PromptCom: Cost Optimization of Language Models Based on Prompt Complexity

Stanford CS224N Custom Project

Yonatan Urman
Department of Electrical Engineering
Stanford University
yurman@stanford.edu

Mengze Gao
Department of Radiology
Stanford University
gaomz@stanford.edu

Abstract

With the rapid expansion of LLM service providers and the growing number of open-source models, each with unique costs and resource demands, optimizing their utilization has become increasingly critical. In this study, we present a method to enhance resource efficiency—both in terms of cost and computational requirements—by aligning queries with the most cost-efficient model capable of adequately answering them. Our approach entails predicting the likelihood that each model from a set of models will effectively handle a given query. Subsequently, we route the query to the model that minimizes a predefined cost function, which can encompass various factors such as cost, computational requirements, environmental footprint, and more. Consequently, we facilitate substantial cost reduction by leveraging more economical models whenever possible while still achieving satisfactory task performance. We showcase the efficacy of our approach on two datasets and demonstrate its superiority in terms of computational savings compared to a recently proposed method that sequentially queries a set of models.

1 Key Information to include

- Mentor: Kaylee Carissa Burns
- Team Contribution: Yonatan Urman: Initial idea, generalizing the code to multiple datasets, running experiments for Tweet eval, writing the report.
  Mengze Gao: Implementing the base training code, running experiments for SST2, writing the report.

2 Introduction

The rapid expansion of Large Language Models (LLMs) across various applications has led to an increase in the availability of LLM service providers, including OpenAI, Google, and AI21, among others. Additionally, there has been an explosion of open-source pre-trained models that are publicly available for use. The prevailing practice involves selecting a single model somewhat arbitrarily and using it consistently. However, this approach results in considerable inefficiencies, squandering cost and computing resources Cos (2023); Bender et al. (2021); Wu et al. (2022), especially considering the diverse nature of queries, many of which could be adequately addressed by smaller, less resource-intensive, and cheaper models. Moreover, this approach engenders a significant environmental footprint due to the suboptimal allocation of compute resources Wu et al. (2021). Additionally, the high cost of accessing state-of-the-art models limits the widespread adaptation of these techniques to many individuals and businesses who cannot afford to use expensive APIs, although, in practice, many use cases can be effectively managed by more affordable models, with only a limited subset of queries requiring the usage of more expensive models.

This situation indicates the need for an efficient approach to leverage multiple LLMs, routing queries to the most appropriate model that minimizes resource usage and cost while ensuring high quality responses. In this work, we address this challenge by developing an efficient model router capable of accurately estimating which models can satisfactorily answer a given query and selecting the one that minimizes a cost function defined by a user.

The financial cost of accessing an LLM from a provider is typically related to factors such as the length of the query, the length of the response, or a fixed price per query. Our project primarily focuses on resource usage optimization. Still, there are many ways to use LLMs more efficiently. Broadly, LLM usage cost reduction strategies can be categorized into three main approaches:

1. **Model Size Reduction**: Techniques like knowledge distillation Gu et al. (2024) can be used to train a model with fewer parameters, requiring less compute power while maintaining similar performance. However, this method often entails a resource-intensive training phase, and the resulting model may be outperformed by the original model.
2. **Prompt-related Optimization**: This approach involves fine-tuning the prompt to optimize usage. Prompt engineering White et al. (2023) alters the prompt to improve model output performance, while other methods focus on concatenating multiple prompts to make calls to the model more efficient. However, this can be challenging, especially when faced with diverse queries.
3. **Model Selection**: This approach aims at designing sophisticated systems to optimize the selection of a model to use for a given query. For example, Chen et al. (2023) introduced the concept of LLM cascade, where models are sequentially queried, returning the response of the first model that adequately responds to the query.

In this work, we focus on approach (3) since (1) may compromise the accuracy of powerful models when downscaled, while approach (2) requires intricate prompt manipulation.

We consider a scenario where we have access to N different trained models, each with an associated cost reflecting the computation required and environmental impact. Our proposed approach aims to minimize resource usage while ensuring satisfactory responses. An additional advantage to our method is that it disentangles users from specific models, improving service efficiency. For instance, if a particular model is unavailable due to network issues or high demand, our method can seamlessly route the request to another model without user intervention. Furthermore, our method can inform users when none of the models can provide a good answer, prompting them to explore alternative solutions and avoiding unnecessary computational resource usage.

Suppose we can accurately estimate which model can answer a given query and select the one with the least cost. In that case, it becomes evident that this approach will yield the highest accuracy with the least resource usage. Therefore, in theory, this approach is the only one capable of producing optimal results in minimizing computation while maintaining accuracy. By assuming access to the models for an initial training phase, we show that it is possible to train a predictor that can estimate a model's likelihood of satisfactorily answering a given query. Then, we show that using such predictor and routing queries to models with minimal cost has the potential for significant computational savings with minimal impact on accuracy.

For instance, on the challenging Tweet eval dataset, the proposed predictor achieves up to 72% accuracy on test data. When used to route queries, it reduces the average cost per query by a factor of 5, while achieving accuracy that is 2% higher than the best single model, demonstrating its high potential for practical use.

3 Related Work

Much of the existing prior work focuses on enhancing the utilization of a single model. For example, Jia et al. (2018); Dao et al. (2022); Dao (2023) concentrate on utilizing hardware accelerators more efficiently. Other works, such as Bai et al. (2021); Xiao et al. (2023), propose quantization methods to expedite inference and training, thus enabling more efficient execution.

Another line of work aims to improve accuracy, thereby enhancing efficiency. Prompt engineering is a notable example, with approaches like chain of thought Wei et al. (2023) and few-shot prompting Brown et al. (2020) focusing on improving accuracy through prompt manipulation.

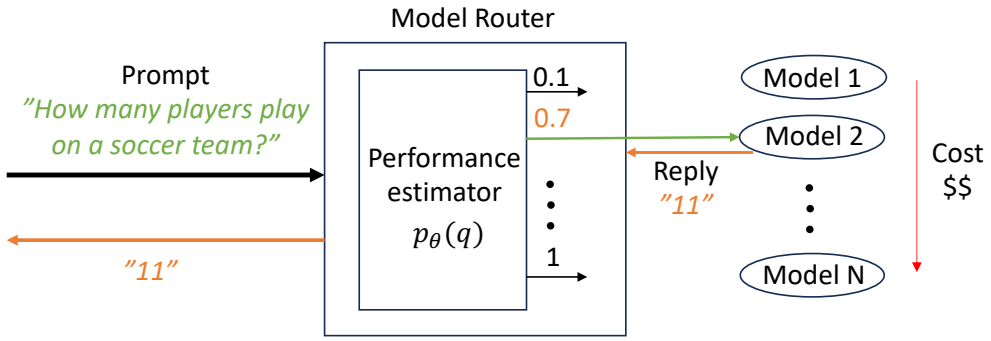


Figure 1: General system overview. Given a prompt, the system estimates the probability that each of the models will answer it correctly. Based on the cost associated with each model and the estimated scores, the system selects a model likely to provide a satisfactory response with minimal cost to the user.

However, our work primarily focuses on efficiently utilizing a set of models without altering prompts. This approach maintains a clean interface with the user, requiring no changes in interaction with the model. Theoretically, this approach could achieve better performance than individual models, as combining different models could cover a broader range of queries.

In Chen et al. (2023), the authors proposed an approach called LLM cascade, which sequentially calls a series of increasingly powerful and expensive models to respond to a query. The system returns the first adequate response based on a pre-trained scoring function that assesses the quality of an answer given a prompt. The results indicate significant savings, with up to a 98% reduction in inference cost compared to the best individual API. Alternatively, it demonstrates potential performance improvement by up to 4% while maintaining the same cost. However, this method may be inefficient for some queries, as it generates unnecessary calls to some models in the set. This inefficiency motivates us to develop a more efficient approach, where we first estimate which models can answer the query and then query the cheapest one.

4 Approach

We propose a model router that, given a query, routes it to a specific model and returns the model's reply. To select which model to send the query to, the router uses a prediction neural network p_θ, which takes the query q as input and estimates the likelihood that each model, out of a given set of N models, can reply correctly. Specifically, $[p_\theta(q)]_i$ estimates the probability that the i-th model will answer the query correctly. Given $p_\theta(q)$, the router selects a model index i that minimizes a cost function:

$$i^* = \arg\min_i c_i$$
$$\text{s.t. } [p_\theta(q)]_i \geq \tau$$

In this formulation, c_i represents the cost associated with using the i-th model, and τ is a threshold determining the minimum confidence level required for a model to be selected. We chose the cost function to be linear in the number of model parameters, i.e., $c_i = \alpha n_i$, where n_i is the number of parameters in the i-th model. We set α such that the cost of the largest model is equal to 1. The method is summarized in Figure 1.

For the predictor $p_\theta(q)$, we first encode the input query using the sentence encoder proposed in Ni et al. (2021). This results in a 768-dimensional vector summarizing the query. We then use a classification neural network comprising two hidden layers of sizes 256 and 128, and an output layer of size N (the number of models in our set). ReLU activations are applied at the output of each hidden layer. During training, we observed that freezing most of the encoder layers and training only the last two layers and the classifier reduced overfitting and provided optimal results.

For training, we assume access to the N different models, which we can query to train p_θ. The training procedure for p_θ is summarized in Algorithm 1. We iterate over a dataset and send each

query to all available models, recording which model answered correctly (f in the algorithm). We then pass the query to p_θ, which estimates which of the N models will respond correctly. We employ binary cross-entropy loss between the model estimates and the actual vector, indicating which models responded correctly. Finally, we backpropagate and update the weights of the predictor θ. At the end of this process, the model p_θ can estimate, given a query, which models from the list can answer the query correctly. It's important to note that we utilize datasets with unique labels as output (e.g., sentiment analysis), where the expected output of each model is one word converted to an index.

Algorithm 1 Model Training Algorithm

Require: N models $\{m_i\}_{i=1}^{i=N}$, Predictor Neural Net p_θ
Require: Dataset containing queries and labels
for all batch in dataset **do**
  $\ell = 0$
  for all (query, label) in batch **do**
    for $i = 1$ **to** N **do**
      $f(i) = \begin{cases} 1 & \text{if } m_i(\text{query}) == \text{label} \\ 0 & \text{otherwise} \end{cases}$
    end for
    $\ell +=$ BinaryCrossEntropyLoss$(p_\theta(\text{query}), f)$
  end for
  Backpropagate and update θ
end for

5 Experiments

We conducted several experiments to evaluate the proposed method and compared it with various baseline approaches. We selected datasets representing tasks where the model is expected to produce a single output corresponding to a label. This selection facilitated precise metric computation and method comparison. We evaluated the following approaches:

1. Our proposed approach: The proposed model router routes the query to the model with minimal cost.
2. Single model: Employing a single model for all queries.
3. Cascade: The method proposed in Chen et al. (2023), which trains a scoring neural network to determine if an output is the correct answer to a query. For the scoring function, we used the same architecture as our predictor. Instead of inputting only the query, we input the concatenated query and model response. The model was trained to estimate if the response is correct (see Appendix). Then, the router sequentially queries the models, starting from the smallest, and returns the output of the first good response as determined by the scoring function.

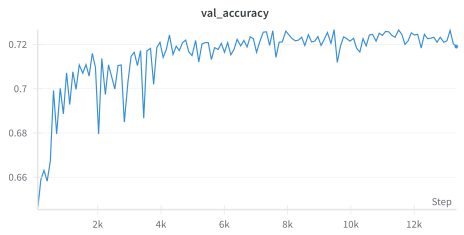
For the set of available models, we set $N = 3$ and utilized the flan-T5 family Chung (2022): *small*, *base*, and *large*, all accessible on Hugging Face at `https://huggingface.co/collections/google/flan-t5-release-65005c39e3201fff885e22fb`. We opted not to employ the llama-2 Hugo Touvron (2023) family of models due to the significant compute resources it required, which were unavailable to us.

5.1 Data

As discussed in Section 5, for evaluation, we are utilizing labeled datasets. For the following datasets, before querying the models we added the prefix: "Please determine the sentiment (choose output from: positive, negative or neutral) in the following: ", and the suffix:". Sentiment:". We tested our results on two datasets:

1. **SST2** dataset from Hugging Face, available at `https://huggingface.co/datasets/sst2/`. This is a corpus with fully labeled parse trees that allows for a complete analysis of the compositional effects of sentiment in language. It was parsed with the Stanford

(a) Train loss curve for prediction model training.



(b) Validation accuracy for prediction model.

Figure 2: Training convergence of the prediction model utilizing a set of $N = 3$ flan-T5 models on the Tweet eval dataset.

parser and includes a total of 215,154 unique phrases from those parse trees, each annotated by 3 human judges. It has 67.3K, 872, 1.82K samples for training, validation and testing (respectively).

2. **Tweet eval** dataset from Hugging Face, available at `https://huggingface.co/datasets/tweet_eval/`. This dataset encompasses seven heterogeneous tasks, all framed as multi-class tweet classification. All tasks have been unified into the same benchmark, with each dataset presented in the same format and with fixed training, validation, and test splits. We utilized the sentiment analysis task for our model. It has 45.6K, 2K, 12.3K samples for training, validation and testing (respectively).

5.2 Evaluation method

We are primarily interested in the tradeoff between compute and performance, aiming to minimize compute while maintaining a satisfactory level of accuracy. To analyze this, we generate a plot where the x-axis represents the cost, and the y-axis represents the accuracy. We position each method on this plot based on its computed average accuracy and cost per prompt, as calculated on the test set of each dataset.

5.3 Experimental details

We followed the approach described in Section 4, utilizing Ni et al. (2021) base model as an encoder to encode the query. On top of this encoder, we added a two-hidden-layers classification model. During training, we freeze all the encoder layers except for the last two. Therefore, we trained only the last two layers of the encoder and the classifier. We used the AdamW optimizer Loshchilov and Hutter (2019) with a learning rate set to 5×10^{-3}. A batch size of 64 was used, and training was conducted for 20 epochs. All models were trained on a single A5000 GPU.

Figure 2 illustrates the training process when training the predictor on the Tweet eval dataset. The final prediction accuracy observed is approximately 72%. Figure 3 depicts the evolution of the confusion matrix of the predictor throughout the training process illustrating the improvement in its estimation capabilities.

In Figure 4, a comparison of different methods on the cost vs. accuracy graph is presented. It is evident that both the proposed approach and the cascade approach outperform any single model. While the cascade approach slightly outperforms our method in terms of accuracy, its associated cost is significantly higher. This outcome was anticipated due to the unnecessary calls to models that cannot answer the query. Additionally, our approach is the only one that can inform the user when it believes no model can respond to the query satisfactorily. For the cascade approach to confirm the absence of a valid response, it must call and access the models, resulting in unnecessary calls.

6 Analysis

As depicted in Figures 4 and 5, our proposed model achieves improved accuracy while requiring significantly less computation. For example, on the Tweet eval dataset, we achieved a fivefold

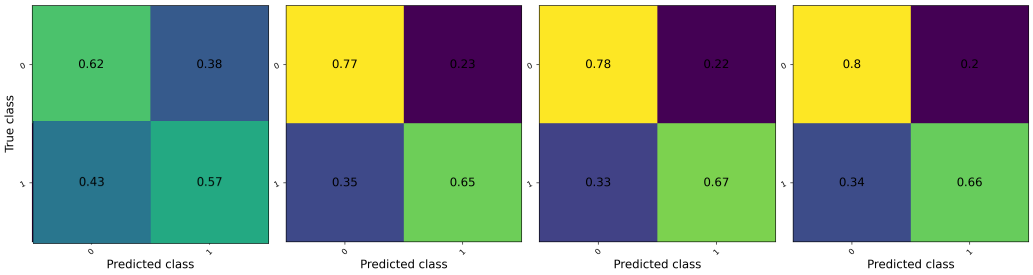


Figure 3: Evolution of the confusion matrix for the predictor on the validation set of Tweet eval during training. 1 indicates correct model predictions for a query, while 0 indicates incorrect predictions. Predictions from multiple models are aggregated.

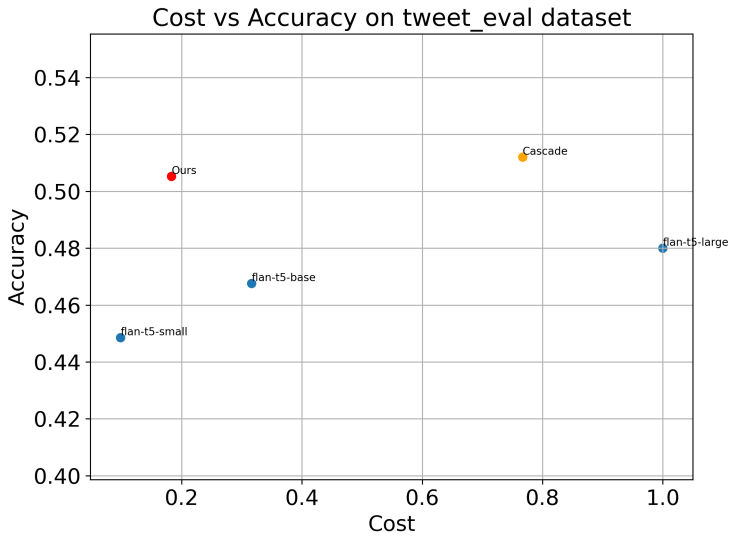


Figure 4: Comparison of Cost vs. Accuracy on the Tweet eval test set. The x-axis represents the average cost per query, while the y-axis indicates accuracy.

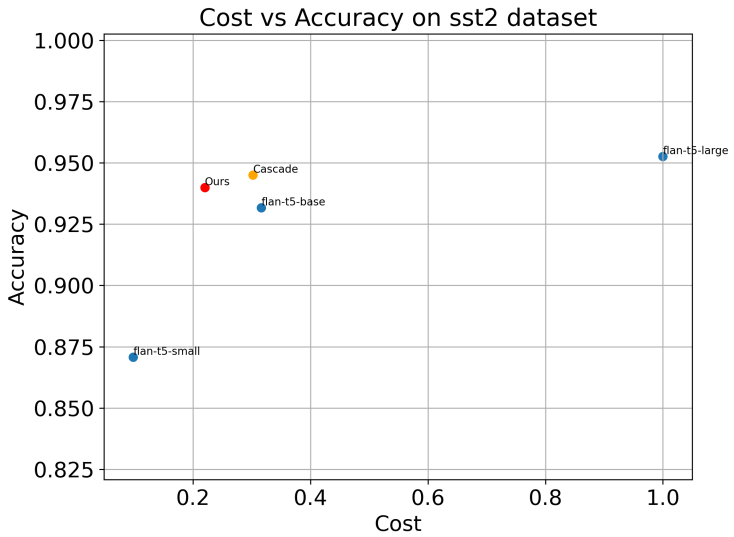


Figure 5: Comparison of Cost vs. Accuracy on the SST2 test set. The x-axis represents the average cost per query, while the y-axis indicates accuracy.

reduction in computing while improving the accuracy of the best model by 2%. It is noteworthy that our approach enhances the performance of the best model.

Upon closer examination of some outputs, we observed cases where smaller models provided correct answers while larger models failed to do so. Our estimator successfully predicted this scenario (or at least predicted that the less expensive models would likely provide correct answers), allowing the router to select these smaller, less expensive models. For instance, the tweet "RT @user Can I politely suggest that we all stop using the term 'alt-right' and use other, more precise words." is classified as neutral, with both the small and base models reaching this classification. However, the larger model classified it as negative, possibly due to overcomplicating the sentence's meaning.

Interestingly, we encountered instances where dataset labels appeared to be incorrect, negatively affecting the performance of larger models despite them providing the correct answers. For example, the tweet "Grayson Allen just gave dude such a sick move" is labeled as negative, though we think the label should be positive (in slang usage). While the larger model correctly classified it as positive, the other models classified it as negative. Our predictor indicated that all models could respond correctly, leading us to query the first model, which provided the correct result (based on the given labels, but not correct in general).

On the easier SST2 dataset, our model did not outperform the best-performing model in terms of accuracy. Upon closer examination, we found no instances where smaller models were correct while larger ones were wrong, limiting the potential for increased accuracy. Nevertheless, our proposed approach achieved similar accuracy with a fivefold reduction in compute, demonstrating significant computational savings.

7 Conclusion

In this project, we tackled the challenge of efficiently utilizing a diverse set of open-source models, each with distinct costs and resource requirements. Our proposed approach introduces a novel method for efficiently leveraging a set of LLMs. By employing a prediction model to estimate the likelihood of each model in the set responding accurately to a query, we route the query to the model that minimizes a cost function.

We conducted thorough analyses on two datasets, demonstrating significant compute savings and improved performance. For instance, on the Tweet eval dataset, our approach reduced compute requirements by 80% compared to running Flan-T5-large alone while achieving a 2% increase in accuracy through the utilization of multiple models. When comparing our method with the cascade approach, we observed that although cascading can yield higher accuracy, it also incurs substantially higher compute demands, making our approach more practical in real-world scenarios.

Exploring more intricate cost functions that incorporate factors such as service provider fees could be interesting for future endeavors. Extending this approach to incorporate API calls beyond open-source models presents another important and interesting extension. Moreover, developing advanced routers capable of aggregating responses from multiple models to enhance accuracy and dynamically altering prompts for greater efficiency could further enrich this line of research.

Another potentially interesting future work would be to combine this approach with the approach presented in Chen et al. (2023). In this case, we will first estimate which models can answer the query; then, if multiple models can, we will sequentially query them and return the first one that receives a high score based on a pre-trained scoring function.

References

2023. Cost estimation of using gpt-3 for real applications. Accessed: 2023-03-31.

Haoli Bai, Lu Hou, Lifeng Shang, Xin Jiang, Irwin King, and Michael R. Lyu. 2021. Towards efficient post-training quantization of pre-trained language models.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA. Association for Computing Machinery.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Lingjiao Chen, Matei Zaharia, and James Zou. 2023. Frugalgpt: How to use large language models while reducing cost and improving performance.

Hyung Won Chung. 2022. Scaling instruction-finetuned language models.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2024. Knowledge distillation of large language models.

Kevin Stone Hugo Touvron, Louis Martin. 2023. Llama 2: Open foundation and fine-tuned chat models.

Zhihao Jia, Matei Zaharia, and Alex Aiken. 2018. Beyond data and model parallelism for deep neural networks. *CoRR*, abs/1807.05358.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization.

Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B. Hall, Daniel Cer, and Yinfei Yang. 2021. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. 2023. A prompt pattern catalog to enhance prompt engineering with chatgpt.

Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga Behram, James Huang, Charles Bai, Michael Gschwind, Anurag Gupta, Myle Ott, Anastasia Melnikov, Salvatore Candido, David Brooks, Geeta Chauhan, Benjamin Lee, Hsien-Hsin S. Lee, Bugra Akyildiz, Maximilian Balandat, Joe Spisak, Ravi Jain, Mike Rabbat, and Kim Hazelwood. 2022. Sustainable ai: Environmental implications, challenges and opportunities.

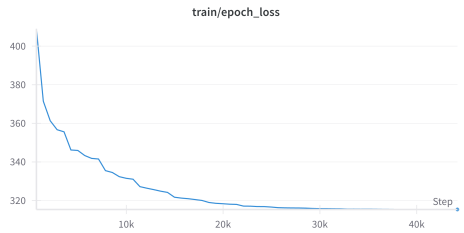
Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga Behram, James Huang, Charles Bai, Michael Gschwind, Anurag Gupta, Myle Ott, Anastasia Melnikov, Salvatore Candido, David Brooks, Geeta Chauhan, Benjamin Lee, Hsien-Hsin S. Lee, Bugra Akyildiz, Maximilian Balandat, Joe Spisak, Ravi Jain, Mike Rabbat, and Kim M. Hazelwood. 2021. Sustainable AI: environmental implications, challenges and opportunities. *CoRR*, abs/2111.00364.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models.

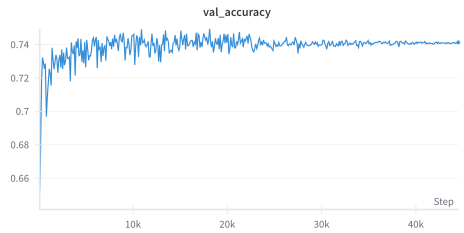
A Appendix

A.1 Score Function Training

To train the score function from Chen et al. (2023), we used the same architecture as our model. The input is the answer concatenated to the query and the output is a single number estimating the probability that the response is correct. In Figure 6, we demonstrate the training process of this function.



(a) Train loss curve for scoring model for the cascade approach



(b) Validation accuracy for scoring model for the cascade approach.

Figure 6: Training convergence of the scoring model for the cascade approach on the tweet eval dataset.