Multi-LLM Routing on Use Case and User Preferences

Cannon Wilson, Abhijnya Menakur. Kumaresh Suresh Babu
University of Wisconsin-Madison

Abstract

Selecting the best Large Language Model (LLM) for specific use cases has become increasingly challenging due to the growing number of models with diverse performance, cost, and latency profiles. This work builds on existing benchmarks to develop a dynamic multi-LLM routing system that adapts to user-defined preferences. By incorporating advanced evaluation models and extending beyond text-based tasks to multimodal applications, our approach democratizes access to LLMs, enabling efficient, cost-effective, and user-centric routing.

1 Introduction

The advent of Large Language Models (LLMs) has revolutionized tasks such as text generation, translation, and sentiment analysis, driving transformative changes across industries from healthcare and finance to education and entertainment. However, the explosion of diverse foundation models presents a significant challenge for users. Balancing trade-offs between cost, latency, and performance to identify the best model for a specific use case remains complex and often technical.

Traditional methods for selecting an LLM rely on manual testing or predefined benchmarks such as GSM8K for math problems or MMLU for reasoning tasks. These approaches are limited in scope, time-intensive, and require substantial expertise. Furthermore, current research predominantly focuses on optimizing a single model's performance or cost efficiency, neglecting opportunities to harness the growing ecosystem of LLMs. While some models excel in performance at high costs, others provide low-cost, low-latency solutions but may underperform in specific tasks. The lack of dynamic systems to navigate these trade-offs leaves users without practical, accessible tools for leveraging state-of-the-art AI systems.

Recent work in multi-LLM routing, such as *RouterBench*, has addressed some limitations by providing benchmarking platforms for evaluating routing strategies. However, these solutions still require manual configuration and testing, making them inaccessible to non-technical users and unsuitable for real-world, dynamic applications.

Building upon these advancements, our research introduces a novel, user-centric approach to multi-LLM routing by addressing key limitations of existing systems:

1. **Dynamic Routing Framework:** We develop a framework that integrates user-defined preferences for cost, latency, and performance into routing decisions. By introducing a judge model to evaluate and rank LLMs dynamically, our system eliminates reliance on manual configurations.

2. **Exploiting Diverse LLMs:** Inspired by kNN-based methods, our system leverages the output embedding space for routing decisions, demonstrating that output embeddings can effectively determine model suitability. Unlike traditional prompt-embedding methods, which require costly evaluations of all candidate models for every prompt, our approach introduces a clustering-based solution to streamline this process.

3. **Noise-Resilient Routing:** To mitigate the impact of dataset noise, we devise a method that routes based on the difference between prompt embeddings and a model's response embeddings. This technique significantly improves routing performance by capturing a model's adaptability to specific inputs.

4. **Multimodal Routing Extensions:** Beyond text-based tasks, our framework extends multi-LLM routing to multimodal domains, including video and image processing, broadening its applicability across diverse industries.

Our system represents a paradigm shift in the way LLMs are utilized, empowering users to seamlessly exploit the strengths of diverse models without technical expertise. Key contributions include:

- Automation of benchmarking and adaptive updates to accommodate the evolving LLM landscape.
- Efficient exploitation of cost-performance-latency trade-offs in real-time.
- Democratization of access to advanced AI capabilities, enabling users across industries to efficiently leverage multi-LLM systems for a wide range of applications.

By integrating dynamic, adaptive routing mechanisms with innovations in embedding-based evaluation and clustering techniques, our research paves the way for a new era of efficient, user-driven AI systems.

2 Literature Survey

- **RouterBench: A Benchmark for Multi-LLM Routing Systems**
  Hu et al. (2024) introduced *RouterBench* as the first standardized benchmarking framework for evaluating routing strategies across diverse LLMs. With over 405,000 inference samples, *RouterBench* highlights the trade-offs between cost and performance, setting the stage for our project to incorporate user-defined preferences into routing decisions.
- **LLM-as-a-Judge with MT-Bench and Chatbot Arena**
  Zheng et al. (2023) proposed the innovative LLM-as-a-Judge model, where LLMs themselves assess output quality, achieving over 80% agreement with human evaluators. Our project integrates a similar judge model to iteratively refine routing decisions by aligning automated rankings with human expectations.
- **FrugalGPT: Cost-Efficient Use of Large Language Models**
  Chen et al. (2023) demonstrated a sequential routing strategy that progressively queries models until a quality threshold is met. This cost-sensitive approach aligns with our goal of balancing budget and performance, enabling dynamic prioritization of user-defined criteria.
- **Blending Is All You Need: Efficient Alternatives to Large LLMs**
  Lu et al. (2024) explored cost-effective alternatives by blending smaller models to achieve performance comparable to larger ones. Incorporating this concept into our routing system allows for flexibility in selecting models of varying sizes and computational demands.

These works collectively inform the design and methodology of our project, which aims to advance the state of multi-LLM routing systems by emphasizing adaptability, multimodal capabilities, and user-centric optimization.

3 Methodology

In the *RouterBench* paper, the authors perform kNN using prompt embeddings. At evaluation time, the best model for a given prompt is found using the best model for the embeddings of k closest prompts in the training data. Their paper and our report use the Mini-LM-l6-v2 encoder to generate embeddings from text.

3.1 Dataset Preparation

We preprocessed and enhanced the *RouterBench* dataset to align it with our routing experiments. First, models that were no longer publicly available (e.g., claude-instant-v1, Yi-34B) were removed, and the following models were selected:

- anthropic/claude-2
- openai/gpt-3.5-turbo-1106
- openai/gpt-4-1106-preview
- mistralai/mistral-7b-instruct
- mistralai/mixtral-8x7b-instruct

Prompts were restricted to English and those where no candidate model generated a correct response were excluded, ensuring 100% accuracy represented the system's best performance. The dataset was enriched with MiniLM-generated embeddings for prompts and responses, along with task descriptions from GPT-3.5-turbo. Filters were applied to remove noisy or irrelevant task descriptions.

FLAN model responses, LMSYS embeddings, and embedding differences (response embeddings minus prompt embeddings) were added to encode task-specific information. K-Means clustering grouped similar embeddings to reduce evaluation

costs, leaving a refined dataset of 2,404 prompts with robust features for experiments like kNN, clustering, and task-aware routing.

Additionally, we created a separate dataset of 1,000 prompts to evaluate router performance based on user preferences for latency, price, and accuracy. Accuracy was measured using the *Judge-win-rate* (decisions by GPT-4 as a judge) and the *Human-win-rate* (user preferences). The prompts, in JSON format, spanned modalities like text, image, audio, and video, with categories such as summarization, text generation, and translation. Based on user preferences and judge scores, the router system selected the best LLM model to return efficient answers tailored to user needs.

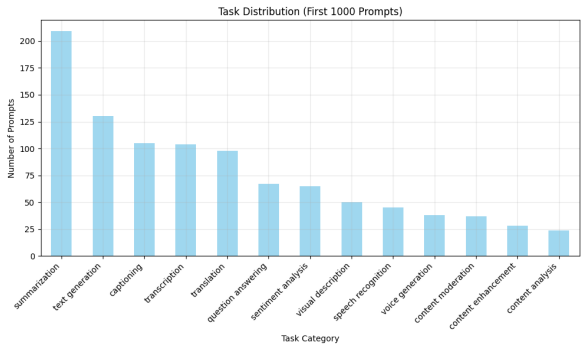

Figure 1: Distribution of Router System Dataset

3.2 kNN

We began by implementing kNN routing on our refined dataset to evaluate its performance across both input and output embedding spaces. In the initial experiment, we used prompt embeddings generated by the Mini-LM encoder and performed kNN by identifying the top-k most similar prompts from the training data, measured using cosine similarity. For a given prompt in the validation set, we predicted the best model based on the most frequently selected best model among the top-k similar prompts. This experiment relied on prompt embeddings to evaluate routing accuracy and explore how well input-space embeddings could inform routing decisions. Next, we extended the experiment to assess whether effective routing could also be performed using the embeddings in the output space, as opposed to the input space. For this, embeddings for the responses generated by LLMs were used for kNN. For each validation prompt, we compared the cosine similarity of its generated response embeddings with those of the training responses to predict the most frequent best model among the top-k similar responses. This allowed us to evaluate whether routing accuracy was consistent across input and output embedding spaces.

In subsequent experiments, we further refined the kNN approach by utilizing response embeddings specific to different models for each prompt. For a validation prompt, we measured cosine similarity between the embeddings of the responses generated by each candidate model and the training responses for the top-k similar prompts. This provided additional insights into the consistency and robustness of output-space-based routing.

To address potential noise in embeddings, we also introduced the concept of embedding difference. This method involved computing the difference between the embeddings of the prompt and the corresponding response. By performing kNN on these embedding differences, we encoded task-specific information and observed improvements in routing performance.

3.3 K-Means

We implemented a novel routing method based on clustering to address the computational cost and inflexibility of kNN. kNN requires evaluating every candidate model on every training prompt, which is computationally expensive and challenging when introducing new models or prompts. To overcome this, we used K-Means clustering, generating N clusters in the desired embedding space (e.g., prompt or response embeddings). Model performance is evaluated only on the prompt closest to each cluster center, representing the entire cluster. This reduces computational costs, with savings approximated as N/D, where D is the total number of prompts.

Clustering experiments were conducted on prompt and response embeddings generated by models such as FLAN-t5-small, GPT-4, and Mistral. Additionally, we explored clustering based on embedding differences, which encode task-specific information by calculating the difference between response and prompt embeddings. Model performance was evaluated using the closest prompt to each cluster center, demonstrating the efficiency and adaptability of this approach.

3.4 Embedding Diff

Routing using embeddings, whether from input (prompt) or output (response), can introduce noise, reducing the clarity of model selection. For example, prompts testing different tasks may have simi-

lar input embeddings, while output embeddings in cases like multiple-choice answers (e.g., `A`, `B`) can lack discriminative power.

To address this, we introduced *embedding diff*, which captures task-specific information by computing the difference between response and prompt embeddings. This method encodes the relationship between input and output spaces, improving task distinction.

We applied embedding diff in kNN and K-Means routing experiments. For kNN, cosine similarity between embedding diff vectors identified the best model from the top-k neighbors. In K-Means clustering, embedding diff vectors formed N clusters, with model performance evaluated based on the cluster centers. This approach reduced noise in routing decisions and improved accuracy.

3.5 Task Description Embeddings

Routing based on *task-aware embeddings* focuses on encoding the underlying task represented by a prompt, enabling routing systems to evaluate model suitability based on task relevance. To generate task descriptions for prompts in *RouterBench*, we used `gpt-3.5-turbo` to summarize the goal of each prompt. However, the model occasionally introduced noise by answering prompts instead of describing tasks, especially for ambiguous or context-lacking inputs. To mitigate this, we applied filters to remove short descriptions and those containing irrelevant keywords, improving the quality of the task descriptions.

These refined task descriptions were used to generate embeddings for routing experiments. In kNN routing, cosine similarity between task description embeddings was used to predict the best model based on top-k neighbors. In K-Means clustering, embeddings were grouped into N clusters, with model performance evaluated based on cluster centers. By leveraging task-aware embeddings, these methods provided a more nuanced and task-specific approach to routing.

3.6 Router System

The first task is designed to assist users in selecting the best model for their specific use case based on task requirements, user preferences, and precomputed performance metrics. The user is allowed to specify a modality (e.g., text, audio, or video), a task category (e.g., translation or sentiment analysis), and their preferences for price, latency, and accuracy. These inputs are matched against a dataset of average model performances to filter and rank available models. Each model is scored based on how well it meets the user's criteria, considering factors such as price, latency, and human or judge win rates. The model with the highest average score is suggested as the best fit for the task. Additionally, the selected model is queried with the user's input prompt, and the resulting response is displayed, providing immediate feedback.

The second task focuses on systematically benchmarking models for various tasks and recording their performance. A predefined dataset of tasks is processed, each with attributes such as modality, category, and user-defined preferences for price, latency, and accuracy. For each task, evaluation is done on all candidate models, calculating metrics such as the cost of processing the task, response latency, and performance in terms of judge and human win rates. The metrics are derived by comparing the model's output to a reference model using a judge model. The results are recorded for each model and task combination, updating a CSV file in real time to ensure data persistence. These tasks collectively enable dynamic model selection and systematic performance evaluation, forming a robust foundation for routing and benchmarking in real-world scenarios.

4 Experimental Setup

4.1 Router System

The models used are assigned an arbitrary cost and the API calls are performed through the openrouter platform which has access to multiple APIs.

Listing 1: Arbitrary Costs

```
REFERENCE_MODEL = Model(
    name="openai/gpt-3.5-turbo",
    input_cost=0.5 / M_TOKENS,
    output_cost=1.5 / M_TOKENS,
)

JUDGE_MODEL = Model(
    name="openai/gpt-4",
    input_cost=30 / M_TOKENS,
    output_cost=60 / M_TOKENS,
)

MODELS = [
    Model(
        name="anthropic/claude-2",
        input_cost=8.0 / M_TOKENS,
        output_cost=24.0 / M_TOKENS,
    ),
    Model(
        name="openai/gpt-3.5-turbo-1106",
        input_cost=1.0 / M_TOKENS,
        output_cost=2.0 / M_TOKENS,
    ),
    Model(
        name="openai/gpt-4-1106-preview",
        input_cost=10.0 / M_TOKENS,
        output_cost=30.0 / M_TOKENS,
    ),
    Model(
        name="mistralai/mistral-7b-instruct",
```

```
        input_cost=0.055 / M_TOKENS,
        output_cost=0.055 / M_TOKENS,
    ),
    Model(
        name="mistralai/mixtral-8x7b-instruct",
        input_cost=0.24 / M_TOKENS,
        output_cost=0.24 / M_TOKENS,
    ),
]
```

5 Results

5.1 kNN

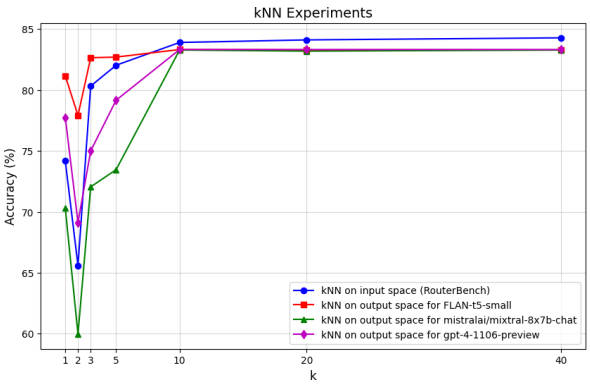

Figure 2: kNN Experiments

It was observed that the routing accuracy using response embeddings (output space) was comparable to that of prompt embeddings (input space). This result highlighted that the output space provides an effective and alternative representation for routing decisions. Additionally, our experiments showed that the size of the LLM generating the response embeddings (e.g., FLAN-t5, Mistral, GPT-4) did not significantly affect the routing accuracy, suggesting that smaller models can also produce embeddings sufficient for effective routing.

Overall, these experiments demonstrated that effective kNN routing can be performed using both input and output embeddings, with comparable accuracy. The findings indicate that embeddings in the output space, even when generated by smaller models, can serve as a reliable basis for routing decisions. These results highlight the versatility of kNN routing methods and their adaptability to different embedding spaces and model configurations.

5.2 K-Means

The results of these experiments showed that clustering performance improved with the size of the embedding model used, which aligns with our expectations. For example, clustering on GPT-4 embeddings generally yielded better routing accuracy compared to clustering on smaller models like

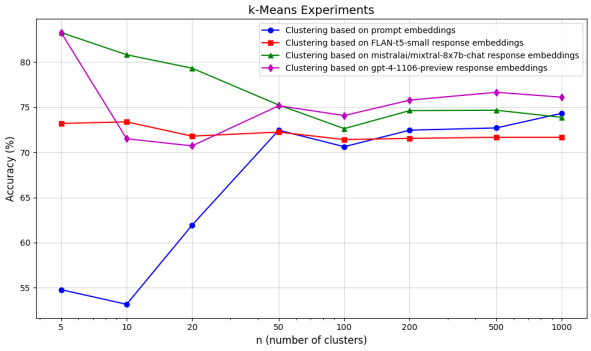

Figure 3: K-Means Experiments

FLAN-t5. However, we observed that the clustering approach, while cost-effective and flexible, achieved slightly lower performance compared to kNN. Specifically, clustering with 1000 clusters on prompt embeddings achieved a routing accuracy of 74.33%, which was approximately 9 percentage points lower than the 83.32% accuracy obtained using kNN on the same embedding data. This trade-off between performance and cost reflects the inherent limitations of clustering compared to kNN, where clustering sacrifices some accuracy for significant computational savings.

Despite its lower accuracy, the clustering method demonstrated remarkable flexibility when applied to new datasets or models. Unlike kNN, which requires extensive evaluations on all training prompts, clustering generalizes better to unseen data since it relies on representative clusters rather than exhaustive model evaluations. This property makes clustering particularly attractive for real-world scenarios where the dataset or the candidate model pool may frequently change. Moreover, the computational efficiency of clustering allowed us to experiment with different embedding spaces and advanced techniques, such as embedding differences, to further refine routing accuracy.

We also observed that increasing the number of clusters improved routing performance to an extent, as it allowed for finer granularity in capturing embedding space structure. However, beyond a certain point, diminishing returns were observed, with marginal improvements in accuracy as the number of clusters increased. While K-Means clustering provided a cost-efficient alternative to kNN, we recognize that more advanced clustering techniques or hybrid approaches may further bridge the gap in accuracy, and we leave these explorations for future work.

In summary, clustering offered a cost-effective

and flexible alternative to kNN for routing, with the ability to generalize well to new datasets or models. Although clustering performance lagged behind kNN by a few percentage points, the significant computational savings and adaptability make it a promising approach for scalable routing systems. Future work may focus on enhancing the clustering algorithm and exploring task-specific embedding techniques to further improve performance while maintaining cost efficiency.

5.3 Embedding Diff

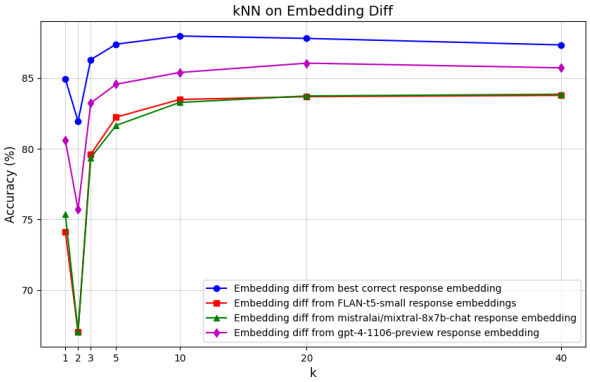

Figure 4: kNN on Embedding Diff

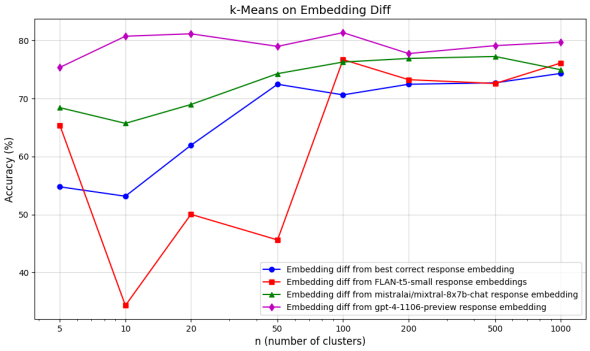

Figure 5: K-Means on Embedding Diff

Our experiments showed that embedding diff enhanced the performance of both kNN and K-Means routing. For kNN, the embedding diff method provided better differentiation between tasks with similar inputs but distinct outputs. For K-Means, the clustering of embedding diff vectors allowed for more precise grouping of related tasks, which translated into improved routing accuracy. These results, visualized in Figure 5, demonstrate that embedding diff offers a novel and effective way to encode task-specific information for routing.

We hypothesize that the success of the embedding diff method lies in its ability to capture the transformation performed by the model to convert an input into an output. This transformation encodes task-specific details that are often lost in direct input or output embeddings. By incorporating this intermediate representation, embedding diff provides a robust alternative to traditional embedding-based routing methods.

In conclusion, the embedding diff method represents a significant advancement in routing systems. It reduces noise in embedding spaces and enhances the precision of routing decisions for tasks with overlapping or non-discriminative embeddings. Future work could explore extending embedding diff to multimodal tasks and further optimizing the clustering and similarity metrics used in this method.

5.4 Task Description Embeddings

The results of these experiments are illustrated in Figure 6 and Figure 7. For kNN, task-aware routing achieved competitive accuracy, demonstrating the potential of task description embeddings to improve model selection. Similarly, K-Means clustering on task-aware embeddings showed promising results, with improved generalization across prompts. However, the performance was slightly lower compared to direct input or output embeddings due to the residual noise in the task descriptions and the limited fine-tuning of the LLM.

We hypothesize that the effectiveness of task-aware embeddings lies in their ability to encode high-level semantic information about the prompt, which is often overlooked in traditional embedding approaches. This method provides a promising avenue for future exploration, especially with the use of fine-tuned LLMs and optimized instructions for task description generation. By reducing noise and improving the fidelity of task descriptions, task-aware embeddings could become a critical component of routing systems for complex, real-world applications.

5.5 Router System

1. **Mistralai/Mixtral-8x7b-instruct - Judge vs Human Win Rate by Task Category:** This graph compares the Judge Win Rate and Human Win Rate for the model `mistralai/mixtral-8x7b-instruct` across various task categories. The Judge Win Rate, determined by an automated evaluation system, reflects how often the model's output was deemed better than a baseline, while the Human Win Rate indicates user

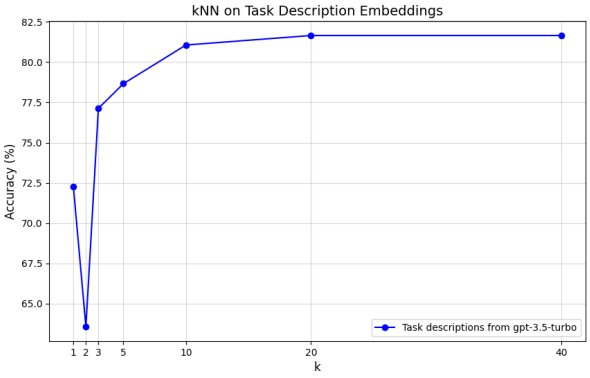

Figure 6: kNN on Task Description Embeddings

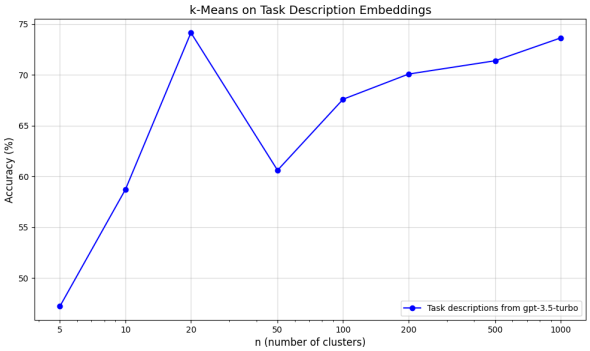

Figure 7: K-Means on Task Description Embeddings

preferences for the model's output. For tasks such as *text generation* and *transcription*, the model performs exceptionally well in both metrics, with slightly higher human win rates. Conversely, categories like *speech recognition* and *summarization* show a clear disparity, with the Judge Win Rate lagging behind human evaluations, suggesting a preference for the model's qualitative performance in these tasks. This indicates that the model aligns well with user expectations in high-performance tasks but requires improvement in lower-performing areas.

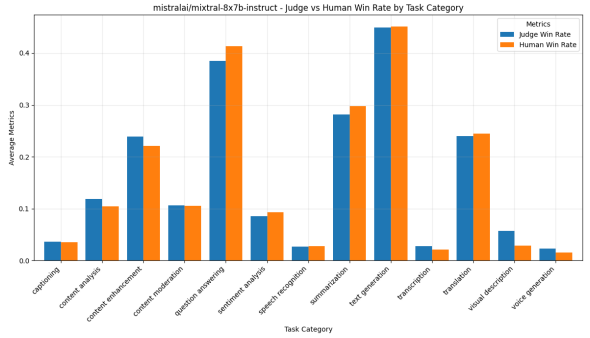

Figure 8: mistralaimixtral-8x7b-instruct - Judge vs Human Win Rate by Task Category

2. **OpenAI/GPT-3.5-turbo-1106 - Judge vs Human Win Rate by Task Category:** Figure 9 illustrates the performance of `openai/gpt-3.5-turbo-1106` across task categories, with metrics again split between Judge Win Rate and Human Win Rate. For categories such as *text generation* and *translation*, the Judge and Human Win Rates are almost identical, indicating that both automated systems and human users recognize the model's strong performance. However, in tasks like *content enhancement* and *content moderation*, the Human Win Rate slightly exceeds the Judge Win Rate, suggesting that users appreciate nuanced outputs more than what automated evaluation captures. This demonstrates GPT-3.5-turbo's broad applicability while highlighting areas for refinement in specific task alignments.

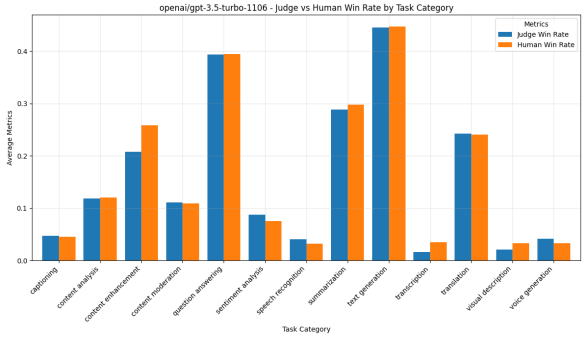

Figure 9: openaigpt-3.5-turbo-1106 - Judge vs Human Win Rate by Task Category

3. **Anthropic/Claude-2 - Judge vs Human Win Rate by Task Category:** Figure 10 depicts the performance of `anthropic/claude-2` across various task categories. Unlike other models, Claude-2 demonstrates a consistently higher Human Win Rate across most tasks, including *question answering*, *text generation*, and *translation*. This highlights its ability to produce human-preferred outputs, particularly in language-intensive tasks. However, for technical or structured tasks like *speech recognition*, the Judge Win Rate is comparatively higher, indicating that Claude-2's outputs may be more consistent or precise in scenarios where objective measures are dominant. This duality underlines Claude-2's potential for tasks requiring either creative flexibility or structural rigor.

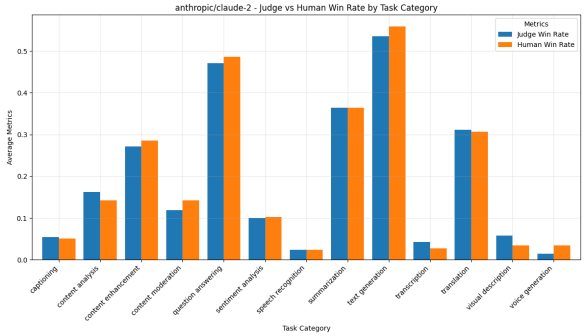

Figure 10: anthropicclaude-2 - Judge vs Human Win Rate by Task Category

4. **Model Evaluation - Judge Win Rate vs Human Win Rate:** Figure 11 compares the overall Judge Win Rate and Human Win Rate across three models: `anthropic/claude-2`, `openai/gpt-3.5-turbo-1106`, and `mistralai/mixtral-8x7b-instruct`. All models perform similarly in terms of Judge Win Rate, with minor variations, but notable differences appear in Human Win Rate. Claude-2 exhibits a marginally higher Human Win Rate compared to the others, reflecting a better alignment with user preferences. GPT-3.5-turbo follows closely, while mistralai/mixtral-8x7b trails slightly, suggesting it is less attuned to user expectations. This graph highlights the trade-offs between automated evaluations and human preferences across models.

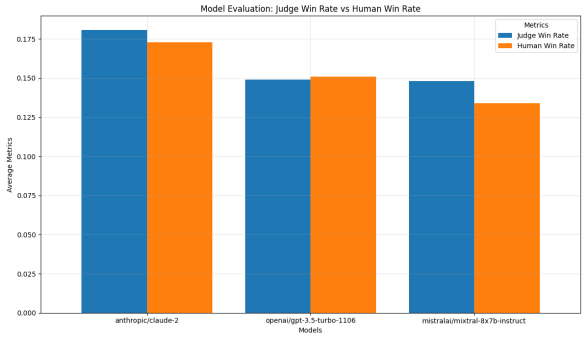

Figure 11: Judge Win Rate vs Human Win Rate

5. **Human Win Rates by Task Category for All Models:** Figure 12 compares the Human Win Rates of all three models (`Claude-2`, `GPT-3.5-turbo`, and `Mixtral-8x7b`) across task categories. Claude-2 demonstrates consistently higher Human Win Rates across tasks like *text generation*, *translation*, and *question answering*, indicating its strong user preference for language-driven tasks. GPT-3.5-turbo performs closely in most categories, especially in structured tasks like *content enhancement* and *content moderation*. Mixtral-8x7b, while competitive, has lower Human Win Rates in categories like *summarization* and *translation*, suggesting that its outputs may lack the refinement or creativity appreciated by users.

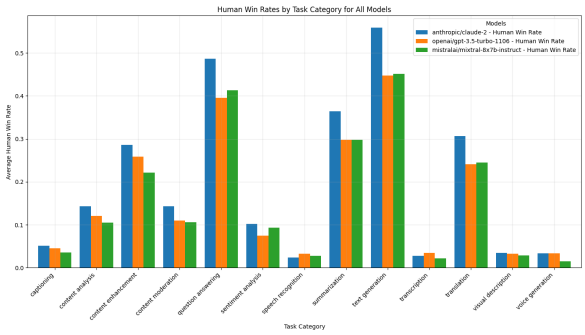

Figure 12: Human Win Rates by Task Category for All Models

6. **Judge Win Rates by Task Category for All Models:** Figure 13 compares the Judge Win Rates of the same three models across all task categories. GPT-3.5-turbo and Claude-2 dominate in categories such as *text generation* and *translation*, with minimal differences. However, Mixtral-8x7b shows slightly lower Judge Win Rates across most categories, particularly in creative or complex tasks like *summarization* and *content enhancement*. Interestingly, all models exhibit relatively low Judge Win Rates in *voice generation* and *visual description*, highlighting these as areas where significant improvement is needed across the board.

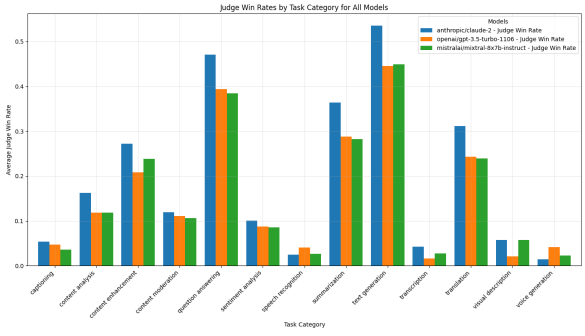

Figure 13: Judge win Rates by Task Category for All Models

6 Conclusion

One result of our work is that routing can be performed using embeddings from the prompt space, the model output space, or using the difference between the two, where the last is the most performant. We showed that k-Means clustering grants significant cost savings as compared to kNN methods but offers somewhat lower performance. Finally, we found that by asking an LLM to describe the task required by a prompt, we were able to perform routing using the embedding for that task description.

Building on the findings from this work, several promising directions emerge for future exploration:

- **Advanced Clustering Techniques:** Explore more sophisticated clustering algorithms, such as hierarchical or density-based clustering, to improve the balance between cost savings and performance.
- **Optimization of Task Descriptions:** Develop fine-tuned LLMs explicitly trained to generate accurate task descriptions, improving the quality and utility of task-aware embeddings for routing.
- **Incorporating Multi-Objective Optimization:** Implement multi-objective optimization techniques to balance cost, accuracy, and latency holistically, providing adaptive solutions for real-world constraints.
- **Dynamic Model Integration:** Introduce mechanisms for dynamically integrating and evaluating new models to ensure adaptability to evolving LLM capabilities.
- **Task-Specific Pretraining for Embedding Diff:** Design pretraining stages focused on optimizing embedding differences for routing-specific tasks, further enhancing performance.
- **Scalability to Larger Datasets and Diverse Languages:** Extend the system to support larger datasets and multilingual capabilities, broadening its applicability across global contexts.
- **Real-Time Adaptive Routing:** Implement real-time routing mechanisms to adapt to dynamic workloads, current system resources, and user preferences.
- **Evaluation Across Real-World Applications:** Validate the system's utility by testing it in real-world scenarios, such as customer service chatbots or domain-specific content generation tasks.

By addressing these areas, the proposed routing methods could evolve into a robust and versatile framework for leveraging diverse LLMs effectively.

7 Contributions

The majority of this project was a collaborative endeavor. Although each component was executed independently, all participants assisted one another in resolving issues. Consequently, all three group members merit equal recognition.

Kincannon Wilson originally presented the project concept. He contributed significantly to the kNN and K-Means techniques, as well as to the Embedding Diff and Task Description Embeddings. His contributions to code design and debugging were highly beneficial. He also developed the Router API system, which was essential for the router infrastructure.

Abhijnya Menakur conducted the literature review. She also conducted the dataset preparation and was instrumental in building the router system. She partially designed the kNN and K-Means code and was responsible for establishing the running environments. She executed the visualization of results and significantly contributed to the report. She produced the PowerPoint presentation and was active in managing the GitHub website as well.

Kumaresh Suresh Babu undertook the reimplementation of the current code. He made substantial contributions to the Embedding Diff and Task Description Embeddings. He primarily focused on enhancing the Router System to improve its user-friendliness and dynamism. He also engaged in dataset manipulation through scripting and was primarily accountable for the report and GitHub repository.

8 References

- Hu et al. (2024). *RouterBench: A Benchmark for Multi-LLM Routing Systems*.
- Zheng et al. (2023). *LLM-as-a-Judge with MT-Bench and Chatbot Arena*.
- Chen et al. (2023). *FrugalGPT: Cost-Efficient Use of Large Language Models*.

- Lu et al. (2024). *Blending Is All You Need: Efficient Alternatives to Large LLMs*.
- Vaswani et al. (2017). *Attention Is All You Need*.
- Reimers and Gurevych (2019). *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*.
- Devlin et al. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*.
- Johnson et al. (2019). *Efficient Processing of Embeddings with FAISS*.
- Wang et al. (2020). *MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-trained Transformers*.
- Paszke et al. (2019). *PyTorch Documentation*.