FASTER, CHEAPER, JUST AS GOOD: COST- AND LATENCY-CONSTRAINED ROUTING FOR LLMS

Javid Lakha[*]
Harvard University

Minlan Yu
Harvard University

Rana Shahout
Harvard University

ABSTRACT

Large Language Models (LLMs) span a broad spectrum of sizes, each presenting distinct trade-offs in cost, latency, and performance that complicate large-scale deployment. Although larger models often provide higher-quality responses for complex prompts, their increased computational overhead and slower inference can degrade user experience in real-time applications. Meanwhile, AI development is moving toward compound AI systems integrating multiple LLMs of different sizes. In such environments, deciding when to invoke smaller or larger models becomes critical, especially under shifting system loads, as we must balance high output quality, tight cost budgets, and acceptable response times. We propose SCORE, a routing system that maximizes response quality while respecting user-specified cost and latency constraints. For each incoming request, SCORE predicts each model's response quality and length and selects the option that best meets current cost, latency, and quality requirements – whether a less expensive model with a lighter load or a more resource-intensive model for complex prompts. By continually adapting these decisions as requests arrive, SCORE balances the system load, enforces budget limits, and maintains user satisfaction through timely, cost-effective, and accurate responses.

1 INTRODUCTION

Recent advances in large language models (LLMs) have driven the development of interactive AI applications. OpenAI's ChatGPT (OpenAI, 2022) exemplifies this trend by supporting conversational interactions across diverse tasks such as language translation, complex question answering, and structured text generation. These tasks rely on understanding and reasoning over extended inputs. Moreover, AI development is moving toward compound AI systems (Zaharia et al., 2024) that integrate multiple components, often LLMs of varying sizes and capabilities.

Model size significantly impacts runtime, cost, and answer quality in transformer-based LLMs. Smaller models often suffice for more straightforward prompts, while larger, more resource-intensive models provide deeper reasoning for complex prompts. Routing all prompts to a single large model can lead to high cost, load, and increased latency, which degrades user experience and restricts real-time applications. For instance, on two NVIDIA H100 GPU, Llama 3.2 Instruct (1B) generates 100 tokens in 0.2 seconds, whereas Llama 3.2 Instruct (70B) requires 3 seconds, reflecting its greater computational complexity. Serving large models in production is thus expensive and slow. As an example, Amazon Bedrock charges \$0.10 per million tokens for Llama 3.2 Instruct (1B) compared to \$0.72 for Llama 3.1 Instruct (70B), illustrating a significant cost disparity.[1]

Recent work (Ong et al., 2024) explores four routing approaches (similarity-weighted matching to training set labels, matrix factorization, a BERT-based classifier, or a call to Llama 3.1 Instruct (8B)) that estimate the quality gains from using larger models. However, their router minimizes cost while meeting fixed performance thresholds and neglects latency – a critical factor in interactive LLM applications. The routing threshold is also static, preventing adaptation to fluctuating costs or system load during inference.

[*]Correspondence to `javidlakha@g.harvard.edu`.
[1]Amazon Bedrock pricing is for the `us-east-1` (North Virginia) region, as of 29 January 2025.

In this paper, we introduce SCORE (SMART COST AND LATENCY OPTIMIZED ROUTER), a routing algorithm designed to maximize response quality under user-specified cost and latency constraints. We define cost as the computational expense of processing a prompt (determined by model size and response length) and latency as the total response time, including queuing delays and model runtime. Our approach operates online and adapts to each model's current load and the overall cost budget.

Our approach leverages the insight that not every response requires a larger, more expensive model. For example, both Llama 3.2 Instruct (1B) and Llama 3.1 Instruct (70B) give correct answers to the prompt *"Which scientific treatise was published in 1687?"*, yet the larger model incurs seven times the cost and 15 times the latency. Conversely, for a more complex prompt such as *"Prove that no positive integers a, b, and c satisfy the equation $a^n + b^n = c^n$ for any integer value of n greater than 2"* the smaller model hallucinates a flawed proof. By contrast, the larger model correctly recognizes this as Fermat's Last Theorem and provides a valid proof sketch. These examples highlight the potential for cost and latency optimization by assigning each prompt to the cheapest model that can answer it well.

We build on this observation by routing prompts to models of varying strengths based on predicted quality, cost, and latency. A cheaper model may not always be faster, depending on factors such as architecture or deployment parallelism. Our system accommodates multiple LLMs of different sizes and capabilities; for illustration, we focus on a two-model system with Llama 3.2 Instruct (1B) for more straightforward tasks and Llama 3.1 Instruct (70B) for more complex prompts (Grattafiori et al., 2024). However, our method generalizes to any number of LLMs. We formulate the routing decision as a constrained optimization problem under incomplete information. We aim to maximize response quality while adhering to cost and latency constraints. Because actual quality, cost, and latency are unknown at decision time, the router will need to predict these to guide model selection, adapting its decisions as new prompts arrive and resource usage evolves.

2 PROBLEM FORMULATION

Deploying LLMs in production requires balancing response quality, computational cost, and latency. Response quality measures how accurately and comprehensively a model addresses a prompt, often evaluated using predefined scoring metrics. Computational cost is proportional to the size of the model and the length of the response, scaled by the model's cost per token. Latency, on the other hand, includes both the processing time—determined by the model's architecture and response length—and the waiting time caused by system load. Importantly, latency is not solely a function of model size; under high load, even smaller models can exhibit significant delays. Our goal is to incorporate load balancing into the router decision process.

Definitions. Let $\{M_i\}$ denote a set of LLMs that vary in size, inference cost and speed. We define $\mathcal{P}$ to be the set of possible prompts; $\mathcal{W}$ to be the set of all model parameters; and $\mathcal{R}$ to be the set of possible responses. Each model M_i maps a prompt and parameters to a response, $M_i : \mathcal{P} \times \mathcal{W} \to \mathcal{R}$.

At time t, for a given prompt $p_t \in \mathcal{P}$, we denote $S(p_t)$ to be the size (i.e. length) of prompt p_t, measured in tokens; $M_i(p_t)$ to be the response from model M_i and $S(M_i(p_t))$ its size; and $Q(M_i(p_t))$ to be the quality score of the response.

Cost and Latency Modeling. The computational cost of processing a prompt with model M_i is primarily influenced by the size of the generated response due to the autoregressive nature of LLMs. We approximate the cost as $C_{M_i}(p_t, M_i(p_t)) = c_{M_i} \cdot S(M_i(p_t))$, where c_{M_i} is the cost per token for model M_i. The latency, on the other hand, depends on both the waiting time for the chosen model at time t, $W_{M_i}(t)$, and the runtime, which is the product of the response size $S(M_i(p_t))$ and the model's per-token processing speed s_{M_i}, $L_{M_i}(M_i(p_t), t) = W_{M_i}(t) + s_{M_i} \cdot S(M_i(p_t))$.

User Policy. The user specifies a policy that includes maximum allowable total cost C_{max} and maximum allowable latency per prompt L_{max}. Additionally, the user provides weights w_C and w_L to prioritize between cost and latency constraints, respectively. The ratio $w_C : w_L$ determines the relative importance of cost versus latency in the optimization process. We seek to maximize the quality of the response to each prompt, subject to the constraints imposed by the user policy.

Predicting Response Quality and Length. Since the actual response qualities $Q(M_i(p_t))$ and response sizes $S_{M_i}(M_i(p_t))$ are unknown prior to processing, we plan to employ predictive models $\hat{Q}_{M_i}$ and $\hat{S}_{M_i}$ for each LLM M_i. In our experimental setup, $\hat{Q}_{M_i}(p_t)$ is scaled between 1 (Very Poor) and 5 (Excellent), and $\hat{S}_{M_i}(p_t)$ ranges from 1 to 512 tokens.

Objective Function. Our objective is to maximize the cumulative predicted quality of responses over all incoming prompts while considering the user-specified cost and latency preferences. Let $t \in [T]$ denote each time period. We formulate the optimization problem as:

$$\max_{d_t} \quad \hat{Q}_{d_t}(p_t) \quad \text{subject to} \quad \sum_{r=1}^{t} c_{d_r} \cdot \hat{S}_{d_r}(p_r) \le C_{\max} \cdot \frac{t}{T} \quad \text{and} \quad W_{d_t}(t) + s_{d_t} \cdot \hat{S}_{d_t}(p_t) \le L_{\max}.$$

Here, $d_t \in \{M_i\}$ is the routing decision: it indicates the model selected to process prompt p_t. The weights w_C and w_L penalize the cost and latency contributions, balancing the trade-offs according to user preferences.

Optimization Approach. To solve the optimization problem, we incorporate the weighted penalties into the objective function using a Lagrangian multiplier λ:

$$\max_{d_t} \quad \hat{Q}_{d_t}(p_t) - \lambda \left(w_C \left(\sum_{r=1}^{t} c_{d_r} \cdot \hat{S}_{d_r}(p_r) - C_{\max} \cdot \frac{t}{T} \right) + w_L \left(W_{d_t}(t) + s_{d_t} \cdot \hat{S}_{d_t}(p_t) - L_{\max} \right) \right).$$

This formulation allows us to adjust λ to enforce the cost and latency constraints, while w_C and w_L determine the relative importance of cost and latency in the optimization.

Routing Decision Function. For each prompt p_t, the routing decision d_t is made by selecting the model that maximizes the net benefit:

$$d_t = \arg\max_{M_i} \left\{ \hat{Q}_i(p_t) - \lambda \Big(w_C \cdot c_{M_i} \cdot \hat{S}_{M_i}(p_t) + w_L \cdot W_{M_i}(t) + w_L \cdot s_{M_i} \hat{S}_{M_i}(p_t) \Big) \right\}.$$

This decision rule accounts for the predicted quality, cost, and latency, weighted according to the user-specified preferences and the Lagrange multiplier λ.

We designed our system to host LLMs of varying sizes across multiple GPUs and to route prompts in a manner that maximizes predicted response quality while adhering to user-specified cost and latency policies. Figure 2 illustrates the architecture in a two-model setting. Our design integrates a router that directs incoming prompts to the appropriate LLM based on predicted response quality, cost, and latency. We use response quality predictors $\{\hat{Q}_{M_i}\}$ to estimate the quality of responses from each model and response length predictors $\{\hat{S}_{M_i}\}$ to predict output token counts, which inform cost and latency estimations. The backend comprises the LLMs $\{M_i\}$, which in our experimental setup include Llama 3.2 Instruct (1B) and Llama 3.1 Instruct (70B).

Response Quality and Length Prediction. We are working on predicting response quality $\hat{Q}_{M_i}(p)$ and length $\hat{S}_{M_i}(p)$. For quality prediction, we intend to build on the methods of Ong et al. (2024), while for length prediction, we plan to adopt the DistilBERT-based bucketing approach from Jin et al. (2023), avoiding the computational cost of TRAIL (Shahout et al., 2024), as this approach requires a call to each of the target models. To train our predictors, we have built a dataset of 100,000 examples sampled from ELI5 (Fan et al., 2019), Natural Questions (Kwiatkowski et al., 2019), and SQuAD (Rajpurkar et al., 2016), with quality scores generated using GPT-4o-mini for scalability. Appendix B includes details.

3 Preliminary Results

Experimental Set-Up. We evaluated SCORE with two LLMs: Llama 3.2 Instruct (1B) and Llama 3.1 Instruct (70B) (Grattafiori et al., 2024). The 70B model provides higher-quality responses at the expense of latency and computational cost, while the 1B model trades lower quality for speed and efficiency. We evaluated how SCORE routes prompts between models to meet user-specified cost and latency constraints. We conducted experiments on three NVIDIA H100 GPUs (80 GB each).

Two GPUs ran Llama 3.1 Instruct (70B) to support its size, input queue, and key-value cache. The third GPU hosted Llama 3.2 Instruct (1B) and managed its queue and cache, as well as the response length and quality predictors for both models. We served the models using vLLM (Kwon et al., 2023), which batches requests and caches attention keys and values. We tested SCORE on 100 prompts sampled from ELI5 (Fan et al., 2019), Natural Questions (Kwiatkowski et al., 2019), and SQuAD (Rajpurkar et al., 2016). We pre-scored each response and set the temperature to 0 to ensure reproducibility. Prompt arrivals followed a Poisson process with a rate of one per second, and we set the cost and latency weights (w_C and w_L) to 1. Our baseline router considers only latency and cost, routing prompts without incorporating response quality.

Results. In our first set of experiments, we fixed the cost constraint and varied the latency constraint. In our second set of experiments, we fixed the latency constraint and varied the cost constraint. We follow Ong et al. (2024) in defining the cost to be the total proportion of response tokens generated by the more powerful model. This set-up assumes that the fixed costs of hosting the model can be amortized across the per-token generation costs. As the latency constraint is only binding for Llama 3.1 Instruct (70B), we express latency as the number of tokens queued for this model (the waiting time) plus the number of tokens in the prompt's response (the runtime).

Our experiments show that, compared to the baseline, by prioritizing prompts that benefit most from the stronger model, SCORE's routing improved mean response quality while respecting latency and cost constraints. We stress that for these experiments, the router had perfect information about the response quality and length. Our next step is to predict these to high accuracy using lightweight models.

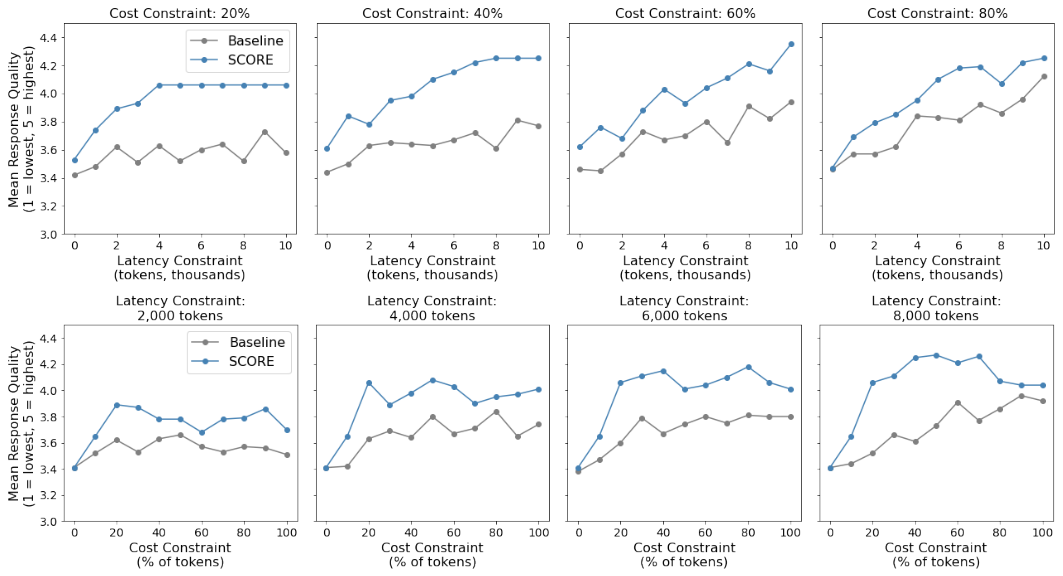

Figure 1: Mean response quality for prompts routed by the baseline router and SCORE under different latency and cost constraints, where the response quality and length predictors operate under perfect information.

4 Conclusion

We presented SCORE, a routing system that maximizes response quality while enforcing user-defined cost and latency constraints in multi-model deployments. Our approach integrates lightweight predictors for response quality and length, enabling the system to dynamically select the most appropriate LLM for each incoming query. Experimental results demonstrate that SCORE efficiently balances load, maintains acceptable response times, and adheres to strict cost budgets. In future work, we plan to implement response quality and length prediction, extend our approach to additional models, and further optimize performance under varying workload conditions.

Acknowledgments

Rana Shahout and Minlan Yu are partially supported by NSF CNS NeTS 2107078. This work was supported in part by ACE, one of the seven centers in JUMP 2.0, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

References

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. ELI5: long form question answering. In *Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL 2019)*, pp. 3558–3567. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1346. URL `https://doi.org/10.18653/v1/p19-1346`.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL `https://arxiv.org/abs/2407.21783`.

Yunho Jin, Chun-Feng Wu, David Brooks, and Gu-Yeon Wei. S3: Increasing gpu utilization during generative inference for higher throughput. *arXiv preprint arXiv:2306.06000*, 2023. URL `https://doi.org/10.48550/arXiv.2306.06000`.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*, 2019.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pp. 611–626, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702297. doi: 10.1145/3600006.3613165. URL `https://doi.org/10.1145/3600006.3613165`.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665*, 2024. URL `https://doi.org/10.48550/arXiv.2406.18665`.

OpenAI. Introducing chatgpt. `https://openai.com/blog/chatgpt`, 2022.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 1606.05250, 2016. URL `https://doi.org/10.48550/arXiv.1606.05250`.

Rana Shahout, Eran Malach, Chunwei Liu, Weifan Jiang, Minlan Yu, and Michael Mitzenmacher. Don't stop me now: Embedding based scheduling for llms. *arXiv preprint arXiv:2410.01035*, 2024. URL `https://doi.org/10.48550/arXiv.2410.01035`.

Matei Zaharia, Omar Khattab, Lingjiao Chen, Jared Quincy Davis, Heather Miller, Chris Potts, James Zou, Michael Carbin, Jonathan Frankle, Naveen Rao, and Ali Ghodsi. The shift from models to compound ai systems. `https://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems`, 2024.

A System Design

Figure 2 illustrates SCORE's architecture in the two-model setting used for our experiments. We modeled prompt arrivals to follow a Poisson process with a rate of one arrival per second, though our system is agnostic to this. The user specifies the maximum allowable total cost (the cost target C_{max}) and maximum allowable latency per prompt (the latency target L_{max}) as well as the cost weight w_C

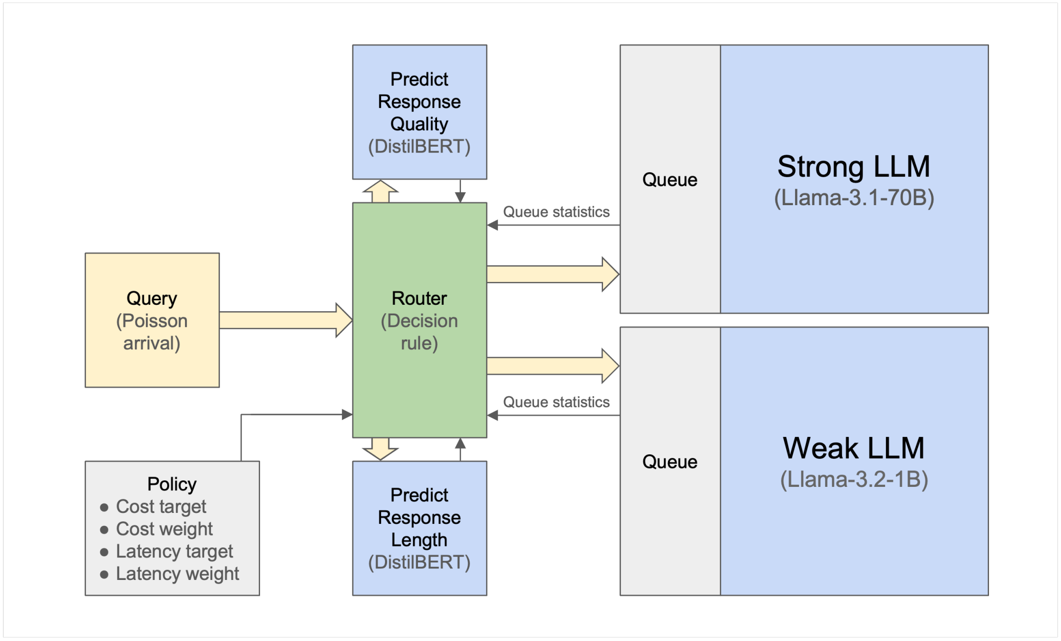

Figure 2: High-level architecture of SCORE in our two-model experimental setting.

and the latency weight w_L which determine the priority of each constraint. The response quality predictors $\{\hat{Q}_{M_i}\}$ estimate the quality of responses from each model and response length predictors $\{\hat{S}_{M_i}\}$ predict output token counts, which inform cost and latency estimations. This is a work in progress and at present our system operates under perfect information. The router directs prompts to one of the models $\{M_i\}$ – either Llama 3.2 Instruct (1B) or Llama 3.1 Instruct (70B) – based on the queue statistics for each model and the predicted response quality and length.

B RESPONSE QUALITY AND LENGTH PREDICTION

To be able to optimally route requests, we need to predict the quality $\hat{Q}_{M_i}(p)$ and the length $\hat{S}_{M_i}(p)$ of the responses from each of the candidate models M_i. In addition to being accurate, it is important that the predictors are lightweight, so that they do not add to the cost and latency of the response.

In the existing literature, Ong et al. (2024) proposes four different methods for predicting response quality in a two model situation: similarity-weighted matching to training set labels, matrix factorization, a BERT-based classifier, or a call to Llama 3.1 Instruct (8B). These models are trained on human preference data, augmented with LLM judge-labeled datasets.

Considering response length prediction, Jin et al. (2023) introduce S3, a system that uses a lightweight DistilBERT-based predictor to estimate the output token length of generated sequences for optimizing memory allocation during LLM inference. Their approach involves bucketing sequence lengths to simplify the prediction task. Notably, they observed that most mispredictions occurred in neighboring buckets. Their training accuracy on the Alpaca dataset was 98.1% and their evaluation accuracy on Natural Questions was 77.13%, suggesting that DistilBERT-based predictors are a viable strategy for token length estimation. Shahout et al. (2024) take a different approach with TRAIL (Token Response and Embedding Adaptive Inference Layer). Their strategy is to use the target LLM itself to predict the length of its response – after generating each token, the embedding is recycled as an input for a classifier that predicts the remaining length of the response. While the use of an autoregressive model may be better placed to predict the length of its response compared to a masked language model, the downside of this strategy is that the prediction requires a call to each of the target models, which we want to avoid.

We intend to train separate response quality and length predictors for each LLM in our system. We think DistilBERT (which has 66M parameters) will make a good starting point, because it offers a balance between efficiency and performance, making it suitable for real-time inference without significant computational overhead.

We built a dataset of 100,000 examples by selecting prompts at random from widely used question-answering datasets:

- **ELI5** (Fan et al., 2019): A dataset containing complex, open-ended questions from the Reddit community "Explain Like I'm Five", requiring detailed explanations.
- **Natural Questions** (Kwiatkowski et al., 2019): A collection of real user prompts from Google Search, paired with answers found in Wikipedia articles.
- **SQuAD** (Rajpurkar et al., 2016): The Stanford Question Answering Dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles, with corresponding answers.

Our intention in selecting data from these sources was to ensure a variety of query types and expected response lengths. For each query in the dataset we generated a response from each of the target LLMs in our system.

To transform the regression task of predicting exact token counts into a classification problem, we intend to follow Jin et al. (2023) and discretize the possible response lengths into equal-width buckets up to a maximum of 512 tokens. We intend to opt for a classification approach because DistilBERT performs poorly on regression tasks; initial experiments with regression yielded unsatisfactory results.

We generated quality scores for each of the responses using GPT-4o-mini, a cheaper variant of GPT-4o. Human labeling of 100,000 examples would have been infeasible due to time and cost constraints. GPT-4o-mini was chosen after a validation study confirmed its high agreement with human raters on small dataset of 100 examples. The rubric used for scoring, included in Appendix C, emphasizes clarity, accuracy, completeness, and relatability, assigning a score from 1 (Very Poor) to 5 (Excellent). This approach ensured scalability while maintaining consistency in evaluations.

C PROMPT

We used the following prompt to elicit GPT-4o-mini's assessment of a model's response quality:

```
Please evaluate the quality of the response provided by a large
language model to an input query. Rate the response on a scale
from 1 to 5 based on the criteria below.

# Input Query:
{query}

# Model Response:
{response}

# Rating Scale and Criteria

**5 (Excellent)**

- The response is exceptionally clear, concise, and complete,
fully addressing the query in a way that is both accurate and easy
to understand.

- It effectively explains complex ideas in a simple, relatable
manner.

- There are no significant factual errors, and the explanation is
```

```
engaging, encouraging further curiosity about the topic.

**4 (Good)**

- The response is clear and mostly complete, with only minor
omissions or areas that could benefit from further simplification.

- It is mostly accurate and appropriately simplified for a general
audience, though the explanation could be slightly more relatable
or accessible.

- The response maintains coherence and engagement but may miss a
small opportunity to fully capture the user's interest.

**3 (Adequate)**

- The response is moderately clear and covers the main points,
though there may be areas where it could be simplified or
rephrased for better understanding.

- It is mostly accurate but may contain minor inaccuracies or lack
sufficient detail, limiting the effectiveness of the explanation.

- The response generally makes sense, though it might feel
slightly off-tone, lacking full relatability or coherence.

**2 (Poor)**

- The response is difficult to understand, incomplete, or somewhat
inaccurate, causing potential confusion.

- It may overuse technical language or fail to break down concepts
sufficiently for the intended audience.

- The response lacks coherence, and its structure or tone may feel
disengaging or irrelevant.

**1 (Very Poor)**
- The response is severely lacking in clarity, completeness, and
accuracy, failing to address the query in a meaningful way.

- It may contain significant factual errors or be entirely off-
topic, confusing, or unhelpful.

- The response is disengaging, incoherent, or overly technical,
making it wholly unsuitable.

Please apply these criteria carefully and respond with a single
integer from the set {1, 2, 3, 4, 5} based on your assessment.
```