RLMEDUSA: REINFORCEMENT LEARNING FOR MULTIPLE DECODING HEADS TO ACCELERATE LLM INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Traditional transformer inference requires step-by-step generation of tokens in which each step is dependent on the previous one, presenting a bottleneck in inference speed. The Medusa technique used LoRA fine-tuning to train multiple decoding heads, each predicting a different number of tokens in advance in order to generate multiple tokens in parallel as part of a draft model that the base model can verify. In this paper, we propose a reinforcement learning based approach to training multiple decoding heads. Our method proposes a reward model scheme that leverages feed-forward networks to estimate token probabilities based on context hidden states and candidate token embeddings. We provide commentary comparing our interpretation of reinforcement learning in language modeling research and how this contrasts with traditional, RLHF-centric interpretations, as well as discuss our experiments with RLMedusa.

INTRODUCTION

Recent years have seen a surge in the capabilities of large-scale language models (LMs) across a wide variety of natural language processing tasks, including machine translation, summarization, and interactive dialogue systems (1; 2; 3). Much of this progress can be attributed to advances in model architectures such as the Transformer (4), the expansion of training data (5; 6), and innovations in hardware accelerators (7; 8). Despite these improvements, the autoregressive nature of many state-of-the-art LMs presents a significant bottleneck for real-time applications. Traditional left-to-right decoding necessitates generating each token sequentially, which can be computationally expensive and slow, especially for lengthy sequences (9; 10). Consequently, there is a growing interest in strategies to accelerate inference without sacrificing fluency or coherence (11; 12).

A promising approach to mitigating slow inference is multi-token parallel prediction—where the next k tokens are predicted in fewer autoregressive steps (13). Notably, *Medusa* (14), introduced by Together AI, demonstrated that one can fine-tune lightweight "heads" on top of a base language model to predict subsequent tokens at different offsets in parallel. This idea draws inspiration from prior work on non-autoregressive generation (15; 16), where partial or full parallel decoding has been explored for tasks such as machine translation. By expanding a "next-token predictor" into a family of "multi-step lookahead" modules, Medusa-style systems can generate multiple tokens concurrently, leading to throughput improvements on modern hardware. The practical impact of such methods has become evident with integration into widely used inference frameworks like vLLM (17) and TensorRT-LLM (18).

However, effectively training multiple heads to predict tokens at different positions—while maintaining the base model's linguistic fidelity—remains a core challenge (19; 14). Previous approaches have largely relied on supervised learning or knowledge distillation from the base model's distributions (20). Although these methods work well in practice, they often require careful hyperparameter tuning and may suffer from exposure bias when predicting tokens far into the future (21; 22). Reinforcement learning (RL) offers a compelling alternative for optimizing multi-step policies, as it enables direct optimization of sequence-level objectives that align with inference speed and quality (23; 24). Indeed, RL-based approaches have a long history in text generation, from policy gradient training of dialogue systems (25; 26) to reward shaping for summarization tasks (27). More

recently, Proximal Policy Optimization (PPO) has become a de facto standard for large-scale RL optimization of language models, offering stability and robustness (28; 29).

In this work, we propose an RL-based methodology for training multiple decoding heads, each capable of predicting tokens at an offset of k steps from the current position in the sequence. Our method is conceptually inspired by Medusa (14) but introduces a two-phase reinforcement learning process designed to simultaneously capture accurate token distributions and preserve the stylistic and contextual fidelity of the base model. Specifically, we first fit a reward model to estimate the probability of a candidate token occurring k steps ahead, given the current hidden state. This reward model is trained on sampled trajectories produced by a base GPT-style language model (1), using a mixture of likely and unlikely token candidates to ensure broad coverage of plausible outputs. We then fine-tune a "second decoding head" via Proximal Policy Optimization, leveraging the reward model to guide the selection of actions (i.e., tokens) that are most likely to align with the base model's multi-step predictions. A KL divergence penalty from the base model is incorporated to retain high-quality, human-like text generation (28; 30).

By combining an RL objective with a carefully constructed reward model, our approach aims to address two key limitations of prior multi-offset decoding systems: (1) insufficient exploration of tokens that are underrepresented by standard supervised training, and (2) excessive drift in style or content when the auxiliary heads deviate from the base model's distributions. We show that this method leads to a more robust "two-tokens-ahead" predictor, reducing the number of autoregressive steps and thereby accelerating inference. While conceptually straightforward, our experiments demonstrate that RL training can effectively adapt large language models to multi-step prediction tasks without degrading their overall coherence.

METHODOLOGY

PROBLEM FORMULATION

Consider a standard language modeling setting where x_i represents the i'th token in a sequence, we want to learn a distribution $p(x_{t+1}|x_{1:t})$, where $x_{1:t}$ represent the sequence $x_1, x_2, \dots x_t$. However, a restriction of this is that tokens must be generated one at a time. To make sampling quicker, we seek to train two distributions, $p_1(x_{t+1}|x_{1:t})$ and $p_2(x_{t+2}|x_{1:t})$, that can be executed independently and concurrently in order to speed up inference. For our purposes, we assume we possess p_1(as most modern transformer architectures represent this distribution), and we seek to learn p_2.

OVERVIEW OF REINFORCEMENT LEARNING APPROACH

We seek to train a reward model $r(e, h)$, where e represents the embedding of a proposed token and h represents the final hidden state of a sequence after processing a sequence of tokens $x_1, x_2 \dots x_t$ through a transformer. We want our reward model $r(e, h)$ to take in this hidden state as well as the proposed embedding of the $t + 2$'th embedding, and output the probability of this token actually being the $t + 2$'th token in a sequence generated by p_1(as we would like p_2 to follow p_1's distribution when 2 tokens are generated). We model this via a standard feed-forward neural network outputting a singular scalar value from 0 to 1.

Now, utilizing this reward function, we initialize p_2 to p_1 and began a training process on the final decoding head of p_2 (note that all other weights are frozen). Assume we have context $x_{1:t}$, and let h represent the final hidden state of this context after processing via a transformer(in our case, we use GPT 2). The RLMedusa objective R can be written as:

$$R(e, h) = r(e, h) - \beta KL(p_2(x_{t+2}|x_{1:t}), p_1(x_{t+1}|x_{1:t})). \tag{1}$$

Note that h represents the final hidden state from processing $x_{1:t}$ through the transformer p_1(though since we freeze all non-decoding head weights of p_2, this should also equal the final hidden state in p_2). At each step e is selected by taking the argmax of p_2's output token probability vector and finding the embedding of this token in p_1's embedding table (which is identical to p_2's embedding table).

TRAINING METHODOLOGY AND DATA COLLECTION

To train the reward function, we desired to create a diverse mix of (proposed $t + 2$'nd token, hidden state) pairs in order. We utilized the Hugging Face Wikipedia dataset and generated GPT-2 completions of 20 tokens for each used sample in the dataset. At each generation step, we maintained the final hidden state, and the probabilities of 9 tokens. These 9 tokens include the highest probability token, 3 of the top 50 probability tokens, and 5 random tokens. This is done in order to ensure that the reward model has a good understanding of which types of tokens have high weightage and low weightage in the final vector. Depsite this, with more compute we would have preferred to train our reward model on more steps and with every candidate token at each step. Then, to train the reward model, we consider pairs, for some k, of the final hidden state for the first k tokens and the token probability for a candidate $k + 2$'nd token in one of our generations. We can then pass in the embedding of this candidate token and the hidden state to our reward model feed-forward neural network, and then use an MSE Loss to compare this with the real token probability and update the reward model.

To train the language model/policy, we utilize our RLMedusa objective on p_2's completions for more Wikipedia base prompts. We train this with gradient ascent.

RESULTS

Despite promising reward curves, we were unable to see strong generations from our multiple decoding heads approach. We suspect this to be the result of constrained compute, as we were only able to train p_2 for 40 epochs on an NVIDIA RTX 2080. In addition, our reward model was trained an Apple M3 Chip. We provide an example of our generations below:

Base Context: The early signs of AML are often vague and nonspecific , and may be similar to those of influenza or other common illnesses . Some generalized symptoms include fever , fatigue , weight loss or loss of appetite , shortness of breath , anemia , easy bruising or bleeding , petechiae (flat , pin @-@ head sized spots under the skin caused by bleeding) , bone and joint pain , and persistent or frequent infections .

RLMedusa Continuation: TheThe symptoms symptoms of of AM AMLL are are similar usually to to those those of of influenza influenza or or other other common common illnesses illnesses . . Some Some generalized generalized symptoms symptoms include include fever fever , , fatigue fatigue , , short shortnessness of of breath breath , , an anemiaemia , , easy easy bruising bruising or or bleeding bleeding , , pet petechechiaeiae ((flat flat , , pin pin @ @−@ @ head head sized sized

While some behavior may be attributed to GPT-2's inabilities compared to modern language models, the repeated tokens clearly indicate insufficient distinction between p_1 and p_2. In the commentary section we discuss the β term on the KL divergence and the more stable training that we observed as a result of it.

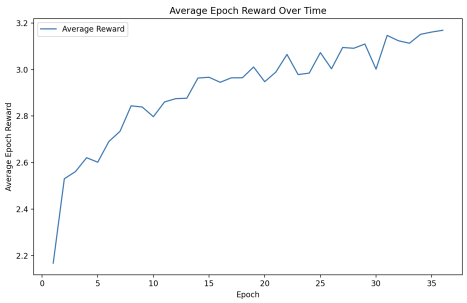

Figure 1: RLMedusaReward Curve

COMMENTARY

A NOTE ON THE KL TERM

The KL term should not be interpreted as if we desire p_1 and p_2 to have similar output distributions. p_1 and p_2 are different distributions trying to model different concepts. However, including a KL term often assists in preventing overfitting on the reward function and unstable training behavior. Note that we desire to keep our β coefficient on the KL term small so it does not misguide the model to train this model.

POTENTIAL REWARD MODELING ALTERNATIVES

In compute-rich environments, instead of fitting a reward model $r(e, h)$, one could directly process the input sequence via p_1, generate two more tokens, and simply reward the model based on the probability of the token corresponding to that selected by p_2. This is much more expensive as running transformer inference is requires orders of magnitudes more FLOPs than inference from our small FFNN for $r(e, h)$, it can provide a degree of exactness that may be plausible in scenarios where one has access to these extra FLOPs. This removal of an approximated reward function has parallels with early, small-scale experiments in RLHF (31), where instead of reward model fitting, human feedback was used as a brute force reward mechanism for each step in the training process. While both direct transformer inference to access token probabilities and direct human feedback are clearer reward signals, they are difficult to scale which is why many elect for fitting a reward model, though the latter is more viable to brute force.

INTERPRETATION AND COMPARISON WITH OTHER RL METHODS IN NLP

RLMedusa's direct analogies with traditional RL make it an intuitive method to understand when considered at a token level. The final hidden state of the context can be interpreted as a state in a traditional RL context, as it is a information-rich, comprehensive representation of the base tokens in a sequence. In addition, the selected token/embedding represents an action in the traditional RL interpretation, as policy (e.g. p_2) must select an appropriate action/token given the current state (context hidden state). Most reinforcement learning work in the realm of language modeling surround RLHF, which utilizes on a multi-arm bandit interpretation of the task at a response level. While token-level RL has not gained much traction in language modeling due to its inability to model larger concept rewards, we believe that token level RL is a natural extension of the Medusa method. Splitting up token generations amongst different heads calls for fine-grained rewards on exactly how each head should interpret a given context, as they each have different tasks and need to generate different types of tokens.

REFERENCES

[1] T. Brown, B. Mann, N. Ryder, *et al.*, "Language Models are Few-Shot Learners," in *NeurIPS*, 2020.

[2] C. Raffel, N. Shazeer, A. Roberts, *et al.*, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," *JMLR*, 2020.

[3] J. Zhang, Y. Zhao, M. Saleh, *et al.*, "PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization," in *ICML*, 2020.

[4] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention Is All You Need," in *NeurIPS*, 2017.

[5] L. Gao, S. Biderman, S. Black, *et al.*, "The Pile: An 800GB Dataset of Diverse Text for Language Modeling," *arXiv preprint arXiv:2101.00027*, 2021.

[6] H. Touvron, T. Lavril, G. Izacard, *et al.*, "LLaMA: Open and Efficient Foundation Language Models," *arXiv preprint arXiv:2302.13971*, 2023.

[7] N. Jouppi, C. Young, N. Patil, *et al.*, "In-Datacenter Performance Analysis of a Tensor Processing Unit," in *ISCA*, 2017.

[8] S. Rajbhandari, J. Rasley, O. Ruwase, *et al.*, "ZeRO-Infinity: Breaking the GPU Memory Wall for Extreme Scale Deep Learning," in *SC*, 2021.

[9] A. Fan, E. Grave, A. Joulin, "Reducing Transformer Depth on Demand with Structured Dropout," in *ICLR*, 2020.

[10] J. Gu, J. Bradbury, C. Xiong, *et al.*, "Non-autoregressive Neural Machine Translation," in *ICLR*, 2018.

[11] J. Li, Y. Liang, D. Li, *et al.*, "Accelerating Autoregressive Decoding of Transformers via Token-level and Phrase-level Caching," in *EMNLP*, 2022.

[12] E. Cho, W. Chang, S. Yun, "Mixing Past and Future Context for Autoregressive Generation," in *ACL*, 2023.

[13] S. Welleck, I. Kulikov, S. Roller, *et al.*, "Neural Text Generation with Unlikelihood Training," in *ICLR*, 2020.

[14] Together AI, "Medusa: Parallel Multi-Offset Decoding for Efficient Language Model Inference," *arXiv preprint*, 2023.

[15] J. Gu, Y. Bai, S. Chang, *et al.*, "Fully Non-Autoregressive Neural Machine Translation: Tricks of the Trade," in *ACL*, 2019.

[16] Y. Qian, Y. Bisk, "Teaching Models New Tricks: A Study on Updating Transformer Models to Predict Non-autoregressive Outputs," in *EMNLP*, 2021.

[17] G. Xiao, G. Zeng, J. Tao, *et al.*, "vLLM: A High-Throughput and Memory-Efficient Inference and Serving Engine for LLMs," *GitHub Repository*, 2023.

[18] NVIDIA, "TensorRT-LLM: High-Performance Deep Learning Inference Optimizer and Runtime Library," *NVIDIA Developer Blog*, 2023.

[19] X. Ren, J. Liu, J. Chen, *et al.*, "Distilling Multi-offset Language Models for Resource-Constrained Devices," *arXiv preprint*, 2022.

[20] G. Hinton, O. Vinyals, J. Dean, "Distilling the Knowledge in a Neural Network," *arXiv preprint arXiv:1503.02531*, 2015.

[21] S. Bengio, O. Vinyals, N. Jaitly, *et al.*, "Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks," in *NeurIPS*, 2015.

[22] M. Ranzato, S. Chopra, M. Auli, *et al.*, "Sequence Level Training with Recurrent Neural Networks," in *ICLR*, 2016.

[23] D. Bahdanau, P. Brakel, K. Xu, *et al.*, "An Actor-Critic Algorithm for Sequence Prediction," in *ICLR*, 2017.

[24] R. Paulus, C. Xiong, R. Socher, "A Deep Reinforced Model for Abstractive Summarization," in *ICLR*, 2018.

[25] J. Li, W. Monroe, T. Shi, *et al.*, "Adversarial Learning for Neural Dialogue Generation," in *EMNLP*, 2017.

[26] N. Jaques, S. Gu, D. Bahdanau, *et al.*, "Sequence Tutor: Conservative Fine-Tuning of Sequence Generation Models with KL-control," in *ICML*, 2017.

[27] N. Stiennon, X. Ouyang, J. Wu, *et al.*, "Learning to Summarize from Human Feedback," in *NeurIPS*, 2020.

[28] J. Schulman, F. Wolski, P. Dhariwal, *et al.*, "Proximal Policy Optimization Algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[29] X. Ouyang, J. Wu, E. Ward, *et al.*, "Training Language Models to Follow Instructions with Human Feedback," *arXiv preprint arXiv:2203.02155*, 2022.

[30] D. M. Ziegler, N. Stiennon, J. Wu, *et al.*, "Fine-Tuning Language Models from Human Preferences," *arXiv preprint arXiv:1909.08593*, 2019.

[31] Timo KaufmannB, Sarah Ball, Jacob Beck, *et al.*, "On the Challenges and Practices of Reinforcement Learning from Real Human Feedback," *Manuscript*, 2023.