When Will the Tokens End? Graph-Based Forecasting for LLMs Output Length

Anonymous ACL submission

Abstract

Large Language Models (LLMs) are typically trained to predict the next token in a sequence. However, their internal representations often encode signals that go beyond immediate next-token prediction. In this work, we investigate whether these hidden states also carry information about the remaining length of the generated output—an implicit form of foresight (Pal et al., 2023). Accurately estimating how many tokens are left in a response has both theoretical and practical relevance. From an interpretability perspective, it reveals that the model may internally track its progress through a generation. From a systems perspective, it enables more efficient inference strategies, such as LLM inference via output-length-aware scheduling(Shahout et al., 2024).

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable ability to generate coherent text, but understanding what latent information they maintain during generation remains a challenge. A key question is whether an LLM internally tracks how much output remains to be produced. This is relevant both for interpretability—understanding a model's sense of progression—and for practical systems such as efficient request scheduling (Qiu et al., 2024; Zheng et al., 2023).

Prior work suggests that transformer hidden states may encode signals beyond immediate next-token prediction. For instance, Pal et al. (2023) showed that a single hidden state can predict several future tokens with notable accuracy, indicating that models internalize aspects of future output. Building on this, Shahout et al. (2024) used intermediate layer embeddings to estimate the number of tokens remaining in a response, identifying layers 10–15 as especially informative.

Accurately estimating the remaining output length offers practical benefits. It enables strategies like adaptive early stopping and intelligent scheduling in multi-user environments. A particularly promising use case is integration with Shortest Job First (SJF) scheduling (Akhtar et al., 2015; Fu et al., 2024), which minimizes latency by prioritizing shorter tasks. In the LLM setting, this allows systems like Orca (Mukherjee et al., 2023) or vLLM (Kwon et al., 2023) to reorder token generation queues dynamically to improve throughput and responsiveness.

Our contributions are:

- **An aggregation-based predictor** that combines hidden states from multiple transformer layers using element-wise operations (e.g., mean, sum) and predicts token-wise output length via a shallow feedforward network.
- **A Layerwise Graph Regressor** that treats each layer's hidden state as a node in a token-specific graph, using a GNN to model inter-layer dependencies for remaining token count prediction.
- **An experimental evaluation** on an instruction-following dataset using a state-of-the-art LLM.
- **An analysis of prediction accuracy** as a function of a token's position relative to the end of the sequence.

We further connect our results to existing interpretability work and discuss what they reveal about internal transformer representations.

2 Method

To predict the number of remaining tokens at each generation step, we consider the task as a regression problem. Let $\mathbf{h}_\ell^t \in R^d$ denote the hidden state (embedding vector) from the ℓ-th layer of the LLM at generation step t, where $\ell \in \{8, 9, \ldots, 15\}$. The prediction target is defined as $y^t = T - t$, where T is the total number of tokens in the generated sequence and t is the current position. The objective is to learn a function f such that:

$$\hat{y}^t = f(\mathbf{h}_8^t, \ldots, \mathbf{h}_{15}^t)$$

We choose to use hidden states from layers 8 to 15 based on empirical findings from TRAIL (Shahout et al., 2024), which showed that these intermediate layers achieve the lowest mean absolute error in output length prediction tasks.

We explore two model architectures for this task:

- **Aggregation.** This baseline follows the TRAIL methodology by leveraging internal hidden states from a large language model (LLM) to predict output lengths. Specifically, we extract token-level hidden states $\mathbf{h}_\ell^t$ from a selected set of layers $\ell \in \{8, \ldots, 15\}$ and aggregate them using a configurable element-wise operation such as mean, sum, or concatenation:

  $$\mathbf{z}^t = \text{Aggregate}(\mathbf{h}_{\ell_1}^t, \ldots, \mathbf{h}_{\ell_k}^t) \in R^d$$

  The aggregated vector $\mathbf{z}^t$ is passed through a lightweight feedforward network ϕ to produce a categorical prediction over discretized bins representing the number of remaining output tokens:

  $$\hat{y}^t = \phi(\mathbf{z}^t)$$

  The model is trained using a cross-entropy loss over these bins as in orig. During evaluation, we compute the expected value of the predicted length by weighting bin midpoints with softmax probabilities. This approach mirrors the core idea of TRAIL (Shahout et al., 2024) by reusing internal representations of the LLM without requiring end-to-end fine-tuning. The implementation supports aggregation modes including mean and sum. It operates purely on precomputed embeddings, ensuring low inference overhead.

- **Layerwise Graph Regressor.** We propose a graph-based regression model for predicting the number of remaining output tokens for each generated token. The model leverages the layerwise structure of transformer hidden states by constructing a token-specific graph where each node corresponds to the hidden embedding $\mathbf{h}_\ell^t \in R^d$ from a selected transformer layer $\ell = 8, \ldots, 15$.

  These embeddings form the node features $\mathbf{x} \in R^{L \times d}$, where L is the number of layers. Nodes are connected using a fully connected topology, resulting in an adjacency matrix $\mathbf{A}$ that captures all pairwise relationships between layers.

  A two-layer Graph Convolutional Network (GCN) is applied to this token-specific graph:

  $$\mathbf{x}^{(1)} = \text{ReLU}(\text{GCN}_1(\mathbf{x}, \mathbf{A}))$$

  $$\mathbf{x}^{(2)} = \text{ReLU}(\text{GCN}_2(\mathbf{x}^{(1)}, \mathbf{A}))$$

  The final node representations $\mathbf{x}^{(2)}$ are aggregated using *global mean pooling* to obtain a compact vector $\mathbf{v}^t \in R^{d'}$:

  $$\mathbf{v}^t = \text{MeanPool}(\mathbf{x}^{(2)})$$

  A fully connected regressor ψ then produces the predicted remaining length:

  $$\hat{y}^t = \psi(\mathbf{v}^t)$$

  This architecture captures inter-layer structural relationships without relying on attention mechanisms, offering a compact and expressive summary of a token's transformer-depth context. The model is trained using the Mean Absolute Error (MAE) loss between predictions $\hat{y}^t$ and ground truth y^t.

Experimental Setup

Dataset For our experiments, we use the Stanford Alpaca instruction-following dataset (Taori et al., 2023), focusing on the first 1,000 examples from the training split. Each example contains a user-issued prompt designed to elicit coherent and aligned responses from instruction-tuned models. Using the Meta-LLaMA-3-8B-Instruct model, we generate responses to these prompts and extract hidden states from selected transformer layers during generation. These hidden representations serve as features for downstream models, allowing us to investigate how internal model dynamics correlate with output sequence length.

Model We use the LLaMA-8B-Instruct model in inference mode to regenerate outputs for the selected prompts.

Training Details We train all models for up to 30 epochs using early stopping and adaptive learning rate scheduling. The optimizer used is AdamW with a learning rate of 1e-3 and a batch size of 16. All training is performed with mixed precision (AMP) to improve computational efficiency. We evaluate models using standard regression metrics, including Mean Absolute Error (MAE) and Normalized MAE (NMAE). For classification-based approaches, we additionally compute the expected value of the predicted output length from the softmax-weighted bin midpoints.

Evaluation Metrics We report the **Mean Absolute Error (MAE)** as our primary evaluation metric. MAE measures the average absolute difference between predicted and true values, providing an interpretable and scale-consistent indication of prediction accuracy:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} |\hat{y}_i - y_i|$$

where $\hat{y}_i$ and y_i represent the predicted and ground-truth number of remaining tokens at generation step i, respectively.This approach has also been adopted in previous studies, and we regard it as a valuable point of reference. (Shahout et al., 2024) (Qiu et al., 2024) To complement MAE, we also report the **Normalized Mean Absolute Error (NMAE)**:

$$\text{NMAE} = \frac{1}{N} \sum_{i=1}^{N} \frac{|\hat{y}_i - y_i|}{y_i}$$

This metric captures relative error, which is particularly informative when the target values (i.e., the number of remaining tokens) vary widely. To avoid division by zero, we exclude instances where $y_i = 0$.

NMAE is especially well-suited for length prediction tasks because it accounts for the scale of the target values. While MAE treats all errors equally, regardless of the true value's magnitude, NMAE penalizes errors relative to the ground truth. For example, an error of 5 tokens is more severe when the true value is 10 than when it is 100. By normalizing the errors, NMAE offers a more nuanced and scale-sensitive evaluation of model performance.

This is particularly important in settings where the target lengths span a wide range — from very short to very long continuations. In such cases, MAE tends to be dominated by absolute errors on longer sequences, potentially masking poor performance on shorter ones. In contrast, NMAE highlights proportional mistakes, which are often more meaningful in practical applications. For instance, overestimating by 5 tokens when only 10 remain may indicate a critical failure in generation control, while the same absolute error on a 100-token continuation is less problematic. We therefore hypothesize that NMAE provides a more balanced and interpretable signal for evaluating length prediction, especially when precise control over short outputs is important.

3 Results

The **Graph model** consistently achieves lower normalized mean absolute error (NMAE) compared to single-layer and aggregation-based baselines across all tested hidden dimensions. For example, at dimension 128, it reduces NMAE by 59% relative to the best baseline (from 0.9442 to 0.3871), and by 26% and 9.5% at dimensions 256 and 512, respectively.

Performance gains persist across different token thresholds. Notably, for sequences with 240 or more remaining tokens, the Graph model reduces NMAE from 0.0635 to 0.0271. In shorter ranges (e.g., [120, 240) and [80, 120)), the reduction is similarly substantial. Even in the most challenging short-token setting ([0, 40)), where MAE is comparable, NMAE drops by over 50%, suggesting improved proportional accuracy. These results highlight the model's robustness across the sequence space.

4 Discussion

Our results reinforce that hidden states in transformer models encode information not only about the next token but also about the overall progress of the generation process. The consistent advantage of the Graph model indicates that combining information across layers captures this signal more effectively than single-layer or pooled representations.

These findings empirically validate the hypothesis posed by Shahout et al. (Shahout et al., 2024), who suggested that integrating multiple layers could enhance predictions. Our model, by leveraging mid-layer embeddings, demonstrates that length-related information is distributed across depth and benefits from structured modeling.

This aligns with broader themes in interpretabil-

Method	MAE	NMAE	Hidden Dim	LTT	UTT
Graph (ours)	**27.04**	**0.3871**	128	0	512
TRAIL	30.98	0.9442	128	0	512
Hidden state aggregation	32.79	1.0374	128	0	512
Proxy Model	110.63	–	128	0	512
Graph (ours)	**23.11**	**0.6754**	256	0	512
TRAIL	28.30	0.9130	256	0	512
Hidden state aggregation	30.62	0.980	256	0	512
Proxy Model	111.96	–	256	0	512
Graph (ours)	**21.13**	**0.7847**	512	0	512
TRAIL	26.94	0.8675	512	0	512
Hidden state aggregation	28.24	0.9574	512	0	512
Proxy Model	110.37	–	512	0	512
Graph (ours)	32.43	**0.0271**	512	≥ 240	400
TRAIL	**19.31**	0.0635	512	≥ 240	400
Graph (ours)	21.47	**0.0133**	512	≥ 120	240
TRAIL	**11.84**	0.0869	512	≥ 120	240
Graph (ours)	29.05	**0.0896**	512	≥ 80	120
TRAIL	**18.19**	0.1836	512	≥ 80	120
Graph (ours)	17.10	**1.1093**	512	≥ 0	40
TRAIL	**14.17**	2.3888	512	≥ 0	40

Table 1: Prediction results across model types and hidden dimensions. (LTT) Lower Token Threshold is the minimum number of tokens remaining; (UTT) is the upper bound.

ity. Each layer may represent different levels of abstraction — from planning and discourse structure to local coherence. Our results suggest that LLMs implicitly maintain a sense of "how much is left," even though they are trained only to predict the next token. Similar to the "Future Lens" findings by Pal et al. (Pal et al., 2023), this foresight can be abstracted as a scalar — the number of tokens remaining.

We also observe that prediction quality varies with token position: the longer the remaining sequence, the stronger the signal. This suggests a potential transition in internal representations throughout generation, which could be further explored in future work.

Interestingly, we also observe cases where a single-layer baseline achieves a lower MSE (Mean Squared Error), while our Graph model yields better NMSE (Normalized MSE). This suggests that predictions based on single layer of hidden states may perform well in fixed-length continuations—e.g., being particularly good at forecasting a fixed offset like 40 tokens ahead—regardless of the total sequence length. In such cases, the layer appears to encode specific knowledge about short-term continuation length.

By contrast, our Graph model, though occasionally less precise at very specific token horizons, performs more consistently across a wide range of sequence lengths. The likely reason is that it integrates information from multiple layers—some of which may contribute noise to local estimates but improve the overall generalization.

The underlying cause might be the model's *fully connected layerwise structure*, which lacks a mechanism to weight or filter layer contributions dynamically. A more expressive mechanism such as attention, or even layer-wise routing, could allow the model to emphasize the most informative layers on a per-token basis. In this sense, attention would serve as a natural mechanism for layer selection, enabling more targeted use of useful depth-level signals while suppressing interference. Incorporating such inductive bias remains an exciting direction for future work.

Limitations

While our results are encouraging, our study has several limitations that suggest caution and point to directions for future work.

Second, our method predicts the number of tokens remaining, but not the content of those tokens. It is a coarse abstraction. There may be cases where the model's internal state captures rich information about upcoming content (as evidenced by Future Lens (Pal et al., 2023)), but predicting an exact length remains difficult — for instance, when the model is planning a response of "about two sentences." In such scenarios, our model may output only an approximate or average length. Additionally, we formulate length prediction as a regression problem; an alternative is to treat it as classification into length bins, as done by Shahout et al. (Shahout et al., 2024). While regression allows finer granularity, classification might yield more stable or interpretable outputs, especially in the presence of outliers.

Third, the reliability of the predictor degrades at extreme sequence lengths. We observed less accurate predictions for particularly long or short outputs. A practical system may need to estimate and report its own uncertainty in such cases. We did not explore confidence calibration or uncertainty estimation, which could be useful in downstream applications such as LLM scheduling — e.g., deferring a prediction if uncertainty is high.

In summary, while we demonstrated the feasibility of predicting token-level output length from hidden states in one setting, further research is needed to test the generality of the approach, improve robustness, and integrate such predictors into practical LLM systems. We also acknowledge that the dataset used in our study is relatively small, which may limit the generalizability of our findings. We hope our findings and methodology serve as a starting point for more work on latent structural knowledge in large language models. We hope our findings and methodology serve as a starting point for more work on latent structural knowledge in large language models.

Ethical Considerations

This research primarily involves analyzing a preexisting language model and does not directly raise severe ethical concerns. We worked with the Alpaca dataset (Taori et al., 2023), which consists of synthetic instruction-response pairs. Although the data was generated by a language model (OpenAI's `text-davinci-003`) and may contain biases or inaccuracies, our use of it is limited to probing model behavior rather than making deployable predictions that affect users. No personal or private information is included in the prompts or outputs.

We note that predicting remaining output length could be used in applications to allocate computing resources or moderate content (e.g., cutting off excessively long answers). If misused, such mechanisms might unfairly truncate or deprioritize certain user inputs. However, in our controlled study, we do not deploy any system — we only analyze performance offline. All experiments were conducted on a private compute environment; we did not involve human subjects or gather new personal data.

In terms of broader impact, improving LLM efficiency via length prediction could benefit users by reducing latency and resource use. However, one should ensure that scheduling based on length predictions does not inadvertently disadvantage complex or long but important queries. There is a minor environmental impact in training the predictors and running the LLM for experiments, but we limited our runs to a relatively small scale (1,000 prompts on an 8B model). We encourage future work to consider energy-efficient methods and to use renewable energy where possible.

Finally, we adhere to the ACL Ethics Policy: we cite the sources of our model and dataset, respect terms of use (LLaMA and Alpaca have appropriate licenses for research use), and open-source our code for transparency. We do not foresee direct harm from this specific research, but as always, further deployment of predictive systems should be tested for fairness and bias (e.g., does the model systematically underpredict lengths for certain types of content, and could that cause harm in a downstream application?).

References

Muhammad Akhtar, Bushra Hamid, Inayat Ur-Rehman, Mamoona Humayun, Maryam Hamayun, and Hira Khurshid. 2015. An optimized shortest job first scheduling algorithm for cpu scheduling. *J. Appl. Environ. Biol. Sci. , 5(12)42-46, 2015*, 5:42–46.

Yichao Fu, Siqi Zhu, Runlong Su, Aurick Qiao, Ion Stoica, and Hao Zhang. 2024. Efficient llm scheduling by learning to rank.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed

Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4.

Koyena Pal, Jiuding Sun, Andrew Yuan, Byron Wallace, and David Bau. 2023. Future lens: Anticipating subsequent tokens from a single hidden state. In *Proceedings of the 27th Conference on Computational Natural Language Learning (CoNLL)*, pages 548–560, Singapore. Association for Computational Linguistics.

Haoran Qiu, Weichao Mao, Archit Patke, Shengkun Cui, Saurabh Jha, Chen Wang, Hubertus Franke, Zbigniew T. Kalbarczyk, Tamer Başar, and Ravishankar K. Iyer. 2024. Efficient interactive llm serving with proxy model-based sequence length prediction.

Rana Shahout, Eran Malach, Chunwei Liu, Weifan Jiang, Minlan Yu, and Michael Mitzenmacher. 2024. Don't stop me now: Embedding based scheduling for llms.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7.

Zangwei Zheng, Xiaozhe Ren, Fuzhao Xue, Yang Luo, Xin Jiang, and Yang You. 2023. Response length perception and sequence scheduling: An llm-empowered llm inference pipeline.

A Appendix

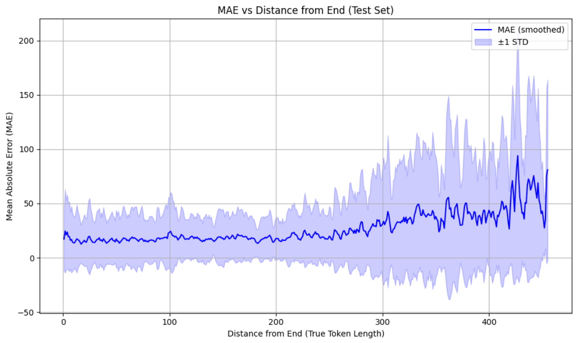

Figure 1: Mean Absolute Error (MAE) as a function of distance from the end of the sequence.

Figure 1 illustrates how prediction accuracy improves as generation progresses. The Mean Absolute Error (MAE) decreases toward the end of the sequence, indicating that the model's internal representations become increasingly informative for estimating the remaining length.