# Doing More with Less – Implementing Routing Strategies in Large Language Model-Based Systems: An Extended Survey

Clovis Varangot-Reille[1, 2, *], Christophe Bouvard[1], Antoine Gourru[2], Mathieu Ciancone[1], Marion Schaeffer[1], and François Jacquenet[2]

[1]Wikit, Lyon, France

[2]Laboratoire Hubert Curien, UMR CNRS 5516, Saint-Etienne, France

[*]*Corresponding author: Clovis Varangot-Reille,* `clovis.varangot{wikit.ai}`

## Abstract

Large Language Models (LLM)-based systems, i.e. interconnected elements that include an LLM as a central component (e.g., conversational agents), are typically monolithic static architectures that rely on a single LLM for all user queries. However, they often require different preprocessing strategies, levels of reasoning, or knowledge. Generalist LLMs (e.g. GPT-4) trained on very large multi-topic corpora can perform well in a variety of tasks. They require significant financial, energy, and hardware resources that may not be justified for basic tasks. This implies potentially investing in unnecessary costs for a given query. To overcome this problem, a routing mechanism routes user queries to the most suitable components, such as smaller LLMs or experts in specific topics. This approach may improve response quality while minimising costs. Routing can be expanded to other components of the conversational agent architecture, such as the selection of optimal embedding strategies. This paper explores key considerations for integrating routing into LLM-based systems, focusing on resource management, cost definition, and strategy selection. Our main contributions include a formalisation of the problem, a novel taxonomy of existing approaches emphasising relevance and resource efficiency, and a comparative analysis of these strategies in relation to industry practices. Finally, we identify critical challenges and directions for future research.

**Keywords**: *Routing, Large Language Model, Optimisation, Cost, Survey*

## 1. Background

In the domain of computer network, a **router** can be defined as "*a device that sends data to the appropriate parts of a computer network*". In LLM-based systems, a router is a
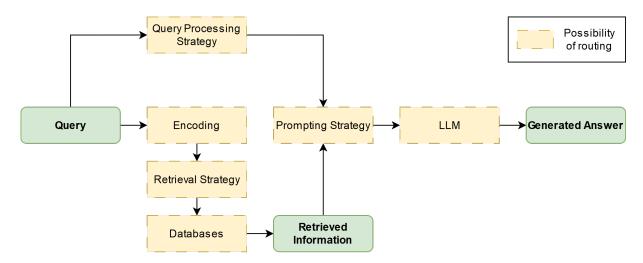
Figure 1: **DynamicRAG: the RAG architecture as an user query-dependent dynamic system** - We propose an architecture that views each step of the RAG framework as a routing opportunity. This approach redefines the architecture of the RAG framework and transforms it into a user-centric system.

system component that routes an item (i.e. user query) to the most appropriate element from a pool of elements candidates to carry out a task.

This paper focuses on one of the most common applications of LLM-based systems: *conversational agents* (Dam et al., 2024). Conversational agents respond to user queries by simulating human conversation (Caldarini et al., 2022; C.-C. Lin et al., 2023). They process the user's query and provide an answer that is relevant, given some metrics, to it (Caldarini et al., 2022). The Retrieval-Augmented Generation (RAG) architecture further improves the relevance of the response by adding an information retrieval step to the process (Gao et al., 2023; P. Lewis et al., 2020). This step involves retrieving the information most relevant to the user query from a knowledge database and including it into the prompt alongside the user query (see Figure 1 for a RAG architecture schema). We can adapt all RAG steps to the user's query by routing it to the most appropriate element at each step (Figure 1). A router allows the whole system to dynamically adapt to different input types (C. Wang et al., 2024).

Given a set of $n$ models $\mathcal{M} = \{M_1, ..., M_n\}$, for a given query $q$, the router function $\mathcal{R}$ aims to maximise the scoring function $s$ (e.g. *accuracy*) while adhering to a budget constraint $B$:

$$\mathcal{R}_{\mathcal{M}}(q) = \underset{M \in \mathcal{M}}{\arg\max}\, s(q, M) \tag{1}$$
$$\text{s.t. } C_M(q) \leq B$$

where $C_M$ is the cost (i.e, $/token$) to call the model $M$ for a query $q$ and $B$ is the user's budget. The budget could be the amount of resources available.

A straightforward optimisation of conversational agent architectures is the selection of the pre-trained LLM which has the ability to answer the user's query. Various LLMs exist

that vary greatly in terms of number of parameters, such as *Llama-3.2-3B* and *Llama-3.1-405B*, as well as models specialised on distinct domains, like *mathstral-7B-v0.1*[1] for mathematical tasks or *Med-Palm*[2] for medical-related tasks. However, most LLM-based systems rely on a single generalist LLM to respond to all user queries. By routing each query to the smallest possible LLM that has the ability to answer correctly to a user query, we can expect to improve the quality of the response by leveraging specialised expertise and optimise costs, as not all tasks require a large LLM. The router allows maximising performance while avoiding the cost of using models that are oversized, have insufficient reasoning capacity, or lack the knowledge needed to provide desired output.

This survey evaluates routing strategies that select the most appropriate element to solve a subtask within an LLM-based system, focusing in particular on the generation step by selecting the appropriate LLM for generation. We exclude approaches that identify the answer closest to the truth among all generated candidate responses (Guha et al., 2024; Si et al., 2023) as well as ensemble methods that combine multiple answers from various candidates to create a meta-answer (J. Hu et al., 2024; D. Jiang et al., 2023; H. Wang et al., 2024). Although these approaches demonstrate effectiveness, they focus solely on maximising performance without addressing cost constraints. Consequently, they do not align with our definition of routing.

This survey is structured into several sections that focus on the critical aspects of routing. We begin by describing the essential elements of routing. Next, we examine the pipeline stage at which routing is implemented in the literature. Finally, we detail and classify routing strategies according to the frameworks used and the resources required. We discuss these strategies considering industrial practices and highlight the key challenges that the field must address.

## 2. Which elements should be optimised for routing?

The primary objectives of routing are to minimise unnecessary resource consumption while maximising performance by using a model or element appropriate to the task. In other words, it seeks to optimise the performance - cost trade-off.

### 2.1 A cost to minimise

Most existing LLM-based systems depend on API calls to closed models, such as those provided by OpenAI. The primary cost to minimise for a router is the price per token. To calculate the total cost of running the pipeline, it is essential to consider all invoked

---

[1] MistralAI's Mathstral
[2] Google's Med-Palm

pre-trained models, including LLMs and embedding models in RAG settings.

Other production-related costs, such as average latency and computational costs, can also be considered Irugalbandara et al., 2024. Latency is measured with respect to the delay between the user's request and the pipeline response. Effective routing could reduce the average latency by routing simple user queries, such as greetings, to LLMs that require fewer computing resources.

As LLM increases in use and size, the power and computing requirements increase significantly Luccioni et al., 2024, increasing the ecological impacts associated with LLM-based systems. This impact is often measured considering the energy consumption (kWh) or the global warming potential (kgCO2eq) of the application. Various tools[3] have been proposed to estimate this environmental footprint.

## 2.2 A performance metric to maximise

The router function is also to maximise a scoring function that evaluates the model ability to produce accurate answers, as explained in equation 1.

There are several scoring function possibilities to maximise. In a traditional supervised learning framework, the evaluation process involves comparing these generated answers with the ground truth. In cases where the data lack ground truth or annotation, including a human evaluator in the evaluation loop allows us to assess whether the response is factually correct, in the expected format and consistent with the expected ground truth Chang et al., 2024. However, evaluating thousands of queries can require significant time and intense human involvement, affecting scalability. In addition, subjective bias might appear while evaluating, such as lack of expertise or preferences. More straightforward strategies include exact matching, partial matching, ROUGE score C.-Y. Lin, 2004, or even semantic similarity to the ground truth Chang et al., 2024; L. Zhang et al., 2024. These metrics may not adequately assess the factual accuracy of the generated content. As a result, previous studies have investigated automatic evaluation methods involving an LLM, where instructions and rating criteria are provided within the prompt along with the generated response Chiang & Lee, 2023. Their alignment with human evaluation remains uncertain, and factors such as instructions and sentence structure play an important role in the generated rating H. Wei et al., 2024. Many frameworks have been proposed, such as Retrieval Augmented Generation Assessment, also known as RAGAS Es et al., 2024. Preference learning, based on the preferences of human reviewers, can be used to assess the quality of responses in addition to traditional methods R. Jiang et al., 2024. Data related to preferences are typically represented as $\lambda_i \prec_x \lambda_j$, indicating that for a given query $x$, the user prefers the generated answer $\lambda_i$ over the alternative $\lambda_j$ Fürnkranz & Hüllermeier, 2012.

---

[3]Ecologits, CodeCarbon, MLCO2, Boavizta

# 3.  When should routing take place?

We propose two stages in the pipeline for the routing process: *pre-generation routing* (see Figure 2) and *post-generation routing* (also known as *cascade routing*) (see Figure 3). Pre-generation routing takes place before generating a response to the user query, while post-generation routing takes place after generating the response.

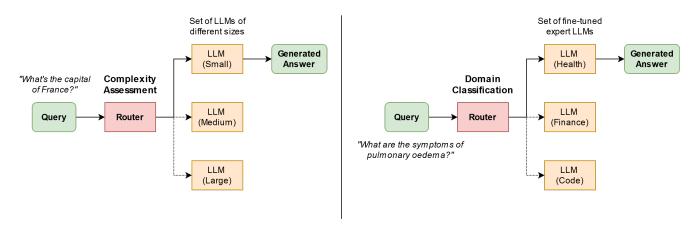## 3.1  Routing as a Pre-generation Step



Figure 2: **Routing as pre-generation step** – Before generating an answer, each LLM ability to provide an appropriate answer is assessed based on the complexity and/or topic of the user's query. *Dotted arrows represent non-selected LLM candidates.*

To implement **pre-generation routing**, we need to infer the ability of the LLM to answer a query *a priori* (see Figure 2). This method minimises latency by not waiting for the LLM response. There are two main approaches to achieve this: 1) infer the domain of knowledge of the query and route it to the associated LLMs trained as domain experts; and 2) assess the LLM candidate ability to answer a query of a given complexity and then route the query to the LLM with sufficient reasoning ability.

In this survey, we define the complexity of a query as the predicted performance score of the LLM for that specific query. The lower the score, the more complex the query. Within the context of RAG-based conversational agents, complexity can be categorised as follows: (a) *low complexity user query* may consist of a simple greeting that does not require retrieval; (b) *intermediate complexity user query* might involve extracting explicit information from a single document; (c) *high complexity user query* may necessitate the extraction of implicit information from multiple documents through reasoning.

## 3.2  Cascading: Routing as a Post-Generation Step

One might conceptualise routing as a **post-generation step**, where each model response is assessed iteratively (or in a cascade manner) by selecting progressively more advanced models until the response is considered pertinent (see Figure 3). In other words, the
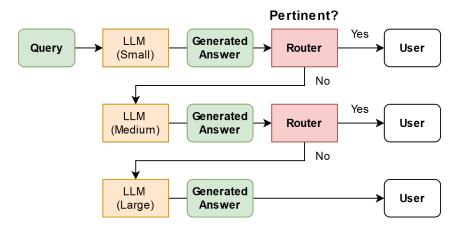
Figure 3: **Routing as post-generation step (or cascade routing)** – The relevance of the use of a larger LLM is determined by the evaluation of the answers generated by the current LLM. Each candidate response is evaluated sequentially. If an answer is deemed inadequate or untrustworthy, the user query is routed to a larger LLM. Typically, the cascade sequence is static.

challenge is no longer to infer the ability of a potential LLM to meet a demand, but to assess the quality of the generated current response. By definition, this approach is less optimal than generating a single response with the pre-generation approach because, in some cases, several answers will be generated for the same query. This entails financial, computational, and latency costs.

The process can be enhanced by hybridising it with the pre-generation step (Dekoninck et al., 2024). We consider it as a **multi-choice cascading** method within the post-generation category.The evaluation of the response determines whether it should be routed to another LLM, similar to a post-generation approach. However, instead of routing the query to the next LLM in the predefined cascade sequence (as shown in Figure 3), the query can be routed to any available model within the set of LLMs at any stage of the cascade. This hybrid process outperforms simple sequential cascading or supervised pre-generation approaches (Dekoninck et al., 2024), but still requires multiple generations per query.

## 4. How routing should be implemented?

Figure 4 represents the different techniques discussed in this survey. The strategies are divided into high- and low-resource strategies. We define a **high-resource strategy** as one that (1) *possibly generates multiple full-sequence responses to the same user query* and/or (2) *uses an LLM with an arbitrary threshold value chosen as a 1B parameter only for the routing process.*
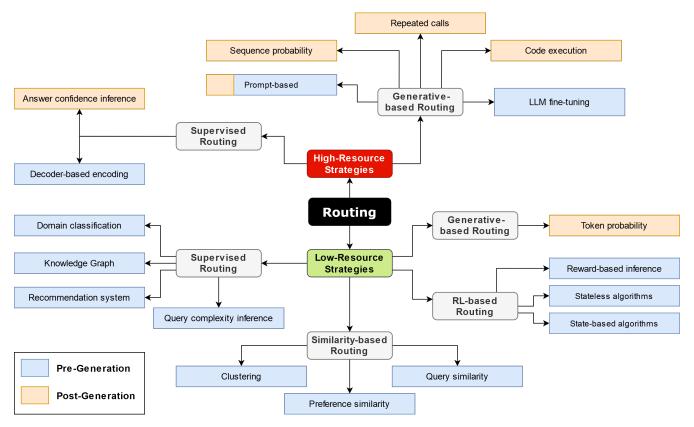
6

Figure 4: **Overview of routing strategies** – Routing strategies are classified according to the resources they require and whether routing occurs before or after the generation step. *RL: Reinforcement Learning.*

## 4.1 High-Resource Strategies

### 4.1.1 Supervised Routing

**Answer confidence inference**

In a cascading approach, at each iteration, a model, such as a fine-tuned *DistilBERT* (Sanh et al., 2020), classifies whether the generated answer aligns with the reference answer (L. Chen et al., 2023). L. Chen et al. (2023) proposed FrugalGPT[4]. This method infers the probability that the generated answer aligns with the reference answer using a *DistilBERT* regression model (Sanh et al., 2020), and compares this probability to an optimised threshold. The process of learning the threshold, denoted as $\tau_i$ for each model, is framed as a constrained optimisation problem. If the score exceeds the threshold, the response generated by $M_i$ is retained; otherwise, a larger model, $M_{i+1}$, is called. They used 12 LLM APIs from 5 providers (OpenAI, AI21, Cohere, ForeFrontAI, Textsynth) with 10M input tokens cost ranging from \$0.2 (*Textsynth's GPT-J*) to \$30 (*GPT-4*). The authors reported that their framework could save between 59% and 98% of costs while maintaining similar accuracy to larger models, such as *GPT-4*. This strategy has often been used for comparisons, but it often underperforms against alternatives, such as LLM-

---

[4]stanford-futuredata/Frugalgpt

based repeated calls routing (Aggarwal et al., 2024) and graph-based supervised routing (T. Feng et al., 2024). Moreover, even lower-resource LLM-based strategies, such as the token probability method (Ramírez et al., 2024), match its efficacy.

**Decoder-only encoding**

Mohammadshahi et al. (2024) proposed the *Routoo Orchestrator*[5], formerly known as the *Leero Orchestrator*. Their strategy differs from the **query complexity inference** approach in that they use a decoder-only LLM to encode the query for supervised training instead of relying on bidirectional language models (i.e., BERT models (Devlin et al., 2019)). They extract the representation of the predefined last token, typically '$<\backslash$s$>$' or '$<$EOS$>$', as these models are autoregressive and process sentences from left to right. This approach has demonstrated promising performance, with top-ranked models on the MTEB leaderboard being decoder-only embedding models such as Nvidia's *NV-Embed-v2* (C. Lee et al., 2024) and BAAI's *bge-en-icl* (C. Li, Qin et al., 2024). However, due to their larger size, these models require significant computing resources. Decoder-only models are typically fine-tuned for dense retrieval (C. Lee et al., 2024; C. Li, Liu et al., 2024; C. Li, Qin et al., 2024; Ma et al., 2024; Muennighoff et al., 2024; L. Wang et al., 2024). It remains unclear whether the *Routoo Orchestrator* employs a non-fine-tuned *Mistral-7B* model (A. Q. Jiang et al., 2023) or has fine-tuned it using the LoRA method (E. J. Hu et al., 2022). Once the input sequence is embedded by the decoder-only model, it is processed through a linear layer alongside an embedding representation of the LLM. The objective is to determine the expected quality score for an LLM $m$ when generating an answer to a specific query $q$. The performance inference model is trained by minimising the cross-entropy loss. During inference, we select the model that maximises the inferred evaluation score, while ensuring that the inference cost $c_m$ remains within a predefined budget. The researchers optimise the selection of models by identifying the subset of LLMs that achieve the highest score for a given set of queries, in order to ensure complementary LLMs for routing. In initial experiments using only open-source models, 56 LLM with parameters of 7 billion, 13 billion, and 34 billion, demonstrated superior accuracy on the MMLU dataset compared to a larger stand-alone LLMs such as *Llama2-70B* (Touvron et al., 2023) (75.9% vs 69.9%) and *Mixtral-8x7B* (A. Q. Jiang et al., 2024)(75.9% vs 70.6%). Notably, these models achieved this accuracy at similar or lower costs: *Routoo* costs \$0.6 per million tokens, while *Llama2-70B* costs \$0.9 per million tokens, and *Mixtral-8x7B* is priced at \$0.6 per million tokens. When *GPT-4-turbo* was included as a routing option, its performance was closely matched (84.9% vs 86.4%) at half the cost (\$10.2 vs \$20 per million tokens), as it was used for approximately 50% of the queries.

---

[5]Leeroo-AI/leeroo_orchestrator

### 4.1.2 Generative-based routing

This class of strategy uses a generative approach specifically for routing. Generative-based routing can potentially take advantage of LLM emergent capability for unsupervised multitasking to generalise to new contexts (Radford et al., 2019; J. Wei et al., 2022).

**Prompt-based routing**

It can be used as either a pre-generation approach or as a post-generation approach.

In a pre-generation approach, the simplest method involves using a pre-trained LLM with prompt-based routing, also known as "function calling". This process entails passing descriptions of the routing options, with or without examples, in the prompt along with the user query. We route to the option returned by the LLM (Ning et al., 2024; Shen et al., 2023). Although it leverages an LLM, it is more energy-efficient and requires fewer resources than fine-tuning an LLM.Ning et al. (2024)[6] prompt *GPT-4* to determine whether a query possesses the necessary characteristics to be addressed by their prompt technique (SoT). Despite using a large LLM, the absence of examples affects inference. Furthermore, this strategy did not surpass a small language model, *RoBERTa* (Zhuang et al., 2021), specifically trained for task classification. Similarly, Shen et al. (2023) proposed *HuggingGPT*[7], a framework for task planning and execution (Shen et al., 2023). An LLM is prompted to select between different models based on their descriptions.
This approach, despite its limitations, offers the advantage of performing inference with minimal examples in the prompt, known as "few-shot inference", or without any examples at all, known as "zero-shot inference". This is particularly useful when limited resources are available for annotation.

In a post-generation approach, the LLM can be prompted to express its uncertainty when responding to user queries (Z. Li et al., 2024). This is known as "verbalised confidence" (Xiong et al., 2024). Z. Li et al. (2024) proposed *Self-Route* to decide whether retrieving traditional chunks using a RAG strategy is sufficient. Otherwise, they suggest using the entire document from which the chunks were extracted. This is an example of how routing can improve another step of a LLM-based conversational agent. The model was instructed to indicate whether the query is answerable based on the retrieved chunks. They used the prompt: "*Write 'unanswerable'*" if the query cannot be answered based on the text. They ran a new generation with a larger context if the question was considered unanswerable. Their findings reveal that implementing this strategy outperformed a naive RAG system in terms of accuracy for the most commonly used LLMs (i.e. *GPT-4o* and *GPT-3.5-turbo*). For *GPT-3.5*, which has a smaller context window (16k), it also outperformed sending large contexts. In contrast, performance between *Self-Route* and sending a large context proved comparable for *GPT-4o* which has a larger context window (128k) while using an average of 61% fewer tokens. Finally, for *gemini-1.5-pro*, which

---

[6]imagination-research/sot
[7]microsoft/JARVIS

features a context window of 1M, sending only large contexts appears to perform better than *Self-Route*, although the latter's performance remains reasonably close. The results for *gemini-1.5-pro* stem from its large context window, which allows the transmission of extensive contexts without truncation. Given these results, it may be tempting to send only large contexts. The authors show how it is possible to save costs while achieving comparable performance with fewer tokens.

However, it has been reported that LLMs are often too confident in expressing their certainty(Xiong et al., 2024), suggesting caution about the reported effectiveness of this method. Furthermore, verbal confidence has been shown to be, in the best case, comparable to random routing to a larger LLM and could even perform worse (Chuang et al., 2024).

### Sequence probability

C.-H. Lee et al. (2024) proposed using the normalised sequence-level probability of a smaller LLM when determining whether to route a query to a larger LLM. The routing is based on the sequence uncertainty. However, this approach tends to rely too heavily on the larger LLM, resulting in an efficacy comparable to that of the larger LLM alone. Consequently, this approach is less effective in reducing call frequency to the larger LLM than the supervised methods discussed earlier.

### LLM fine-tuning

If numerous resources are available, an LLM can be fine-tuned for routing tasks like classification or regression task (Liu et al., 2024; Ong et al., 2024), or for code generation to use providers API (Patil et al., 2024). Routing can be achieved by fine-tuning an LLM to add new routing-related tokens to its vocabulary: uncertainty-related tokens (Chuang et al., 2024) or token identifiers for domain experts (Chai et al., 2024). The LLM can also be fine-tuned for domain classification (Liu et al., 2024), but adding new domains will require a retraining of the model.

Liu et al. (2024) suggested using *Qwen1.5-1.8B-Chat* (Bai et al., 2023), which they fine-tuned for a domain-classification task[8]. This meta-model categorises the prompt, and the corresponding pre-trained expert associated with that category then generates a response. Similarly, Ong et al. (2024) fine-tuned a *Llama-3-8B model* (Grattafiori et al., 2024) on a scoring task designed to evaluate both the complexity of a query and the model's ability to answer it through LLM evaluation (Ong et al., 2024). This score is subsequently converted into the probability that the user has a preference for the larger model answer. While Ong et al. (2024) found that this strategy is not significantly more efficient regarding cost or quality compared to other non-LLM techniques, such as matrix factorisation, Liu et al. (2024) demonstrated a marked improvement in performance across all datasets through supervised fine-tuning. The accuracy increased from 15% to nearly 100% for the

---

[8]godcherry/ExpertTokenRouting

MMLU dataset (Liu et al., 2024). The authors' differing levels of optimism regarding the fine-tuning of an LLM for classification or regression tasks may stem from Ong et al. (2024) comparing this technique with other optimised approaches, while Liu et al. (2024) evaluated it against the same standalone LLM without fine-tuning.

Patil et al. (2024) approached the problem as a code generation task.[9] They fine-tuned a *Llama-7B model* to direct queries to the appropriate model via API calls. To achieve this, they undertook the following steps: first, they created a dataset of API calls to model libraries, including Torch Hub, TensorFlow Hub v2, and HuggingFace. Second, they generated instructions to call a specific API based on in-context examples and the API documentation, utilising *GPT-4* in accordance with the Self-Instruct paradigm (Y. Wang et al., 2023). To enhance the LLM's ability to utilise the retrieved information in a RAG context, they incorporated the API documentation for a specific model into the training instructions. Finally, they trained a *Llama-7B-based* model to generate the API call code. The model can perform zero-shot inference without documentation retrieval or RAG-based inference with the top-1 documentation retrieved. In a zero-shot context, without fine-tuning, *Llama-7B* fails to accomplish the task, scoring O%. It outperforms larger models after fine-tuning, achieving an average improvement of 35 points over *GPT-3.5-turbo-0301* and 46 points over *GPT-4-0314*.

**Repeated calls**

In a post-generation approach, we could route to the next model whenever the uncertainty of the LLM for its answer is high. Smaller LLMs tend to give consistent answers to simple questions but show inconsistencies when confronted with more complex questions (Yue et al., 2024). Thus, the uncertainty of the model can be assessed by repeatedly querying the LLM with the same prompt, using temperatures ranging from 0.4 (Yue et al., 2024) to 1 (Aggarwal et al., 2024). By analysing the consistency of the responses, researchers can assess the level of uncertainty (Aggarwal et al., 2024; Yue et al., 2024). Aggarwal et al. (2024) introduced a three-step approach called *Automix*[10] (Aggarwal et al., 2024). The models used by the authors were *GPT-3.5*, *Llama-2-13B* (Touvron et al., 2023), and *Mistral-7B-Instruct-v0.2* (A. Q. Jiang et al., 2023), as the smaller language models, and *GPT-4* as the larger language model. Given a set of $N$ unique models with increasing size $\mathcal{M} = \{M_1, ..., M_N\}$: (1) using the smaller model $M_j$ (j=1) to generate an answer $A_{i,j}$ to a query $q_i$ based on a related context $C_i$; (2) initiating a self-verification process with the same model $M_i$ with a few-shot meta-prompt (*verification prompt*) to determine whether $A_{i,j}$ aligns with the provided context $C_i$. To estimate a confidence score of $M_j$ aligning with $C_i$, they produced $A_{i,j}$ $k$ times ($A_{i,j}^k$, k > 1) at a high temperature and calculated the proportion of $A_{i,j}^k$ aligning with $C_i$; (3) based on this confidence score, the router either retained the current answer $A_i$ or call a larger model $M_j$ (j>i). This procedure repeats until the confidence score reaches a satisfactory level or the entire set of LLM $\mathcal{M}$

---

[9]ShishirPatil/gorilla
[10]automix-llm/automix

series has been tested. The authors employed two routing strategies: a more complex one based on a Partially Observable Markov Decision Process (POMDP) and a simpler one grounded in a confidence cost/quality trade-off threshold.

On the other hand, Yue et al. (2024) did not assess the correctness of the answers finding this concept too challenging to evaluate. Instead, they scored the responses based on the ratio of identical answers among $k$ samples for the same query, which reflects answer consistency[11]. They assessed response consistency across different representations of thought by comparing responses generated by Chain-of-Thought (CoT) and Programme-of-Thought (PoT) prompts, with further details provided in Appendix A. The *Mixture-of-Thoughts* (MoT) representation employs both prompting strategies by mixing samples from CoT and PoT. For each query, they generated $k$ samples for each prompting approach (PoT, CoT, or MoT). The algorithm then calculates a consistency score, which is defined as the proportion of identical answers among $k$ samples).

The authors reported that *Automix* (Aggarwal et al., 2024) achieves a higher F1 score on the QASPER and COQA datasets and superior accuracy across a range of costs compared to *HybridLLM* (Ding et al., 2024), *FrugalGPT*, (L. Chen et al., 2023) or standalone models such as *GPT-4* and *Llama-2-13B* (Touvron et al., 2023), particularly the routing based on the POMDP approach. Even in low-resource scenarios with a small training dataset size, their method significantly outperforms both *HybridLLM* (Ding et al., 2024) and *FrugalGPT* (L. Chen et al., 2023). These findings are supported by Yue et al. (2024), who demonstrated that including various thought-based reasoning (MoT) when deciding whether to call a larger LLM resulted in similar accuracy at lower cost compared to assessing consistency between CoT or PoT samples alone (Yue et al., 2024). This underlines the need for diversity and complementarity between routing options. In both studies, users have to call an LLM multiple times for each query, and despite their efficiency, these approaches prove to be resource-intensive.

**Code execution**

This strategy applies exclusively to code generation tasks. It directly evaluates the confidence in the answer by executing the generated code. *EcoAssistant*[12], an iterative multi-agent code generator designed to query external knowledge for question and answering, has been proposed by J. Zhang et al. (2023). In this generator, an LLM interacts with a code executor to generate and execute code. If the code generated by the smaller LLMs does not succeed, the request will be forwarded to a larger LLM. Here, success, rather than uncertainty, serves to guide routing. To determine the correctness of the generated code for a given query, the authors implemented an evaluation using *GPT-4* alongside the success of code execution. They used two architectures: *GPT-3.5-turbo* + *GPT-4* and *Llama-2-13B*-chat (Touvron et al., 2023) + *GPT-3.5-turbo* + *GPT-4*. The authors

---

[11]MurongYue/LLM_MoT_cascade
[12]JieyuZ2/EcoAssistant

reported that their framework generated more successful code snippets at a lower cost than using *GPT-4*. Although more expensive than using *GPT-3.5-turbo*, *EcoAssistant* significantly exceeded the percentage of successful code generation..

In all the high-resources routing approaches discussed, the savings from the routing process may be outweighed by the implementation requirements of high-resource routing strategies.

## 4.2 Low-Resource Strategies

The drawbacks of the routing strategies presented in the previous section can be addressed using low-resource strategies. They attempt to avoid such resource requirements while maintaining routing performance. Most of them are pre-generation, except the last one.

### 4.2.1 Similarity-based Routing

**Query similarity**
The primary concept behind these approaches is to route a user's query to the LLM that has the best performance on similar queries answered in previous interactions (e.g. cosine similarity).
In its simplest form, the approach routes the user's query to the elements that have successfully responded to the $n$ most similar previous queries, with the assumption that similar queries require similar processing. Stripelis et al. (2024) propose a 1NN router that routes the user query to the LLM identified as producing the most appropriate response for the most similar query in the training data, based on cosine similarity. However, this strategy fails to capture complex relationships between user queries and expert answers, and performs worse than randomly selecting from available models. Manias et al. (2024) suggests *Semantic-Router*, a framework developed by Aurelio AI[13], which uses $n$ examples. *Semantic-Router* detects intentions by routing the user's requests to an intention that is associated with $n$ similar previous requests. The authors demonstrated that *Semantic-Router* performed similarly to prompt-based intent detection, achieving around 90% accuracy without requiring LLM inference.
Similarly, Jang et al. (2023) proposes a method that routes to different expert adapters based on the training tasks that are most similar to the user query. Each training task is embedded and associated with one or more expert IDs. During the inference process, they identify the most similar training tasks and the most frequently associated expert IDs from the training tasks most similar to the user's query. They showed improved performance on non-generative tasks by using a smaller language model (t5-3B (Raffel et al., 2020)) compared to a fine-tuned multi-task t0-3B model (Sanh et al., 2022). However,

---

[13]aurelio-labs/semantic-router

their framework showed inferior performance on generative tasks.

Malekpour et al. (2024) applied this approach within a text-to-SQL framework by examining previous successes among similar SQL queries. They proposed selecting the cheapest LLM that yields a higher proportion of SQL generation matching the ground truth among the first $n$ most similar SQL queries. To ensure minimal performance, the authors implemented a threshold that represents the minimum proportion of queries a LLM must success to be considered. They utilised three LLMs as routing options: *gpt-4o*, *gpt-4o-mini*, and a quantised version of *Llama3.1-8B-instruct* (Grattafiori et al., 2024).The authors demonstrated that they could achieve performance close to that of the best stand-alone model for this task (*gpt-4o*)—60.1% compared to 61.0%—at a lower cost (1.1 times cheaper). However, it is important to emphasise that most of the time, the strategy routed queries to the best LLM (*gpt-4o*) -81%. A routing strategy must not only minimise costs while maximising quality, but also avoid achieving higher quality by exclusively routing to the larger LLM. In addition, the choice of LLM did not include a specialist code generation LLM, which could potentially have led to a cheaper option with similar performance. Despite these limitations, employing similarity retrieval based on previous successes or utterances effectively reduces costs while maintaining performance. These approaches require minimal resources.

The query representation can be improved by optimising the encoding process using contrastive learning. The optimisation ensures that similar samples are close together in the multidimensional semantic space, while dissimilar samples are far apart (Le-Khac et al., 2020).

C.-H. Lee et al. (2024) applied it to Dialogue State Tracking (DST) by fine-tuning a *SenBERT-based* bi-encoder (Reimers & Gurevych, 2019) within a framework called *OrchestraLLM*. DST is a task whose objective is to extract the user's intention and dialogue-related information in a structured representation (see Annex A for an example). Some dialogues may be too complex to be handled with a smaller LLM, and the conversation processing may therefore require a larger LLM. The authors assigned a series of utterances to either the smallest or largest LLM, depending on which model could accurately generate the correct dialogue state representation. In each iteration, they selected the appropriate LLM using a *K-NN* algorithm applied to the embedding of the new instance. The majority vote from the k closest neighbours determined which LLM had been most successful in similar past examples. *OrchestraLLM* shows a slight improvement in accuracy compared to a direct call to the larger LLM, while also reducing the number of calls to this model by 50% to 80%. Optimising the representation strategy with the contrastive loss improves the assignment ratio to a smaller model by 8 points. Alternatively, S. Chen et al. (2024) proposes a framework called *RouterDC*, which improves user query representation through multi-level contrastive learning. They map user queries into a shared space, learning embeddings of both the LLM representation and the query. This multi-

level loss consists of two parts. The first is the contrastive loss between the query and the LLM to evaluate positive/negative LLM sets based on their performance compared to the query. The second is the contrastive loss between similar queries to ensure they are located closer together in this space. Subsequently, they generate a selection probability distribution over the set of LLMs by applying a softmax function to the similarity between the user query and the learnable LLM representation embeddings. During inference, it selects the LLM that maximises this similarity probability. The options include a mix of generalist and expert LLMs: *Mistral-7B* (A. Q. Jiang et al., 2023), *MetaMath-Mistral-7B* (Yu et al., 2024), *zephyr-7b-beta* (Tunstall et al., 2024), *Chinese-Mistral-7B, dolphin-2.6-mistral-7b, Llama-3-8B* (Grattafiori et al., 2024), and *dolphin-2.9-llama3-8b.* Additionally, *RouterDC* is compared to *Zooter*, a strategy described in section 4.2.3 (Lu et al., 2024), which employs a **Reward-based inference** approach. It is also evaluated against a majority voting method, a multiclass classification model, and a clustering technique for embeddings. Except for the MMLU dataset, *RouterDC* outperforms the best stand-alone LLMs and performs as well as or better than other routing methods on different datasets. It achieves the highest average performance across out-of-distribution datasets by effectively approximating the results of the best performing models on each dataset.

**Clustering previous interactions**

Incorporating unsupervised learning can enhance similarity-based routing by identifying the relevant cluster associated with the user query. Once the closest cluster is determined, the query is directed to the LLM that has exhibited the best performance in previous interactions within that cluster. The two studies that proposed cluster-based routing use a *k-means* algorithm. Pichlmeier et al. (2024) proposed the *Expert Router* framework, which employs TF-IDF encoding reduced to 100 dimensions through singular value decomposition. Using k-means is more energy efficient than using an LLM, but the researchers didn't train it from scratch. Instead, they used a pre-trained *k-means* model trained on the C4 dataset (Raffel et al., 2020) - a 300GB cleaned version of the Common Crawl web crawling corpus (Gururangan et al., 2023). While Pichlmeier et al. (2024) do not describe how they assign an LLM to each cluster, Srivatsa et al. (2024) evaluate which LLM has the most frequently generated answers that match the ground truth for each cluster derived from the training data set. The authors evaluated the framework's feasibility rather than its performance by testing latency, response time, and session throughput. They demonstrated that the infrastructure remains robust under high-load scenarios. Their work underscores the potential necessity of training on large datasets to enable smaller algorithms to effectively capture the information contained in queries.

When Srivatsa et al. (2024) trained the K-means algorithm from scratch for the task, the clusters did not generalise effectively from the training dataset to the test dataset. The size and diversity of the training data set appear to be key factors when training

clustering methods for routing. The authors did not observe any impact on the results from altering the encoding strategy, whether using a dense representation strategy with *RoBERTa* (Zhuang et al., 2021) or a sparse strategy with *TF-IDF*. Another significant point to highlight in this latest study is the selection of LLM for routing: the researchers did not compare a smaller LLM with a substantially larger one, using only smaller models such as *gemma-7b and gemma-7b-it* (Gemma-Team, Mesnard et al., 2024), *metamath-7b* (Yu et al., 2024), *mistral-7b-it* (A. Q. Jiang et al., 2023), *llama-2-13B-chat and llama-2-7b* (Touvron et al., 2023). The study did not involve topic experts, such as those in mathematical tasks, and relied on non-specialist models. Consequently, the clusters formed lacked sufficient distinctiveness to differentiate between the LLMs; often, the LLM assigned to a cluster was also the best-performing model across the dataset. Therefore, selecting complementary models based on topic or parameter values is essential for effective routing in the context of LLMs.

**Preference similarity**

User preferences can improve similarity retrieval by incorporating additional information. By analysing which model was favoured for similar queries, one can infer the likelihood of a larger model being preferred. Both strategies used preference data to develop a ranking-based algorithm. Ong et al. (2024) used preference data from the Chatbot Arena Leaderboard [14], a preference-based LLM ranking interface, to propose *RouteLLM*: a range of routing strategies based on user preferences[15] [16]. They propose to reformulate the routing problem between a smaller, *Mixtral-8x7B*, versus a larger LLM, *GPT-4-1106-preview*, as a binary classification task. This task involves predicting the probability of the larger LLM being preferred for a specific query. To determine this probability, the researchers employ a Bradley-Terry (BT) algorithm (Bradley & Terry, 1952) for similarity-weighted ranking. They weight the queries from the training dataset according to their similarity to the user's query and subsequently use these weighted queries to learn the BT coefficients. These coefficients enable the estimation of the probability of preferring a larger LLM for a specific query. The model parameters are learned through maximum likelihood estimation based on the preference data. Once this probability is inferred, the appropriate route is selected according to a cost threshold ($\alpha \in [0, 1]$), optimised to balance cost and quality.

In parallel, Zhao et al. (2024) introduced an ELO-based algorithm called *Eagle* (Elo, 1978), which aims to rank LLMs through a dual-component structure that includes both a global ELO ranking and a local ELO ranking. First, they computed a global ELO rating for each LLM by leveraging all available pairwise comparison information to establish an overall ranking. Then, they derived a local ELO ranking using pairwise information from the $N$ nearest neighbour queries, which were identified through cosine similarity. The algorithm selects the optimal LLM for a specific query by calculating a weighted sum of

---

[14]Chatbot Arena Leaderboard
[15]lm-sys/RouteLLM
[16]https://huggingface.co/routellm

its global and local ELO rankings, while ensuring the selected model's cost remains within the user's budget.

*Eagle* demonstrated notable superiority over various supervised models, including linear SVM, MLP, and KNN, showcasing a marginal performance improvement of 5.14% over MLP and 4.73% over KNN across multiple benchmarks (i.e., ARC-Challenge, Wino-Grande, HellaSWAG, etc.) (Zhao et al., 2024). Similarly, Ong et al. (2024) demonstrated that their proposed strategies not only matched but, in many instances, surpassed the performance of BERT-based classifiers and fine-tuned LLMs. These results highlight the potential of using information about user preferences to train routing algorithms.

### 4.2.2 Supervised Routing

Similarity-based routing can struggle in complex tasks due to its unsupervised nature, especially when dealing with similar domains of knowledge or when there is significant noise. Thus, once the ability of an LLM to successfully respond to user queries or a specific domain of knowledge has been evaluated, pre-generation routing can be viewed from a supervised paradigm.

**Recommendation System**

The matrix factorization approach from Ong et al. (2024) extrapolate users' preferences from $\mathcal{M}$, a set of various LLM, and $\mathcal{Q}$, a set of queries. This method attempts to deduce a hidden scoring function $s : \mathcal{M} \times \mathcal{Q} \rightarrow \mathbb{R}$ where the score $s(LLM_n, q_i)$ for a $LLM_n$ and a query $q_i$ reflects user preference. For instance, if $LLM_n$ is better than $LLM_a$, $s(LLM_n, q_i) > s(LLM_a, q_i)$. Matrix factorisation proves to be more efficient than a classification-based or LLM-based approach for routing between a smaller and a larger model, particularly in terms of call savings and cost-quality trade-offs. Specifically, while using *GPT-4-1106-preview* over *Mixtral-8x7B* (A. Q. Jiang et al., 2024), it was possible to achieve 80% quality gain on an open-ended question dataset called MT-Bench by relying on only 30% of calls to *GPT-4-1106-preview*. In contrast, the random assignment required 78% of calls to this model. The results are similar when comparing *Claude-3-Opus* to *Llama-3-8B* (Grattafiori et al., 2024) (42% required) on this dataset.

**Domain Classification**

A straightforward approach to routing is domain-based routing, which directs queries to specific expert models based on their domain, such as, for example, health-related or financial queries. This strategy focuses on the level of granularity associated with the domain of knowledge of the query, rather than its specific details.

Two studies implemented a BERT-based classifier model (Simonds et al., 2024; Y. Wang et al., 2024). Notably, Simonds et al. (2024) proposed an implementation that in-

volves fine-tuning a DeBERTA-v3-large model (He et al., 2023a) for domain classification, called *MoDEM*. This implementation encompasses domains including Math, Health, Science, Coding, and *Other*. Based on the identified domain, queries are routed to the appropriate expert LLM for Health with *Palmyra-Med-70B*, for Math with *Qwen2.5-72B-Math-Instruct* (Yang, Zhang et al., 2024), for Science with *Qwen2.5-72B-Instruct* (Yang, Yang et al., 2024), for Coding with *Qwen2.5-72B-Instruct*, and for Other with *Meta-Llama-3.1-70B-Instruct*. Similarly, Y. Wang et al. (2024) found that integrating domain-based classification routing by fine-tuning a BERT-based model effectively matched the performance of the best LLM from their set of small specialised LLM options (*Llama-3-Smaug-8B, Mathstral-7B-v0.1, Qwen2-7B-Instruct* (Yang, Yang et al., 2024), and *Gemma-2-9b-it* (Gemma-Team, Riviere et al., 2024)). The router achieved 52.2%, compared to 52.0% for *Gemma-2-9b-it* (Y. Wang et al., 2024). It was less effective than a much larger LLM (*Llama-3-70B* (Grattafiori et al., 2024)), however, perform similarly to moderately larger models such as *Mixtral-8x7B-Instruct-v0.1* (A. Q. Jiang et al., 2024) and *Yi-1.5-34B-Chat*. It seems that there is a marge of improvement as Simonds et al. (2024)'s approach achieved an MMLU accuracy of 87.7%, which is closer to that of Llama-3.1-405B (88.6%) than Qwen 2.5-72B (86.1%), while using a pool of expert 70B models at a cost per token four times lower than that of the 405B model (Simonds et al., 2024).

BERT-based models do not appear to be necessary for effective domain-based routing. Jain et al. (2024) use a domain classification task, termed *Composition-of-Experts*, which they enhance with a two-step routing approach. In the first step, a *k-NN* model maps queries to knowledge domains such as finance, coding, medicine or Russian language. If classifier uncertainty —measured by entropy— exceeds a predefined threshold, the model assigns the user query to a 'general' category. In the second step, a mixed integer linear programme allocates categories to experts, aiming to minimise costs while adhering to budget constraints related to the available billions of parameters. For certain datasets, such as Arena-Hard or MT-Bench, this framework achieved scores comparable to larger language models, like *Llama-3-70B-Instruct* (Grattafiori et al., 2024) and *Qwen2-72B-Instruct* (Yang, Yang et al., 2024), using fewer average active parameters (approximately 30-50 compared to 70).

This underscores the necessity of defining well-specified domains when implementing domain-based classification routing instead of relying on a default method (Simonds et al., 2024). This strategy demonstrates high accuracy when routing across various domains on the MMLU Multi-Domain benchmark, achieving scores between 77% and 97% in Health, Math, Science, and Coding. However, performance significantly declines in the 'Other' domain, where it drops to 53%, indicating that reliance on a default approach may not be effective. Employing a higher level of granularity and focusing specifically on the knowledge domain improves the router's generalisation ability on out-of-distribution datasets (Y. Wang et al., 2024). This approach allows less restrictive routing rules by linking a routing candidate to more general concepts (i.e., the knowledge domain), resulting in a

3.6% increase in accuracy compared to direct mapping of queries to experts.

**Query complexity inference**

Focusing on user queries can uncover more intricate relationships than those established through domain knowledge. By classifying user queries according to their complexity and assessing whether a candidate LLM can handle each category, the routing process becomes a classification task (Ding et al., 2024; Malekpour et al., 2024; Srivatsa et al., 2024; Stripelis et al., 2024; C. Wang et al., 2024).

C. Wang et al. (2024) demonstrate that employing the granularity level of the query captures more specific information. Their query-level routing significantly outperforms domain-based routing on datasets related to the training data, achieving 64.3% compared to 52.2%. Remarkably, despite using a set of small LLMs (under 9B parameters), it surpasses the performance of a much larger LLM, Llama-3-70b (Grattafiori et al., 2024). However, it is less efficient than the domain-based router on out-of-distribution datasets, scoring 67.1% compared to 69.9%. This illustrates the bias-variance trade-off, where high performance on related data comes at the cost of generalisation ability, and vice versa.

Ding et al. (2024) suggested determining whether a difference in answer quality exists between smaller and larger models for a given query by computing the BART score (Yuan et al., 2021). This score calculates the probability of generating a sequence of $m$ tokens $\mathbf{y} = \{y_1, ..., y_m\}$ based on a reference sequence of $n$ tokens $\mathbf{x} = \{x_1, ..., x_n\}$ with a seq2seq model, called $BART$ (M. Lewis et al., 2020). To train their classification model, the authors implemented a series of progressively refined strategies. They began with a **deterministic strategy** using binary cross-entropy loss with hard labels, which indicate whether the BART score of the smaller LLM is equal to or exceeds that of the larger LLM. Building on this, they introduced a **probabilistic strategy**, replacing hard labels with soft labels derived from the average BART scores of ten responses per model, accommodating the non-deterministic nature of LLMs. To further enhance this approach, they incorporated a **transformation step** to address label imbalance, as the smaller LLM typically scores lower. By introducing a relaxation constant, $t$, they reduced the influence of the larger LLM, ensuring a more balanced label set. This constant was optimised to maximise the pairwise score difference between the two models. The study demonstrated that when there is a significant difference in parameter sizes between two large language models (LLMs), such as FLAN-T5 (800m) (Chung et al., 2024) compared to LLaMA-2-13B (Touvron et al., 2023), 40% of queries were directed to the smaller model, resulting in only a 10% decrease in quality. In cases where the parameter size differences were more subtle, such as between LLaMA-2-13B and GPT-3.5-turbo, 20% of queries were assigned to the smaller model, with a reduction in quality of less than 1%. The use of probabilistic strategies seems to outperform deterministic strategies. The method used to label the training data is crucial. Its deterministic strategy has been compared with

LLM-based repeated calls routing (Aggarwal et al., 2024) and token probability-based routing (Ramírez et al., 2024). The findings suggest that *HybridLLM* is less effective than these alternatives.

One could classify based on previous successes rather than relying on sequence probability (Malekpour et al., 2024; Stripelis et al., 2024). Malekpour et al. (2024) demonstrated that training a *DistilBERT* (Sanh et al., 2020) to classify the cheapest LLM capable of handling a query could reduce costs by 1.4 times, albeit with a loss of nearly six performance points compared to *GPT-4o-mini* (55.2% vs 61.0%). They showed that this strategy resulted in lower successful SQL generation compared to the similarity retrieval method discussed in 4.2.1. However, it is important to note that this classification strategy relied more on smaller LLMs, which performed poorly in generating SQL queries. A better selection of LLM might yield different results (i.e., SQL Expert LLM). Stripelis et al. (2024) trained a BERT-based model (Devlin et al., 2019) and discovered that this strategy reduces costs by 30% and latency by 40% compared to stand-alone generalist (*Fox-1.6B* (Z. Hu et al., 2024)) or experts LLM across various domains, including biomedical (e.g. *BioLlama-7B*), coding (e.g. *CodeLlama-7B* (Rozière et al., 2024)), and mathematics (e.g. *MathDeepSeek-7B*). Their integration of a 2-layer MLP, although suboptimal, highlighted the need for more complex architectures for query-based classification (i.e. BERT architectures).

For Srivatsa et al. (2024), LLM answer ability to a specific query can be scored by evaluating the robustness of generated responses [17]. This assessment involves generating ten responses and comparing them to the reference answer (Srivatsa et al., 2024). An LLM $M_N$ from a set $\mathcal{M} = \{M_1, ..., M_N\}$ is considered adequate for a query if the majority of its generated answers are consistent with the ground truth. During inference, the task can be approached as a multi-label classification task, where multiple outputs correspond to all LLMs deemed reliable for $q_i$, or as separate binary classifications, where one classifier is assigned to each LLM. Once this was accomplished, the authors implemented various strategies to identify the most suitable LLM for routing: selecting the model with the highest confidence score, randomly choosing from those models that surpassed a predetermined arbitrary confidence threshold, and employing a Random Forest algorithm that used the confidence scores of all models—alongside the confidence score of the initial gold reference model—as input. Various LLMs were included: *gemma-7b and gemma-7b-it* (Gemma-Team, Mesnard et al., 2024), *metamath-7b* (Yu et al., 2024), *mistral-7b-it* (A. Q. Jiang et al., 2023), *Llama-2-13B-chat and llama-2-7b* (Touvron et al., 2023). Although the router exhibited reduced latency, it did not achieve the same accuracy as the best-performing LLM, *gemma-7b*.

Alternatively, one can directly infer the performance score of the LLM for a given user

---

[17] kvadityasrivatsa/llm-routing

query, making it a regression task. Q. J. Hu et al. (2024) derive a performance score $Performance_{i,j}$ for an LLM $M_j$ on a query $q_i$ while considering its budget (Q. J. Hu et al., 2024). This is defined as $Performance_{i,j} = \lambda \cdot P_{i,j} - cost_j$, where $P_{i,j}$ represents the predicted ability of $M_j$ to answer to $q_i$, $\lambda$ represents the user's willingness to pay and $cost_j$ denotes the cost of calling $M_j$.[18] The authors evaluated the value of $P_{i,j}$ in the training set using *GPT-4* evaluation or exact matching and used it to train the regression model, but did not provide details. It was demonstrated that using a regression model as a routing mechanism resulted in performance scores that were comparable to those of larger language models, such as GPT-4 or Claude V2, on standard datasets (i.e., MMLU, MBPP, and GSM8K) and RAG-related datasets, but at less cost. Conversely, Sakota et al. (2024) proposed strategies for selecting the most suitable LLM based on performance criteria. These strategies include selecting the model with the highest performance score, irrespective of cost; opting for a model that exceeds a specified performance threshold while being the less expensive; or choosing the model that achieves the highest score within the user's budget by solving an integer linear programming problem. The different models used are OpenAI's: *text-ada-001, text-babbage-001, text-curie-001 and text-davinci-002*. Maximising the performance score without considering cost achieves an accuracy equivalent to the best-performing model, *text-davinci-2*. However, this approach incurs a cost comparable to that of routing to the highest-performing model, resulting in an 11% cost reduction. By implementing either the threshold approach or the cost-sensitive method, they achieved comparable accuracy at a significantly reduced cost (approximately a 62% decrease).

Shnitzer et al. (2024) proposed an approach to identify which LLM might perform best on a new task that was not seen during training. They implemented a collection of binary classifiers to determine whether a model $M_i$ within a set of $n$ LLMs $\mathcal{M} = \{M_1, ..., M_n\}$, could provide an answer that matches the reference answer for a given query. However, the methodology for labelling the training set is not explicitly described. The routing approach infer which LLM will have the highest proportion of expected correct answers for a new task $d$. To generalise on data outside the distribution, the researchers developed an out-of-distribution confidence model. This model estimates the probability that a binary classifier's prediction is correct for a specific data point within a given task, thereby reflecting the model's uncertainty. It employs a regression approach to predict this probability by analysing the distance between the new task and previously encountered tasks. They demonstrated that these strategies outperformed the best model on average, *Llama-2-70B* (Touvron et al., 2023), by enabling the selection of smaller models that could provide adequate answers.

The routing process extends beyond simply directing to various pre-trained LLM; it also encompasses routing to retrieval strategies, prompting techniques, and even systems.

---

[18]withmartian/routerbench

Actually, Jeong et al. (2024) demonstrate similar results to those commented earlier with their *Adaptive-RAG* framework[19]. They route to alternative RAG systems, including no-retrieval, single-step naive RAG, and multi-step RAG methods. Complexity is assessed by generating answers for a set of queries: if the no-retrieval architecture returns the correct answer, the query is labelled as low complexity; if no-retrieval fails while the single and multi-step architectures succeed, the complexity is rated as moderate; and if the single-step fails, it is labelled as high complexity. Subsequently, a *T5-Large* model (Raffel et al., 2020) is trained to classify user queries into one of three complexity categories. Although this strategy tends to favour more complex architectures over a naive RAG approach, it is comparable in matching the reference response within the same latency. The performance remains consistent when using either *Flan-T5-XXL* (Chung et al., 2024) or *GPT-3.5-turbo* for generation. Similarly, in Ning et al. (2024)'s study, the model determines whether a query can be answered using their *Squeletton-of-Thought* (SoT) prompting approach, treating it as a binary classification task[20]. The authors of the study found that incorporating a router between prompts is more effective than applying *SoT* to all queries. However, no comparison was made between the efficacy of their approach and that of a basic prompt as a baseline. It is therefore difficult to ascertain the relative efficacy of this approach.

When explicitly detailed, most studies use a model from the *BERT* family as a backbone, adapting it to supervised training: *BERT-based* (Devlin et al., 2019; Ong et al., 2024; Stripelis et al., 2024), *RoBERTa* (Ning et al., 2024; Srivatsa et al., 2024; Zhuang et al., 2021), *DistilBERT* (Malekpour et al., 2024; Sakota et al., 2024; Sanh et al., 2020; Srivatsa et al., 2024). The choice of backbone model must align with the difficulty of the task of interest. T. Feng et al. (2024) demonstrated that replacing the *DeBERTa* model (He et al., 2023b) with *RoBERTa* (Zhuang et al., 2021) enhanced the performance of the *HybridLLM* framework. Other methods have implemented *T5* (Jeong et al., 2024; Raffel et al., 2020; Srivatsa et al., 2024), multi-layers perceptron (Q. J. Hu et al., 2024; Stripelis et al., 2024), *Random Forest* (Srivatsa et al., 2024), *linear optimisation programmes* (Dekoninck et al., 2024), or *K-nearest neighbour* (Q. J. Hu et al., 2024; Shnitzer et al., 2024).

Supervised pre-generation routing proves to be both rapid and effective (Ding et al., 2024; Srivatsa et al., 2024). Furthermore, studies highlighting the ineffectiveness of classification-based routing in improving the cost / quality trade-off emphasise the importance of complementary capabilities among the available LLMs Srivatsa et al., 2024. This limitation is discussed in the section 5.2.3. Yue et al. (2024) found that a *RoBERTa* model (Zhuang et al., 2021), fine-tuned to route between the smaller *GPT-3.5* and the larger *GPT-4*, was less effective than a specific prompt engineering case discussed pre-

---

[19]starsuzi/Adaptive-RAG
[20]imagination-research/sot

viously. This finding indicates that the effectiveness of a routing framework may be influenced by the available routing candidates. The use of strategies based on query information captures more specific information and significantly outperforms strategies based on domain knowledge. However, it shows reduced efficiency on out-of-distribution datasets, highlighting the bias-variance trade-off for performance Y. Wang et al., 2024.

**Knowledge Graph**

Most methods discussed in this survey have one major drawback: they cannot generalise to new routing options (e.g. models, tools). As noted by T. Feng et al. (2024), these methods rely on a transductive learning framework, which involves learning specific rules from a corpus and applying them to particular cases, similar to those encountered during training. However, in a rapidly evolving environment where new tools frequently emerge, maintaining this type of architecture becomes costly, necessitating frequent retraining whenever new tools are introduced. To address this issue, T. Feng et al. (2024) proposes an *inductive graph framework, GraphRouter*. This framework focuses on learning contextual information regarding the interactions between tasks and tools, ultimately enhancing generalisation. The graph consists of three types of nodes: task nodes, query nodes, and LLM nodes. To initialise task and LLM nodes, we first generate their descriptions and include additional calling cost information for the LLM nodes. We then encode these descriptions using a BERT-like model (Devlin et al., 2019). Query nodes are also encoded using the same model. The relationships among these nodes are represented by edge features. The task-query edge indicates whether a query is related to a specific task, such as a math-related query in the context of GSM8K. Additionally, the LLM-query edge provides a score that reflects the performance of an LLM while considering a desired cost by concatenating cost and performance. They employ a two-layer graph attention network with a 32-dimensional hidden layer to update the representations of nodes using information from their local neighbourhoods. Subsequently, the task of selecting an LLM for a query $i$ can be reframed as an edge prediction between the query $i$ and LLM nodes. The routing options included a set of open-source LLMs, featuring various Llama-based models such as *Llama-3-7b and Llama-3-70b* (Grattafiori et al., 2024), *Llama-2-7b* (Touvron et al., 2023), NousResearch's 34b LLM, *Mistral-7b* (A. Q. Jiang et al., 2023), and *Qwen-1.5-72b* (Bai et al., 2023). They demonstrated a performance surpassing that of the largest LLM on a multi-task dataset while requiring lower costs. Their results also exceeded those of prompt-based routing, *HybridLLM* (Ding et al., 2024), *FrugalGPT* (L. Chen et al., 2023), and a bandit-based model. Notably, with new LLM options, they achieved a performance improvement of 4 points over *FrugalGPT* and 21 points over *HybridLLM*.

### 4.2.3 Reinforcement Learning (RL)-based Routing

Supevised routing, similarity-based routing and generative-based routing strategies may not be able to adapt to new routing options or a new context, such as changes in the way users express themselves. This limitation arises because the routing process needs to learn from interactions Sutton & Barto, 2014. In RL, the router acts as an agent with a set of actions. In the context of optimising the generation step in a conversational agent, an action involves selecting a specific model. Based on the current state or context (user query), the model must identify the appropriate action to maximise a reward function while adhering to constraint functions, such as the cost.

**Stateless algorithms**

Stateless algorithms provide a low-resource alternative by mapping rewards directly to actions. One such approach is *Stochastic Learning Automaton* (Sikeridis et al., 2024) from the *PickLLM* framework. It considers a finite set of actions, specifically the LLM candidates. Each candidate has a probability associated with it that indicates its likelihood of being selected, determined by the reward returned. After each interaction between the user and the system, each probability is updated using a very basic linear reward-inaction scheme (Narendra & Thathachar, 1974): increasing the probability of selecting the LLM with the highest reward, while decreasing the probability of other candidates. A learning rate controls this process. They used a set of open source models, including *Mistral-7B* (A. Q. Jiang et al., 2023), *WizardLM-70B, Llama-2-13B* and *Llama-2-70B* (Touvron et al., 2023). They used the LLM-as-a-Judge (i.e., *GPT-3.5-turbo*) framework alongside a reward model to evaluate accuracy. This strategy significantly reduces both cost and latency compared to using the most expensive option or randomly selecting a model. Specifically, the SLA option reduces latency more consistently than the Q-learning option.In terms of accuracy, the model performed slightly better than Llama-2-70B and Mixtral-8x7B (A. Q. Jiang et al., 2024) on a medical subset and a computer science subset. On other datasets, such as open-ended question-answering and financial datasets, it matched their performance. However, especially when compared to Mixtral-8x7B, it underperformed on a subset of Reddit. This discrepancy may be due to the absence of Mixtral-8x7B in the set of options provided.

Instead of relying on action probabilities, one could optimise selection using expected rewards given the selection of an LLM candidate, referred to as *Q-values* (Kaelbling et al., 1996). In Q-learning, an agent iteratively updates its *Q-values* based on the rewards it receives from the environment after performing actions (Kaelbling et al., 1996). The agent selects the LLM candidate with the highest expected reward. However, focusing solely on highest rewards lead the algorithm to favour previously successful actions. To address this issue, Sikeridis et al. (2024) introduced an epsilon-greedy approach, which balances exploration and exploitation. This technique selects a random action with a probability of $\epsilon$ and chooses the action with the highest expected reward, as indicated by

*Q-values*, with a probability of $(1 - \epsilon)$ (Sikeridis et al., 2024). This strategy effectively reduces both cost and latency in comparison to purely random selection or routing only to the most expensive LLM. Performance remains comparable to the **Stochastic Learning Automaton** approach in terms of accuracy.

Stateless strategies effectively converge on the LLM candidate that best fits the data and resource requirements (Sikeridis et al., 2024). It simplifies the identification of the most appropriate LLM in a given static context. However, it comes at the expense of the ability to select actions based on the current query.

**State-based algorithms**

State-based algorithms differ from stateless algorithms in that they consider contextual information when selecting actions. Unlike stateless algorithms, which identify a single action that maximises expected reward, state-based algorithms consider multiple actions that may be optimal depending on the context.

Nguyen et al. (2024) proposed the *Meta-LLM* framework which incorporates a contextual multiarmed bandit (MAB) model with a discrete set of actions, or arms, that can be selected at each step. An arm is an LLM option to which the router can route. Each arm has an unknown reward distribution. At each iteration, the model chooses an arm and receives a reward, allowing it to update the Q-values. The objective is to select the optimal arm while maximising the reward. Nguyen et al. (2024) implemented a framework to address classification tasks by considering both accuracy and cost within their reward function. Initially, they used OpenAI LLMs as options: *text-ada-001, text-babbage-001, text-curie-001 and text-davinci-002*. MAB model achieved comparable accuracy to the most expensive and high-performing LLM, *text-davinci-002* (91.0 versus 90.9), while incurring significantly lower costs (0.3 versus 4.8 $ per 10,000 queries). The authors explain this by saying that some queries were correctly classified by small LLMs but not by large ones. They conducted alternative experiments using a more heterogeneous set of providers, including *Amazon's Titan Lite, Cohere Command-Light, Llama-2-7B (Touvron et al., 2023)*, and *Claude Instant*. In these trials, the MAB matched the performance of the most performant LLM, *Llama-2-7B* (91.6 versus 91.5), and even outperformed the most expensive model, *Claude Instant* (91.6 versus 87.6), while requiring drastically lower costs: *MetaLLM* at 0.5 $ per 10,000 queries, *Claude Instant* at 2.5 $ per 10,000 queries, and *Llama-2-7B* at 2.4 $ per 10,000 queries. This strategy effectively reduced costs while matching or exceeding the best available LLMs performance. It is important to investigate how the inclusion of state-based algorithms improves the router ability to adapt to its environment.

**Reward-based inference**

Hari & Thomson (2023) noted that it would be impractical to create a Q-table to map the actual reward $r_{M_i}$ for selecting an LLM $M_i$ over all queries. An alternative approach is

to estimate future $r_{M_i}$ using a training set. In their *Tryage* framework, Hari & Thomson (2023) infer expert model losses (rewards) by training a *BERT-small* model (Bhargava et al., 2021; Turc et al., 2019) on a dataset containing queries and corresponding expert model losses. The list of expert models includes *ClinicalBERT* (Alsentzer et al., 2019), *SECBERT, FinancialBERT, PatentBERT, CodeBERT* (Z. Feng et al., 2020) and *RoBERTa* (Zhuang et al., 2021), among others.

This strategy was taken a step further by Lu et al. (2024) with *Zooter*, a regression model trained from *mdberta-v3-base* (He et al., 2023b) through knowledge distillation from a reward model, *QwenRM* (Bai et al., 2023). The router encompasses a series of open-source models: *WizardMath* (H. Luo et al., 2025), *WizardCoder* (Z. Luo et al., 2024), *WizardLM* (Xu et al., 2024), *Llama-2-Chat* (Touvron et al., 2023), *OpenChat* (G. Wang et al., 2024), and *Vicuna*, all of which have 13 billion parameters. During the training process, the answer for a set of queries is generated using the LLMs, and a corresponding reward is attributed to each using a reward model. The distribution of rewards is normalised by a softmax function and used to train a student model (*mdberta-v3-base*), where the Kullback-Leibler divergence is used as the distillation loss.

Using a reward model proves to be an effective approach to the routing problem. Hari & Thomson (2023) discovered that their framework achieved greater routing effectiveness to the "ideal" expert model (50.8% accuracy) compared to a stand-alone *GPT-3.5* (23.6%) or the fine-tuned LLM *Gorilla* (Patil et al., 2024) (10.8%). There are no explicit details on the selection criteria for the ideal expert model. In addition to demonstrating the effectiveness of reward models, Lu et al. (2024) also confirms two hypotheses from previous sections (Lu et al., 2024).

- The first hypothesis highlights the importance of integrating complementary LLMs within the routing process. The scores of the independent models vary significantly depending on the dataset, with some models excelling on specific datasets; for instance, *WizardLM* performs well on the Flask Dataset, while *Llama-2-Chat* (Touvron et al., 2023) performs on AlpacaEval. Nevertheless, *Zooter* achieves an average performance comparable to the best model across each dataset.

- The second hypothesis emphasises the need for variability in LLM parameter sizes. In a benchmark comprising MMLU, GSM8K, and HumanEval, all open-source models exhibited poor performance, which constrained Zooter's overall effectiveness. However, Zooter's performance closely aligns with that of *GPT-4*, apart from the last benchmark, where *GPT-4* significantly outperforms it (32.3 for *Zooter* versus 88.3 for *GPT-4*). Changing the reward model ranking did not improve *Zooter*'s performance on this particular benchmark. This difference may be related to the complexity of the tasks assigned, which may be too challenging for 13B parameter models, given that *GPT-4* is much larger than the LLMs integrated into the router.

Reward-based supervised training enhances the representation of independent LLM

abilities in the embedding latent space (Lu et al., 2024).

### 4.2.4 Generative-based routing

A common post-generation routing approach is to generate a response, assess confidence and determine whether further generation by an LLM is required. An optimal method would simplify this process by assessing confidence during the generation phase itself.

**Token probability**

Actually, Ramírez et al. (2024) achieves this in a low-resource manner by measuring *output uncertainty* or *margin*. Using the list of possible tokens returned by a LLM for the token in the first position of the generation, researchers calculate the margin between the probabilities of the first and second most likely tokens. This method has the advantage of assessing confidence without generating the entire output, making it the most resource-efficient option within the post-generation/cascade category. This approach is akin to the BART score discussed in section 4.2.2; however, the BART score requires the complete final sequence of tokens from the generated answer and does not consider the probabilities of less likely tokens at the same position. A larger language model is called if the margin exceeds a threshold established by a budget-based criterion. They used different pairs of small-large models: *Mistral-7B-Instruct-v0.2* (A. Q. Jiang et al., 2023) & *Mixtral-8x7B-Instruct-v0.1* (A. Q. Jiang et al., 2024), *Llama-2-13B*-hf & *Llama-2-70b-hf* (Touvron et al., 2023), and finally *davinci-002 (GPT-3)* and *GPT-4*. This approach has proven to be effective in maximising performance while minimising costs across all pairs of language models compared to other methods, including regression models, *HybridLLM* (Ding et al., 2024) and *FrugalGPT* (L. Chen et al., 2023). The differences are most significant when using the small-large LLM pair *GPT-3* and *GPT-4*. *Frugal-GPT* performs better on several datasets of classification task (e.g. ISEAR, RT-Pol) with smaller LLM pairs (*Mistral-7B-Instruct-v0.2 & Mixtral-8x7B-Instruct-v0.1*, *Llama-2-13B*-hf & *Llama-2-70b-hf*), but not on Q/A and reasoning tasks.

## 5. Discussion

A wide range of routing strategies exists, from similarity learning to LLM fine-tuning. Most of the strategies presented in this paper are based on the pre-generation approach for routing, employing various lightweight strategies (i.e., similarity learning, supervised learning, RL). On the other hand, post-generation strategies tend to be resource-intensive, as they generally require the generation of several responses.

This survey highlights that the routing problem can be effectively addressed through low-resource solutions, which do not require extensive costs, even for strategies that can generalise to new routing options (T. Feng et al., 2024).

## 5.1 Industrial Consideration

In addition to understanding the scientific state of the art, it is essential to consider the industrial landscape. In other words, *what current strategies are companies implementing and disseminating in light of the findings from this review?* Most of the modules used by companies are based on a pre-generation routing approach, primarily aiming to direct the user's query to specific tools, prompts, or scripts. These modules are generally based on a rather simplistic methodology, including conditional approach[21], similarity routing[22], or simply a prompt-based classification system[23] Some companies leverage more sophisticated architecture by employing an LLM to generate synthetic data used to implement a classification-based routing with a small classifier[24]. Many of the mainstream LLM providers propose prompt-based routing, such as AWS[25] or OpenAI[26]. With this survey, our goal is to improve the transfer of various lightweight routing strategies identified from research to industry.

## 5.2 Key challenges

### 5.2.1 Going beyond financial costs

Most studies focus on optimising the trade-off between financial costs and answer quality L. Chen et al., 2023, while overlooking other significant expenses, such as computational and ecological costs. Computational requirements and environmental impacts are intricately linked. Models with a larger number of parameters generally require more floating-point operations, resulting in increased energy consumption Desislavov et al., 2023; Kaack et al., 2022. It is essential to minimise not only the financial costs but also the computational requirements given the urgent need to mitigate the impact of LLMs on climate change Kaack et al., 2022; Luccioni et al., 2024. This can be accomplished by employing routing modules that activate more resource-intensive components only when necessary. Future work should consider incorporating the computational and environmental costs discussed in section 2.1 into the cost function $C_M$ from the equation 1.

### 5.2.2 Standardisation of routing strategy experiments

There is a clear lack of standardisation in terms of the datasets, metrics and evaluation methodology used. This makes identifying the most appropriate routing methods for their application difficult for readers. Furthermore, few studies have compared their routing approaches with those of other works. The authors often compare these strategies with

---

[21]Haystack's ConditionalRouter, Haystack's FileTypeRouter
[22]aurelio-labs/semantic-router
[23]LangChain's router, Llamaindex's router
[24]lamini-ai/llm-routing-agent
[25]awslabs/multi-agent-orchestrator
[26]openai/swarm

non-routing approaches or other methods they have developed themselves. There is urgent need of a comprehensive benchmark that would allow to compare routing approaches in the same context. Future research could greatly benefit from established routing evaluation frameworks [27] (Q. J. Hu et al., 2024).

In future studies, researchers should always include comparison of their proposed stategy with the following baselines: *1) random routing, 2) the best-performing LLM for each query (the gold routing), 3) the stand-alone overall best performing LLM, and 4) alternative routing strategies discussed in this survey.* These comparisons will help us understand whether routing success comes from available routing options or routing architecture. The difference between the overall best-performing LLM and the gold routing gives us the possible margin of improvement. Also, researchers must look at the number of calls made to each set of options: a routing strategy may perform well while also costing less, simply because it *almost always* routes to the most expensive LLM. Including stand-alone LLMs from the available options as comparison with the custom routing approach will show the capabilities of the models on the different datasets.

### 5.2.3 Using complementary routing options

Routing to candidates with complementary rather than redundant skills is essential to optimise routing performance. For example, multiple models of the same size (e.g., 7B parameters) trained on a generalist corpus may exhibit some complementarity due to differences in their training datasets. However, we cannot expect significant performance enhancements for queries on specific or complex topics requiring advanced reasoning. The primary objective of routing is to maximise quality. Incorporating models with varying parameter sizes or expert models allows the routing strategy to adapt effectively to various contexts. In addition, when evaluating different routing approaches, it is critical to consider the complementarity between routing options as a confounding factor.

### 5.2.4 Consider all steps in the LLM-based system as routing possibilities

Almost all of the studies focused on the generation step by selecting the most appropriate LLM to answer the user's query. However, current LLM-based systems typically include several additional steps. The routing approach can be effectively applied during the embedding step in the RAG architecture (Figure 1). This includes routing between different embedding strategies, such as dense or sparse vectors, or the selection of fine-tuned embedding models (Gao et al., 2023). It also allows routing to databases tailored to specific topics, selection of appropriate similarity functions such as cosine similarity or BM25, and selection of the most appropriate prompting approaches (Gao et al., 2023). This framework can also be extended to facilitate routing between static knowledge sources, such as databases, and dynamic knowledge sources, such as website searches. Some authors have studied the use of routing for additional steps, such as context size selection (Z. Li

---

[27]lm-sys/RouteLLM

et al., 2024), prompt strategies (Ning et al., 2024), and even alternative pipeline designs (Jeong et al., 2024). Viewing these systems as dynamic systems rather than traditional static models facilitates optimisation at each stage and enhances modularity (Gao et al., 2024). This transforms query answering into a singular dynamic process that depends on the specific query being addressed.

### 5.2.5 Towards autonomous adaptive routing strategies

A significant drawback of the routing processes investigated in this survey is the need to retrain or adapt the entire routing system when a new routing option is introduced, such as an additional LLM, a novel prompting strategy, or a new pipeline option. Consequently, routing becomes a non-adaptive process reliant on its initial configuration. Future research should assess the feasibility of autonomous processes that decide when to exploit established actions and when to explore new options. This addresses the *exploration-exploitation dilemma*, which involves balancing the use of current knowledge with the exploration of new routing options to gain additional information Berger-Tal et al., 2014. The research community should consider the routing process as an autonomous adaptive agent capable of adjusting to available resources and, more broadly, to its environment.

## 6.  Conclusion

Routing in an LLM-based system can be defined as a process that aims at selecting components of the system which maximise performance while minimising cost. The definition of the cost to be minimised and the score function to be maximised is essential to the construction of a routing algorithm. We classified routing strategies as pre-generation or post-generation. We highlighted that implementing a routing strategy to maintain performance while minimising cost can be efficient and does not necessarily require high resources. We highlighted the need to develop products based on the lightweight strategies discussed in this survey. We emphasised that there is a need to work on well-designed benchmarks across routing strategies to assess which approach offers the best potential by proposing key baseline comparisons. We also discussed the importance of considering computational and environmental costs in addition to financial costs. Finally, we explore future perspectives for routing improvement through the complementarity of routing options and its potential confounding effect, we consider LLM-based systems as dynamic systems, and highlight the need for the router to be able to autonomously generalise to new routing options.

## References

Aggarwal, P., Madaan, A., Anand, A., Potharaju, S. P., Mishra, S., Zhou, P., Gupta, A., Rajagopal, D., Kappaganthu, K., Yang, Y., Upadhyay, S., Faruqui, M., & .,

M. (2024). AutoMix: Automatically mixing language models. *The 38th Annual Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=e6WrwIvgzX

Alsentzer, E., Murphy, J., Boag, W., Weng, W.-H., Jindi, D., Naumann, T., & McDermott, M. (2019). Publicly available clinical BERT embeddings. *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, 72–78. https://doi.org/10.18653/v1/W19-1909

Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., Fan, Y., Ge, W., Han, Y., Huang, F., Hui, B., Ji, L., Li, M., Lin, J., Lin, R., Liu, D., Liu, G., Lu, C., Lu, K., ... Zhu, T. (2023). Qwen technical report. *CoRR*, cs.CL/2309.16609v1. https://arxiv.org/abs/2309.16609

Berger-Tal, O., Nathan, J., Meron, E., & Saltz, D. (2014). The exploration-exploitation dilemma: a multidisciplinary framework. *PloS one*, 9(4), e95693. https://doi.org/10.1371/journal.pone.0095693

Bhargava, P., Drozd, A., & Rogers, A. (2021). Generalization in NLI: Ways (not) to go beyond simple heuristics. *Proceedings of the Second Workshop on Insights from Negative Results in NLP*, 125–135. https://doi.org/10.18653/v1/2021.insights-1.18

Bradley, R. A., & Terry, M. E. (1952). Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39, 324–345. https://doi.org/https://doi.org/10.2307/2334029

Caldarini, G., Jaf, S., & McGarry, K. (2022). A literature survey of recent advances in chatbots. *Information*, 13(1). https://doi.org/10.3390/info13010041

Chai, Z., Wang, G., Su, J., Zhang, T., Huang, X., Wang, X., Xu, J., Yuan, J., Yang, H., Wu, F., & Yang, Y. (2024). An expert is worth one token: Synergizing multiple expert LLMs as generalist via expert token routing. *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 11385–11396. https://doi.org/10.18653/v1/2024.acl-long.614

Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., Chen, H., Yi, X., Wang, C., Wang, Y., Ye, W., Zhang, Y., Chang, Y., Yu, P. S., Yang, Q., & Xie, X. (2024). A survey on evaluation of large language models. *ACM Trans. Intell. Syst. Technol.*, 15(3). https://doi.org/10.1145/3641289

Chen, L., Zaharia, M., & Zou, J. (2023). FrugalGPT: How to use large language models while reducing cost and improving performance. *CoRR*. https://arxiv.org/abs/2305.05176

Chen, S., Jiang, W., Lin, B., Kwok, J., & Zhang, Y. (2024). RouterDC: Query-based router by dual contrastive learning for assembling large language models. *The 38th Annual Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=7RQvjayHrM

Chiang, C.-H., & Lee, H.-y. (2023). A closer look into using large language models for automatic evaluation. *Findings of the Association for Computational Linguistics: EMNLP 2023*, 8928–8942. https://doi.org/10.18653/v1/2023.findings-emnlp.599

Chuang, Y.-N., Zhou, H., Sarma, P. K., Gopalan, P., Boccio, J., Bolouki, S., & Hu, X. (2024). Learning to route with confidence tokens. *CoRR*, cs.CL/2410.13284v1. https://arxiv.org/abs/2410.13284

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Castro-Ros, A., Pellat, M., Robinson, K., . . . Wei, J. (2024). Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70), 1–53. http://jmlr.org/papers/v25/23-0870.html

Dam, S. K., Hong, C. S., Qiao, Y., & Zhang, C. (2024). A complete survey on LLM-based AI chatbots. *CoRR*, cs.CL/2406.16937v2. https://arxiv.org/abs/2406.16937

Dekoninck, J., Baader, M., & Vechev, M. (2024). A unified approach to routing and cascading for LLMs. *CoRR*, cs.CL/2410.10347v1. https://arxiv.org/abs/2410.10347

Desislavov, R., Martínez-Plumed, F., & Hernández-Orallo, J. (2023). Trends in AI inference energy consumption: Beyond the performance-vs-parameter laws of deep learning. *Sustainable Computing: Informatics and Systems*, 38, 100857. https://doi.org/https://doi.org/10.1016/j.suscom.2023.100857

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, 4171–4186. https://doi.org/10.18653/v1/N19-1423

Ding, D., Mallick, A., Wang, C., Sim, R., Mukherjee, S., Rühle, V., Lakshmanan, L. V. S., & Awadallah, A. H. (2024). Hybrid LLM: Cost-efficient and quality-aware query routing. *The 12th International Conference on Learning Representations*. https://openreview.net/forum?id=02f3mUtqnM

Elo, A. E. (1978). *Ratings of chess players past and present*. Arco Pub.

Es, S., James, J., Espinosa Anke, L., & Schockaert, S. (2024). RAGAS: Automated evaluation of retrieval augmented generation. *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics*, 150–158. https://aclanthology.org/2024.eacl-demo.16

Feng, T., Shen, Y., & You, J. (2024). GraphRouter: A graph-based router for LLM selections. *CoRR*, cs.AI/2410.03834v1. https://arxiv.org/abs/2410.03834

Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D., & Zhou, M. (2020). CodeBERT: A pre-trained model for programming and natural languages. *Findings of the Association for Computational Linguistics: EMNLP 2020*, 1536–1547. https://doi.org/10.18653/v1/2020.findings-emnlp.139

Fürnkranz, J., & Hüllermeier, E. (2012). Preference learning. In C. Sammut & G. I. Webb (Eds.), *Encyclopedia of machine learning and data mining* (pp. 1–7). Springer US. https://doi.org/10.1007/978-1-4899-7502-7_667-1

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Guo, Q., Wang, M., & Wang, H. (2023). Retrieval-augmented generation for large language models: A survey. *CoRR*, cs.CL/2312.10997v5. https://arxiv.org/abs/2312.10997

Gao, Y., Xiong, Y., Wang, M., & Wang, H. (2024). Modular RAG: Transforming RAG systems into LEGO-like reconfigurable frameworks. *CoRR*, cs.CL/2407.21059v1. https://arxiv.org/abs/2407.21059

Gemma-Team, Mesnard, T., Hardin, C., Dadashi, R., Bhupatiraju, S., Pathak, S., Sifre, L., Rivière, M., Kale, M. S., Love, J., Tafti, P., Hussenot, L., Sessa, P. G., Chowdhery, A., Roberts, A., Barua, A., Botev, A., Castro-Ros, A., Slone, A., . . . Kenealy, K. (2024). Gemma: Open models based on gemini research and technology. *CoRR*, cs.CL/2403.08295v4. https://arxiv.org/abs/2403.08295

Gemma-Team, Riviere, M., Pathak, S., Sessa, P. G., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard, T., Shahriari, B., Ramé, A., Ferret, J., Liu, P., Tafti, P., Friesen, A., Casbon, M., Ramos, S., Kumar, R., Lan, C. L., Jerome, S., . . . Andreev, A. (2024). Gemma 2: Improving open language models at a practical size. https://arxiv.org/abs/2408.00118

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., . . . Ma, Z. (2024). The llama 3 herd of models. https://arxiv.org/abs/2407.21783

Guha, N., Chen, M. F., Chow, T., Khare, I. S., & Re, C. (2024). Smoothie: Label Free Language Model Routing. *The 38th Annual Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=pPSWHsgqRp

Gururangan, S., Li, M., Lewis, M., Shi, W., Althoff, T., Smith, N. A., & Zettlemoyer, L. (2023). Scaling expert language models with unsupervised domain discovery. *CoRR*, cs.CL/2303.14177v1. https://arxiv.org/abs/2303.14177

Hari, S. N., & Thomson, M. (2023). Tryage: Real-time, intelligent routing of user prompts to large language models. *CoRR*, cs.LG/2308.11601v2. https://arxiv.org/abs/2308.11601

He, P., Gao, J., & Chen, W. (2023a). Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. https://arxiv.org/abs/2111.09543

He, P., Gao, J., & Chen, W. (2023b). DeBERTav3: Improving deBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. *The 11th International Conference on Learning Representations*. https://openreview.net/forum?id=sE7-XhLxHA

Hu, E. J., yelong shen, Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2022). LoRA: Low-rank adaptation of large language models. *International Conference on Learning Representations.* https://openreview.net/forum?id=nZeVKeeFYf9

Hu, J., Wang, Y., Zhang, S., Zhou, K., Chen, G., Hu, Y., Xiao, B., & Tan, M. (2024). Dynamic ensemble reasoning for LLM experts. *CoRR*, cs.AI/2412.07448v1. https://arxiv.org/abs/2412.07448

Hu, Q. J., Bieker, J., Li, X., Jiang, N., Keigwin, B., Ranganath, G., Keutzer, K., & Upadhyay, S. K. (2024). Routerbench: A benchmark for multi-LLM routing system. *Agentic Markets Workshop at ICML 2024.* https://openreview.net/forum?id=IVXmV8Uxwh

Hu, Z., Zhang, J., Pan, R., Xu, Z., Han, S., Jin, H., Shah, A. D., Stripelis, D., Yao, Y., Avestimehr, S., He, C., & Zhang, T. (2024). Fox-1 technical report. *CoRR*, cs.CL/2411.05281v2. https://arxiv.org/abs/2411.05281

Irugalbandara, C., Mahendra, A., Daynauth, R., Arachchige, T., Dantanarayana, J., Flautner, K., Tang, L., Kang, Y., & Mars, J. (2024). Scaling down to scale up: A cost-benefit analysis of replacing OpenAI's LLM with open source SLMs in production. *2024 International Symposium on Performance Analysis of Systems and Software*, 280–291. https://doi.org/10.1109/ISPASS61541.2024.00034

Jain, S., Raju, R., Li, B., Csaki, Z., Li, J., Liang, K., Feng, G., Thakkar, U., Sampat, A., Prabhakar, R., & Jairath, S. (2024). Composition of experts: A modular compound AI system leveraging large language models. *CoRR*, cs.LG/2412.01868v1. https://arxiv.org/abs/2412.01868

Jang, J., Kim, S., Ye, S., Kim, D., Logeswaran, L., Lee, M., Lee, K., & Seo, M. (2023). Exploring the benefits of training expert language models over instruction tuning. *Proceedings of the 40th International Conference on Machine Learning*, 14702–14729. https://dl.acm.org/doi/abs/10.5555/3618408.3619008

Jeong, S., Baek, J., Cho, S., Hwang, S. J., & Park, J. (2024). Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 7036–7050. https://doi.org/10.18653/v1/2024.naacl-long.389

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., & Sayed, W. E. (2023). Mistral 7b. *CoRR*, cs.CL/2310.06825v1. https://arxiv.org/abs/2310.06825

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., de las Casas, D., Hanna, E. B., Bressand, F., Lengyel, G., Bour, G., Lample, G., Lavaud, L. R., Saulnier, L., Lachaux, M.-A., Stock, P., Subramanian, S., Yang, S., . . . Sayed, W. E. (2024). Mixtral of experts. https://arxiv.org/abs/2401.04088

Jiang, D., Ren, X., & Lin, B. Y. (2023). LLM-Blender: Ensembling large language models with pairwise ranking and generative fusion. *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 14165–14178. https://doi.org/10.18653/v1/2023.acl-long.792

Jiang, R., Chen, K., Bai, X., He, Z., Li, J., Yang, M., Zhao, T., Nie, L., & Zhang, M. (2024). A survey on human preference learning for large language models. *CoRR*, cs.CL/2406.11191v2. https://arxiv.org/abs/2406.11191

Kaack, L. H., Donti, P. L., Strubell, E., Kamiya, G., Creutzig, F., & Rolnick, D. (2022). Aligning artificial intelligence with climate change mitigation. *Nature Climate Change*, 12(6), 518–527. https://doi.org/10.1038/s41558-022-01377-7

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4(1), 237–285.

Lee, C., Roy, R., Xu, M., Raiman, J., Shoeybi, M., Catanzaro, B., & Ping, W. (2024). Nv-embed: Improved techniques for training llms as generalist embedding models. *CoRR*, cs.CL/2405.17428v1. https://arxiv.org/abs/2405.17428

Lee, C.-H., Cheng, H., & Ostendorf, M. (2024). OrchestraLLM: Efficient orchestration of language models for dialogue state tracking. *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1434–1445. https://aclanthology.org/2024.naacl-long.79

Le-Khac, P., Healy, G., & Smeaton, A. (2020). Contrastive representation learning: A framework and review. *IEEE Access*, 8, 193907–193934. https://doi.org/10.1109/ACCESS.2020.3031549

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7871–7880. https://doi.org/10.18653/v1/2020.acl-main.703

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 9459–9474. https://dl.acm.org/doi/abs/10.5555/3495724.3496517

Li, C., Liu, Z., Xiao, S., Shao, Y., & Lian, D. (2024). Llama2Vec: Unsupervised adaptation of large language models for dense retrieval. *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 3490–3500. https://doi.org/10.18653/v1/2024.acl-long.191

Li, C., Qin, M., Xiao, S., Chen, J., Luo, K., Shao, Y., Lian, D., & Liu, Z. (2024). Making text embedders few-shot learners. *CoRR*, cs.IR/2409.15700. https://arxiv.org/abs/2409.15700

Li, Z., Li, C., Zhang, M., Mei, Q., & Bendersky, M. (2024). Retrieval augmented generation or long-context LLMs? a comprehensive study and hybrid approach. *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, 881–893. https://doi.org/10.18653/v1/2024.emnlp-industry.66

Lin, C.-C., Huang, A. Y. Q., & Yang, S. J. H. (2023). A review of ai-driven conversational chatbots implementation methodologies and challenges (1999–2022). *Sustainability*, 15(5), 1–13. https://doi.org/10.3390/su15054012

Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. *Text Summarization Branches Out*, 74–81. https://aclanthology.org/W04-1013/

Liu, J., Gong, R., Zhang, M., He, Y., Cai, J., & Zhuang, B. (2024). ME-Switch: A memory-efficient expert switching framework for large language models. *CoRR*, cs.LG/2406.09041v2. https://arxiv.org/abs/2406.09041

Lu, K., Yuan, H., Lin, R., Lin, J., Yuan, Z., Zhou, C., & Zhou, J. (2024). Routing to the expert: Efficient reward-guided ensemble of large language models. *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics:* 1964–1974. https://aclanthology.org/2024.naacl-long.109

Luccioni, S., Trevelin, B., & Mitchell, M. (2024). The environmental impacts of AI – policy primer. *Hugging Face Blog.* https://doi.org/10.57967/hf/3004

Luo, H., Sun, Q., Xu, C., Zhao, P., Lou, J., Tao, C., Geng, X., Lin, Q., Chen, S., Tang, Y., & Zhang, D. (2025). Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *CoRR.* https://arxiv.org/abs/2308.09583

Luo, Z., Xu, C., Zhao, P., Sun, Q., Geng, X., Hu, W., Tao, C., Ma, J., Lin, Q., & Jiang, D. (2024). Wizardcoder: Empowering code large language models with evol-instruct. *The 12th International Conference on Learning Representations.* https://openreview.net/forum?id=UnUwSIgK5W

Ma, X., Wang, L., Yang, N., Wei, F., & Lin, J. (2024). Fine-tuning llama for multi-stage text retrieval. *SIGIR '24: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2421–2425. https://doi.org/10.1145/3626772.3657951

Malekpour, M., Shaheen, N., Khomh, F., & Mhedhbi, A. (2024). Towards optimizing SQL generation via LLM routing. *NeurIPS 2024 Third Table Representation Learning Workshop.* https://openreview.net/forum?id=VYvYR7U7s3

Manias, D. M., Chouman, A., & Shami, A. (2024). Semantic routing for enhanced performance of LLM-assisted intent-based 5G core network management and orchestration. *CoRR*, cs.NI/2404.15869v1. https://arxiv.org/abs/2404.15869

Mohammadshahi, A., Shaikh, A., & Yazdani, M. (2024). Routoo: Learning to route to large language models effectively. *CoRR*, cs.CL/2401.13979v3. https://arxiv.org/abs/2401.13979

Muennighoff, N., Su, H., Wang, L., Yang, N., Wei, F., Yu, T., Singh, A., & Kiela, D. (2024). Generative representational instruction tuning. *CoRR*, cs.CL/2402.09906v2. https://arxiv.org/abs/2402.09906

Narendra, K. S., & Thathachar, M. A. L. (1974). Learning automata - a survey. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-4(4), 323–334. https://doi.org/10.1109/TSMC.1974.5408453

Nguyen, Q. H., Hoang, D. C., Decugis, J., Manchanda, S., Chawla, N. V., & Doan, K. D. (2024). MetaLLM: A high-performant and cost-efficient dynamic framework for wrapping LLMs. *CoRR*, cs.LG/2407.10834v2. https://arxiv.org/abs/2407.10834

Ning, X., Lin, Z., Zhou, Z., Wang, Z., Yang, H., & Wang, Y. (2024). Skeleton-of-thought: Prompting LLMs for efficient parallel generation. *The 12th International Conference on Learning Representations*. https://openreview.net/forum?id=mqVgBbNCm9

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. *CoRR*, cs.LG/2406.18665v3. https://arxiv.org/abs/2406.18665

Patil, S. G., Zhang, T., Wang, X., & Gonzalez, J. E. (2024). Gorilla: Large language model connected with massive APIs. *The 38th Annual Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=tBRNC6YemY

Pichlmeier, J., Ross, P., & Luckow, A. (2024). Performance characterization of expert router for scalable LLM inference. *CoRR*, cs.CL/2404.15153v2. https://arxiv.org/abs/2404.15153

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8), 9. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(1), 1–67. http://jmlr.org/papers/v21/20-074.html

Ramírez, G., Birch, A., & Titov, I. (2024). Optimising calls to large language models with uncertainty-based two-tier selection. *First Conference on Language Modeling*. https://openreview.net/forum?id=T9cOYH0wGF

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 3982–3992. https://doi.org/10.18653/v1/D19-1410

Rozière, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X. E., Adi, Y., Liu, J., Sauvestre, R., Remez, T., Rapin, J., Kozhevnikov, A., Evtimov, I., Bitton, J., Bhatt, M., Ferrer, C. C., Grattafiori, A., Xiong, W., Défossez, A., . . . Synnaeve,

G. (2024). Code llama: Open foundation models for code. https://arxiv.org/abs/2308.12950

Sakota, M., Peyrard, M., & West, R. (2024). Fly-Swat or Cannon? Cost-effective language model choice via meta-modeling. *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, 606–615. https://doi.org/10.1145/3616855.3635825

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2020). DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *CoRR*, cs.CL/1910.01108v4. https://arxiv.org/abs/1910.01108

Sanh, V., Webson, A., Raffel, C., Bach, S., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Raja, A., Dey, M., Bari, M. S., Xu, C., Thakker, U., Sharma, S. S., Szczechla, E., Kim, T., Chhablani, G., Nayak, N., Datta, D., . . . Rush, A. M. (2022). Multitask prompted training enables zero-shot task generalization. *International Conference on Learning Representations*. https://openreview.net/forum?id=9Vrb9D0WI4

Shen, Y., Song, K., Tan, X., Li, D., Lu, W., & Zhuang, Y. (2023). HuggingGPT: Solving AI tasks with chatGPT and its friends in hugging face. *37th Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=yHdTscY6Ci

Shnitzer, T., Ou, A., Silva, M., Soule, K., Sun, Y., Solomon, J., Thompson, N., & Yurochkin, M. (2024). Large language model routing with benchmark datasets. *CoRR*, cs.CL/2309.15789v1. https://arxiv.org/abs/2309.15789

Si, C., Shi, W., Zhao, C., Zettlemoyer, L., & Boyd-Graber, J. (2023). Getting MoRE out of mixture of language model reasoning experts. *Findings of the Association for Computational Linguistics: EMNLP 2023*, 8234–8249. https://doi.org/10.18653/v1/2023.findings-emnlp.552

Sikeridis, D., Ramdass, D., & Pareek, P. (2024). PickLLM: Context-aware RL-assisted large language model routing. *CoRR*, cs.LG/2412.12170v1. https://arxiv.org/abs/2412.12170

Simonds, T., Kurniawan, K., & Lau, J. H. (2024). MoDEM: Mixture of domain expert models. *CoRR*, cs.CL/2410.07490v1. https://arxiv.org/abs/2410.07490

Srivatsa, K. A., Maurya, K., & Kochmar, E. (2024). Harnessing the power of multiple minds: Lessons learned from LLM routing. *Proceedings of the Fifth Workshop on Insights from Negative Results in NLP*, 124–134. https://aclanthology.org/2024.insights-1.15

Stripelis, D., Xu, Z., Hu, Z., Shah, A. D., Jin, H., Yao, Y., Zhang, J., Zhang, T., Avestimehr, S., & He, C. (2024). TensorOpera router: A multi-model router for efficient LLM inference. *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, 452–462. https://doi.org/10.18653/v1/2024.emnlp-industry.34

Sutton, R. S., & Barto, A. G. (2014). Temporal-difference learning. *Reinforcement Learning: An Introduction*, 143–166. https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., . . . Scialom, T. (2023). Llama 2: Open foundation and fine-tuned chat models. *CoRR*. https://arxiv.org/abs/2307.09288

Tunstall, L., Beeching, E. E., Lambert, N., Rajani, N., Rasul, K., Belkada, Y., Huang, S., Werra, L. V., Fourrier, C., Habib, N., Sarrazin, N., Sanseviero, O., Rush, A. M., & Wolf, T. (2024). Zephyr: Direct distillation of LM alignment. *First Conference on Language Modeling*. https://openreview.net/forum?id=aKkAwZB6JV

Turc, I., Chang, M., Lee, K., & Toutanova, K. (2019). Well-read students learn better: The impact of student initialization on knowledge distillation. *CoRR*, cs.CL/1908.08962v1. https://arxiv.org/abs/1908.08962v1

Wang, C., Zhang, B., Sui, D., Tu, Z., Liu, X., & Kang, J. (2024). A survey on effective invocation methods of massive LLM services. *CoRR*, cs.SE/2402.03408v2. https://arxiv.org/abs/2402.03408

Wang, G., Cheng, S., Zhan, X., Li, X., Song, S., & Liu, Y. (2024). Openchat: Advancing open-source language models with mixed-quality data. *CoRR*. https://arxiv.org/abs/2309.11235

Wang, H., Polo, F. M., Sun, Y., Kundu, S., Xing, E., & Yurochkin, M. (2024). Fusing models with complementary expertise. *The 12th International Conference on Learning Representations*. https://openreview.net/forum?id=PhMrGCMIRL

Wang, L., Yang, N., Huang, X., Yang, L., Majumder, R., & Wei, F. (2024). Improving text embeddings with large language models. *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 11897–11916. https://doi.org/10.18653/v1/2024.acl-long.642

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., & Hajishirzi, H. (2023). Self-instruct: Aligning language models with self-generated instructions. *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 13484–13508. https://doi.org/10.18653/v1/2023.acl-long.754

Wang, Y., Zhang, X., Zhao, J., Wen, S., Feng, P., Liao, S., Huang, L., & Wu, W. (2024). Bench-CoE: A framework for collaboration of experts from benchmark. *CoRR*, cs.AI/2412.04167v1. https://arxiv.org/abs/2412.04167

Wei, H., He, S., Xia, T., Wong, A., Lin, J., & Han, M. (2024). Systematic evaluation of LLM-as-a-judge in LLM alignment tasks: Explainable metrics and diverse prompt templates. *CoRR*, cs.CL/2408.13006v1. https://arxiv.org/abs/2408.13006

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean,

J., & Fedus, W. (2022). Emergent abilities of large language models. *Transactions on Machine Learning Research.* https://openreview.net/forum?id=yzkSU5zdwD

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q. V., & Zhou, D. (2024). Chain-of-thought prompting elicits reasoning in large language models. *Proceedings of the 36th International Conference on Neural Information Processing Systems.* https://dl.acm.org/doi/10.5555/3600270.3602070

Xiong, M., Hu, Z., Lu, X., LI, Y., Fu, J., He, J., & Hooi, B. (2024). Can LLMs express their uncertainty? an empirical evaluation of confidence elicitation in LLMs. *The 12th International Conference on Learning Representations.* https://openreview.net/forum?id=gjeQKFxFpZ

Xu, C., Sun, Q., Zheng, K., Geng, X., Zhao, P., Feng, J., Tao, C., Lin, Q., & Jiang, D. (2024). WizardLM: Empowering large pre-trained language models to follow complex instructions. *The 12th International Conference on Learning Representations.* https://openreview.net/forum?id=CfXh93NDgH

Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., Zhou, C., Li, C., Li, C., Liu, D., Huang, F., Dong, G., Wei, H., Lin, H., Tang, J., Wang, J., Yang, J., Tu, J., Zhang, J., Ma, J., . . . Fan, Z. (2024). Qwen2 technical report. *CoRR*, cs.CL/2407.10671v4. https://arxiv.org/abs/2407.10671

Yang, A., Zhang, B., Hui, B., Gao, B., Yu, B., Li, C., Liu, D., Tu, J., Zhou, J., Lin, J., Lu, K., Xue, M., Lin, R., Liu, T., Ren, X., & Zhang, Z. (2024). Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *CoRR.* https://arxiv.org/abs/2409.12122

Yu, L., Jiang, W., Shi, H., YU, J., Liu, Z., Zhang, Y., Kwok, J., Li, Z., Weller, A., & Liu, W. (2024). Metamath: Bootstrap your own mathematical questions for large language models. *The 12th International Conference on Learning Representations.* https://openreview.net/forum?id=N8N0hgNDRt

Yuan, W., Neubig, G., & Liu, P. (2021). BARTScore: Evaluating generated text as text generation. *Advances in Neural Information Processing Systems.* https://openreview.net/forum?id=5Ya8PbvpZ9

Yue, M., Zhao, J., Zhang, M., Du, L., & Yao, Z. (2024). Large language model cascades with mixture of thought representations for cost-efficient reasoning. *The 12th International Conference on Learning Representations.* https://openreview.net/forum?id=6okaSfANzh

Zhang, J., Krishna, R., Awadallah, A. H., & Wang, C. (2023). EcoAssistant: Using LLM assistant more affordably and accurately. *CoRR*, cs.SE/2310.03046v1. https://arxiv.org/abs/2310.03046

Zhang, L., Jijo, K., Setty, S., Chung, E., Javid, F., Vidra, N., & Clifford, T. (2024). Enhancing large language model performance to answer questions and extract information more accurately. *CoRR*, cs.CL/2402.01722v1. https://arxiv.org/abs/2402.01722

Zhao, Z., Jin, S., & Mao, Z. M. (2024). Eagle: Efficient training-free router for multi-LLM inference. *CoRR*, cs.LG/2409.15518v2. https://arxiv.org/abs/2409.15518

Zhuang, L., Wayne, L., Ya, S., & Jun, Z. (2021). A robustly optimized BERT pre-training approach with post-training. In S. Li, M. Sun, Y. Liu, H. Wu, K. Liu, W. Che, S. He & G. Rao (Eds.), *Proceedings of the 20th chinese national conference on computational linguistics* (pp. 1218–1227). https://aclanthology.org/2021.ccl-1.108/

# A. Appendix A - More information about different concepts addressed by the different studies

## A.1 Example of Dialogue State Tracking

- **Iteration 1:**

  - **User:** What are the hours for the Natural History Museum?
  - **Assistant:** The Natural History Museum is open from 10 AM to 5 PM daily.
  - **Dialogue State** $S_1$: { "topic": "museum-hours", "entity": "Natural History Museum" }

- **Iteration 2:**

  - **User:** How about the Science Museum?
  - **Assistant:** The Science Museum is open from 9 AM to 6 PM every day.
  - **Dialogue State** $S_2$ $(+S_1)$: { "topic": "museum-hours", "entity": "Science Museum" }

- **Iteration 3:**

  - **User:** Are there any special exhibits at the Natural History Museum this weekend?
  - **Assistant:** Yes, there is a special exhibit on ancient civilizations at the Natural History Museum this weekend.
  - **Dialogue State** $S_3$ $(+S_1 + S_2)$: { "topic": "special-exhibits", "entity": "Natural History Museum", "time-frame": "this weekend" }

## A.2 Prompting approaches used by Yue et al. (2024) (Yue et al., 2024)

### A.2.1 Chain-of-Thoughts (CoT)

This approach was proposed by J. Wei et al. (2024) where intermediate reasoning is generated in natural language to solve a problem (J. Wei et al., 2024). Here is an example from the work of Yue et al. (2024):

```
1 Question: Kobe and Pau went to a restaurant...
2 Answer: Pau ordered 5 x 2 = 10 fried chickens in total. Therefore, Pau ate
    10 x 2 = 20 pieces of fried chicken. Ans = 20
3
4 Question: Joelle has 5 orchids and 4 African daisies on her balcony...How
    many petals do the daisies have compared to the orchids?
5 Answer:
```

### A.2.2 Program-of-Thoughts (PoT)

This approach has been proposed by Gao et al. ([2023](#)), where the intermediate reasoning steps are translated into Python code (Gao et al., [2023](#)). Here is an example from the work of Yue et al. ([2024](#)):

```python
#Question: Kobe and Pau went to a restaurant...
#Python code, return ans
kobe_order=5
pau_order=kobe_order*2
pau_eaten =2*pau_order

#Question: Joelle has 5 orchids and 4 African daisies on her balcony...How
    many petals do the daisies have compared to the orchids?
#Python code, return ans
```

This example differs slightly from the original version of Gao et al. ([2023](#)) where the intermediate steps are in the form of natural language *and* Python code. For example:

```python
# Kobe ordered 5 pairs of fried chicken
kobe_order=5
```

# B. Appendix B - Descriptive table of the routing approaches

Table 1: A description of the various works included in the survey

| Study | Task | Benchmarks | Compared Strategies | Resources needed | Routing Step | Routing approach |
|---|---|---|---|---|---|---|
| Aggarwal et al. (2024) | Short and Multi-Choice Q/A, Text understanding, Reasoning | QASPER,QUALITY, COQA, MUTUAL, DIPLOMAT | Stand-alone models (*GPT-4, Llama-2-13B*), FrugalGPT ((L. Chen et al., 2023)), HybridLLM ((Ding et al., 2024)) | High | Post | Repeated calls |
| (Chai et al., 2024) | Multi-domain Q/A | MMLU Expert | Prompting methods, gold routing, LLM-Blender | High | Pre | LLM fine-tuning |
| L. Chen et al. (2023) | Short Q/A, Text classification | HEADLINES, OVERRULING, COQA | Stand-alone models (e.g., *GPT-4, J1-Large, Xlarge, FAIRSEQ*), best LLM per query | High | Post | Answer confidence inference |

Table 1: A description of the various works included in the survey (Continued)

| Study | Task | Benchmarks | Compared Strategies | Resources needed | Routing Step | Routing approach |
|-------|------|------------|---------------------|------------------|--------------|------------------|
| S. Chen et al. (2024) | Multi-domain and Multi-Choice Q/A, Code generation | MMLU, GSM8K, CMMLU, ARC-Challenge, HumanEval, PreAlgebra, MBPP, C-EVAL | stand-alone models (e.g., *MetaMath-Mistral-7B, Chinese-Mistral-7B*), majority voting, ZOOTER (Lu et al., 2024), multi-class classification and clustering | Low | Pre | Query similarity |
| Chuang et al. (2024) | Multi-domain and Multi-Choice Q/A, Code generation | MMLU, OpenbookQA, GSM8K, MedQA | Verbalised confidence, logits-based uncertainty, random routing | High | Post | LLM fine-tuning |
| Dekoninck et al. (2024) | Multi-domain and Multi-Choice Q/A | ARC-Challenge, MMLU-Pro, MixEval, GSM8k | Linear interpolation, linear optimisation programs (query-based classification, cascading) | High | Post | Answer confidence inference |
| Dekoninck et al. (2024) | Multi-domain and Multi-Choice Q/A | ARC-Challenge, MMLU-Pro, MixEval, GSM8k | Linear interpolation, linear optimisation programs (query-based classification, cascading) | Low | Pre | Query complexity inference |

Table 1: A description of the various works included in the survey (Continued)

| Study | Task | Benchmarks | Compared Strategies | Resources needed | Routing Step | Routing approach |
|-------|------|-----------|---------------------|------------------|--------------|------------------|
| Ding et al. (2024) | Instructions | MixInstruct | Random routing, smallest LLM, largest LLM | Low | Pre | Query complexity inference |
| T. Feng et al. (2024) | Multi-domain Q/A, Text understanding, Summarization | Alpaca, GSM8K, SQUAD, Multi-News | FrugalGPT (L. Chen et al., 2023), HybridLLM Ding et al. (2024), prompt routing, smallest LLM, largest LLM, gold routing, bandit-based model | Low | Pre | Knowledge Graph |
| Hari & Thomson (2023) | Multi-Domain Text Corpus | Expert corpus (e.g., Pile-CC, Pubmed Central, ArXiv ) | Stand-alone model (*GPT-3.5-turbo*), Gorilla ((Patil et al., 2024)) | Low | Pre | Reward-based inference |
| Q. J. Hu et al. (2024) | Multi-domain and Multi-Choice Q/A, Instructions, Reasoning | MMLU, MT-Bench, MBPP, HellaSwag, Winogrande, GSM8K, Arc-Challenge | Stand-alone models (e.g., *WizardLM-13B, Claude-V2, Llama-70B*), K-NN, MLP, linear interpolation, gold routing | Low | Pre | Query complexity inference |
| Jain et al. (2024) | Multi-domain Q/A | MMLU Pro, GSM8K | Stand-alone models (e.g., *Qwen2-7B-Instruct, Gemma-2-9B-it*) | Low | Pre | Domain classification |

Table 1: A description of the various works included in the survey (Continued)

| Study | Task | Benchmarks | Compared Strategies | Resources needed | Routing Step | Routing approach |
|---|---|---|---|---|---|---|
| Jang et al. (2023) | Q/A, Reasoning, Text understanding, Text classification, Instructions | RTE, CB, ANLI, COPA, HellaSWAG, Storycloze, WinoGrande, WSC, WiC, Big-Bench, wiki-auto, HGen, COVID-QA, ELI5 | Stand-alone models (e.g., T0-11B, GPT-3, T0-3B) | Low | Pre | Query similarity |
| Jeong et al. (2024) | Q/A, Text understanding | Single-step Q/A (e.G., SQuAD-v1.1, TriviaQA), Multi-step Q/A (e.g., MuSiQue, HotpotQA) | Adaptive Retrieval, Self-RAG, gold routing | Low | Pre | Query complexity inference |
| C.-H. Lee et al. (2024) (1) | DST | MultiWOZ, SGD | Prompt-DST, IC-DST and DS2-T5 | Low | Pre | Query similarity |
| C.-H. Lee et al. (2024) (2) | DST | MultiWOZ, SGD | Prompt-DST, IC-DST and DS2-T5 | High | Post | Sequence probability |
| Z. Li et al. (2024) | Q/A, Fact extraction, Text understanding | NarrativeQA, QASPER, MultiFieldQA, HotpotQA,etc. | Naïve RAG, long context | High | Post | Prompt-based routing |

Table 1: A description of the various works included in the survey (Continued)

| Study | Task | Benchmarks | Compared Strategies | Resources needed | Routing Step | Routing approach |
|---|---|---|---|---|---|---|
| Liu et al. (2024) | Multi-domain and Multi-choice Q/A, Code generation | MMLU, GSM8K, MATH, HumanEval, MBPP, C-Eval and C-MMLU | Stand-alone models (e.g., *Dolphin-2.2.1-Mistral-7B, Llama-2-13B-Chat, MetaMath-13B*) | High | Pre | LLM fine-tuning |
| Lu et al. (2024) | Q/A, Instructions, Multi-domain Q/A, Code generation | AlpacaEval, FLASK, MT-Bench, MMLU, GSM8K, HumanEval | Stand-alone models (e.g., *WizardCoder*, *Vicuna*, *GPT-4*), overall best LLM, gold routing | Low | Pre | Reward-based inference |
| Malekpour et al. (2024) | Text-to-SQL | BIRD | stand-alone models (*GPT-4o, GPT-4o-mini* and *Llama-3.1-8b*) | Low | Pre | Query similarity, Query complexty inference |
| Manias et al. (2024) | Intent detection | Custom Dataset | / | Low | Pre | Query similarity |
| Mohammadshahi et al. (2024) | Multi-Domain Q/A | MMLU | Stand-alone models (e.g., *Llama-2-70B, Mistral-7B, Mixtral-8x7B, GPT-3.5*) | High | Pre | Decoder-based encoding |

Table 1: A description of the various works included in the survey (Continued)

| Study | Task | Benchmarks | Compared Strategies | Resources needed | Routing Step | Routing approach |
|---|---|---|---|---|---|---|
| Nguyen et al. (2024) | Text classification | IMDB, SST-2 | Stand-alone models (e.g., *text-ada-001, text-babbage-001, Claude Instant, Cohere Command-Light*) | Low | Pre | State-based algorithms |
| Ning et al. (2024) | Q/A | FastChat, LLMZoo | No routing | Low | Pre | Query complexity inference |
| Ning et al. (2024) | Q/A | FastChat, LLMZoo | No routing | High | Pre | Prompt-based routing |
| Ong et al. (2024) | Multi-Domain Q/A, Instructions | MT Bench, MMLU, GSM8K | Random routing | Low | Pre | Query complexity inference, preference similarity, recommendation system |
| Ong et al. (2024) | Multi-Domain Q/A, Instructions | MT Bench, MMLU, GSM8K | Random routing | High | Pre | LLM fine-tuning |
| Patil et al. (2024) | Code generation | Custom API calling dataset | Stand-alone models (*Llama-7B, pt-3.5, GPT-4*) | High | Pre | LLM fine-tuning |
| Pichlmeier et al. (2024) | Feasibility | / | / | Low | Pre | Clustering previous interactions |

Table 1: A description of the various works included in the survey (Continued)

| Study | Task | Benchmarks | Compared Strategies | Resources needed | Routing Step | Routing approach |
|---|---|---|---|---|---|---|
| Ramírez et al. (2024) | Text classification, Multi-Domain and Multi-Choice Q/A, Fact Extraction | Wikifact, Openbook, ISEAR, FEVER, bAbI, Natural Questions, SST-2, CR, RT-Polarity | FrugalGPT ((L. Chen et al., 2023)), HybridLLM ((Ding et al., 2024)), query-based classification | Low | Post | Token probability |
| Sakota et al. (2024) | Text classification, Multi-Domain and Multi-Choice Q/A | MMLU, GSM8K, WikiFact, RAFT, LegalSupport, etc. | Gold routing, stand-alone models (e.g., *text-ada-001*, *text-babbage-001*, *text-curie-001*) | Low | Pre | Query complexity inference |
| Shen et al. (2023) | Task Planning | Custom Dataset | / | High | Pre | Prompt-based routing |
| Shnitzer et al. (2024) | Multi-Domain Q/A, Instructions, Text classification | HELM, MixInstruct | Stand-alone models (*Open-Assistant,Vicuna*), *MLM-Scoring, SimCLS, SummaReranker, PairRanker*, overall best LLM, gold routing, | Low | Pre | Query complexity inference |
| Sikeridis et al. (2024) | Q/A | HC3 | Random routing, stand-alone models (*Mixtral-8x7B, Llama2-70B*) | Low | Pre | Stateless algorithms |

Continued on next page

Table 1: A description of the various works included in the survey (Continued)

| Study | Task | Benchmarks | Compared Strategies | Resources needed | Routing Step | Routing approach |
|---|---|---|---|---|---|---|
| Simonds et al. (2024) | Multi-Domain and Multi-Choice Q/A, Code generation | MMLU, MMLU Pro, GPQA, HumanEval, College Math, MATH, GSM8k, Olympiad Bench | Stand-alone models (e.g., *Llama 3.1 405B, Qwen 2.5-72B, etc.*) | Low | Pre | Domain Classification |
| Srivatsa et al. (2024) | Multi-domain Q/A | MMLU, GSM8K | Stand-alone models (e.g., *gemma-7b, mistral-7b*), random routing, gold routing | Low | Pre | Clustering previous interactions, query complexity inference |
| Stripelis et al. (2024) | Multi-domain Q/A, Code generation | Ai2-ARC, GSM8K, MBPP, PubMedQA | Random routing, stand-alone models (e.g., *BioLlama-8B, Fox-1.6B, MathDeepSeek-7B*), gold routing | Low | Pre | Query similarity, Query complexity inference |
| Y. Wang et al. (2024) | Multi-domain and Multi-Choice Q/A / Code generation / Reasoning | MMLU Pro, GSM8K, Winogrande, Big Bench Hard, MMMU, MMStar | Stand-alone models (*Gemma-2-9b-it, Llama-3-Smaug-8B, Mathstral-7B-v0.1, Qwen2-7B-Instruct*) | Low | Pre | Domain classification, Query complexity inference |

Table 1: A description of the various works included in the survey (Continued)

| Study | Task | Benchmarks | Compared Strategies | Resources needed | Routing Step | Routing approach |
|-------|------|------------|---------------------|------------------|--------------|------------------|
| Yue et al. (2024) | Multi-Domain Q/A, Text classification, Reasoning, Text understanding | GSM8K, ASDIV, TabMWP, Big-Bench Hard, CREPE | Stand-alone models wit CoT (*GPT-3.5, GPT-4*) | Low | Pre | Query complexity inference |
| Yue et al. (2024) | Multi-Domain Q/A, Text classification, Reasoning, Text understanding | GSM8K, ASDIV, TabMWP, Big-Bench Hard, CREPE | Stand-alone models wit CoT (*GPT-3.5, GPT-4*) | High | Post | Repeated calls |
| J. Zhang et al. (2023) | Instructions | ToolBench | stand-alone models (GPT-3.5, GPT-4) | High | Post | Code execution |
| Zhao et al. (2024) | Multi-Domain and Multi-Choice Q/A, Code generation , Instructions | MMLU, Hellaswag, GSM8K, ARC-Challenge, Winogrande, MBPP, MT-Bench | Linear SVM, KNN and MLP | Low | Pre | Preference similarity |