuanced manner.

In MoAI, each LLM is seen as an autonomous expert with a specific skill or range of skills. For example, one model might be specialized in generating creative text, while another could excel in synthesizing technical information or understanding language. This specialization allows MoAI to select the best model to handle each query or a specific part of it.

Example: Imagine a MoAI system used for customer support. When a customer asks a complex technical question, the gating network might redirect the query to an LLM specialized in technical terms, ensuring an accurate and well-informed response. If a question requires an empathetic and customer-oriented answer, another LLM specializing in human interactions might be called upon.

There are two types of layers in Mixture of Experts (MoE) architectures:

MoE Dense (Dense Mixture of Experts): In this configuration, all available experts are activated during each pass of the model. This method, while powerful in terms of modeling capacity, can be extremely computationally expensive as it requires running each expert for every input.

The output of the dense MoE layer can be formulated as:

$$\mathcal{F}_{\text{dense}}^{\text{MoE}}(\mathbf{x};\Theta,\{\mathbf{W}_i\}_{i=1}^N) = \sum_{i=1}^{N} \mathcal{G}(\mathbf{x};\Theta)_i f_i(\mathbf{x};\mathbf{W}_i),$$

$$\mathcal{G}(\mathbf{x};\Theta)_i = \text{softmax}(g(\mathbf{x};\Theta))_i = \frac{\exp(g(\mathbf{x};\Theta)_i)}{\sum_{j=1}^{N}\exp(g(\mathbf{x};\Theta)_j)},$$

where g(x;Θ) represents the gating value before the softmax operation.

MoE Sparse (Sparse Mixture of Experts): In contrast to MoE Dense, the MoE Sparse layer only activates a subset of experts for each pass. This selection is made by the gating network, which chooses the most appropriate experts based on the input. This method is more efficient in terms of computation and resources, as it reduces the number of experts involved, thus limiting the computational load.

The output of the MoE Sparse layer can be formulated as:

$$\mathcal{G}(\mathbf{x};\Theta)_i = \text{softmax}(\text{TopK}(g(\mathbf{x};\Theta)+\mathcal{R}_{\text{noise}},k))_i,$$

$$\text{TopK}(g(\mathbf{x};\Theta),k)_i = \begin{cases} g(\mathbf{x};\Theta)_i, & \text{if } g(\mathbf{x};\Theta)_i \text{ is in the top-}k \text{ elements of } g(\mathbf{x};\Theta), \\ -\infty, & \text{otherwise.} \end{cases}$$

4.1.2 Comparison and Preference for Mixture of AI

MoE Dense is often used in applications where maximum precision is critical, as activating all experts can improve the model's overall performance by aggregating diverse responses. However, its high computational cost limits its use, especially in environments with limited resources or where response time is a critical factor.

MoE Sparse, on the other hand, is preferred for scenarios where balancing performance and cost is important. By activating only the necessary experts, this method allows for high-quality responses without unnecessarily utilizing all available resources. This approach is particularly suited for systems like Mixture of AI, which aim to maximize efficiency while maintaining high performance standards.

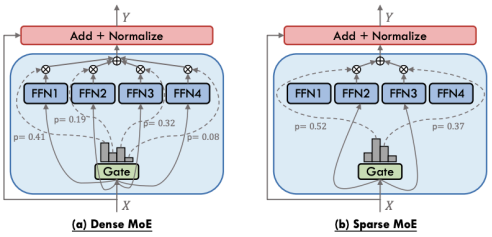


Fig4. MoE Dense vs Sparse Architecture

Comparison between MoE Sparse and Mixture of AI

Among the two types of Mixture of Experts (MoE) layers described, the MoE Sparse layer is the one that most closely resembles the operation of Mixture of AI.

In the MoE Sparse layer, a subset of experts is selected at each pass based on the specific needs of the task, significantly reducing computational load. Similarly, Mixture of AI uses a gating model to determine which large language models (LLMs) to mobilize for a given query. This gating model assesses the potential performance of each model and selects the ones most likely to provide the most relevant and highest-quality answers, while also considering resource constraints and other operational parameters.

The main advantage of MoE Sparse is the reduction in computational costs by activating only a subset of experts at each step, which is particularly useful when computational cost is a major concern. Analogously, Mixture of AI optimizes resource usage by activating only the necessary LLMs for each specific task, thereby saving energy and costs while maintaining high performance.

In both cases, special attention is given to balancing the workload among the experts or models to avoid overloading some components while leaving others underutilized. For MoE Sparse, this is addressed by an auxiliary loss function that balances the distribution of tokens across the experts. In Mixture of AI, this could be implemented by the gating model, which ensures a balanced distribution of queries, thereby optimizing the use of the different available LLMs.

The parallelism between MoE Sparse and Mixture of AI lies in their shared approach to managing computational complexity through selective resource allocation. While MoE Dense engages all experts at every step, MoE Sparse, and by extension Mixture of AI, strives to use only a specialized subset, thereby reducing costs and improving efficiency.

This approach ensures that resources are used optimally, maximizing performance while minimizing costs. It also allows for easier scaling, as new LLMs can be added or removed from the pool without requiring major changes to the architecture.

However, this approach can suffer from load balancing issues, where certain models may be over-utilized while others remain underutilized. It is also crucial to ensure that the gating model is sufficiently performant to make reliable and fast decisions.

4.2 Learning Algorithm

This section provides an overview of the different components of the Mixture of AI architecture, followed by a more detailed introduction to the problem our technology aims to solve. Unlike traditional Mixture of Experts (MoE) models, which operate at the level of activations or sub-networks, Mixture of AI operates at the level of full models. We define the interactions between multiple language models (LLMs) to process a single query.

In Mixture of AI, each query is routed through a set of nnn language models {M1,M2,…,Mn}, orchestrated by a central routing model called the gating model. The role of this routing model is to determine the optimal combination of LLMs to activate for each query, based on predefined collaboration instructions known as *mix instructions*.

Let's assume the input is a column vector $x \in R^D$. For each model M_i, we have a transformation of the input given by $M_i(x)$. The gating model uses a gating function $G(x;\Theta)$, parameterized by Θ, to determine the weights applied to the outputs of the activated models. This can be formally described as:

$$\mathbf{MoAI}(x) = \sum_{i=1}^{n} G(x;\Theta)_i M_i(\mathbf{x})$$

where G(x;Θ)i represents the probability or weight assigned to the model MiM_iMi for the query x.

The gating function G can be described by a linear transformation followed by a softmax function as:

$$\mathbf{G}(x;\Theta) = \mathbf{softmax}(W_g X + b_G),$$

Where:

- W_g is the weight matrix of the gating model,
- b_G is the bias vector of the gating model,
- The softmax function ensures that the resulting values are transformed into a probability distribution, with each $\mathbf{G}(x;\Theta)$ representing the probability or weight for each model M_i.

Mix Instructions are guidelines provided by the developer to determine how the LLMs interact with each other to solve a given query. These instructions define specific roles for each model, such as:

- **Proposers (Pi)**: Models responsible for generating initial responses. They may offer different perspectives on a given

question. The output of a proposer (P_i) for an input xxx is denoted as $P_i(x)$.

- **Aggregators (Ai)**: Models that combine the responses provided by the proposers to generate a final response. The aggregator A_i uses the outputs of the proposers to produce an aggregated response:

$$A_i(\{P_j(x)\})=A_i(P_1(x),P_2(x),P_3(x),...,P_k(x)),$$

where k is the number of proposers.

The gating model determines the proportions of each model used based on the input x, and the mix instructions define the collaboration process. Specifically, the gating model assigns weights to each model (proposers and aggregators) depending on the input and task, ensuring the most appropriate models are activated and their outputs combined in a manner that optimizes the overall result.

4.3 Routing/Gating

4.3.1 Routing/Gating Strategy

The gating mechanism of Mixture of AI (MoAI) is the core of the system, playing a central and essential role in query processing. Every routing decision passes through this component, which determines which model among the many available experts should be used to respond to a given query.

Mathematical Modeling

The gating process is defined by a dynamic function G, which selects the most appropriate language model for each query x. The selection is based on a set of weighted criteria that define a composite score for each model i, denoted as score $i(x)$.

The selected model iii is the one that maximizes this score:

$$\mathbf{G(x)}=argmax_{i\in\{1,2,...,N\}} score_i(x),$$

Where score$i(x)$ is calculated as:

$$score_i(x)= \alpha C_i(x) + \beta Q_i(x) + \gamma P_i(x) + \delta F_i(x) + \epsilon G_i(x)$$

With:

- $C_i(x)$: The cost associated with using model i,
- $Q_i(x)$: The quality of the expected response from model i,
- $P_i(x)$: The level of data privacy provided by model i,
- $F_i(x)$: The speed of the response from model i,
- $G_i(x)$: The carbon footprint associated with using model i.

The coefficients α,β,γ,δ,ϵ represent the relative importance of each criterion and can be adjusted to meet strategic priorities and specific user requirements.

4.3.2 Gating Model Training

The gating model of Mixture of AI is a central component that employs a hybrid architecture, integrating a large language model (LLM) with traditional neural networks. This combination leverages the deep contextual understanding of LLMs and the specific optimization capabilities of neural networks for routing decisions.

Training Data

To train our gating model, we primarily use a large multilingual corpus, including conversations and queries from various sources. A representative dataset has been selected, consisting of both real and synthetic queries, to cover a wide range of use case scenarios.

- Multilingual Corpus: This includes conversation data in multiple languages, with the majority in English (81%), followed by Chinese (3.1%) and French (2.2%). We have ensured a diverse sample set to capture linguistic and cultural specificities.
- Sample Filtering: We filtered the data to include various types of queries an end-user might ask, resulting in a database of approximately 10,000 comparisons across 22 different models.

To further diversify and enrich the training data, two additional datasets are included:

- Gold-Labeled Data (Dgold): High-quality samples manually validated by experts.
- AI-Judge Labeled Data (Djudge): Queries annotated using a state-of-the-art language model (GPT-4) to evaluate the quality of responses.

Training Process

The training process for the gating model follows these steps:

1. Query Encoding: User queries are first encoded into feature vectors {x} using a pre-trained LLM. This step provides a rich semantic representation of the textual inputs.
2. Metadata Integration: In addition to the query feature vectors, we integrate metadata {m} about the models, including cost, speed, carbon footprint, and privacy levels.
3. Neural Network: A neural network takes the combined vector c=[x;m] as input and is trained to predict the probability p of assigning each query to a particular expert model:
   p=softmax(Wc+b)
   where Wc and b are the parameters of the neural network.
4. Selection and Tuning: The model is validated using an independent test set, and hyperparameters are adjusted to optimize overall performance.

Performance Evaluation

The evaluation of the Mixture of AI gating model is a critical step to ensure the quality and reliability of the system. This evaluation focuses on several key aspects to assess the model's performance, robustness, and adaptability.

Academic Benchmarks
We use recognized academic benchmarks to evaluate the model's capabilities in varied and complex contexts. These benchmarks include:

MT Bench: Designed to evaluate text understanding and generation capabilities in complex multilingual scenarios. The test includes open-ended questions, where the responses are evaluated by LLM models or human judges. Scores are measured in terms of response quality, relevance, and contextual accuracy.

MMLU (Massive Multitask Language Understanding): This benchmark contains questions from 57 different subjects, totaling 14,042 questions. It assesses the model's ability to comprehend and process information across diverse knowledge domains, ranging from mathematics to history.

GSM8K: A set of mathematical problems at the school level, containing over 1,000 problems. This benchmark tests the model's reasoning and logical abilities.

…

Contamination Check
To ensure the integrity of the evaluation results, we perform rigorous checks to avoid data contamination, ensuring that test set data has not been included in the training set. This verification is essential for evaluating the model's ability to generalize to new data, or situations and questions it has not previously encountered. The procedure includes:

Overlap Detection: Identifying similarities between the training and test datasets using text matching and semantic verification techniques.

Proactive Exclusion: Excluding training data that shows excessive similarity to the identified test datasets, ensuring that the model is tested on genuinely new cases.

Optimization and Generalization
To maximize efficiency and generalization, we use advanced data augmentation and supervised learning techniques. The addition of Dgold and Djudge helps address gaps in the initial data and improves the similarity scores between the training datasets and benchmarks. These similarity scores serve as a metric for evaluating the model's ability to generalize to new query types, thus facilitating continuous improvement of the gating model's performance.

5. Dataset

5.1 Benchmark Dataset

The Benchmark Dataset is essential for training the gating model. In this section, we first describe the architecture for designing this dataset and then show the framework for training the gating model.

The **benchmark dataset creation system** is designed to evaluate the performance of LLMs based on their responses to queries. It includes several key components:

Question Database: This database contains a diverse set of queries designed to test the LLMs in a wide variety of real-world scenarios. These queries cover different domains of knowledge, including open-ended questions, factual queries, reasoning tasks, and multi-turn dialogues. The dataset includes both common and complex questions to assess the models' generalization and problem-solving abilities.

Mix-Instruction Database:

The mix-instructions are predefined directives that guide how the LLMs should collaborate with each

other when responding to a query. The instructions define specific roles for different models, such as proposer, aggregator, or evaluator, ensuring that each model participates in the problem-solving process according to its strengths.

LLMs for Responses: Multiple LLMs are used to generate answers to the queries in the question database. These models are chosen based on their capabilities, such as language understanding, reasoning, and problem-solving. Each model generates a response to each query according to the mix-instructions provided.

Advanced AI Model for Rating: A highly advanced AI model is used to rate the quality of the responses generated by the LLMs. This AI model evaluates the responses based on factors such as correctness, relevance, coherence, and fluency. The rating model is trained on a large set of labeled data and is designed to provide an objective measure of the response quality.

Human Supervision for Validation: To ensure the accuracy and reliability of the ratings, human supervisors are employed to validate the ratings provided by the AI model. The supervisors verify the AI's evaluations and correct any discrepancies, ensuring that the benchmark dataset reflects the true quality of the model outputs.

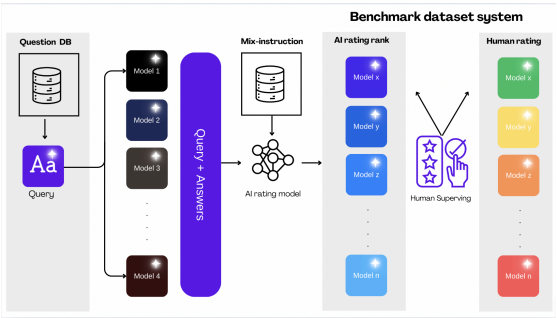


Fig 5.Benchmark dataset Architecture

5.1.1.2 Database of Mix-Instructions
5.1.1.2.1 Mixture of AI Basic

Mix-Instructions
The mix-instructions database is a central element of the Mo-AI system, allowing the configuration and customization of AI model behavior according to specific user needs. These instructions define how the model should prioritize its actions in terms of costs, speed, performance, environmental impact, and privacy. They are divided into two main categories: simple mix-instructions and compound mix-instructions.

Simple Mix-Instructions

Simple mix-instructions are precise, targeted directives that focus the system's attention on a single parameter when choosing an AI model to respond to a query. They play an essential role in optimizing system performance by allowing it to focus on a specific objective, whether it's cost reduction, processing speed, response quality, minimizing environmental impact, or data protection.

Cost Optimization
Cost optimization aims to minimize expenses associated with AI model usage while maintaining acceptable response quality. This instruction is crucial in scenarios where financial resources are limited, or in environments where cost management at scale is necessary.

When a cost optimization instruction is given, the system analyzes available models and selects one that offers a good compromise between cost and performance for a given task. If the task is simple, the system prioritizes less expensive models. For more complex tasks, it may adjust its strategy by selecting a slightly more expensive but better-performing model if necessary to ensure an adequate response.

Example: For a simple query like "What is the capital of France?", the system might choose a less expensive model like Gemma 2B, as the question is basic and doesn't require a highly sophisticated model. For a more complex question like "Explain the economic implications of Brexit", the system might opt for a slightly more expensive but better-performing model like Claude 3 Sonnet to ensure a more complete and accurate response.

Speed
Speed emphasizes response time reduction. This instruction is essential in time-critical situations, such as real-time response systems, financial services, or emergency response applications.

Under a speed instruction, the system prioritizes models capable of processing and generating responses quickly, even if this means higher costs or additional resource usage. The system evaluates each model's speed in terms of latency and ability to process large amounts of data quickly.

For an urgent query such as "What are the immediate measures to take in case of a cyberattack?", the system might choose a very fast model like Llama 3.1-70B, even if it costs more, because response speed is paramount in this context.

Performance

Performance focuses on the accuracy, quality, and relevance of the provided response. This is the key parameter in situations where accuracy is crucial, such as scientific research, medical diagnostics, or complex data analysis.

With a performance instruction, the system favors models that offer the best results in terms of accuracy and detail. These models are often more expensive and slower, but they are indispensable for tasks where accuracy cannot be compromised.

Example: For a complex query like "Comparative analysis of Keynes and Friedman's economic theories", the system might choose a highly performant model like GPT-4o, which is capable of providing a detailed and well-documented response, even if it requires more time and resources.

Environmental Impact (Green)
Environmental impact aims to minimize the ecological footprint of AI model usage. This instruction is important in contexts where sustainability is a priority, or when users seek to reduce their environmental impact while using technology.

Under an environmental instruction, the system favors models that are lighter, require less computing power, and thus consume less energy. These models are generally smaller in size or use algorithms optimized for energy efficiency.

Example: For a query like "Give me examples of daily ecological gestures", the system might select a lightweight model like Gemma 2B, which is designed to operate with a reduced carbon footprint.

Privacy
Privacy focuses on data protection and personal information security. This instruction is essential in fields like healthcare, finance, or any sector where sensitive data is at stake.

With a privacy instruction, the system selects models that offer high guarantees in terms of data security, such as models operating in local mode or with strict data management protocols. These models are Open Source.

Examples of Simple Mix-Instructions
- "Minimize my costs by 80% while maintaining quality."
- "Use the most economical models available, with cost optimization at 20%."
- "Prioritize cost savings at 60%, even if it reduces performance."
- "Provide the response as quickly as possible"
- "Favor models offering fast response time with speed above 300 tokens/s."
- "Maximize speed 70% of the time."
- "Ensure the highest quality with performance optimized at 90%."
- "Use the most powerful model to achieve 100% performance and reliability."
- "Focus on obtaining the best possible result with 75% performance optimization."
- "Choose a model with 70% reduced environmental impact."
- "Favor energy-efficient models with 50% impact reduction."
- "Use a lightweight model requiring less computing power."
- "Ensure maximum data privacy at 100%."
- "Use a model guaranteeing sensitive data privacy."
- "Prioritize privacy at 90%, even if it results in low performance"

Compound Mix-Instructions

Compound mix-instructions are more complex as they combine multiple parameters to create directives that meet specific requirements. They allow achieving a balance between several priorities based on user needs.

Examples of Compound Mix-Instructions:
- "Optimize my costs at 60% while ensuring 70% speed": Here, the AI must balance between cost savings and maintaining satisfactory response speed.
- "Ensure 80% performance and 90% privacy, even if it increases costs": This instruction requires the AI to choose models that are both performant and secure, without too much concern for costs.
- "Reduce environmental impact to 50% while maintaining 70% performance": The AI must find a compromise between energy efficiency and performance.
- "Prioritize 80% speed with 60% privacy, even if it leads to high costs": Instruction for an environment where speed and security are essential, with secondary costs.

5.1.1.2.1 Mixture of AI Advanced

- **Mix-instructions**

Chain of Thought (CoT) : Imagine a assembly line where each worker performs a specific task before passing the product to the next. In CoT, each AI model acts like a worker. One model generates an initial idea, a second model refines it, and so on until a final result is obtained.
How it works**:** Each model takes the output of the previous model as input and produces an output that will serve as input for the next model. It's a sequential and direct approach.
Example**:** To answer a complex question, a first model might identify keywords, a second might search for relevant information in a database, and a third might formulate a coherent answer.

Tree of Thought (ToT)

This is like exploring a maze with multiple possible paths. Instead of following a single path, ToT explores multiple options simultaneously. Each model can generate multiple ideas from the same input, creating branches in a tree.
How it works: The main model divides the problem into subproblems and creates branches for each subproblem. Each branch is explored by a different model. In the end, the results of all branches are compared to find the best solution.
Example: To solve a math problem, ToT could explore different approaches, such as factorization, using trigonometric formulas, etc.

Parallel Thought Chains

Imagine several teams working independently on different aspects of a project. In parallel thought chains, each model works on a specific part of the problem without interacting directly with the others.
How it works: Each model receives the same input and produces an independent output. The results of all models are then combined to obtain a final answer.
Example: To translate a text, one model could focus on grammar, another on vocabulary, and a third on translating specific terms.

Iterative Loops

This is a bit like refining a drawing: you start with a rough sketch, then add details in each iteration. In iterative loops, models work collaboratively to gradually improve the solution.
How it works: One model generates an initial solution, which is then evaluated by another model. Based on the evaluation, the solution is improved, and the process is repeated until a stopping criterion is met.
Example: To design a new product, one model could generate initial ideas, another model could evaluate the feasibility of these ideas, and so on until a final design is obtained.

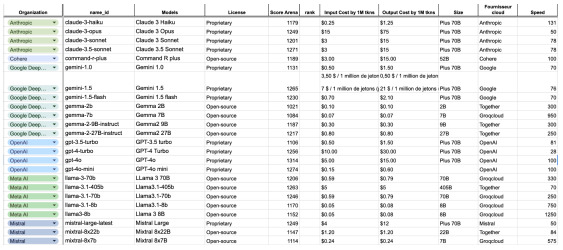
Hierarchical CoT

This is like a hierarchical organization where each level has a specific task. The main model delegates subtasks to specialized models, which can in turn delegate subtasks to other models.
How it works**:** The main model divides the problem into subproblems and assigns each subproblem to a specialized model. The results of the specialized models are then combined to obtain a final solution.
Example**:** To answer a science question, a main model could identify the scientific domain, a specialized model could search for relevant scientific articles, and another model could extract the relevant information from these articles.

5.2 Models

5.2.1 Model List

This section encompasses a vast collection of artificial intelligence models, each possessing specific characteristics and addressing various needs according to the mix-instructions defined by users. This database is designed to offer maximum flexibility in selecting the most appropriate models, taking into account complex criteria such as cost, performance, speed, environmental impact, and privacy.



Organization	name_id	Models	License	Score Arena	rank	Input Cost by 1M tkns	Output Cost by 1M tkns	Size	Fournisseur cloud	Speed
Anthropic	claude-3-haiku	Claude 3 Haiku	Proprietary	1179		$0.25	$1.25	Plus 70B	Anthropic	131
Anthropic	claude-3-opus	Claude 3 Opus	Proprietary	1249		$15	$75	Plus 70B	Anthropic	50
Anthropic	claude-3-sonnet	Claude 3 Sonnet	Proprietary	1201		$3	$15	Plus 70B	Anthropic	78
Anthropic	claude-3.5-sonnet	Claude 3.5 Sonnet	Proprietary	1271		$3	$15	Plus 70B	Anthropic	78
Cohere	command-r-plus	Command R plus	Open-source	1189		$3.00	$15.00	52B	Cohere	100
Google Deep…	gemini-1.0	Gemini 1.0	Proprietary	1131		$0.50	$1.50	Plus 70B	Google	70
						3,50 $ / 1 million de jeton	0,50 $ / 1 million de jeton			
Google Deep…	gemini-1.5	Gemini 1.5	Proprietary	1265		7 $ / 1 million de jetons	21 $ / 1 million de jetons	Plus 70B	Google	76
Google Deep…	gemini-1.5-flash	Gemini 1.5 flash	Proprietary	1230		$0.70	$2.10	Plus 70B	Google	70
Google Deep…	gemma-2b	Gemma 2B	Open-source	1021		$0.10	$0.10	2B	Together	300
Google Deep…	gemma-7b	Gemma 7B	Open-source	1094		$0.07	$0.07	7B	Groqcloud	950
Google Deep…	gemma-2-9B-instruct	Gemma2 9B	Open-source	1187		$0.30	$0.30	9B	Together	300
Google Deep…	gemma-2-27B-instruct	Gemma2 27B	Open-source	1217		$0.80	$0.80	27B	Together	250
OpenAI	gpt-3.5-turbo	GPT-3.5 turbo	Proprietary	1106		$0.50	$1.50	Plus 70B	OpenAI	81
OpenAI	gpt-4-turbo	GPT-4 Turbo	Proprietary	1256		$10.00	$30.00	Plus 70B	OpenAI	28
OpenAI	gpt-4o	GPT-4o	Proprietary	1314		$5.00	$15.00	Plus 70B	OpenAI	100
OpenAI	gpt-4o-mini	GPT-4o mini	Proprietary	1274		$0.15	$0.60	-	OpenAI	100
Meta AI	llama-3-70b	LLama 3 70B	Open-source	1209		$0.59	$0.79	70B	Groqcloud	330
Meta AI	llama-3.1-405b	Llama3.1-405b	Open-source	1263		$5	$5	405B	Together	70
Meta AI	llama-3.1-70b	Llama3.1-70b	Open-source	1248		$0.59	$0.79	70B	Groqcloud	250
Meta AI	llama-3.1-8b	Llama3.1-8b	Open-source	1170		$0.05	$0.08	8B	Groqcloud	750
Meta AI	llama3-8b	Llama 3 8B	Open-source	1152		$0.05	$0.08	8B	Groqcloud	1250
Mistral	mistral-large-latest	Mistral Large	Proprietary	1249		$4	$12	Plus 70B	Mistral	50
Mistral	mistral-8x22b	Mistral 8x22B	Open-source	1147		$1.20	$1.20	22B	Together	84
Mistral	mistral-8x7b	Mistral 8x7B	Open-source	1114		$0.24	$0.24	7B	Groqcloud	575

Fig 6.LLM list with characteristics

Model Characteristics

Each model in the database is evaluated and cataloged based on several key criteria:

- Cost

Input Cost per Million Tokens:This parameter measures the cost of processing 1 million input tokens. It allows comparison of different models' cost-effectiveness based on query size.
Output Cost per Million Tokens: This cost is associated with generating 1 million output tokens. It is crucial for applications where text generation is intensive.
Example: A model with an input cost of $0.10 per million tokens and an output cost of $0.50 per million tokens would be ideal for tasks requiring large amounts of input processing with moderate output.

- Performance

Model Benchmark Scores: Each model is evaluated in a competitive framework based on its ability to solve complex tasks, with a score reflecting its overall quality.
Ranking:The model's rank is established based on its performance score relative to other models in

the database. This ranking is used to guide model selection according to specific needs.
Example:A model with a score of 1287 would be ranked among high-performance models, ideal for tasks requiring maximum precision and quality.

- Speed

Processing Speed:This criterion measures how long a model takes to respond to a query. It is often expressed in milliseconds or in relative terms compared to other models and plays a crucial role in environments where latency is a critical factor.
Latency: A model's speed can be decisive in real-time applications, such as virtual assistants or algorithmic trading systems.
Example: A model with a relative speed of 900 tokens/s would be extremely fast, making it ideal for applications where response time is essential.

- Privacy

License: The nature of the license (proprietary or open-source) influences the degree of transparency, control, and security of the model. Open-source models are often preferred for projects requiring increased customization and control.
Privacy Level: Some models are optimized to handle sensitive data, ensuring enhanced security for applications dealing with confidential information.

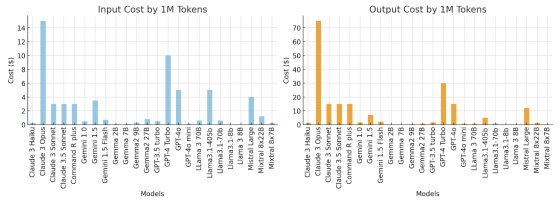


Fig7.These graphs provide a quick visualization of the differences between models in terms of costs.

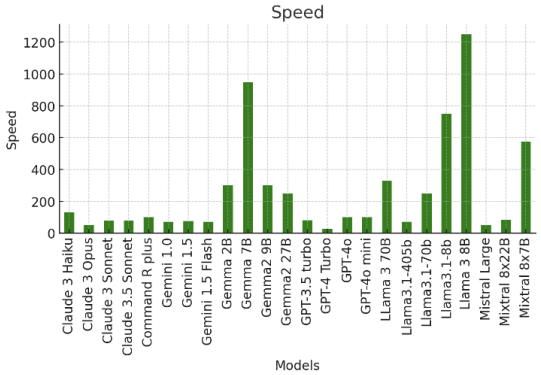


Fig. 8: These graphs provide a quick visualization of the differences between the models in terms of speed

- Environmental Impact (Green)

Model Size: Smaller models with fewer parameters generally consume less energy for training and inference. This size is an indicator of the model's ecological footprint.
Environmental Optimization:Some models are specially designed to minimize their ecological impact, whether through increased efficiency or the use of greener computational resources.
Example: A 7B parameter model hosted on a cloud infrastructure optimized for green energy would be preferred for tasks where environmental sustainability is an important criterion.

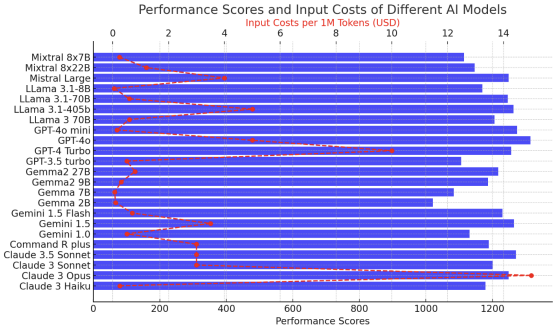


*Fig9.:Here's a graph that visualizes the **Performance Scores** (in blue) and **Input Costs** per million tokens (in red) for various AI models.*

This chart helps compare the models based on both their performance and cost, making it easier to assess trade-offs between cost and efficiency.

Fig9. LLM benchmarks

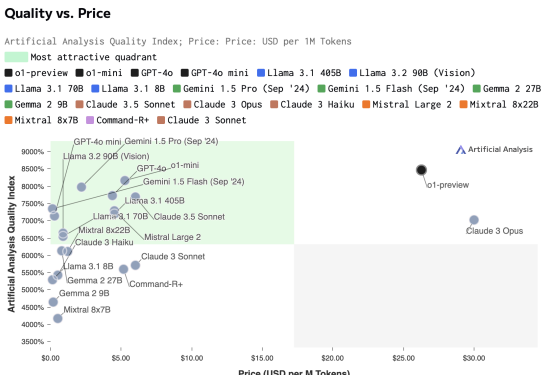


Fig10. Artificial analysis benchmarks about quality vs price

	MODEL	AVERAGE	MMLU KNOWLEDGE	GPQA EXPERT	MMMU VISION	HELLASWAG CONTEXT	HUMANEVAL CODING	BBHARD REASONING	GSM8K K6 MATH	MATH UNI MATH
1	Claude 3.5 Sonnet	82.25%	88.70%	59.4%	68.3%	89%	92%	93.10%	96.40%	71.10%
2	GPT-4o (0513)	81.69%	88.70%	53.6%	69.1%	94.20%	90.20%	91.30%	89.80%	76.60%
3	GPT-4 Turbo (0409)	79.10%	86.50%	48.0%	63.1%	94.20%	90.20%	87.60%	9%	72.20%
4	Llama 3.1 (405B)	79.01%	88.60%	51.1%	64.5%	87%	89%	81.3%	96.80%	73.80%
5	Mistral Large 2 (0724)	78.80%	84%	35.1%	–	89.20%	92%	87.30%	93%	71%
6	Claude 3 Opus	77.35%	86.80%	50.4%	59.4%	95.40%	84.90%	86.80%	95%	60.10%
7	Llama 3.1 70B	75.65%	86%	46.7%	60.6%	87%	80.50%	81.30%	95.10%	68.0%
8	Gemini 1.5 Pro	73.61%	81.90%	46.2%	62.2%	92.50%	71.90%	84%	91.70%	58.50%
9	GPT-4 (0314)	71.15%	86.40%	35.7%	56.8%	95.30%	67%	83.10%	92%	52.90%
10	GPT-4o Mini	70.70%	82%	40.2%	59.4%	–	87.20%	–	–	70.20%
11	Claude 3 Sonnet	69.85%	79%	46.4%	53.1%	89%	73%	82.90%	92.30%	43.10%
12	Gemini 1.5 Flash	68.63%	78.90%	39.5%	56.1%	81.30%	67.50%	89.20%	68.80%	67.70%
13	Claude 3 Haiku	66.10%	75.20%	40.1%	50.2%	85.90%	75.90%	73.70%	88.90%	38.90%
14	Llama 3.1 8B	64.29%	73.0%	32.8%	–	74.20%	72.60%	61%	84.50%	51.90%
15	Mistral Nemo 12B	41.30%	68%	8.72%	–	83.5%	–	–	–	5%

Fig11. LLM benchmarks

5.2.2 Inference Servers

Inference servers play a central role in the artificial intelligence ecosystem by enabling the execution of pre-trained AI models in real-world environments. Unlike the training phase, where the goal is to learn models from large datasets, inference consists of using these models to make predictions or generate content based on new input data.

The performance of inference servers is therefore essential for providing real-time or near-real-time responses, especially in applications such as virtual assistants, recommendation systems, or image and text generation platforms.

a. Operation of Inference Servers

Inference servers handle the execution phase of an already trained AI model using optimized hardware and software architectures. They are designed to process requests with low latency while minimizing resource consumption to offer competitive scalability and operating costs. These servers generally fall into three main categories:

On-premise Servers

These solutions are installed locally in a company's private data centers. They offer complete control over data and resources used but require high initial infrastructure and maintenance costs. They are particularly popular in regulated sectors or for companies with very specific needs.

Cloud-based Servers

Major cloud computing providers such as AWS, Microsoft Azure, and Google Cloud offer flexible, on-demand inference server infrastructures. This approach allows for high scalability and reduced initial investment costs by only paying for resources used. Cloud services are ideal for startups and companies seeking rapid implementation with minimal infrastructure management.

Hybrid Servers

Combining the advantages of both previous approaches, these servers allow switching between on-premise and cloud environments as needed. This enables maintaining control of sensitive data while benefiting from the flexibility and scalability of cloud resources.

b. Characteristics of Inference Servers

Inference servers must meet several requirements to optimize the performance of generative AI models:

Low Latency

Real-time applications, such as AI assistants or image generation, require very short response times. High-performing servers guarantee latency in the millisecond range, allowing them to handle millions of simultaneous requests.

High Throughput

For generative models such as GPT or DALL·E, the number of queries processed per second (QPS) must be high to ensure continuous service availability. Inference servers are therefore designed to maximize the number of requests processed in parallel.

Hardware Optimization

The use of specialized processing units, such as GPUs (Graphics Processing Units), TPUs (Tensor Processing Units), or ASICs (Application-Specific Integrated Circuits), accelerates the calculations needed for neural networks, thus reducing processing times.

Memory Management

Large models, such as transformer networks, require extensive memory for temporary storage of weights and activations. Inference servers optimize this management to avoid bottlenecks, particularly during inference phases where many network layers are involved.

c. Challenges of Inference Servers

With the rise of generative AI models, inference servers must address several technical and economic challenges:

Energy Consumption

Training large AI models is already very energy-intensive, and repeated inference adds additional pressure. Server providers invest in energy-efficient architectures to reduce the carbon footprint of AI systems, particularly those used at scale.

Cost

The cost of running models on inference servers can become prohibitive, especially when these models are used for commercial purposes with millions of users. Additionally, infrastructure costs vary depending on latency, throughput, and specific AI application requirements.

Security and Privacy

With the increasing use of AI models in sensitive environments (finance, healthcare, etc.), data security is a major priority. Inference servers must offer solid guarantees in terms of data encryption in transit and at rest, as well as mechanisms to prevent leaks of sensitive information from inferences.

d. Major Inference Server Providers

Major players in this field distinguish themselves through their infrastructure capabilities, optimization of their services for AI models, and customization they offer to businesses. Among the major providers, we find:

- Amazon Web Services (AWS)

AWS offers AI-optimized EC2 instances and infrastructures based on GPUs and FPGAs (Field Programmable Gate Arrays), enabling high-performance real-time inference. AWS also offers SageMaker, a complete development platform for model training and inference.

- Google Cloud

With its TPUs dedicated to AI inference, Google Cloud showcases an optimized infrastructure for AI applications, particularly in computer vision and natural language processing (NLP). Vertex AI allows managing the entire lifecycle of AI models, from training to inference.

- Microsoft Azure

In partnership with OpenAI, Microsoft Azure offers a wide range of inference services via Azure Machine Learning and Cognitive Services. The partnership with OpenAI provides access to cutting-edge generative models, such as GPT-4, with simplified integration via robust APIs.

GroqCloud

Specializing in hardware architectures for AI inference, GroqCloud offers innovative solutions for businesses requiring ultra-low latency, particularly in critical environments like finance and healthcare.

- Together AI

Backed by Nvidia, Together AI enables developers to build and customize AI models through several services:

Inference: Execution of over 100 open-source models on serverless or dedicated instances

Fine-tuning: Adjustment of generative AI models with proprietary data while maintaining data ownership

GPU Clusters: Access to GPU clusters optimized for training and fine-tuning

Custom Models: Ability to train models from scratch with different architectures

Role in Mo-AI

In the Mixture of AI (Mo-AI) architecture, inference servers form the backbone of model deployment and selection. Since Mo-AI leverages a mix of different AI models (predictive, generative, etc.) based on task requirements, the inference layer is responsible for dynamically redirecting tasks to the most appropriate model, whether for image generation, text analysis, or other functions. This dynamic redirection ensures that the optimal model is selected not only in terms of performance but also in terms of cost and energy efficiency, in line with Green AI principles.

List of Inference Servers:

GroqCloud: Known for low latency, ideal for real-time applications

Together: Provides scalable infrastructure for both training and inference

OpenAI: Offers high performance for large language model inference

Anthropic AWS: Focuses on AI security with robust privacy standards

Google Cloud: Offers versatile inference capabilities with built-in optimization tools

Microsoft Azure: Flexible and scalable with excellent enterprise tool integration

Mistral: Specialized in efficient inference for large-scale generative models

Cohere: Offers fast inference services focusing on textual applications

Private Servers: For companies with specific privacy requirements or unique infrastructures

6. Experiments

The purpose of this section is to present in detail the experiments conducted to evaluate the effectiveness of the Mixture of AI system. This system relies on a gating model (or selection model) capable of choosing the best artificial intelligence model for a given task, as well as mixed model configurations optimized for different metrics (cost, speed, performance, environmental impact, privacy).

6.1 System Setup

To ensure a rigorous evaluation of our approach, we established an experimental framework based on the following aspects:

The use of distinct questions covering various domains
Implementation of a gating model designed to analyze these prompts and determine the most appropriate AI model to respond
Comparisons with a classical AI assistant and a random selection model to verify that our system offers a more judicious choice than alternatives based on static or non-optimized decisions
Selected prompts come from several application domains:
Complex mathematical problem solving
Creative text generation
Multilingual translation
General knowledge questions
Tabular data analysis and statistical inference.

These different tasks will allow evaluation of whether the gating model knows how to choose the right model based on context and task complexity.

Gating Model Evaluation
The gating model plays a central role in our approach. It analyzes prompt characteristics and uses specific criteria to select the most suitable AI model for each situation. To test the effectiveness of this process, we submitted 100 prompts/questions to the gating model and measured the accuracy of its selections.
Three main configurations were tested:
Gating Model: Our dynamic selection system that chooses a model based on prompt requirements
Classical AI Assistant: An assistant using a single model for all questions, without distinction between tasks
Random Model Selector: A model that randomly selects from a predefined set, without basing decisions on prompt characteristics.

6.2 Benchmark Results

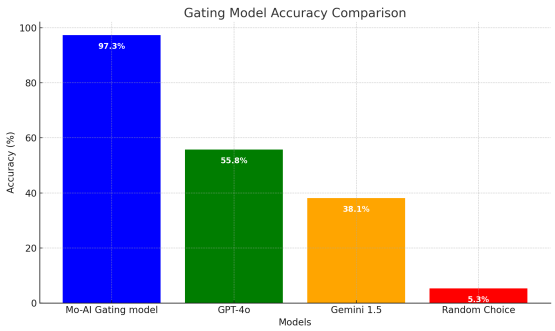


Fig12. Gating model accuracy comparaison

This comparison highlights the clear advantage of the Mo-AI Gating model.

6.3 Main results

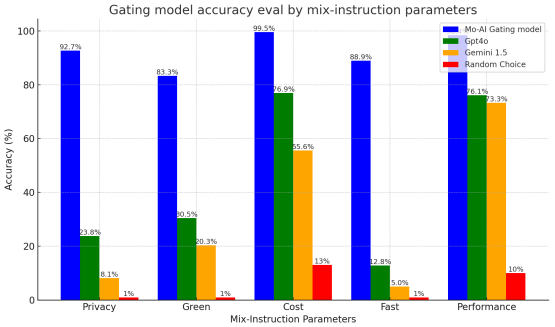


Fig13. Gating model accuracy eval by mix-instructions

This analysis focuses on evaluating the accuracy of different AI models, including the Mo-AI Gating model, GPT-4o, Gemini 1.5, and Random Choice, across five key instruction parameters: Privacy, Green (Environmental Efficiency), Cost, Speed (Fast), and Performance.

6.4 Mixture Analysis

Evaluation of Mix Models
The purpose of evaluating mix models is to test the performance of combinations of AI models on various tasks, particularly on standardized benchmarks. The idea behind using mixed models is that combining multiple specialized models can yield better results than using a single generalist model.
The mixed models will be tested on several datasets and benchmarks, including the well-known MMLU. This benchmark is particularly well-suited for testing the capabilities of AI models across a wide range of topics, from linguistic comprehension to solving more technical problems.

Mix Model 1: Opelamis C
Setup

- Models: GPT-4o + Llama 3.1 8B + Claude 3.5 Sonnet
- Type of mix: Basic mixture
- Mix instructions: Cost optimization 50%

Opelamis C (Mo-AI mixed model) Evaluation: Quality vs Price

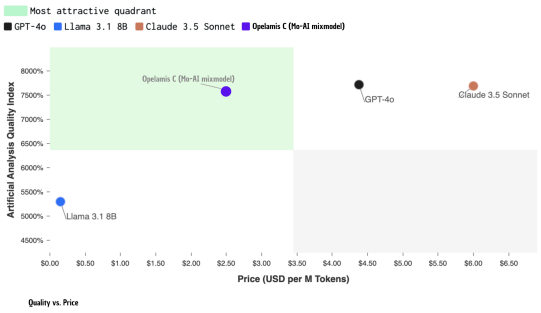


Fig14. Opelamis C evaluation Quality vs Price

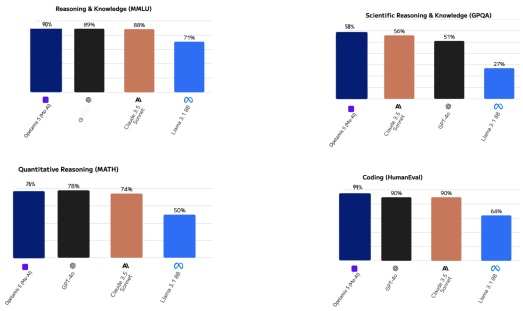


Fig15. Opelamis C result on Benchmark

Mix Model 2: Opelamis S
- Setup
Models: GPT-4o + Llama 3.1 8B + Claude 3.5 Sonnet
Type of mix: Basic mixture
Mix instructions: Increase speed 80%

Opelamis S (Mo-AI mixed model) Evaluation :Quality vs Price
Overall Performance:
Opelamis S is a hybrid model designed to maximize efficiency by combining the strengths of three powerful models: GPT-4o for precision and understanding, Llama 3.1 8B for speed, and Claude 3.5 Sonnet for analytical depth. This mix aims to increase speed without sacrificing the quality of analysis.

Speed: Thanks to the integration of Llama 3.1 8B and optimized mix instructions, Opelamis S has a clear advantage in terms of speed, achieving an 80% increase in speed. It generates a higher number of tokens per second than each of the individual models.
Quality: By combining the fine analytical qualities of Claude 3.5 Sonnet and the precision of GPT-4o, Opelamis S offers a balanced solution that provides both strong understanding performance and extended analytical capability.
Cost: As a mixed model, Opelamis S optimizes cost by using each model where it excels, enabling more economical management while delivering superior performance in terms of quality and speed.

GPT-4o

Strengths:
Precision: GPT-4o is recognized for its ability to produce highly precise and nuanced responses. It excels in understanding complex contexts and tasks requiring analytical rigor.
Applications: Ideal for tasks where response quality is paramount, such as complex content creation, translation, or in-depth analysis.
Limitations:
Speed: GPT-4o is not the fastest model, especially compared to Llama 3.1 8B. Its response time may be a bottleneck in environments requiring immediate results.
Cost: Generally, GPT-4o has a relatively high cost per million tokens due to its superior precision.

Llama 3.1 8B

Strengths:
Speed: Llama 3.1 8B is the fastest model in the group, capable of generating tokens at a very high speed. This makes it ideal for real-time tasks or applications where speed is essential.
Cost: Llama 3.1 8B is also highly competitive in terms of cost, with a per-million-tokens price significantly lower than GPT-4o and Claude 3.5 Sonnet.
Limitations:
Analysis Quality: While fast, Llama 3.1 8B's analytical performance and text understanding fall short of GPT-4o and Claude 3.5 Sonnet. It may lack precision in complex or nuanced tasks.

Claude 3.5 Sonnet

Strengths:
Analytical Depth: Claude 3.5 Sonnet excels in tasks requiring detailed analysis. It can generate deep and nuanced responses, particularly in contexts needing complex deductions.
Response Quality: Its answers are often longer and more detailed, making it a good choice for applications like research or in-depth consultations.
Limitations:
Speed: Claude 3.5 Sonnet is slower than the other models, making it less suitable for scenarios requiring fast responses.
Cost: Its price per million tokens is typically higher, which can be a disadvantage in contexts where cost optimization is critical.

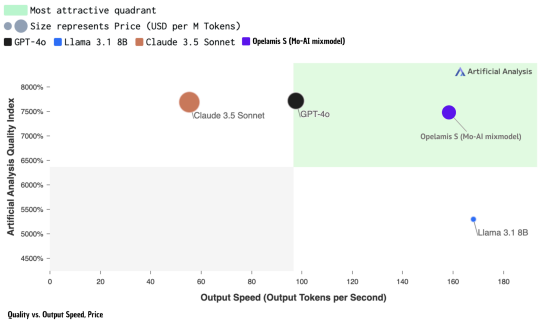


Fig16. Opelamis S evaluation Quality vs Price vs Speed

Opelamis S positions itself as a versatile model with an ideal compromise between speed, quality, and cost. It is particularly well-suited for environments where responsiveness is essential, without sacrificing precision and analysis.

Mix Model 3: Ged
Setup

- Models: GPT-4o + GPT-4o + GPT-4o + GPT-4o
- Type of mix: Advanced mixture
- Mix instructions: Parallelized Thought Chains

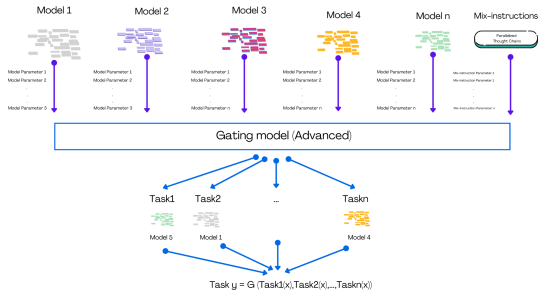


Fig17. Parallelized Thought Chains architecture

Here's how MoAI operates to create the mixmodel **Ged**:
Process:
Model 1: This instance might focus on providing a general, broad understanding of the problem or question at hand, establishing the core ideas.
Model 2: This instance could delve deeper into technical or highly detailed aspects, ensuring the response has solid and specific information.
Model 3: Another GPT-4o instance might analyze contextual elements, providing the background or historical perspective of the topic.
Model 4: The final instance could explore alternative interpretations, counterarguments, or complex nuances.
The **gating model** receives the outputs of all four models and evaluates them based on relevance, diversity, and the ability to address the core question.

It assigns **weights** to each model's output, considering how they complement one another. For instance, if Model 2's technical depth is essential for this task, it might weigh that output higher.
The gating model also ensures the final answer is coherent, even though it incorporates multiple viewpoints.
The final response is a synthesized answer that:
- Balances depth and breadth of information, pulling from both the highly specific and the broader contextual perspectives.
- Ensures no critical angle is missed, whether technical details, historical context, or alternative interpretations.
- Presents a unified response while incorporating the variety offered by the four models.

Mix model 4: Valé

- **Setup**

Models : GPT-4o + GPT-4o + Claude 3.5 Sonnet + Gemini-1.5 Flash
Type of mix : Advanced mixture
Mix instructions : **Chain of Thought**

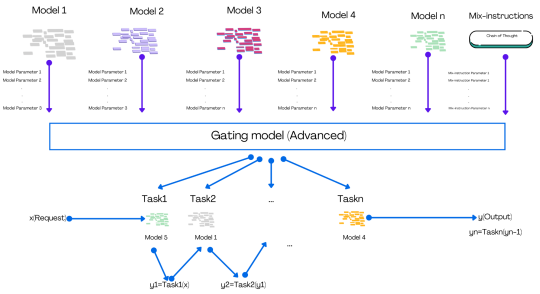


Fig18. Chain of Thought architecture

Here's how MoAI Operates to Create the Mix Model Valé:

In this advanced CoT setup, the Valé model leverages the strengths of multiple LLMs to form a highly structured response. By using the Chain of Thought approach, the model performs step-by-step reasoning, where each model contributes a unique layer of understanding.

Process:

Model 1 (GPT-4o):

Focus: Initial problem interpretation and question decomposition.

Role: This instance begins by breaking down the question into its core elements and identifies the key aspects that need to be addressed. It sets up a logical structure for the answer, helping subsequent models understand the sequence of reasoning needed.

Model 2 (GPT-4o):

Focus: Detailed analysis and intermediate reasoning.

Role: This instance takes the question breakdown from Model 1 and delves into detailed analysis. It focuses on technical accuracy and in-depth reasoning, particularly on quantitative or complex data aspects, enhancing the answer's depth.

Model 3 (Claude 3.5 Sonnet):

Focus: Contextual and interpretive analysis.

Role: Claude 3.5 Sonnet adds layers of contextual understanding and interpretation. This model considers the broader implications, synthesizing historical background, real-world applications, and ethical considerations, and ensuring the response is not overly narrow.

Model 4 (Gemini-1.5 Flash):

Focus: Synthesis, nuance, and finalization.

Role: Gemini-1.5 Flash evaluates the reasoning chain up to this point, identifying any gaps or overlooked nuances. It integrates counterpoints, alternative interpretations, or additional insights, ensuring the final response is well-rounded and nuanced.

Gating Model's Role:

The gating model orchestrates the process, ensuring that each model's output aligns with the overall Chain of Thought. It uses the following mechanisms:

Sequential CoT Execution: The gating model ensures that each model builds on the previous one's output in a coherent sequence. It guides the process from initial interpretation to detailed analysis, contextual synthesis, and nuanced finalization.

Relevance and Cohesion Checks: At each stage, the gating model checks that the response aligns with the original question and maintains logical coherence. If there's any inconsistency or if one model's output deviates from the core answer, the gating model may loop back to a previous stage or adjust the chain.

Adaptive Weighting: Each output's weight is adjusted based on its importance in the chain. For example, if Gemini-1.5 Flash's nuanced insights are vital for the question, the gating model will emphasize that output. This adaptive weighting ensures that critical perspectives are prioritized without sacrificing coherence.

Final Output:

The final answer produced by Valé is a meticulously crafted response that:

Follows a logical reasoning chain from problem interpretation to final synthesis.

Balances depth and breadth by integrating highly technical analysis with broader contextual insights.

Addresses multiple perspectives, incorporating initial interpretations, detailed analysis, contextual elements, and nuanced viewpoints.

Presents a coherent, unified response that respects the Chain of Thought process, with each model's contribution reinforcing the others

Mix model 5: Paulo

- **Setup**

Models : GPT-4Turbo + GPT-4o + Claude 3.5 Sonnet +Mistral-Large 2
Type of mix : Advanced mixture
Mix instructions : Tree of Thought

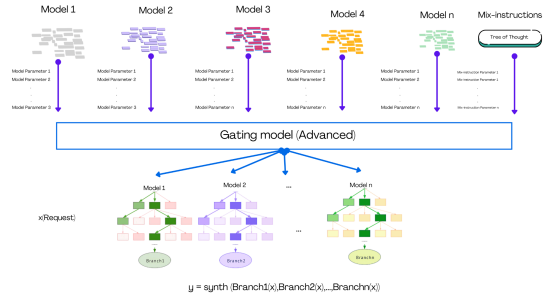


Fig19. Three of Thought architecture

How MoAI Operates to Create the Mix Model Paulo:

In this advanced setup, the Paulo model uses a "Tree of Thought" (ToT) approach, where the solution process is structured like a branching tree. Each model explores different "branches" of reasoning based on the initial question, enabling a wide exploration of possibilities and potential answers. This structure allows the model to examine multiple directions and refine the final response by selecting the most relevant branches.

Process:

Model 1 (GPT-4Turbo):

Focus: Initial exploration and branching.

Role: GPT-4Turbo initiates the response by generating a set of initial ideas, hypotheses, or approaches to the question. It quickly explores broad and diverse potential paths that can serve as different "branches" in the thought tree. These branches set up multiple possible directions for subsequent models to evaluate and expand on.

Model 2 (GPT-4o):

Focus: Technical deepening of branches.

Role: GPT-4o takes each of the branches proposed by GPT-4Turbo and adds depth to them, performing more detailed analyses. It might explore the technical or logical implications of each branch, discarding paths that seem less viable and strengthening those that appear promising. This model refines the initial branches into a more structured set of options.

Model 3 (Claude 3.5 Sonnet):

Focus: Contextual enrichment and further branching.

Role: Claude 3.5 Sonnet takes the refined branches from GPT-4o and assesses their broader implications, including ethical, social, and historical contexts. This model also introduces additional sub-branches where needed, expanding on the paths that could benefit from a deeper contextual layer. Claude's exploration brings alternative perspectives and nuances into the thought tree, helping identify branches with strong potential.

Model 4 (Mistral-Large 2):

Focus: Critical evaluation and pruning.

Role: Mistral-Large 2 evaluates all branches and sub-branches, deciding which paths are most relevant and viable for the final answer. This model applies a "pruning" approach, where less relevant branches are cut, and the focus narrows on the most promising paths. Mistral-Large 2 ensures the final branches represent the best of all possible approaches explored, refining the thought tree to a concise, high-quality set of conclusions.

Gating Model's Role:

The gating model orchestrates the process, enabling the Tree of Thought to grow and branch effectively, with the following key functions:

Branch Tracking and Management: The gating model manages the branches created by each model, ensuring no path is left unexplored until it has been evaluated. It ensures that branching is methodical and efficient, directing models to expand or prune specific areas of the thought tree as necessary.

Dynamic Pruning and Selection: As each model contributes to the thought tree, the gating model dynamically prunes weaker branches. It ensures that only the most promising paths continue to receive attention, maximizing efficiency and avoiding redundant exploration.

Prioritization of Branches: The gating model weighs each branch's relevance, determining which ideas should be prioritized based on the question's requirements. For example, if technical depth is critical, branches developed by GPT-4o are prioritized; if contextual insights are more relevant, Claude's branches receive more weight.

Final Output:

The final answer produced by Paulo is a carefully curated response that:

Presents a refined path through the thought tree, focusing on the most relevant branches.

Integrates multiple perspectives and layers of analysis, from broad ideas to technical depth and contextual insights.

Eliminates irrelevant paths, providing a concise and high-quality answer by selecting only the best branches.

Reflects a well-rounded understanding of the question, synthesizing both in-depth exploration and a broad view of possible solutions.

Mix model 6: Juni

- **Setup**

Models : GPT-4 Turbo + GPT-4o + Claude 3.5 Sonnet +Mistral-Large 2
Type of mix : Advanced mixture
Mix instructions : **Loop Iteration of Thought**

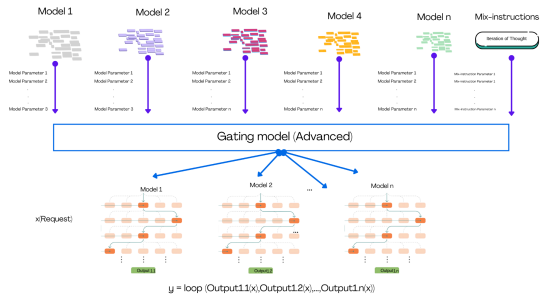


Fig20. Loop Iteration of Thought architecture

How MoAI Operates to Create the Mix Model Juni:

In this configuration, the Loop Iteration of Thought (IoT) approach is employed. This method involves a continuous, looped refinement process, where each model contributes iteratively in cycles until a desired quality threshold is met. Each iteration loops through the models, with each one building on the outputs of the others, refining and enhancing the response in a structured, ongoing manner.

Process:

Model 1 (GPT-4Turbo):

Focus: Initial response generation and core idea formation.

Role: GPT-4 Turbo begins the loop by creating a foundational answer. This first iteration is a general response covering key concepts and establishing a framework that subsequent models will build upon. The output is broad and foundational, setting the stage for detailed, iterative refinement.

Model 2 (GPT-4o):

Focus: Technical depth and specificity.

Role: GPT-4o takes GPT-4Turbo's initial output and iterates on it, focusing on deepening the technical accuracy and adding detailed insights. This model enhances the response with specific information and corrects any inaccuracies, providing a more solid basis for the subsequent models to refine further.

Model 3 (Claude 3.5 Sonnet):

Focus: Contextualization and diverse perspectives.

Role: Claude 3.5 Sonnet reviews the technically enriched response and adds layers of context, historical background, or ethical considerations as appropriate. This model ensures the response is well-rounded, incorporating diverse viewpoints that enrich the answer with social, cultural, or philosophical nuances.

Model 4 (Mistral-Large 2):

Focus: Coherence, synthesis, and final polish.

Role: Mistral-Large 2 synthesizes the responses from the previous models, ensuring coherence and clarity. It integrates all aspects into a single, unified answer while enhancing readability and flow. Mistral-Large 2's iteration provides a polished, high-quality response, ready for evaluation by the gating model.

Looping Mechanism with Gating Model:

The gating model plays a crucial role in managing the looped iterations, ensuring that each pass through the models adds meaningful improvements to the response. Here's how it operates:

Iteration Trigger: After each full cycle through the models, the gating model assesses the response's quality against predefined criteria. If the response does not meet the quality threshold, the gating model initiates another loop, restarting the process with GPT-4 Turbo.

Adaptive Refinement: The gating model dynamically adjusts the focus of each model in subsequent loops, guiding them to refine particular aspects that need improvement. For instance, if the response lacks depth, it might direct GPT-4o to focus more intensively on technical details in the next cycle.

Convergence Check: As the response reaches higher quality, the gating model monitors for convergence, ensuring that further iterations do not lead to redundant refinement. It aims to prevent overfitting the answer and stops the loop once the response is comprehensive and well-balanced.

Final Output:

The final output from Juni, after several looped iterations, is a highly refined answer that:

Reflects continuous improvement from multiple iterative passes, with each model adding layers of accuracy, context, and polish.

Balances technical detail, contextual depth, and coherence, achieving an optimal mix of different perspectives.

Adapts to the question's complexity, with the number of iterations flexibly determined by the gating model's evaluation.

Minimizes redundancy, as the gating model stops the loop once further iterations no longer add significant value.

7. MoAI Memory

Large Language Models (LLMs) like GPT-4, Claude, and others have shown remarkable capabilities in natural language understanding, generation, and problem-solving. However, as their applications become increasingly integrated into real-world scenarios, the inefficiencies of their memory systems have become more apparent. LLMs traditionally retain full conversational

histories, leading to a rapid accumulation of tokens, which in turn increases computational load and cost.
The use of large language models (LLMs) is transforming various industries through their capacity for conversational, generative, and analytic tasks. Managing the memory of LLMs effectively is essential for both reducing costs and improving performance. We will explain here the MoAI Memory, an innovative memory optimization technique tailored for AI systems, emphasizing token reduction, memory structure choices, and the integration of memory tuning for enhanced accuracy.

7.1 LLM Background

7.1.1 Overview

LLMs are based on deep learning architectures, particularly transformer models, which leverage vast datasets to learn language patterns. These models, such as OpenAI's GPT and Google's Gemini, are trained through a process known as unsupervised learning, where they predict the next word in a sentence based on the preceding words. Despite their advanced capabilities, traditional LLMs have limitations, particularly in maintaining contextual awareness across extended dialogues.The need for memory in conversational AI arises from the inherent nature of human conversation, where context and previous interactions significantly influence ongoing dialogue. In applications like customer service, personal assistants, and interactive storytelling, the ability to recall past interactions is crucial for delivering personalized and relevant responses. The integration of memory mechanisms in LLMs can enhance user engagement and satisfaction by creating a more coherent conversational experience.
LLMs generate text by predicting tokens sequentially, utilizing probabilistic models that analyze patterns in the input data. The process begins with an initial input, where the model computes a distribution over potential next tokens and selects the most probable one. This prediction relies heavily on the context provided by previous tokens, allowing the model to generate coherent and contextually appropriate responses.To improve the quality of token generation, memory can be integrated into the model's architecture. By retaining key information from previous interactions, LLMs can leverage this contextual memory during token prediction, leading to responses that reflect an understanding of user preferences, prior topics discussed, and specific user needs. This integration allows for a more nuanced interaction, as the model can reference past conversations and build upon them, ultimately enhancing the conversational flow.

7.1.2 Memory Architectures

Memory in LLMs can be categorized into short-term and long-term memory systems. Short-term memory retains context from the current session, enabling the model to recall recent exchanges and maintain coherence. Long-term memory, on the other hand, stores information across multiple sessions, allowing the model to remember user-specific details, preferences, and historical interactions.

Several techniques are employed to manage memory effectively:

Embedding Memory: This involves encoding important information into the model's latent space. Through embedding, key pieces of information become integrated into the model's understanding, enabling efficient retrieval during conversations.

Memory Slots: The use of structured memory slots allows the model to categorize and store information based on topics or interactions. Each slot can hold relevant details that the model can access when generating responses.

Attention Mechanisms: Attention mechanisms play a crucial role in memory access, allowing the model to focus on relevant parts of its memory when generating responses. By assigning varying degrees of importance to different memories, the model can tailor its output based on the context of the conversation.

7.2-MoAI Memory: Optimizing Large Language Model Memory for Efficiency

The MoAI Memory integrates several key strategies for optimizing memory use, enhancing both performance and accuracy in language models. Each technique addresses specific challenges associated with memory management and contributes to the overall efficiency of the model.

7.2.1. Dynamic Memory Allocation (DMA)

Dynamic Memory Allocation (DMA) enables the model to adjust its memory resources according to the demands of the task at hand, providing a flexible approach to handling contextual information.

Dynamic Memory Allocation can be modeled by a function M(x), representing memory usage in relation to task x. This function adjusts the memory resources allocated based on the contextual parameters of x.

Adaptive Scaling:

The DMA system evaluates the context of each task to determine optimal memory needs. For example, when processing lengthy conversations or data-rich queries, MoAI Memory dynamically expands memory resources to support complex contextual understanding. In less demanding scenarios, it scales down memory usage to minimize computational overhead, thus conserving resources.

Let C(x) be the contextual complexity of task x, which can be defined as a function of several variables, including input sequence length L(x) and data volume D(x). The allocated memory M(x) is then proportionally adjusted to this contextual complexity:

$$M(x) = k \cdot f(C(x)) = k \cdot (a \cdot L(x) + b \cdot D(x))$$

where k, a, and b are adjustment constants. Thus, if C(x) is high (e.g., for a task requiring long-term context), M(x) increases. For simpler tasks, M(x) decreases, thereby optimizing memory usage.

Relevance-Driven Storage:

To avoid memory clutter, the framework evaluates the relevance of information in real time. Using relevance assessment algorithms, it prioritizes storing only pertinent data, while discarding or temporarily archiving less relevant information. This approach reduces latency and ensures quick access to essential details, enabling the model to maintain a focused context.

Relevance-based storage can be represented by a data selection function S(r,t), which evaluates the relevance r of information in real-time t. Each data point is assigned a relevance weight P(x), and only data with P(x)>δP(x) (a relevance threshold) are stored. This allows filtering out non-essential information and focusing on critical elements.

$$S(r,t) = \{d \in D : P(d) > \delta\}$$

where d represents data, and D the set of available data. This reduces latency, as only relevant data S(r,t) is readily accessible.

Optimized Retrieval:

By allocating memory dynamically and only retaining relevant information, DMA reduces the model's need to scan through extensive, non-essential data. This prioritization enhances retrieval speed and responsiveness, which is especially valuable for real-time applications.

The optimized retrieval function, R(S(r,t)), relies on S(r,t) to limit search to relevant data. Retrieval is therefore modeled as a function proportional to the number of elements in S(r,t), reducing retrieval time:

$$R(S(r,t)) = \alpha \cdot |S(r,t)|$$

where $|S(r,t)|$ is the size of the relevant data set, and α is a retrieval factor that depends on memory architecture.

7.2.2. Memory Summarization

Memory Summarization compresses extensive historical data into concise formats, allowing the model to retain continuity without the burden of storing every detail.

Context-Aware Summarization:

The summarization algorithms identify and extract the most significant information from past interactions. By summarizing based on conversational context, MoAI Memory maintains continuity in dialogue while removing extraneous content, ensuring that the model can refer back to important points as needed.

Let H be the historical data containing multiple conversational points. We define S(H) as the summarized version of H, such that:

$$S(H) = \sum_{i=1}^{n} w_i \cdot h_i$$

where each hi represents a significant data point in H, and wi is a weight based on the contextual relevance of hi. The weights wi are determined dynamically, assigning higher values to points with greater significance in the conversational context. This ensures that S(H) maintains continuity while excluding irrelevant details.

Vector-Based Compression:

Utilizing vector embeddings, MoAI Memory compresses summarized information into lower-dimensional spaces, reducing memory demands and allowing for faster access. This compression method maintains the integrity of contextual understanding while minimizing the

memory footprint, thereby enhancing the model's efficiency.

To reduce memory load, MoAI Memory applies vector-based compression by mapping summarized information S(H) into a lower-dimensional space. Let V(H) represent the vector embedding of S(H) in a high-dimensional space:

$$V(H) \in \mathbb{R}^d$$

Using a compression function C, we map V(H) to a lower-dimensional space Vc(H) where dc<d:

$$V_c(H) = C(V(H))$$

The compression function C preserves essential contextual information while minimizing Vc(H)'s dimensionality. This vector-based reduction enables faster access and lower memory demands, supporting efficient data retrieval.

Iterative Refinement:

The refinement process continuously updates the summarized memory. Define St(H) as the summary at time t and ΔSt as an update based on new data Nt:

$$S_{t+1}(H) = S_t(H) + \alpha \cdot \Delta S_t$$

where α is a learning rate controlling the influence of new information Nt on the summary. As more relevant information is processed, S(H) adapts, enhancing accuracy and responsiveness. This iterative model ensures that the memory summary reflects the most recent and essential data, optimizing the model's relevance to current tasks.

The summarization process is iterative, allowing for updates to stored summaries over time. As new information becomes relevant, previous summaries are refined to reflect the latest context, ensuring that memory remains accurate and responsive to current needs.

7.2.3. Hierarchical Memory Structures

The MoAI Memory framework utilizes a hierarchical structure, categorizing information into short-term and long-term memories to optimize accessibility and relevance.

Short-Term Memory (STM):

STM serves as the model's immediate working memory, storing recent interactions for quick reference. This memory type is designed for high-speed retrieval and is periodically refreshed to focus on the most current and relevant information. STM enables the model to handle immediate context effectively, improving the coherence and relevance of responses.

Short-Term Memory (STM) holds recent interactions and focuses on high-speed, contextually relevant data for immediate responses. Formally, STM can be represented as a sliding window MSTM(t), which stores information for a limited timeframe or based on relevance thresholds.

STM Window: The contents of STM are defined as :

$$M_{STM}(t) = \{s_i \mid t - \Delta t \leq t_i \leq t\}$$

where si represents a recent interaction and ti is its timestamp. This window is dynamically updated, ensuring only the most recent and pertinent data points are retained.

Weight-Based Relevance: To prioritize recent yet relevant data, each interaction si in STM is weighted by a decay factor λ such that:

$$w_{STM}(s_i) = e^{-\lambda(t-t_i)}$$

where higher wSTM(si) values indicate greater relevance. This exponential decay applies to recent data, enabling STM to maintain high coherence by prioritizing information from more recent interactions, improving real-time performance.

Adaptive Refresh Rate: STM's contents are periodically refreshed, clearing low-relevance data points and retaining those exceeding a certain relevance threshold θSTM:

$$M_{STM}(t+1) = \{s_i \in M_{STM}(t) \mid w_{STM}(s_i) \geq \theta_{STM}\}$$

This adaptive mechanism ensures the STM remains focused on the immediate conversational context without being overwhelmed by older data.

Long-Term Memory (LTM):

LTM stores information that is relevant over extended periods, such as user preferences,

historical interactions, and foundational knowledge. This structure enables the model to retain important details without overwhelming short-term processes. Access to LTM allows the model to provide responses that are contextually informed by past interactions, enhancing the user experience in ongoing engagements.

LTM Representation: The information stored in LTM is indexed as MLTM, with entries lj and corresponding relevance scores wLTM(lj):

$$M_{LTM} = \{(l_j, w_{LTM}(l_j)) \mid w_{LTM}(l_j) \geq \theta_{LTM}\}$$

where θLTM is the threshold defining data worth retaining in long-term storage. High-relevance entries lj have stable, lasting weights that reflect their contextual importance to user-specific needs or recurring tasks.

Decay Control and Retention: Unlike STM, LTM entries have lower decay rates μ\muμ, which are influenced by interaction frequency and relevance across multiple sessions. For an entrylj accessed repeatedly, we have:

$$w_{LTM}(l_j) = \beta f_{access}(l_j) - \mu(t - t_{last})$$

where faccess(lj) tracks access frequency, tlast is the last access time, and β adjusts weighting according to access importance. This decay control mechanism prevents important historical data from fading while enabling gradual phasing out of less relevant information.

Cross-Memory Linking:

The hierarchical system includes a linking mechanism that bridges STM and LTM, ensuring fluid interaction between the two memory types. This feature allows the model to draw upon long-term context while considering the immediate needs of short-term tasks, resulting in responses that are informed by both recent and historical information.

Linking Function: The linking function L:

$$M_{STM} \times M_{LTM} \to \mathbb{R}$$

quantifies the relevance of specific LTM entries to current STM data. Given an STM entry sk and an LTM entry lj, the linking score L(sk,lj) is calculated as:

$$L(s_k, l_j) = \cos(\theta(s_k), \theta(l_j)) \cdot (w_{STM}(s_k) + w_{LTM}(l_j))$$

where cos(θ(sk),θ(lj) represents the cosine similarity between embeddings of sk and lj, indicating contextual alignment. A higher score signifies that lj should be brought into STM's active context, aiding in coherent responses across memory tiers.

Dynamic Contextual Retrieval:

If L(sk,lj)≥θCML (a cross-memory threshold), the LTM entry lj is temporarily loaded into STM for the current task. This dynamic retrieval mechanism ensures that historical knowledge supports immediate tasks without burdening the STM with excess information.

$$M_{STM}(t) \leftarrow M_{STM}(t) \cup \{l_j \mid L(s_k, l_j) \geq \theta_{CML}\}$$

7.2.4. Reinforcement-Based Memory Retention

MoAI Memory leverages reinforcement learning to determine the importance and retention duration of specific memories, optimizing what is stored based on usage patterns and interaction feedback.

Retention Scoring System:

A scoring mechanism evaluates memories based on their frequency of access, relevance, and user feedback. Memories with higher scores are retained longer, while those with lower scores are eventually removed or archived. This approach ensures that the model's memory is constantly refined, adapting to user needs over time.

Frequency of Access: The frequency of access faccess(mi) for a memory mi is computed over a time period T, where T is the total number of interactions. The access frequency is defined as:

$$f_{access}(m_i) = \sum_{t=1}^{T} \mathbb{I}(m_i, t)$$

where I(mi,t) is the indicator function that returns 1 if memory mi is accessed during time step t and 0 otherwise. This term helps capture the memory's engagement level over time.

Relevance: The relevance r(mi) of memory mi is evaluated based on the cosine similarity between the current context vector c(t) and the memory vector mi, which represents the content of memory mi. This can be expressed as:

$$r(m_i) = \cos(\mathbf{c}(t), \mathbf{m}_i)$$

where

$$\cos(\mathbf{c}(t), \mathbf{m}_i) = \frac{\mathbf{c}(t) \cdot \mathbf{m}_i}{\|\mathbf{c}(t)\| \|\mathbf{m}_i\|}$$

User Feedback: The user feedback ffeedback is incorporated as an additional component in the retention score. It is represented as a function of the user's expressed satisfaction or relevance feedback regarding memory mi. If the feedback is positive (such as a thumbs-up, approval, or further queries), the memory retention is boosted. Formally, the feedback score can be expressed as:

$$f_{feedback}(m_i) = \alpha \cdot \mathbb{I}(positive\_feedback) + \beta \cdot \mathbb{I}(negative\_feedback)$$

where α and β are scaling constants that adjust the influence of positive and negative feedback. Positive feedback increases the retention score, while negative feedback decreases it, guiding the system toward memory refinement based on user preferences.

Overall Retention Score: The overall retention score S(mi) for a memory mi is a weighted sum of the frequency of access, relevance, and user feedback components:

$$S(m_i) = \lambda_1 \cdot f_{access}(m_i) + \lambda_2 \cdot r(m_i) + \lambda_3 \cdot f_{feedback}(m_i)$$

,

where λ1, λ2 and λ3 are weight factors that control the influence of each component. This final score determines the retention duration for memory mim_imi. Memories with higher scores are retained longer, while those with lower scores are either archived or removed.

Memory Retention and Decay:

The retention score directly influences the memory retention period. For memories with a high retention score, the decay rate is slow, meaning they are retained for a longer period. Conversely, memories with a low retention score undergo faster decay and are eventually discarded.

Decay Function: The decay D(mi) of a memory mi over time is modeled using an exponential decay function, where memories with lower retention scores decay faster:

$$D(m_i) = e^{-\gamma \cdot (1 - S(m_i))}$$

where γ is the decay constant. The value of S(mi) determines the retention time-memories with higher scores experience slower decay and are retained longer, whereas memories with lower scores decay rapidly.

Threshold-Based Removal: To decide when to remove a memory mi, a threshold θdecay is set. If the decay value of memory mi falls below this threshold, it is removed from active memory storage:

$$\mathbb{I}(remove, m_i) = \mathbb{I}(D(m_i) < \theta_{decay})$$

Reinforcement Learning for Memory Retention Optimization

Reinforcement Learning (RL) is used to continuously optimize the memory retention process. The model receives rewards or penalties based on how well the retention system improves over time in terms of user satisfaction and interaction efficiency.

Reward Function: The reward function R(t) at each time step is based on the performance of the memory system in predicting or adapting to user needs. A higher reward is given when the system successfully retains and utilizes memories that improve the interaction. This can be defined as:

$$R(t) = \sum_{i=1}^{N} \mathbb{I}(successful\_memory\_usage_i) - \gamma \cdot \mathbb{I}(inefficient\_memory\_usage)$$

where N is the number of memory interactions in a given time step, and I(successful_memory_usagei) is an indicator function that returns 1 if memory mi improves the user experience, and 0 otherwise.

Policy Update: The RL agent updates its policy π based on accumulated rewards. The policy governs how memory retention scores are adjusted over time, ensuring that the system becomes better at prioritizing certain types of memories. The policy update can be formulated as:

$$\pi'(t) = \pi(t) + \alpha \cdot \Delta R(t)$$

7.2.5. Efficient Garbage Collection

Efficient garbage collection is a critical aspect of MoAI Memory's overall architecture, ensuring that the system remains performant over time by systematically managing the memory it utilizes. This process involves clearing out outdated or unused memories to free up resources for new, more relevant information while retaining the flexibility to access past data when necessary. The garbage collection strategy within MoAI Memory is designed to be both resource-efficient and contextually sensitive, providing an optimal balance between memory preservation and efficient resource use.

Automated Clean-Up Cycles

MoAI Memory implements automated clean-up cycles, which operate periodically to manage memory usage. These cycles scan the stored memories to identify which memories are no longer useful, marking them for removal or archival. The process helps to ensure that memory usage does not exceed the system's capacity, preventing issues like memory overload, which could negatively impact the system's performance.

Cycle Triggering: The clean-up cycle is triggered based on time intervals or a predefined memory usage threshold. The decision to initiate the garbage collection process can be made based on one or more of the following criteria:

Time-based Trigger: The system may decide to start a clean-up cycle after a set period, such as every 24 hours or after a certain number of interactions.

Memory Utilization Threshold: The cycle may be initiated when memory utilization exceeds a certain threshold, indicating the need to free up space for new data.

These triggers ensure that memory management is performed regularly, without waiting for the system to become overwhelmed with outdated or unnecessary memories.

Memory Relevance Scoring: During each clean-up cycle, the model evaluates the relevance of stored memories using the retention scoring system discussed earlier. Memories that receive low relevance scores (i.e., those that have not been accessed recently, are contextually irrelevant, or have received negative user feedback) are flagged for potential removal.
The evaluation formula for memory relevance within the clean-up cycle can be defined as:

$$R_{clean}(m_i) = \lambda_1 \cdot f_{access}(m_i) + \lambda_2 \cdot r(m_i) + \lambda_3 \cdot f_{feedback}(m_i)$$

where:

- $f_{access}(m_i)$ measures how frequently the memory is accessed,
- $r(m_i)$ measures the relevance of the memory in the current context,
- $f_{feedback}(m_i)$ captures user feedback indicating satisfaction or relevance.

Memory Removal Criteria: After evaluating the memories, the clean-up process determines which memories fall below a relevance threshold θrelevance, indicating that they are no longer useful. Memories that fall below this threshold are marked for removal or archival. The process of memory removal can be formalized as:

$$\mathbb{I}(remove, m_i) = \mathbb{I}(R_{clean}(m_i) < \theta_{relevance})$$

where $\mathbb{I}(remove, m_i)$ is an indicator function that returns 1 if memory m_i should be removed and 0 otherwise.

Archiving Process: The archived memories are moved to a secondary memory store, separate from the active working memory. This store may be organized by metadata such as the last accessed time, relevance history, or task-related category. These archived memories can be retrieved and re-integrated into the active memory store when they become relevant again. The retrieval process from the archive can be modeled as:

$$\text{Retrieve}(m_i) = \mathbb{I}(R_{clean}(m_i) > \theta_{retrieve})$$

where $\theta_{retrieve}$ is a threshold that determines when an archived memory is re-integrated into the active memory store.

Memory Retrieval and Relevance Re-assessment: When an archived memory is retrieved, it undergoes a re-assessment of its relevance in the current context. This allows the system to determine whether the memory is truly valuable again or if it should be archived once more. This re-assessment uses the same relevance scoring system discussed earlier, ensuring that archived memories are only reactivated when they are deemed valuable.

Acknowledgements

We thank Louis-nicolas Roussel for their contributions to the MoAI dataset,Emmanuel BOGDI for contributing to the data collection script and platform interface, Vincent Delmas and others researchers for valuable feedback and thoughtful discussions.

References

Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, A. Hassan Awadallah, R. W White, D. Burger, C. Wang. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation arXiv preprint arXiv:2308.08155.

S. Rasal, E.J. Hauer. 2024. Navigating Complexity: Orchestrated Problem Solving with Multi-Agent LLMs. arXiv preprint arXiv:2402.16713.

Q. Wang, Z. Wang, Y. Su, H. Tong, Y. Song. 2024a. Rethinking the Bounds of LLM Reasoning: Are Multi-Agent Discussions the Key? arXiv preprint arXiv:2402.18272.

S. Abdelnabi, A. Gomaa, S. Sivaprasad, L. Schönherr, M. Fritz. 2023. LLM-Deliberation: Evaluating LLMs with Interactive Multi-Agent Negotiation Games. arXiv preprint arXiv:2309.17234.

C.-M. Chan, W. Chen, Y. Su, J. Yu, W. Xue, S. Zhang, J. Fu, Z. Liu. 2023. Chateval: Towards Better LLM-Based Evaluators Through Multi-Agent Debate. arXiv preprint arXiv:2308.07201.

Y. Zhang, X. Yang, C. Bai, F. Wu, X. Li, X. Li, Z. Wang. 2024c. Towards Efficient LLM Grounding for Embodied Multi-Agent Collaboration. arXiv preprint arXiv:2405.14314.

Q. Wu, Y. Wu, G. Bansal, J. Zhang, Y. Zhang, S. Zhang, L. Jiang, X. Zhang, Z. Li. 2024. Autodefense: Multi-Agent LLM Defense Against Jailbreak Attacks. arXiv preprint arXiv:2403.04783.

Y. Zeng, Y. Wu, X. Zhang, H. Wang, Q. Wu. 2024. Autogen: A Multi-Agent LLM Framework for Efficient Problem Solving and Application Enhancement. arXiv preprint arXiv:2405.11106.

J. Echterhoff, Y. Liu, A. Alessa, J. McAuley, Z. He. 2024. Cognitive Bias in High-Stakes Decision-Making with LLMs. arXiv preprint arXiv:2403.00811.

Z. Yang, W. He, S. Lin, P. Gupta, D. P. Kingma, and K. Cho. 2020. Mixture of Experts: A Brief Survey and Directions for Future Research. arXiv preprint arXiv:2010.11220.

M. F. Carbin, A. S. Mao, and A. F. Hall. 2021. MOE: A Mixture of Experts for Transformer Models. arXiv preprint arXiv:2105.12571.

P. Shankar, M. Greff, and J. Schmidhuber. 2022. Adaptive Mixture of Experts with Reinforcement Learning for Sequence Modeling. In Proceedings of the 39th International Conference on Machine Learning, volume 130, pages 11837–11848.

T. J. S. Cui, M. R. Yao, and F. T. Ma. 2024. Scaling Transformers with Mixture of Experts: A New Paradigm in Efficient Large-Scale Language Models. arXiv preprint arXiv:2401.01576.

J. Parisotto and R. Salakhutdinov. 2017. Neural Map: Structured Memory for Deep Reinforcement Learning. In Proceedings of the 31st International Conference on Machine Learning (ICML), volume 70, pages 4095–4103.

D. Chen, H. Zhang, and W. Li. 2024. Memory-Enhanced Pretrained Models for Multitask Learning and Knowledge Preservation. arXiv preprint arXiv:2403.12983.

A. N. Lee, D. K. Lim, T. J. Wang. 2024. Mixture of Expert Systems and Multi-Agent Coordination: A Hybrid Approach for Advanced LLM Applications. arXiv preprint arXiv:2404.10586.

A. N. Lee, D. K. Lim, T. J. Wang. 2024. Mixture of Expert Systems and Multi-Agent Coordination: A Hybrid Approach for Advanced LLM Applications. arXiv preprint arXiv:2404.10586.

Y. Zhang, H. Zeng, Z. Fan, Y. Chen. 2024. Multi-Task Routing in Mixture-of-Expert LLMs for Real-Time Performance. arXiv preprint arXiv:2402.09511.

Ong, I., Almahairi, A., Wu, V., Chiang, W.-L., Wu, T., Gonzalez, J. E., Kadous, M. W., & Stoica, I. (2024). RouteLLM: Learning to Route LLMs with Preference Data. arXiv preprint arXiv:2406.18665.

Li, J., Consul, S., Zhou, E., Wong, J., Farooqui, N., Ye, Y., Manohar, N., Wei, Z., Wu, T., Echols, B., Zhou, S., & Diamos, G. (2024). Banishing LLM Hallucinations Requires Rethinking Generalization. arXiv preprint arXiv:2406.17642.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, Q. Zhao, P. Chi, N. Le, E. Chi. 2022. Chain of Thought Prompting Elicits Reasoning in Large Language Models. arXiv preprint arXiv:2201.11903.

N. Chiang, P. He, Y. Li, S. Yu. 2023. Parallel Chain of Thought Processing: Accelerating Sequential Reasoning in Large Models. arXiv preprint arXiv:2305.11292.