Proactive Inference Scheduling via Output Length Prediction in LLMs

Abstract

Efficiently predicting the output length of large language models (LLMs) is crucial for optimizing processing power and memory allocation. This paper presents a scalable length prediction framework using a BERT-based model trained on the Castillo dataset(Perez-Ramirez et al., 2025) using data from Llama-3.2-1B, Llama-3.2-3B, and Llama-3.1-8B. The framework integrates classification and regression approaches to estimate token lengths and standard variance. It demonstrates strong generalization and robust performance across multiple LLMs, including on the unseen dataset. This work provides an effective solution for resource-aware deployment of LLMs.

1 Introduction

Large language models (LLMs) have driven transformative advances in natural language processing due to their remarkable capabilities in language understanding and generation (Brown et al., 2020; Raffel et al., 2020; Chowdhery et al., 2023). When fine-tuned with instructions, LLMs have been widely adopted in applications such as question answering, dialogue systems, and code generation (Ouyang et al., 2022; Chen et al., 2021; Achiam et al., 2023; Team et al., 2023), serving millions of users daily in both production and everyday scenarios. However, the substantial computational and memory overhead poses significant challenges to the scalability and cost-efficiency of inference processes.

Most LLMs adopt an autoregressive architecture, where the model encodes input text and predicts subsequent tokens by computing their log probabilities based on the preceding context. Under high-concurrency and latency-sensitive settings, efficient management of computational and memory resources becomes a core bottleneck in LLM service systems. Recent systems research has emphasized memory management of attention key-value (KV) caches (Kwon et al., 2023) and reactive scheduling strategies to accommodate runtime demand fluctuations (Duan et al., 2024; Patel et al., 2024; Agrawal et al., 2024). However, these approaches are constrained by the inherent randomness of the generation process.

Proactive scheduling strategies aim to predict the output length of LLMs in advance to enable preemptive scheduling. This work proposes a fundamental framework for output length prediction by training a BERT-based model on the Castillo dataset using data from Llama-3.2-1B, Llama-3.2-3B, and Llama-3.1-8B.

2 Related Work

Research on LLM output length prediction remains limited. Existing studies primarily focus on enhancing resource allocation efficiency in LLM-as-a-Service (LMaaS) systems by predicting generation lengths for scheduling CPUs, GPUs, and other hardware resources.

Zheng et al. (Zheng et al., 2023a) fine-tuned an LLM by appending an instruction to the user input, prompting the model to predict its own output length. However, this approach is intrusive to user input, may affect the generated content, and was only evaluated on a single model. Ke Cheng et al. (Cheng et al., 2024) explored optimized resource allocation in LMaaS by employing a small BERT model to predict LLM output lengths using a random forest approach. They also implemented an online learning mechanism, collecting and retraining on requests where prediction errors exceeded 10 tokens or 10% of the actual output length.

Haoran Qiu et al. developed the μ-serve system (Qiu et al., 2024), demonstrating that small models can achieve high accuracy in output length prediction. They found that classifying predicted lengths into five buckets offered the best trade-off for hardware scheduling—finer classifications provided better granularity but led to lower accuracy, negatively impacting schedulers. Their method utilized BERT as a proxy model, followed by a linear classification head. Training involved joint fine-tuning of BERT and the classifier, followed by separate training of the classifier, effectively leveraging BERT's language understanding while maintaining accuracy and training efficiency.

Perez-Ramirez, D. F. et al. (Perez-Ramirez et al., 2025) introduced the Castillo dataset, which consists of prompt–output length pairs across various models, providing a valuable resource for further training and evaluation of generation length prediction models.

3 Dataset Setting

Upon inspection, the dataset provided in the original task fig. 1 exhibits a distribution concentrated at both ends. Further analysis reveals that the LLama-3.2-1B-Instruct model suffers from performance degradation, leading to issues such as token repetition and corrupted outputs. These anomalies significantly hinder model training and limit generalization ability Layaq, Bairam (2020).

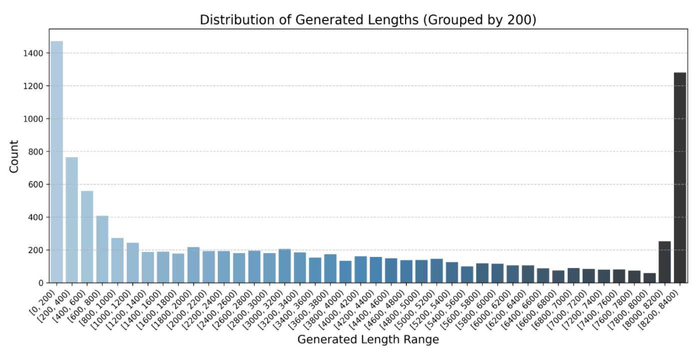

Figure 1: Generation length distribution

To address this issue, we refined the dataset by removing entries with output tokens exceeding 15,000 and incorporated the Castillo dataset for model training. This dataset includes prompt–output length pairs from various LLMs, comprising seven open-source prompt datasets with a total of 15,000 pairs. The datasets encompass open-ended instruction data (Dolly, ShareGPT, Alpaca) and code-oriented data (Mbpp, Apps, DS-1000, BigCodeBench). During training, we utilized six datasets, excluding Alpaca, and evaluated overall test performance on these six datasets. Additionally, we tested on Alpaca to demonstrate generalization capability. Each dataset entry includes a prompt, its mean output (averaged over 10 generations), the standard deviation of outputs, and the LLM used for generation,the example case for dataset could be seen at Appendix A.1.

Thus, we adopted the Castillo dataset and selected three models of varying sizes—Llama-1B, 3B, and 8B—for training. The token length distribution of Llama-3 1B/3B/8B within the Castillo dataset is shown in fig. 2, where a more balanced distribution is observed, facilitating model training. Relevant information including the mean output and the standard deviation of the output from the Castillo dataset (trimmed top 1%) are presented in fig. 3&fig. 4, illustrating that different models generate varying output lengths for the same prompt.

Table 1: Token length statistics for various datasets

Name	Samples	Mean	Min.	P25	P50	P75	P99	Max.
DollyDataset	2000	125.9	36	44	50	146	795.2	4003
ShareGPT	2000	260.5	36	48	64	168	2534.0	4003
Alpaca	2000	53.7	39	45	49	57	114.0	397
Mbpp	974	153.5	88	109	131	173	336.3	2265
Apps	2000	545.0	87	307.7	441	650	2105.0	2534
DS-1000	1000	317.2	67	170.5	283	395	1018.3	2109
BigCodeBench	1140	179.8	87	137	164	205	398.4	1251

4 Classification Model

4.1 Model Configuration

In this part, we try to build a classification model to predict the ouput length. Haoran Qiu et al. (Perez-Ramirez et al., 2025) demonstrated in their research that a small proxy model can achieve high accuracy with high efficiency. We therefore use BERT as the basic model of the classification model, the precise model architecture is presented in fig. 5.

We employ **BERT** as the backbone to encode and process the input prompts. The tokenized inputs are passed through BERT's transformer-based encoder, which extracts high-dimensional contextual representations via its hidden layers. These representations are then mapped to model-specific information (e.g., the identity of the LLM that generated the answer) through a downstream classification module. The model encoder allows the classification model to learn from dataset of other models while not being falsely guided. It also allows the model to be easily applied on other model's dataset with few training.

The classification module consists of:

- Two linear layers with ReLU activation (applied to the first linear layer)
- Two dropout layer for regularization
- A final linear classifier head to predict the target labels

BERT's hidden dimension is first reduced to a fixed size (`end dimension`) before applying the specified linear transformations. This architecture facilitates robust feature extraction and task-specific adaptation while minimizing overfitting.

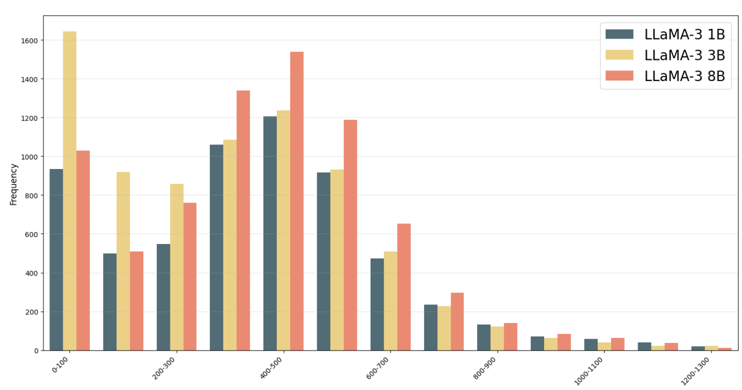

Figure 2: Castillo Dataset Generated Length Distribution (trimmed top 1%)

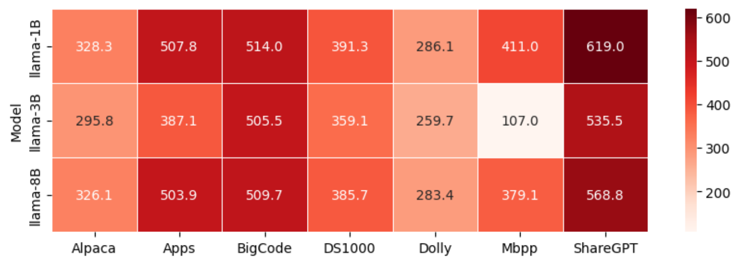

Figure 3: Castillo dataset: Mean output length in different models

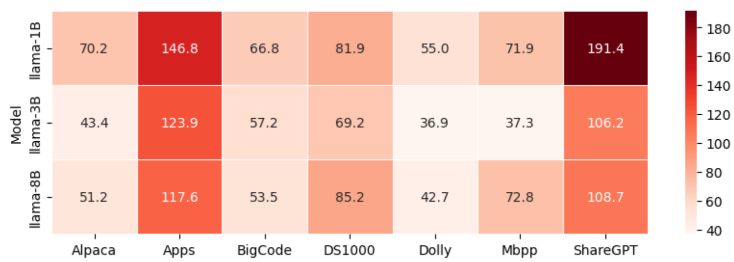

Figure 4: Castillo dataset: std of output length in different models

4.2 Data Binning Strategy

In practical applications, bucket classification must be configured based on specific task requirements and hardware constraints. To evaluate the precision under varying granularity levels, we employed multiple bucket sizes, including divisions per 50, 100, 200, 500 and 1000 tokens.

4.3 Model Training Performance

The loss and accuracy curves during training are depicted in Figure 6. We show only Llama-3.2-1B (200 tokens/bin) for brevity; other models follow similar trends.The model exhibits overfitting, likely due to the limited dataset size Despite incorporating a dropout layer and other tricks, the overfitting issue remains unresolved. Furthermore we present the result of training under different bin setting in fig. 7table 2.

Table 2: Comparison of classification results across different models and bin sizes.

Models	Bin Size=50		Bin Size=100		Bin Size=200		Bin Size=500		Bin Size=1000	
	Acc(%)	F1	Acc(%)	F1	Acc(%)	F1	Acc(%)	F1	Acc(%)	F1
llama-1B	16.81	0.164	31.21	0.305	50.66	0.499	68.90	0.679	89.78	0.8844
llama-3B	19.23	0.187	33.63	0.335	50.55	0.505	72.64	0.720	92.19	0.9099
llama-8B	16.15	0.151	28.35	0.280	49.12	0.484	67.25	0.667	91.86	0.9020

It can be easily and observed that the Acc decline as the bin size grows, which is easy to understand: since there are more bins to choose from, it is harder for the model to allocate the prompts into the right bin. According to the result, the model show similar accuracy on different models, demonstrating its robustness. The Acc(accuracy) reaches around 50% on 200tokens/bin, around 70% on 500tokens/bin and around 90 % on 1000tokens/bin, which is really impressive.

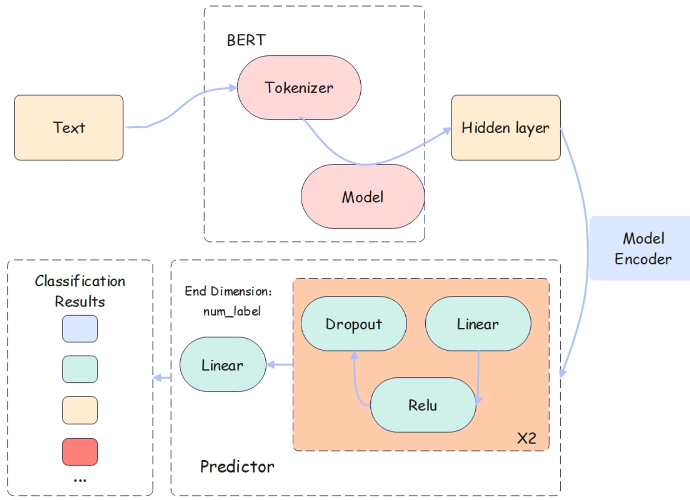

Figure 5: model architecture of classification model

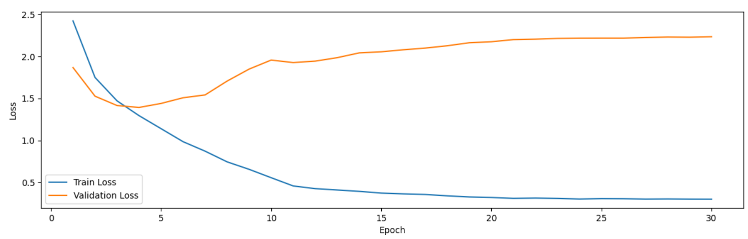

Figure 6: Training curve on Llama-3-1B dataset

4.4 Generalization Experiments

To test our model's generalization capacity,the model is tested on Alpaca without any exposure during training. The result is presented in table 3. Compared to table 2, on the Alpaca dataset the model has degenerate accuracy on small bins with fewer tokens(50,100), while has relevantly stable results on large bins with more tokens (200,500,1000), demonstrating strong generalization capability. Notably, the model performs the strongest generalization capability on Llama-3.2-3B.

Table 3: Classification results across different models and bin sizes on Alpacha dataset

Models	Bin Size=50		Bin Size=100		Bin Size=200		Bin Size=500		Bin Size=1000	
	Acc(%)	F1	Acc(%)	F1	Acc(%)	F1	Acc(%)	F1	Acc(%)	F1
llama-1B	10.00	0.094	17.70	0.164	35.45	0.361	71.50	0.714	93.60	0.927
llama-3B	13.35	0.135	24.35	0.254	41.65	0.435	74.50	0.747	96.55	0.952
llama-8B	11.50	0.105	21.75	0.215	34.80	0.354	75.45	0.756	94.55	0.943

5 Regression model

5.1 Model Configuration

Based on the classification model, we build the regression model presented in fig. 8.The model configuration is similar to that of the classification model, except the last linear layer output of the regression model has only two dimensions including the predicted length and the predicted standard division(std). By using the standard division, we can further quantify

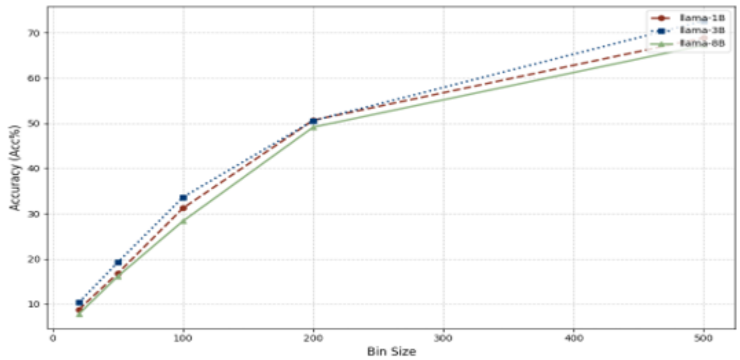

Figure 7: classification over different models

the variation in output length across identical prompts, enabling optimized allocation of memory and computational resources in practical deployments.

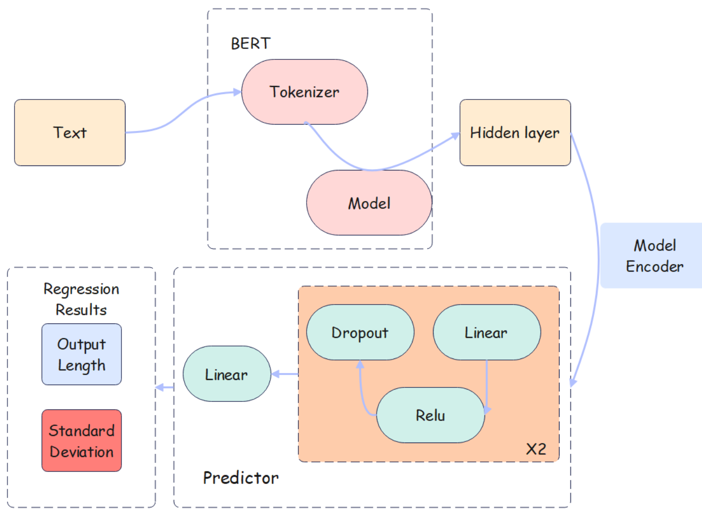

Figure 8: regression model configuration

5.2 Baseline model

(Zheng et al., 2023b) add prompt after the users' questions to let the LLM predict the length of the generated text before answer the questions. Inspired by this research, the paper employs the prompt method as the baseline model.

Systematic prompt engineering reveals that the Llama-3.2-1B and Llama-3.2-3B models fail to understand task instructions, resulting in poor performance metrics. Conversely, the larger Llama-3.1-8B model effectively processes engineered prompts. Thus, we employ Llama-3.1-8B to compare our method against the baseline (prompt engineering), with the specific prompt template provided in Appendix A.2.

5.3 Model Training Performance

The loss&accuracy curves on Llama-3-1B dataset throughout the training are shown in fig. 9. It can be observed that the regression model do not have overfitting problem, owing to our model can get more information from the exact output length than classification results.

The result on the test dataset across different LLM models and the result of the baseline models are presented in table 4

Table 4: Comparison of regression results across different methods and models. Of which "mean" represents the mean output token length, and "std" represents the standard deviation of output token length.

Models	Our regression model				Baseline(Prompt)	
	MSE(mean)	MAE(mean)	MSE(std)	MAE(std)	MSE(mean)	MAE(mean)
llama-1B	75378.30	121.02	108624.30	80.84	/	/
llama-3B	45735.30	119.18	28182.20	54.68	/	/
llama-8B	82152.72	128.28	19193.33	51.60	3210768.61	384.03

The experimental results demonstrate consistent mean absolute error (MAE 120 tokens) across all model variants, suggesting our regression model's high accuracy and robustness on different models.

Notably, the 3B model achieves significantly higher stability (MAE mean=51.60) compared to other counterparts, which we attribute to its more concentrated token distribution patterns during generation. Alsothe Llama-3.2-1B model demonstrates significantly poorer variance prediction performance, primarily due to its frequent output degradation during generation.

Compared to the baseline model, our regression model has profound advantage on accuracy(128.28 vs. 384.01 on MAE and 82152.72 vs. 3210768.61 on MSE). Our model is also capable of predicting the standard division of the prompt which is unlikely to be achieved by baseline model (prompt engineering).

5.4 Generalization Experiments

Following the approach in classification part, we evaluate the regression model on the Alpaca dataset. The results are presented in table 5

Table 5: Regression results across different models and bin sizes on Alpacha dataset

Models	Regression Results on Alpaca			
	MSE(mean)	MAE(mean)	MSE(std)	MAE(std)
llama-1B	70542.27	163.87	48814.96	70.78
llama-3B	73246.13	164.94	16033.02	53.45
llama-8B	78596.34	175.10	21076.86	52.87

Compared to the results on test dataset presented in table 4, the prediction of the std is stable, even better on Alpaca dataset, while the accuracy of the predicted mean length decrease slightly(120 to 160 on MAE approximately). This demonstrates our regression model's outstanding generalization capability.

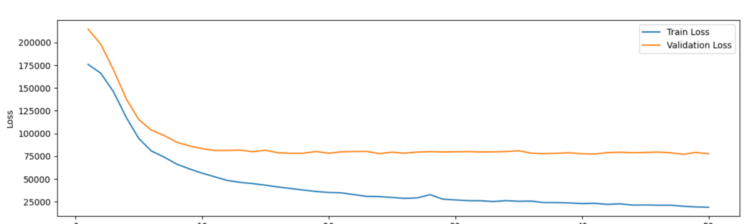

Figure 9: Training curve on Llama-3-1B dataset

6 Conclusion

Our findings mark a promising step toward proactive inference optimization in LLMs, and we believe the proposed framework can serve as a foundation for future advancements in resource-aware LLM serving systems.

6.1 Innovations

In this work, we proposed a novel output length prediction framework for large language models (LLMs) by leveraging the newly released Castillo dataset. Compared to prior approaches, our contributions are multifaceted:

- **Adoption of the Castillo dataset**: We utilized the recently open-sourced Castillo dataset, which provides comprehensive output length statistics across multiple LLMs, enhancing prediction accuracy and generalization.
- **Multi-model validation**: Our prediction framework was validated on Llama-3-1B, 3B, and 8B models, showing consistent and robust performance across different model scales.
- **Incorporation of variance prediction**: Beyond predicting the mean output length, our regression model also predicts the standard deviation, enabling more refined resource allocation strategies.
- **Generalization experiments**: We demonstrated strong generalization capabilities by evaluating on previously unseen datasets (e.g., Alpaca), confirming the model's adaptability.
- **Unified prediction framework**: We developed an integrated predictor architecture capable of supporting both classification and regression tasks, which can be reused across different LLM backends.See example case in Appendix B.

6.2 Future Work

While our method demonstrates strong empirical performance, several important research directions merit further investigation to advance this line of work. First, architectural improvements could address persistent overfitting issues observed despite using larger datasets and proxy models, potentially through more sophisticated regularization techniques or neural architecture search. Second, the development of specialized Chain-of-Thought (CoT) benchmarks is critically needed to properly evaluate length generalization capabilities, given the unique characteristics and computational demands of CoT reasoning. Finally, practical deployment considerations including computational efficiency, memory constraints, and robustness against overthinking phenomena require systematic study to enable real-world applications. These research directions would not only strengthen the current framework but also contribute broadly to the field's understanding of reasoning in large language models.

References

Achiam Josh, Adler Steven, Agarwal Sandhini, Ahmad Lama, Akkaya Ilge, Aleman Florencia Leoni, Almeida Diogo, Altenschmidt Janko, Altman Sam, Anadkat Shyamal, others . Gpt-4 technical report // arXiv preprint arXiv:2303.08774. 2023.

Agrawal Amey, Kedia Nitin, Panwar Ashish, Mohan Jayashree, Kwatra Nipun, Gulavani Bhargav, Tumanov Alexey, Ramjee Ramachandran. Taming Throughput-Latency Tradeoff in LLM Inference with Sarathi-Serve // 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24). Santa Clara, CA: USENIX Association, VII 2024. 117–134.

Brown Tom, Mann Benjamin, Ryder Nick, Subbiah Melanie, Kaplan Jared D, Dhariwal Prafulla, Neelakantan Arvind, Shyam Pranav, Sastry Girish, Askell Amanda, Agarwal Sandhini, Herbert-Voss Ariel, Krueger Gretchen, Henighan Tom, Child Rewon, Ramesh Aditya, Ziegler Daniel, Wu Jeffrey, Winter Clemens, Hesse Chris, Chen Mark, Sigler Eric, Litwin Mateusz, Gray Scott, Chess Benjamin, Clark Jack, Berner Christopher, McCandlish Sam, Radford Alec, Sutskever Ilya, Amodei Dario. Language Models are Few-Shot Learners // Advances in Neural Information Processing Systems (NeurIPS). 33. 2020. 1877–1901.

Evaluating Large Language Models Trained on Code. // . 2021.

Cheng Ke, Hu Wen, Wang Zhi, Du Peng, Li Jianguo, Zhang Sheng. Enabling Efficient Batch Serving for LMaaS via Generation Length Prediction. 2024.

Chowdhery Aakanksha, Narang Sharan, Devlin Jacob, Bosma Maarten, Mishra Gaurav, Roberts Adam, Barham Paul, Chung Hyung Won, Sutton Charles, Gehrmann Sebastian, Schuh Parker, Shi Kensen, Tsvyashchenko Sasha, Maynez Joshua, Rao Abhishek, Barnes Parker, Tay Yi, Shazeer Noam, Prabhakaran Vinodkumar, Reif Emily, Du Nan, Hutchinson Ben, Pope Reiner, Bradbury James, Austin Jacob, Isard Michael, Gur-Ari Guy, Yin Pengcheng, Duke Toju, Levskaya Anselm, Ghemawat Sanjay, Dev Sunipa, Michalewski Henryk, Garcia Xavier, Misra Vedant, Robinson Kevin, Fedus Liam, Zhou Denny, Ippolito Daphne, Luan David, Lim Hyeontaek, Zoph Barret, Spiridonov Alexander, Sepassi Ryan, Dohan David, Agrawal Shivani, Omernick Mark, Dai Andrew M., Pillai Thanumalayan Sankaranarayana, Pellat Marie, Lewkowycz Aitor, Moreira Erica, Child Rewon, Polozov Oleksandr, Lee Katherine, Zhou Zongwei, Wang Xuezhi, Saeta Brennan, Diaz Mark, Firat Orhan, Catasta Michele, Wei Jason, Meier-Hellstern Kathy, Eck Douglas, Dean Jeff, Petrov Slav, Fiedel Noah. PaLM: Scaling Language Modeling with Pathways // Journal of Machine Learning Research. 2023. 24, 240. 1–113.

Duan Jiangfei, Lu Runyu, Duanmu Haojie, Li Xiuhong, Zhang Xingcheng, Lin Dahua, Stoica Ion, Zhang Hao. MuxServe: flexible spatial-temporal multiplexing for multiple LLM serving // Proceedings of the 41st International Conference on Machine Learning. 2024. (ICML'24).

Kwon Woosuk, Li Zhuohan, Zhuang Siyuan, Sheng Ying, Zheng Lianmin, Yu Cody Hao, Gonzalez Joseph E., Zhang Hao, Stoica Ion. Efficient Memory Management for Large Language Model Serving with PagedAttention // Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles. 2023.

Layaq Shaheen, Bairam Dr. Manjula. A Recapitulation of Imbalanced Data // International Journal of Innovative Technology and Exploring Engineering. 01 2020. 9. 452–455.

Ouyang Long, Wu Jeffrey, Jiang Xu, Almeida Diogo, Wainwright Carroll, Mishkin Pamela, Zhang Chong, Agarwal Sandhini, Slama Katarina, Ray Alex, Schulman John, Hilton Jacob, Kelton Fraser, Miller Luke, Simens Maddie, Askell Amanda, Welinder Peter, Christiano Paul F, Leike Jan, Lowe Ryan. Training language models to follow instructions with human feedback // Advances in Neural Information Processing Systems. 35. 2022. 27730–27744.

Patel Pratyush, Choukse Esha, Zhang Chaojie, Shah Aashaka, Goiri Íñigo, Maleki Saeed, Bianchini Ricardo. Splitwise: Efficient generative llm inference using phase splitting // 2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA). 2024. 118–132.

Perez-Ramirez Daniel F., Kostic Dejan, Boman Magnus. CASTILLO: Characterizing Response Length Distributions of Large Language Models. 2025.

Qiu Haoran, Mao Weichao, Patke Archit, Cui Shengkun, Jha Saurabh, Wang Chen, Franke Hubertus, Kalbarczyk Zbigniew T., Başar Tamer, Iyer Ravishankar K. Power-aware deep learning model serving with µ-serve // Proceedings of the 2024 USENIX Conference on Usenix Annual Technical Conference. USA: USENIX Association, 2024. (USENIX ATC'24).

Raffel Colin, Shazeer Noam, Roberts Adam, Lee Katherine, Narang Sharan, Matena Michael, Zhou Yanqi, Li Wei, Liu Peter J. Exploring the limits of transfer learning with a unified text-to-text transformer // Journal of machine learning research. 2020. 21, 140. 1–67.

Team Gemini, Anil Rohan, Borgeaud Sebastian, Alayrac Jean-Baptiste, Yu Jiahui, Soricut Radu, Schalkwyk Johan, Dai Andrew M, Hauth Anja, Millican Katie, others . Gemini: a family of highly capable multimodal models // arXiv preprint arXiv:2312.11805. 2023.

Zheng Zangwei, Ren Xiaozhe, Xue Fuzhao, Luo Yang, Jiang Xin, You Yang. Response Length Perception and Sequence Scheduling: An LLM-Empowered LLM Inference Pipeline. 2023a.

Response length perception and sequence scheduling: An llm-empowered llm inference pipeline. // . 36. 2023b. 65517–65530.

A Appendix / supplemental material

A.1 Dataset Entry Example (Castillo)

Prompt Format


```
{
  "sample_id": "b4d89e...",
  "prompt_id": "a3f9...",
  "model": "meta–llama/Llama–3.3–70B–Instruct",
  "dataset": "Alpaca",
  "prompt_text": "What is the capital of France?",
  "longest_response": "...",
  "shortest_response": "...",
  "input_size": 8,
  "output_sizes": [12, 13, 14, ..., 10],
  "output_mean": 12.3,
  "output_std": 1.5,
  "output_percentiles": {"25": 11.0, "50": 12.0, "75": 13.5, "99": 15.0},
  "top_k": 50,
  "top_p": 0.9,
  "temp": 0.7,
  "category": "QA",
  "gen_time": 1.84
}
```


A.2 Prompt for baseline model

Prompt Format

Prompt:

Predict the length of your answer before answering my question, use the format:"[length(only one number)]: (your answer)". The length of your answer should be as close to the prediction of the length you give me as possible. Remember to follow the format. + question.

Example Prompt

Prompt:

Predict the length of your answer before answering my question, use the format:"[length(only one number)]: (your answer)." The length of your answer should be as close to the prediction of the length you give me as possible. Remember to follow the format. Who are you

Example Answer

Answer:

55: I'm an artificial intelligence model known as Llama. Llama stands for "Large Language Model Meta AI."

A.3 Loss curve of regression model

Shown in fig. 1fig. 2

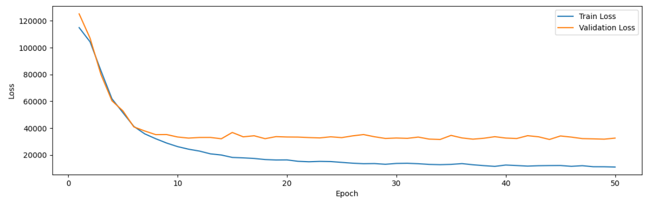

Figure 1: Loss curve of regression model on Llama-3-3B dataset

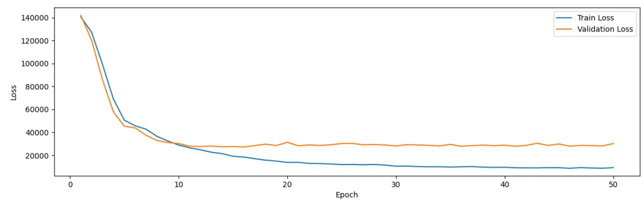

Figure 2: Loss curve of regression model on Llama-3-8B dataset

B Unified Models case

Example Test Case

```
print("\nExample predictions:")

example_prompt = "Introduce Nanjing Univerity In China." # input prompt here
example_model = "llama–3.2–3B" # you can choose model here
# regression results
reg_prediction = predict_response_length(example_prompt, example_model,
task='regression')
print(f"Regression prediction: Expected response mean length:
{reg_prediction['predicted_mean']:.2f} tokens, "
        f"Expected std: {reg_prediction['predicted_std']:.2f} tokens")
# classification results
cls_prediction = predict_response_length(example_prompt, example_model,
task='classification')
print(f"Classification prediction: P99 token length class:
{cls_prediction['predicted_p99_class']}, "
        f"Range: {cls_prediction['predicted_p99_range']}")
```