CARGO: A Framework for Confidence-Aware Routing of Large Language Models

Amine Barrak*, Yosr Fourati*[‡], Michael Olchawa*, Emna Ksontini[†], Khalil Zoghlami*
*Department of Computer Science and Engineering, Oakland University, Rochester, MI, USA

[†]University of North Carolina Wilmington, Wilmington, NC, USA

[‡]Mediterranean Institute of Technology (MEDTECH), Tunis, Tunisia

Email: aminebarrak@oakland.edu

Abstract—As large language models (LLMs) proliferate in scale, specialization, and latency profiles, the challenge of routing user prompts to the most appropriate model has become increasingly critical for balancing performance and cost. We introduce CARGO (Category-Aware Routing with Gap-based Optimisation), a lightweight, confidence-aware framework for dynamic LLM selection. CARGO employs a single embeddingbased regressor trained on LLM-judged pairwise comparisons to predict model performance, with an optional binary classifier invoked when predictions are uncertain. This two-stage design enables precise, cost-aware routing without the need for humanannotated supervision. To capture domain-specific behavior, CARGO also supports category-specific regressors trained across five task groups: mathematics, coding, reasoning, summarization, and creative writing. Evaluated on four competitive LLMs (GPT-40, Claude 3.5 Sonnet, DeepSeek V3, and Perplexity Sonar), CARGO achieves a top-1 routing accuracy of 76.4% and win rates ranging from 72% to 89% against individual experts.

These results demonstrate that confidence-guided, lightweight routing can achieve expert-level performance with minimal overhead, offering a practical solution for real-world, multi-model LLM deployments.

Index Terms—Prompt Routing, Lightweight Model Router, Confidence-Aware Inference, Large Language Models (LLMs).

I. INTRODUCTION

Large language models (LLMs) have advanced significantly in recent years, driven by improvements in model architectures, training methodologies, and large-scale data availability [1]. These developments have expanded their capabilities beyond traditional natural language processing (NLP) tasks, allowing complex reasoning, code generation, knowledge retrieval, and multimodal understanding [2]. Modern LLMs incorporate billions of parameters, self-attention mechanisms, and reinforcement learning to enhance response quality based on human feedback [3].

However, LLMs are optimized for different objectives, some excel in efficiency and factual accuracy, while others prioritize creativity and depth of reasoning [4]. For instance, models tailored for coding tasks emphasize precise syntax generation and debugging assistance [5], whereas those optimized for creative writing focus on coherence and stylistic variation [6]. These differences arise from variations in training data, alignment techniques, and external knowledge integration.

As LLMs are deployed across diverse applications, users submit queries that span domains such as mathematics, coding,

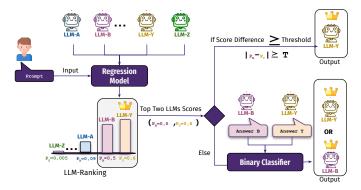


Figure 1: Overview of CARGO. A single embedding regressor scores all candidate LLMs; if the top–two scores are within threshold τ , a classifier selects the better one.

creative writing, and business analysis. Each domain imposes distinct requirements on reasoning abilities, factual accuracy, and language fluency [7], [8]. No single model consistently outperforms others in all categories due to differences in optimization strategies and architectural design. A model adept at numerical reasoning may struggle to generate coherent narratives, while a model fine-tuned for conversational fluency may lack precision in coding-related tasks [4], [9].

Moreover, within the same domain, task complexity varies significantly. A business-related query can involve simple data retrieval or complex financial forecasting, each requiring different levels of reasoning and contextual understanding [10]. Similarly, coding tasks range from syntax corrections to large-scale algorithm optimization, where different models excel in specific areas [11].

Most deployments address this challenge by maintaining a portfolio of LLM experts and then routing each query to whichever model promises the best quality [12]. However, contemporary routing frameworks exhibit three limitations that undermine their practical utility. (1) *Latency overhead*. Some routing frameworks employ sequential cascade strategies: prompts are first processed by small models and, if confidence is insufficient, passed step-by-step to larger ones. This incremental procedure causes latency to accumulate, pushing the worst-case response time toward the sum of all model calls [13]. (2) *Limited expert models diversity*. Many cost-aware systems reduce routing to a binary choice between one "small" and one "large"

model, overlooking the mid-sized, domain-specialist experts that are increasingly common in production environments [14]. (3) *High supervision and scalability constraints* Routers still rely on thousands of labeled prompt—response pairs or reward scores for calibration. Because the cost of label acquisition grows with both the breadth of domains and the size of the expert pool, every expansion or task shift triggers an expensive retraining cycle, hindering operational scalability [12], [14].

Recent benchmarking, further show that existing routing mechanisms, have difficulty generalizing to complex tasks and up-to-date models [15].

Motivated by these shortcomings, this paper introduces a confidence-aware routing framework (CARGO) structured around three complementary steps, each carefully designed to resolve key gaps identified in existing approaches. In the first stage, we avoid the common dependence on costly human-annotated data by using candidate LLMs themselves to evaluate response quality. This idea is supported by recent studies demonstrating that LLM-generated assessments closely match human judgements, especially in complex reasoning and knowledge-rich contexts [16], [17]. The second stage involves a lightweight embedding-based regression model. This model simultaneously ranks all candidate experts in a single inference pass. Finally, if the two best candidates receive close regression scores, a refinement step invokes a binary classifier trained on pair-wise preferences to break the tie. This overall process is illustrated in Figure 1. We provide the source code as a fully reproducible framework to support future research¹.

This work is guided by the following research questions:

- **RQ1** (Individual Model Performance): How do selected expert LLMs (GPT-4o, Claude 3.5 Sonnet, DeepSeek V3, Perplexity Sonar) individually perform on our curated five-domain dataset?
- RQ2 (Comparative Routing Efficacy): How effectively does our confidence-aware routing framework select optimal LLMs compared to individual expert models?
- RQ3 (Category-Specific Routing Behavior): How does routing performance vary across task categories, and does category-specific modeling further enhance accuracy?

II. RELATED WORK

The rapid proliferation of large language models (LLMs), accelerated by Transformer architectures, has significantly expanded accessibility but introduced substantial complexity in model selection, with platforms like Hugging Face now hosting over 200,000 models [18]. As no single model is universally optimal for all tasks [4], end-users face considerable difficulty identifying the best fit among diverse architectures, training data, and intended applications. The distinct strengths and weaknesses exhibited by each model necessitate intelligent selection strategies to maximize performance.

Large Language Models (LLMs) often excel in specialized domains when adapted or fine-tuned for specific tasks. For instance, OpenAI's Codex, fine-tuned for programming, solves

¹https://sites.google.com/view/cascon2025

28.8% of HumanEval coding benchmark problems in a single attempt, whereas the base GPT-3 model solves nearly none [19]. Similarly, Google's Minerva, trained on mathematical content, achieves state-of-the-art accuracy on STEM question-answering tasks [20], surpassing general LLMs in quantitative reasoning. Instruction-tuned models like InstructGPT [21] further highlight the impact of domain-specific fine-tuning, with a 1.3-billion-parameter variant preferred by human evaluators over the original 175-billion-parameter GPT-3 across diverse prompts. In other words, no single model performs best across all tasks. Instead, performance depends on training strategies, reasoning capabilities, and task alignment, meaning a model's effectiveness relies on how it was trained, its ability to reason, and how well it fits the requirements of a specific task.

Given the diverse strengths of individual LLMs across tasks, researchers have explored LLM Ensemble—a paradigm that strategically integrates multiple models to leverage their complementary capabilities [22]. Instead of relying on a single model, ensembles aggregate outputs from multiple models to improve accuracy and robustness. Basic ensemble methods include majority voting, where outputs from the same or different LLMs are combined, and the final answer is selected by agreement or similarity [23]. Advanced methods such as DEEPEN (Deep Parallel Ensemble) enhance aggregation by integrating token-level predictions from heterogeneous LLMs at each generation step. DEEPEN aligns probability distributions across models by mapping them into a shared representation space based on relative representation theory. The combined predictions are then translated back into the probability space of a designated primary model, allowing the system to handle vocabulary mismatches [24].

However, employing ensembles of multiple LLMs can significantly increase computational overhead, latency, and resource demands, especially if models are used indiscriminately. Recent studies, such as Bench-CoE [12] and Routing to the Expert [25], explore dynamic routing methods to efficiently manage these demands by selecting the best model or a subset of models based on query characteristics. Bench-CoE [12] leverages benchmark evaluations to train routers that assign queries to expert models, while Routing to the Expert [25] employs reward-guided mechanisms to route inputs effectively. Despite their dynamic approaches, these methods rely heavily on ground truth datasets to train their routers. This dependency introduces challenges, particularly in acquiring large-scale labeled data, and limits generalization to inputs outside the training data distribution.

Dynamic model selection presents an appealing alternative by intelligently selecting specialized models in real-time. By using lightweight classifiers or meta-models to route queries, dynamic selection efficiently combines accuracy with reduced computational overhead. In this paper, we propose an innovative dynamic routing method that employs LLM-based judges instead of traditional ground truth datasets. We leverage multiple LLM evaluators to train a classifier that dynamically selects the optimal LLM for each input, significantly improving routing effectiveness, particularly in multi-category scenarios.

III. METHODOLOGY

We present CARGO, a confidence-aware routing framework that selects the most suitable LLM for each prompt. As shown in Figure 2: (1) We first collect prompts across diverse domains and obtains responses from multiple LLMs. (2) A pairwise labeling procedure, scores these responses using multiple LLMs as judges. (3) The resulting labeled dataset trains a routing regression models, either global (all domains combined) or category-specific. (4) we finally compares framework performance with individual LLMs.

A. Prompt Dataset Preparation

Our benchmark draws prompts from Hugging Face datasets in five categories: Mathematics, Coding, Reasoning & Knowledge, Text Summarization, and Creative Writing.

Each category was initially populated with approximately 1,000 prompts, aggregated from one or more sources. The prompts were shuffled to increase randomness and reduce source-specific patterns. To address overlap across datasets, we applied MinHashLSH [26], a similarity detection technique, to identify and remove duplicate or highly similar prompts.

After filtering and deduplication, the final prompt counts per category were: 1,085 for mathematics, 1,453 for coding, 958 for reasoning, 1,210 for summarization, and 1,069 for English creative writing. The sets were reshuffled to prevent ordering bias. Table I summarizes the datasets used to evaluate LLMs across these task categories.

Table I: Datasets used for evaluating LLM ranking

Category	Dataset Source	Subset Size
	OpenThoughts-math [27],	
Mathematics	AI2 ARC [28], MBPP [29] HumanEval [19],	1,085
Coding	Alpaca [30] MMLU-Pro [31],	1,453
Reasoning & Knowledge	BBH [32]	958
Text Summarization	XSum [33]	1,210
Creative Writing	10k Prompts Ranked [34]	1,069

1) LLM Response Collection

To obtain responses, each prompt was individually submitted through the OpenRouter API [35], a unified interface that streamlines querying across multiple language models. We configured each query with a *default temperature of "0.7"*, which strikes a balance between creativity and hallucination [36]. We selected four distinct LLMs for their complementary strengths and diverse architectures:

- **ChatGPT-4o**: Chosen for strong general-purpose reasoning and fluent multilingual output [37].
- Claude 3.5 Sonnet: Selected for advanced reasoning and large context handling [38].
- **DeepSeek V3**: Included for fast inference and factual accuracy via a Mixture-of-Experts (MoE) that activates only a subset of components per query [39].
- **Perplexity Sonar**: Used for accurate, concise responses with real-time web retrieval [40].

Each model's output was captured and stored alongside the corresponding prompt.

B. Pairwise Annotation with LLM Judges

Inspired by recent ranking methodologies, we propose a pairwise labeling approach, which leverages multiple LLMs as judges. Our approach builds upon traditional pairwise ranking strategies [41], [42], incorporating a jury of independent LLM evaluators to reduce bias and improve stability [17], [43].

1) Approach Overview

For each prompt, we collect responses from M distinct LLMs. We then:

- a. Generate All Pairs: Construct every possible pair of responses (R_i, R_j) .
- b. **Multiple LLM Judges:** Present each pair to N different LLM judges, ensuring the judges remain blind to which LLM authored each response.
- c. Comparison Criteria: Judges compare pairs based on clarity, accuracy, and completeness, returning a score indicating which response is preferred.
- d. **Score Aggregation:** Pairwise comparisons yield cumulative scores that position each response in a final ranking.

By relying on this llm-based annotation approach rather than on hand-crafted gold answers, the framework can scale naturally to domains where ground truth is ambiguous or absent.

2) Scoring Mechanism

Each judge J_k assigns one of three possible scores $a_{i,j}^{(k)}$ when comparing response R_i to R_j :

$$a_{i,j}^{(k)} = \begin{cases} 1 & \text{if judge } J_k \text{ prefers response } R_i, \\ 0.5 & \text{if } R_i \text{ and } R_j \text{ are equally good (tie),} \\ 0 & \text{if judge } J_k \text{ prefers response } R_j. \end{cases}$$
 (1)

For each pair (R_i, R_j) , we aggregate judge scores as $s_{i,j}$:

$$s_{i,j} = \sum_{k=1}^{N} a_{i,j}^{(k)}; \quad S_i = \sum_{j \neq i} s_{i,j}.$$
 (2)

Here, N is the total number of judges. A higher $s_{i,j}$ indicates that R_i consistently outperformed R_j in pairwise evaluations. The global score S_i for a given response R_i is derived by summing its pairwise scores against all other responses. A larger S_i signals stronger overall performance relative to the other M-1 responses.

3) Ranking Outcomes

Finally, we rank all responses in descending order of their total scores S_i . The top-scoring response is considered the "best" solution for that prompt, followed by others in decreasing order of preference. By incorporating multiple independent judges, we mitigate single-model biases.

Example: Consider four responses $\{A, B, C, D\}$ to a single prompt. We compare each pair (e.g., A vs. B, A vs. C, A vs. D, etc.), presenting them to four independent LLM judges. If three judges prefer A over B while one finds them equally good, A accumulates 3.5 points and B gets 0.5 for that pair. Repeating this for all six pairs yields a total score per response, which determines the final ranking.

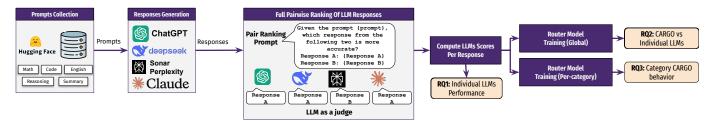


Figure 2: Overview of **CARGO**. Prompts from five task categories are answered by LLMs. A *pairwise LLM-judging stage* ranks responses and produces labeled data to train confidence-aware regression routers—either *global* or *category-specific*.

4) Evaluation Protocol

We employ three evaluations to assess the reliability and fairness of the pairwise LLM-based annotation scheme using a stratified sample of 250 prompts (50 prompts per category). Inter–judge agreement: To assess the consistency of our LLM jury, we compute pairwise Cohen's κ between all judge pairs and report the mean. Agreement scores range from 59.50% to 64.06%, with an average of 62.06%, corresponding to $\kappa=0.62$. This reflects *substantial* agreement², indicating a stable and reliable ranking signal.

Self-preference bias: We assess fairness by measuring the self-win rate, the frequency with which a judge selects its own output. With four candidates, the expected rate under random selection is 25%. Observed rates are: gpt-4o (26.3%), claude-3.5-sonnet (30.4%), deepseek-chat (31.1%), and sonar (24.6%). Overall, results suggest most judges behave fairly, with limited bias. Human validation: To assess external validity, two independent professionals reviewed 250 prompts sampled across five categories. For each, they selected the best response from four anonymized LLM outputs. We compare their choices to the top-ranked outputs from our annotator using Cohen's κ . Observed averaged agreement corresponds to $\kappa=0.72$. These results indicate strong alignment between human judgments and the automatic rankings.

C. Routing Model Training

We trained regression models to predict the optimal LLM response using two primary strategies: *global* (combining all categories) and *category-specific*. Below, we detail our data preparation method, describe the regression models used, and outline the experimental setup.

Embedding and Scoring Preparation: For each prompt, embeddings were generated using OpenAI's text-embedding-ada-002 model. Scores from various LLMs were normalized to sum to 1, enabling fair comparability across prompts and models. These embeddings served as input features for regression models, with the normalized scores used as regression targets.

<u>Classification Models:</u> We constructed a dataset of LLM comparisons by generating examples where, for each prompt, one model outperformed another. Each entry contains the prompt embedding, metadata for two LLMs, and a label indicating the

better one. This dataset was used to train binary classifiers to predict the better of two given LLMs for a specific prompt.

Regression Models: We employed four regression models to predict LLM scores: Random Forest (RF), Ridge Regression, XGBoost, and a Multi-Layer Perceptron (MLP). All models were trained using Ada embeddings as input features, with normalized LLM scores.

Global Model Training: In the global training approach, prompts from all categories (mathematics, coding, reasoning, summarization, and English) were combined into a single, unified dataset. Regression models were trained on this combined dataset to predict LLM performance across diverse tasks.

<u>Category-Specific Training:</u> In the category-specific approach, separate regression models were trained independently for each task category. Each model used only prompts from its respective domain, allowing specialized predictions tailored specifically to that task.

Evaluation Methodology: We partitioned our dataset into 80% training and 20% validation sets, employing stratified sampling to preserve category proportions in both sets.

We evaluated the regression models based on their ability to predict LLM performance using the following metrics:

- Mean Squared Error (MSE): Quantifies the prediction error between the model's output scores and the actual normalized performance scores for each LLM.
- Accuracy of Predicting the Best Model (Top-1 Accuracy): The percentage of prompts for which the regression model correctly identified the top-performing LLM.
- **Top-2 Inclusion Rate** (**Top-1-or-2 Accuracy**): The proportion of prompts for which the true top-performing LLM appeared within the model's top two predictions.
- Win Rate: Reflects how frequently the router model either correctly selected the optimal LLM or an LLM with superior actual performance. Formally defined as:

Win Rate =
$$\frac{1}{\text{Total}} \left(0.5 \times N_{\text{spec}} + N_{\text{better}} \right)$$
 (3)

Where:

- $N_{\rm spec}$: Number of times the specific LLM was picked.
- N_{better} : Number of times top LLM was preferred.
- Total: Total number of prompts evaluated.

Example (CARGO vs. DeepSeek Win Rate): If *DeepSeek* is selected 4 times, better-performing models 3 times, and worse models once out of 8, its win rate: (4*0.5+3)*100/8 = 62.5%.

²Standard interpretation: $\kappa < 0.20$ (slight), 0.21–0.40 (fair), 0.41–0.60 (moderate), 0.61–0.80 (substantial), > 0.80 (near–perfect).

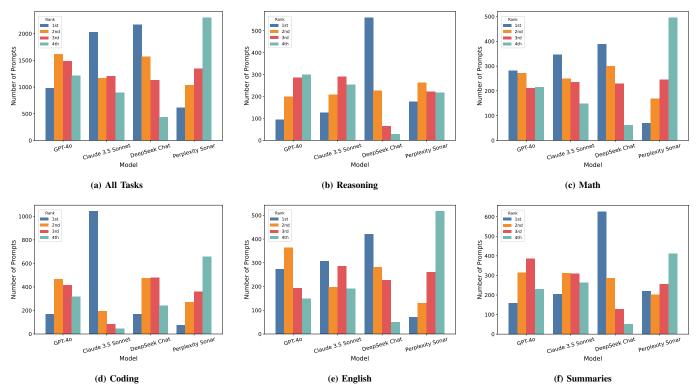


Figure 3: Rank frequencies (1 = best, 4 = worst) awarded to each model: (a) all tasks combined, (b) reasoning, (c) math, (d) coding, (e) English-language tasks, and (f) summarization.

IV. RESULTS

A. RQ1: Task-Specific Performance of Individual LLMs

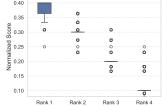
This research question examines the task-specific performance and scoring characteristics of four prominent large language models (LLMs): **GPT-40**, **Claude 3.5 Sonnet**, **DeepSeek**, and **Sonar**, evaluated over 5,289 prompts across multiple task categories. We present three complementary analyses: (i) distribution of normalized scores by individual models, (ii) frequency of top-ranking performance across different tasks, and (iii) stratification of scores based on assigned ranks.

Normalized Score Distribution by Model: Figure 4,(a) illustrates the distribution of normalized scores assigned by the PairRanker method for each of the four models across all prompts. The scores for all models fall within a range of 0.08 to 0.40. Distribution patterns emerge, reflecting differences in median performance and score dispersion. Specifically, DeepSeek and Claude 3.5 Sonnet show concentrated high scores, while Sonar exhibits broader, lower distributions.

Task-Specific Rank Frequencies: We further analyze model performance at a granular task-specific level, reporting how often each model achieved the highest rank. Across the benchmark set, **DeepSeek** most frequently obtained the top rank (2,164 instances), followed by **Claude 3.5 Sonnet** (2,025 instances). In contrast, **GPT-40** and **Sonar** were ranked first less often, with 977 and 609 instances, respectively.

In task-specific evaluations, DeepSeek excelled in *reasoning* (560 top ranks) and *math* (388 top ranks). Claude 3.5 Sonnet





(a) Score distributions per LLM (b) Score distributions by rank Figure 4: Normalised Score trends across prompts: (a) by LLM; (b) by rank (Rank-1 to Rank-4).

demonstrated clear dominance in *coding*, achieving the highest rank in 1,043 instances—significantly outperforming other models. In *English-language tasks*, DeepSeek secured the top rank most frequently (420 instances), ahead of Claude 3.5 Sonnet (306), GPT-40 (273), and Sonar, which consistently occupied lower ranks, including 517 instances at rank 4. DeepSeek also emerged as the leader in *summarization*, with 627 top-rank occurrences. Sonar's performance consistently placed it lower across all task categories, particularly in language-intensive evaluations. Detailed rank distributions across task categories are shown in Figure 3.

Score Stratification by Rank: Figure 4,(b) analyzes normalized scores across all prompts, stratified by assigned ranks. Rank 1 scores consistently cluster around a median of approximately 0.39. Rank 2 scores exhibit a broader distribution with a median around 0.30, while Ranks 3 and 4 exhibit progressively

lower medians. These patterns indicate that models with Rank 1 and Rank 2 typically produce high-quality outputs.

F1: Task-Specific LLM Performance Insights

DeepSeek and Claude outperformed others—DeepSeek in reasoning/summarization, Claude in coding. Scoring shows the top two LLMs achieve a high mean score (0.30–0.39), making them strong candidates per prompt.

B. RQ2: Performance of the ensemble vs. individual models

This research question investigates whether an ensemblebased approach outperforms individual models in predicting the best-performing LLM for a given prompt.

<u>Classification Models Performance:</u> We evaluated multiple binary classification models to determine their effectiveness in predicting the superior LLM given pairs of models and a prompt embedding. Specifically, we compared Random Forest, Logistic Regression, XGBoost, MLP, and Ridge classifiers, assessing each using accuracy, precision, recall, F1-score, and Area Under the ROC Curve (AUC). Table II summarizes the performance metrics.

The MLP classifier achieved the best performance overall, yielding the highest accuracy (0.826), precision (0.844), recall (0.805), F1-score (0.824), and AUC (0.903). Random Forest and XGBoost followed closely, demonstrating competitive results, while Logistic Regression and Ridge classifiers performed noticeably lower across all metrics.

Table II: Performance Comparison of Binary Classifiers

Model	Accuracy	Precision	Recall	F1-score	AUC
Random Forest	0.767	0.770	0.769	0.769	0.856
Logistic Regression	0.643	0.650	0.638	0.644	0.685
XGBoost	0.763	0.774	0.752	0.763	0.846
MLP	0.826	0.844	0.805	0.824	0.903
Ridge Classifier	0.641	0.648	0.635	0.641	0.641

F2: Binary Classification Between Two LLMs

For a given prompt, a binary classifier can determine the better LLM between a pair with up to **82.6% accuracy**.

Regression Models Performance: To address this, we trained multiple lightweight regression models, including Random Forest, Ridge Regression, XGBoost, Support Vector Regression (SVR), and Multi-Layer Perceptron (MLP), using a dataset composed of normalized prompt embeddings and corresponding performance scores for GPT-40, Claude 3.5 Sonnet, DeepSeek, and Perplexity Sonar.

We report each model's average MSE, Top-1 prediction accuracy, and Top-1-or-2 accuracy, which reflects the proportion of cases in which the model's top prediction matches either the true best or the true second-best LLM. Table III summarizes the performance of each regression model. Random Forest and Ridge Regression demonstrated the highest Top-1 accuracy, reaching approximately 58%, while Random Forest also achieved the highest Top-1-or-2 accuracy at 83.65%.

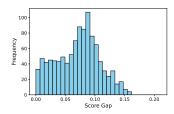
Regression Analysis of Predicted Score Gaps: We analyzed the regression predictions by computing the

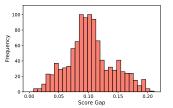
Table III: Predictive performance of regression models.

Model	Avg. MSE	Top-1 Acc.	Top-1-or-2 Acc.
Random Forest	0.0079	58.41%	83.65%
XGBoost	0.0082	57.84%	83.27%
Ridge	0.0079	58.13%	82.42%
SVR	0.0087	56.14%	79.77%
MLP	0.0110	51.51%	75.33%

gap between the top predicted LLM (Rank 1) and the next-best options (Rank 2 and Rank 3). We focused this analysis on the Random Forest model, given its best performance. We compute the score gaps:

$$g_{1-2}(p) = \hat{y}_{(1)}(p) - \hat{y}_{(2)}(p), \qquad g_{1-3}(p) = \hat{y}_{(1)}(p) - \hat{y}_{(3)}(p).$$





(a) Rank-1 vs Rank-2

(b) Rank-1 vs Rank-3

Figure 5: Predicted score gaps from RF regression: (a) top-2 LLMs; (b) top-1 vs. third-ranked LLM.

Figure 5 presents histograms of these predicted score gaps, illustrating how closely the top predictions are spaced. *The Rank 1 vs Rank 2 gap has a mean of 0.072*, whereas the Rank 1 vs Rank 3 gap averages 0.162. These values show that the regressor assigns its top two candidates close scores.

F3: Regression Performance Analysis

Random Forest achieved the best prediction accuracy among all regressors, with top-1 and top-1-or-2 accuracies of 58.41% and 83.65%, respectively. Score gap analysis shows that the predicted scores for the top two LLMs are often close.

Confidence-Based Routing Decision: Our regression analysis revealed that the mean score gap between the top two predicted LLMs is only 0.07, suggesting that many prompts yield two closely ranked candidates. To exploit this observation without incurring the cost of always querying multiple models, we introduce a *confidence-based routing policy* guided by the predicted score gap.

For a prompt p, let $\hat{y}_{(1)}(p)$ and $\hat{y}_{(2)}(p)$ denote the highest and second-highest scores predicted by the regressor. We define the confidence gap as

$$g(p) = \hat{y}_{(1)}(p) - \hat{y}_{(2)}(p).$$

Given a threshold τ , we apply the following decision rule:

- 1) If $g(p) \ge \tau$, the router issues a *single* query to the top-ranked LLM.
- 2) Otherwise, the router:
 - a) Sends queries to both the top- and second-ranked LLMs.

b) Runs a binary classifier to select the superior response from these two queries.

To identify a suitable τ , we sweep values from 0.01 to 0.20 and report the following metrics:

- Coverage Accuracy: The percentage of prompts where the set of LLMs chosen for evaluation (usually the top-1 predicted LLM, and the top-2 if the model is uncertain) includes the actual best-performing LLM.
- Overall Selection Accuracy: The percentage of prompts where the router's final decision, either the top-1 predicted LLM or the one selected between the top two using a classifier, matches the LLM that actually performed best.

We evaluated our confidence-based LLM routing strategy using a two-stage system where a Random Forest regressor first predicts the quality scores of each LLM given a prompt embedding. When the gap between the top two scores falls below a threshold τ , a second-stage MLP classifier is invoked to select between them.

Figure 6 shows how performance evolves as τ increases. Coverage accuracy improves from 58.4% at $\tau=0.01$ to 78.5% at $\tau=0.10$, while overall selection accuracy rises from 58.3% to 74.4% over the same range. Beyond $\tau=0.10$, both metrics plateau, with marginal gains up to 81.2% coverage and 76.4% accuracy at $\tau=0.20$. This trend highlights a favorable trade-off where the regressor is used in confident cases, and the classifier is invoked only in uncertain ones, resulting in strong accuracy without added routing complexity.

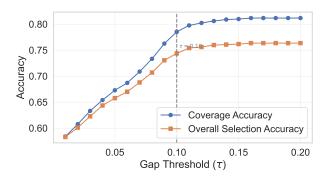


Figure 6: Accuracy–Confidence trade-off from variation analysis of the regression gap between the top two predicted LLMs.

Win Rate Comparison: CARGO vs. LLMs: We analyse the win rate, which quantifies how often a model selects an LLM that performs better or equally well compared to a reference LLM on a given prompt. It reflects the model's ability to identify the most suitable LLM response. *Ensemble models consistently achieve higher or comparable win rates than any single LLM across all evaluated cases.* As shown in Table IV, Random Forest achieves the highest win rates among regressors for gpt-40 (72.87%) and perplexity (84.12%), and ties for the best result on deepseek (61.15%). Once combined with the MLP classifier, the pipeline further improves performance. At $\tau = 0.12$, it reaches 82.28%, 73.53%, 71.36%, and 89.18% respectively across the four LLMs, with an average

win rate of 79.09%. These results confirm that the two-stage pipeline is more effective at selecting the better LLM than any standalone model or one-stage ensemble.

Table IV: Win rates (%) of CARGO models versus each LLM.

Model	GPT-40	Claude 3.5	DeepSeek	Perplexity
	Regre	essor-Only Mod	lels	
Random Forest	72.87	63.42	61.15	84.12
XGBoost	73.11	63.33	60.59	83.41
Ridge	73.25	63.33	60.26	83.13
SVR	71.25	61.81	59.22	81.47
MLP	69.42	59.83	55.39	78.12
C	ARGO Rout	ing Pipelines (I	RF + MLP)	
$\tau = 0.07$	78.69	70.04	67.67	87.19
$\tau = 0.10$	81.62	72.92	70.79	88.94
$\tau = 0.12$	82.28	73.53	71.36	89.18

F4: Confidence-Based Routing Performance

The confidence-based two-stage pipeline (Random Forest regressor + MLP classifier) substantially improves LLM selection, achieving **76.4% overall accuracy** at $\tau=0.10$. It also consistently outperforms any single LLM, with a **mean win rate of 79.1%** across the expert pool.

C. RQ3: Category-Specific Ensemble Task Behavior

This section investigates how ensemble-based routing strategies perform across different task categories. By analyzing both regression and classification results, we aim to understand whether certain categories benefit more from ensemble decision **Category Classification Accuracy:** We trained a Random Forest to predict each prompt's high-level category (Coding, English, Math, Reasoning, Summaries). The classifier achieved 96 % accuracy on held-out data, with overall precision, recall, and F1 all approximately 0.96 (see Table V).

Table V: Random Forest category classification performance

Category	Precision	Recall	F1-score
Coding	0.97	0.99	0.98
English	0.88	0.92	0.90
Math	0.99	0.95	0.97
Reasoning	0.98	0.93	0.96
Summaries	0.98	0.99	0.98
Overall	Accuracy = 0.9603		

Regression Model Performance by Category: As shown in Table VI, regression models exhibit consistent yet varied performance across the five task categories. Among them, Ridge Regression and Random Forest emerge as the most stable performers, both achieving strong average MSE.

Top-1 Accuracy varies more significantly across categories. While models may not always precisely identify the single best-performing LLM, they remain consistently effective at narrowing down the top two candidates. This is reflected in the noticeably higher Top-1-or-2 Accuracy across all categories. Notably, tasks such as *Coding* and *Reasoning* exhibit both high Top-1-or-2 Accuracy (up to 0.919 and 0.864, respectively) and

relatively strong Top-1 Accuracy, with values exceeding 0.74 and 0.63 in the best cases.

Table VI: Regression Model Performance by Categorium	gorv
--	------

Category	Model	Avg. MSE	Top-1 Acc.	Top-1-or-2 Acc.
	Random Forest	0.0093	0.518	0.748
	XGBoost	0.0095	0.509	0.743
Summaries	MLP	0.0120	0.413	0.642
	Ridge	0.0092	0.514	0.748
	SVR	0.0098	0.472	0.706
	Random Forest	0.0078	0.500	0.750
	XGBoost	0.0081	0.485	0.719
English	MLP	0.0099	0.459	0.704
_	Ridge	0.0076	0.500	0.770
	SVR	0.0085	0.459	0.745
	Random Forest	0.0079	0.474	0.745
	XGBoost	0.0083	0.454	0.750
Math	MLP	0.0077	0.510	0.750
	Ridge	0.0077	0.490	0.735
	SVR	0.0089	0.495	0.765
	Random Forest	0.0062	0.747	0.912
	XGBoost	0.0066	0.747	0.912
Coding	MLP	0.0064	0.762	0.919
	Ridge	0.0062	0.755	0.905
	SVR	0.0074	0.740	0.919
	Random Forest	0.0073	0.616	0.859
	XGBoost	0.0077	0.638	0.859
Reasoning	MLP	0.0078	0.621	0.853
	Ridge	0.0076	0.627	0.859
	SVR	0.0084	0.638	0.864

Confidence-Based Routing Decision by Category: We aim to improve the prediction accuracy of the top candidate LLM selected to answer a given prompt per category. Specifically, we analyze how the routing behavior varies across different macrocategories based on the predicted score gap between the top two LLM candidates. We use the Random Forest regressor as the primary router, since Random Forest performed consistently well across all categories in earlier evaluations. The MLP classifier applied in this routing pipeline is the same as in RQ2, trained once globally on the full dataset.

As shown in Figure 7, increasing the threshold τ steadily improves the Overall Selection Accuracy in all categories up to a saturation point. Most categories exhibit a rapid gain in performance between $\tau=0.05$ and $\tau=0.12$, after which the improvement flattens. The *Coding* and *Reasoning* categories achieve the highest accuracy, peaking at 0.864 and 0.818 respectively. In contrast, *Summaries* and *English* show more gradual increases and lower plateaus, reaching maximum accuracies of 0.683 and 0.673 respectively.

F5: Ensemble Pipeline Performance by Category

It was straightforward to predict the category of a given prompt, achieving 96% accuracy using a Random Forest classifier. Although the regression models alone were not always sufficient to reliably select the best-performing LLM, Top-1-or-2 Accuracy reached 86.4% for Coding and 81.8% for Reasoning. Overall Selection Accuracy increased consistently with a higher score gap threshold τ , reaching 86.4% in coding and 81.8% in reasoning.

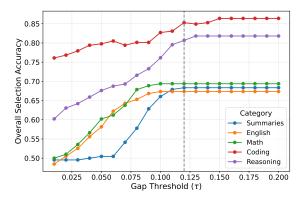


Figure 7: Overall Selection Accuracy vs. score gap threshold τ for each prompt category.

V. DISCUSSIONS

A. Accuracy vs. Computational Cost

To clarify when lightweight routing models (regressors and binary classifiers) should be invoked, we relate their execution cost to the resulting selection accuracy. In our setup, each prompt always goes through a single regressor R, which scores all candidate LLMs. A binary classifier B is optionally used to resolve close decisions between the top two candidates.

Let g(p) denote the score difference between the top-1 and top-2 LLMs predicted by R on prompt p. The classifier is called only if $g(p) < \tau$, where τ is a user-defined threshold controlling the router's conservativeness. The expected number of lightweight model calls per prompt is then:

$$\mathbb{E}[C(\tau)] = C_R + \underbrace{\Pr[g(p) < \tau]}_{\text{classifier usage}} \cdot C_B, \tag{4}$$

where $C_R=1$ and $C_B=1$ are unit costs for the regressor and the classifier. The classifier usage term is measured empirically on the test set.

Global results. For the global regressor, increasing τ causes the router to fall back to the binary classifier more often, leading to higher accuracy. Between $\tau=0.04$ and $\tau=0.08$, the classifier is invoked on 23% to 55% of prompts, and accuracy rises from 64.4% to 70.7%. This is the most efficient part of the curve, where each additional classifier call yields a meaningful gain. After $\tau=0.10$, accuracy gains slow down while cost rises sharply. At $\tau=0.13$, accuracy reaches 75.9% with over 80% classifier usage, and saturates near 76% beyond that. Using the classifier on half the prompts captures most of the benefit, making $\tau\in[0.06,0.10]$ a practical trade-off.

Per-category results. Using a separate regressor per category improves the balance between accuracy and classifier usage. At $\tau=0.13$, Coding reaches 84.9% accuracy with 79% classifier usage, and Reasoning reaches 81.8% with 94%. In contrast, Summaries and English need the classifier on nearly every prompt but still stay below 70% accuracy. This highlights differences in how well the regressor can distinguish top candidates across domains.

Baselines. We set baseline models for comparison. The Top-1 method simply selects the highest-scoring LLM from R; it is cheap but often inaccurate. The $Top-1 \lor Top-2$ strategy measures how often the true best LLM appears among the top two predictions. While this shows that the regressor can often narrow down the correct choice to a small set, it does not resolve which of the two to pick-choosing at random still risks failure in half the cases. A classifier is therefore essential to make accurate selections within this narrowed set. All-pairs **voting** serves as a strong baseline, invoking the classifier B on every pair of candidate LLMs and selecting the model with the most wins. For M=4, this results in $\binom{4}{2}=6$ classifier calls per prompt. While this approach achieves competitive accuracy, its cost scales quadratically with M, making it impractical for large ensembles and potentially introducing delays that violate service-level agreements (SLAs).

Table VII compares all strategies in terms of selection accuracy and expected lightweight cost per prompt, assuming M=4 candidate LLMs. Both global and category-aware versions of CARGO offer a balanced trade-off, capturing most of the gains of Top-2 or all-pairs voting while requiring far fewer classifier calls.

Table VII: Accuracy and expected number of lightweight model calls per prompt (4 LLMs).

Routing Strategy	Expected Calls	Accuracy
Top-1 only (global R)	1R	58.4 %
Top-1 or Top-2 (accept both)	1R	83.7 %
All-pairs voting $\binom{4}{2} = 6B$	6B	82.6 %
Global CARGO ($\tau=0.13$)	1R + 0.946B	75.99 %
Per-category CARGO ($\tau = 0.13$):		
Summaries	1R + 1.000B	68.3 %
English	1R + 0.995B	67.3 %
Math	1R + 1.000B	69.4 %
Coding	1R + 0.794B	84.9 %
Reasoning	1R + 0.938B	81.8 %

B. Scalability: New Experts and New Domains

A practical router should scale with expanding model portfolios and evolving tasks with minimal retraining. To quantify this scalability, we conduct two ablation studies that extend the original benchmark: (i) adding a fifth expert and (ii) introducing an unseen domain.

Experiment 1: Adding a Fifth Expert. We added *Gemini* $\overline{2.5\text{-}Pro}$ to the original set of four expert models, generated responses for 250 prompts, and re-labeled the resulting pairs. Adding a fifth model introduces 4 new pairs per prompt, each judged by 4 LLMs, resulting in $250(prompts) \times 4(pairs) \times 4(judges) = 4,000$ additional comparisons. The average Cohen's kappa remained at 0.60, and human agreement with the new top-ranked model held steady at 71%.

Experiment 2: Adding a sixth domain. We added a *Translation* slice with 50 prompts from SoftAge-AI³. All four experts produced responses, and each prompt generated 6 unique pairs,

judged by 4 LLMs, yielding 50×6 (pairs) $\times4$ (judges) = 1,200 additional comparisons. Judge agreement for translation was $\kappa=0.77$, and human alignment with LLM ranking was $\kappa=0.67$, comparable to earlier domains.

These results show that *CARGO* accommodates both portfolio and task expansion without extra human labeling, while preserving agreement, fairness, and alignment. Such plug-and-play scalability is essential for real-world deployments with evolving model inventories and workloads.

VI. THREATS TO VALIDITY

Internal Validity. Our ranking methodology revealed that some models consistently outperformed others, leading to imbalanced rank distributions. This poses a threat to internal validity by potentially biasing the learned models toward dominant LLMs. To reduce this effect, we employed stratified sampling to preserve category and rank diversity in training and test sets. Construct Validity. Relying on a single LLM to evaluate responses can introduce self-preference bias, as shown in [44]. To mitigate this, we used multiple independent LLMs as judges for pairwise comparisons, with final decisions made by majority vote to prevent any model from evaluating its own output. We also reported inter-judge agreement to assess consistency, quantified self-preference bias across models, and validated our ranking methodology through human agreement studies, thereby strengthening the reliability of the evaluation.

External Validity. While our analysis is based on a curated dataset, there remains a risk that the findings may not generalize to other domains or prompt distributions. To mitigate this, we collected prompts from multiple sources and ensured coverage across diverse categories, increasing the likelihood that our results extend beyond a single task or domain. Additionally, the list of expert LLMs was selected in early 2025 based on thencurrent market leaders. As the LLM landscape evolves rapidly, newer models may outperform our selected experts. However, our methodology is model-agnostic and remains applicable regardless of which LLMs are included in the expert pool.

Conclusion to Validity. Since our evaluation relies on LLM-based judges, ensuring generalizability is essential. To address this, we tested the scalability of our system by adding a sixth domain and new expert models. The router maintained consistent performance, indicating robustness to domain and model change.

VII. CONCLUSION

We introduced **CARGO**, a lightweight, confidence-aware routing framework for dynamically selecting the most suitable large language model (LLM) per prompt. CARGO integrates an embedding-based regressor with a fallback binary classifier, enabling cost-sensitive control over routing while supporting both global and category-specific configurations. CARGO was evaluated on four state-of-the-art expert LLMs across five task domains. It achieved a top-1 routing accuracy of 76.4% globally and up to 86% in specific domains. Against expert models, our two-stage pipeline consistently outperformed individual LLMs, achieving a mean win rate of 79.1%.

³https://huggingface.co/datasets/SoftAge-AI/prompt-eng_dataset

Unlike prior routing systems that depend on large transformer-based routers (e.g., BERT) and extensive labeled datasets, CARGO relies on lightweight models and LLM-based judgments, requiring significantly less training data. We further demonstrated the robustness of CARGO by extending it to a new domain and adding an expert model, while maintaining performance and agreement levels. These results show that accurate and adaptive LLM selection can be achieved with minimal overhead, making CARGO well-suited for real-world, performance-sensitive deployments where routing speed, model cost, and adaptive accuracy are essential.

VIII. ACKNOWLEDGEMENT

This work was conducted in collaboration with **zuvu.ai**, an industrial partner developing AI-powered tools.

REFERENCES

- [1] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong et al., "A survey of large language models," arXiv preprint arXiv:2303.18223, vol. 1, no. 2, 2023.
- [2] J. Chen, H. Lin, X. Han, and L. Sun, "Benchmarking large language models in retrieval-augmented generation," in *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence*. AAAI Press, 2024. [Online]. Available: https://doi.org/10.1609/aaai.v38i16.29728
- [3] M. Shao, A. Basit, R. Karri, and M. Shafique, "Survey of different large language model architectures: Trends, benchmarks, and challenges," *IEEE Access*, 2024.
- [4] R. K. Joshi, P. Priya, V. Desai, S. Dudhate, S. Senapati, A. Ekbal, R. Ramnani, and A. Maitra, "Strategic prompting for conversational tasks: A comparative analysis of large language models across diverse conversational tasks," arXiv preprint arXiv:2411.17204, 2024.
- [5] L. Chen, Q. Guo, H. Jia, Z. Zeng, X. Wang, Y. Xu, J. Wu, Y. Wang, Q. Gao, J. Wang et al., "A survey on evaluating large language models in code generation tasks," arXiv preprint arXiv:2408.16498, 2024.
- [6] C. Gómez-Rodríguez and P. Williams, "A confederacy of models: A comprehensive evaluation of llms on creative writing," arXiv preprint arXiv:2310.08433, 2023.
- [7] Z. Chen and H. Qi, "Large language models for mathematical analysis," arXiv preprint arXiv:2501.00059, 2024.
- [8] M. U. Hadi, q. a. tashi, R. Qureshi, A. Shah, A. Muneer, M. Irfan, A. Zafar et al., "A survey on large language models: applications, challenges, limitations, and practical usage," 2023.
- [9] N. Sinha, V. Jain, and A. Chadha, "Are small language models ready to compete with large language models for practical applications?" arXiv preprint arXiv:2406.11402, 2024.
- [10] R. Koncel-Kedziorski, M. Krumdick, V. Lai, V. Reddy, C. Lovering, and C. Tanner, "Bizbench: A quantitative reasoning benchmark for business and finance," arXiv preprint arXiv:2311.06602, 2023.
- [11] W. Yan, H. Liu, Y. Wang, Y. Li, Q. Chen, W. Wang, T. Lin, W. Zhao, L. Zhu, H. Sundaram et al., "Codescope: An execution-based multilingual multitask multidimensional benchmark for evaluating llms on code understanding and generation," arXiv preprint arXiv:2311.08588, 2023.
- [12] Y. Wang, X. Zhang, J. Zhao, S. Wen, P. Feng, S. Liao, L. Huang, and W. Wu, "Bench-coe: a framework for collaboration of experts from benchmark," arXiv preprint arXiv:2412.04167, 2024.
- [13] L. Chen, M. Zaharia, and J. Zou, "Frugalgpt: How to use large language models while reducing cost and improving performance," arXiv preprint arXiv:2305.05176, 2023.
- [14] D. Ding, A. Mallick, C. Wang, R. Sim, S. Mukherjee, V. Ruhle, L. V. Lakshmanan, and A. H. Awadallah, "Hybrid Ilm: Cost-efficient and quality-aware query routing," arXiv preprint arXiv:2404.14618, 2024.
- [15] Q. J. Hu, J. Bieker, X. Li, N. Jiang, B. Keigwin, G. Ranganath, K. Keutzer, and S. K. Upadhyay, "Routerbench: A benchmark for multi-llm routing system," arXiv preprint arXiv:2403.12031, 2024.
- [16] A. Bavaresco, R. Bernardi, L. Bertolazzi, D. Elliott et al., "Llms instead of human judges? a large scale empirical study across 20 nlp evaluation tasks," arXiv preprint arXiv:2406.18403, 2024.
- [17] P. Verga, S. Hofstatter, S. Althammer, Y. Su, A. Piktus, A. Arkhangorodsky, M. Xu, N. White, and P. Lewis, "Replacing judges with juries: Evaluating Ilm generations with a panel of diverse models," arXiv preprint arXiv:2404.18796, 2024.

- [18] S. N. Hari and M. Thomson, "Tryage: Real-time, intelligent routing of user prompts to large language models," arXiv:2308.11601, 2023.
- [19] M. Chen *et al.*, "Evaluating large language models trained on code,"
- [20] E. Dyer and G. Gur-Ari, "Minerva: Solving quantitative reasoning problems with language models," *June*, vol. 30, p. 2022, 2022.
- [21] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray et al., "Training language models to follow instructions with human feedback," Advances in neural information processing systems, vol. 35, pp. 27730–27744, 2022.
- [22] Z. Chen, J. Li, P. Chen, Z. Li, K. Sun, Y. Luo, Q. Mao, D. Yang, H. Sun, and P. S. Yu, "Harnessing multiple large language models: A survey on llm ensemble," arXiv preprint arXiv:2502.18036, 2025.
- [23] J. Li, Q. Zhang, Y. Yu, Q. Fu, and D. Ye, "More agents is all you need," arXiv preprint arXiv:2402.05120, 2024.
- [24] Y. Huang, X. Feng, B. Li, Y. Xiang, H. Wang, T. Liu, and B. Qin, "Ensemble learning for heterogeneous large language models with deep parallel collaboration," *Advances in Neural Information Processing Systems*, vol. 37, pp. 119838–119860, 2024.
- [25] K. Lu, H. Yuan, R. Lin, J. Lin, Z. Yuan, C. Zhou, and J. Zhou, "Routing to the expert: Efficient reward-guided ensemble of large language models," arXiv preprint arXiv:2311.08692, 2023.
- [26] E. Zhu, "Datasketch: Probabilistic data structures for big data," https://github.com/ekzhu/datasketch, 2016.
- [27] O. T. Team, "OpenThoughts-114k-math," https://huggingface.co/datasets/ open-r1/OpenThoughts-114k-math, January 2025.
- [28] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord, "Think you have solved question answering? try arc, the ai2 reasoning challenge," arXiv:1803.05457v1, 2018.
- [29] J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le et al., "Program synthesis with large language models," arXiv preprint arXiv:2108.07732, 2021.
- [30] Tarun, "Python code instructions 18k alpaca," https://huggingface.co/datasets/iamtarun/python_code_instructions_18k_alpaca, 2023.
- [31] Y. Wang, X. Ma, G. Zhang et al., "Mmlu-pro: A more robust and challenging multi-task language understanding benchmark," in *The Thirty*eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track, 2024.
- [32] M. Suzgun, N. Scales, N. Schärli, S. Gehrmann, Y. Tay, H. W. Chung, A. Chowdhery, Q. V. Le, E. H. Chi, D. Zhou, and J. Wei, "Challenging big-bench tasks and whether chain-of-thought can solve them," arXiv preprint arXiv:2210.09261, 2022.
- [33] S. Narayan, S. B. Cohen, and M. Lapata, "Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization," *ArXiv*, vol. abs/1808.08745, 2018.
- [34] Data is Better Together Collective, "10k prompts ranked," https://huggingface.co/datasets/data-is-better-together/10k_prompts_ranked, 2024, hugging Face dataset.
- [35] OpenRouter, "API Reference," 2025. [Online]. Available: https://openrouter.ai/docs/api-reference/overview
- [36] M. Renze, "The effect of sampling temperature on problem solving in large language models," in *Findings of the Association for Computational Linguistics: EMNLP* 2024, 2024, pp. 7346–7356.
- [37] OpenAI, "Gpt-4o technical report," https://openai.com/index/gpt-4o, 2024, accessed: 2025-06-01.
- [38] Anthropic, "Claude 3.5 sonnet release notes," https://www.anthropic.com/ news/claude-3-5-sonnet, 2024, accessed: 2025-06-01.
- [39] DeepSeek, "Deepseek-v3 technical overview," https://github.com/ deepseek-ai/DeepSeek-V3, 2024, accessed: 2025-06-01.
- [40] Perplexity AI, "Introducing sonar models," https://www.perplexity.ai/ blog/sonar-models, 2024, accessed: 2025-06-01.
- [41] D. Jiang, X. Ren, and B. Y. Lin, "Llm-blender: Ensembling large language models with pairwise ranking and generative fusion," arXiv preprint arXiv:2306.02561, 2023.
- [42] J. Luo, X. Chen, B. He, and L. Sun, "Prp-graph: Pairwise ranking prompting to llms with graph aggregation for effective text re-ranking," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024, pp. 5766–5776.
- [43] S. Badshah and H. Sajjad, "Reference-guided verdict: Llms-as-judges in automatic evaluation of free-form text," *arXiv:2408.09235*, 2024.
- [44] P. Wang, L. Li, L. Chen, Z. Cai, D. Zhu, B. Lin, Y. Cao, Q. Liu, T. Liu, and Z. Sui, "Large language models are not fair evaluators," arXiv preprint arXiv:2305.17926, 2023.