Interactive Text Games: Lookahead Is All You Need!

Hosein Rezaei, James Walker, Frank Soboczenski
University of York
{hosein.rezaei, james.walker, frank.soboczenski}@york.ac.uk

Abstract

The cross-modal grounding of LLMs has recently garnered significant attention, while grounding them in textual interactions has been less explored. As the first of its kind, the GLAM framework utilises LLMs as agents in interactive text-based games to investigate their grounding capabilities. However, it faces the challenge of low computational efficiency, which hinders further experiments. This paper proposes the use of Lookahead models for action selection, demonstrating through empirical results that the approach can substantially improve training speed, achieving performance gains relative to the size of the action space.

1 Introduction

A well-known limitation of Large Language Models (LLMs) is that their language is grounded only in textual contexts and not in real-world phenomena (Bender and Koller, 2020; Harnad, 2024). Thus, researchers are trying to ground them into perception, e.g. visual modalities (Reich and Schultz, 2024; Li et al., 2024b) and 3D environments (Liu et al., 2024; Li et al., 2024a). However, opposing viewpoints argue that learning meaning from text alone is still valuable (Pavlick, 2023; Lyre, 2024; Bommasani et al., 2022). An intermediate approach hypothesises that grounding in unimodal text is beneficial but not in raw sequential form, rather, in goal-oriented interactions (Chai et al., 2019), or as is called, conversational grounding (Shaikh et al., 2024).

A recent attempt in this regard is *Grounded LAnguage Models* (GLAM) (Carta et al., 2023) that uses LLMs as agents to play an interactive text-based game and examines their language grounding capabilities. In a Reinforcement Learning (RL) setup, a prompt is created including the goal, hints, observations, and a final question about the next step of the game. The agent is expected to select the next action, not by generating an output but by predicting the probability of action tokens. In fact, the LLM ranks a set of potential responses (actions). It then uses game rewards for parameter optimisation. So, through textual interaction with the environment, the agent learns what different words mean in terms of functionality. However, this approach suffers from computational inefficiencies, making further research in this direction practically challenging.

The main reason behind this is that GLAM requires a full LLM forward pass to determine the rank of each action. This stems from the autoregressive nature of LLM's, in which billions of computations are performed in each run, just to predict a single next token. Intuitively, this effort seems useful for guessing which tokens might appear at subsequent positions. Although these guesses are unreliable for generating responses (since they overlook dependencies between tokens), they can still be useful for ranking, because they help filter out many tokens of vocabulary that are unlikely and assign higher scores to the more probable tokens.

This paper examines the above idea by proposing efficient variations of Lookahead LLMs (Xia et al., 2024), where they predict not only the next immediate token, but also the second, third, ... up to K next tokens. Using future tokens, the likelihood of all actions can be approximated with fewer forward passes. Analytically, it reduces the training time of GLAM by a factor of the number of actions. The experiments presented here demonstrate that a more than 2x improvement is achieved.

The contributions of this paper are as follows.

- Novel efficient variants of Lookahead LLMs are proposed that can be used to predict multiple future tokens in one forward pass.
- The use of Lookahead LLMs is proposed to approximate the rank of a set of potential responses and is demonstrated in text-based games for interactive language grounding.

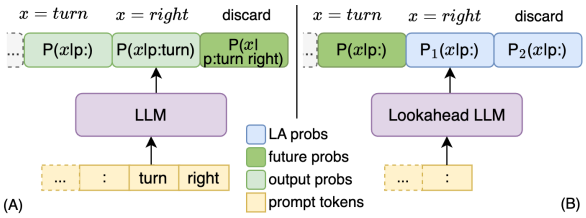

Figure 1: A) GLAM runs LLM once per action, looking up action tokens in output. B) Using Lookahead LLM, the model is called once, and tokens of all actions are queried in the output. The dotted sections show previous tokens which are removed for the sake of space.

2 Background

Using LLMs as agents in interactive games has become a popular trend (Hu et al., 2024). However, few studies address grounding (Ichter et al., 2023; Lin et al., 2024), and even fewer focus on unimodal text-based games like GLAM. Most of the works mentioned above use LLM-generated responses to extract valid actions. In contrast, GLAM directly uses output probabilities to assess the likelihood of actions and samples from them. In this respect, it is the only and first of its kind. A similar study is (Yao et al., 2020) however, it uses LLM to generate actions and then uses a Deep Reinforcement Relevance Network (DRRN) for ranking.

As discussed in Sections 1, and 3, GLAM's long runtime limits experimentation with larger LLMs and games with larger action spaces, which may contribute to overfitting and hinder language grounding improvements. To address these limitations, this work proposes the use of Lookahead LLMs, an active area of research also known as Speculative Decoding (SD) (Xia et al., 2024) or Parallel Decoding (Santilli et al., 2023). Most of these approaches aim to improve efficiency of inference and generation (Xia et al., 2024). Their common paradigm, Guess-And-Verify, drafts future tokens first and later verifies them, either by the same drafter model (Self Drafting) or with a more powerful LLM (Independent Drafting).

Nevertheless, not all works are considered in the survey. For example, (Qi et al., 2020) adds K self-attention blocks to predict K future tokens, increasing the size of the model. To reduce GPU load, Skippy Simultaneous Speculative Decoding (S3D) (Zhong and Bharadwaj, 2024) appends K masked tokens to the prompt and skips some mid-layers for cost-effective drafting. However, it also incorporates Tree Attention, adding complexity.

Although most SD proposals use an autoregressive drafter, ParallelSpec (Xiao et al., 2024) uses Lookahead models for drafting. Similarly to one of the models proposed in this study, it extends the input with K additional mask tokens so that it outputs the same number of extra tokens. The output is then compared with that of a target model to compute loss in a knowledge distillation setup.

(Kim et al., 2024) studied Blockwise Parallel Decoding (BPD) (Stern et al., 2018) improving its quality with two refinements. However, of particular relevance to ours, it did not alter the Lookahead drafter, consisting of K+1 extra layers on top of the decoder. Similarly, LlamaMultiToken (Gloeckle et al., 2024) splits the N attention blocks into two sets of size K and $N - K$, the first being used for future tokens and the latter for the original operation of the model. Then it uses multiple heads with separate losses to optimise the parameters.

Overall, the above efforts deal with various levels of complexity, mainly because their major concern is generation. However, in this paper, the main concern is obtaining multiple future predictions to increase ranking speed via approximation via *simpler* and *more efficient* models.

3 Methodology

In order to choose the next action in each step, GLAM creates one prompt per action and runs the LLM to compute the exact probability of each token in each action given the prompt (containing the goal and observations); see Figure 1. The formal definition of the problem is the same as provided in Section 3.1 of (Carta et al., 2023), but simply put, considering $\mathcal{A}$ as the set of actions, the probability of each $a_i \in \mathcal{A}$ is calculated by Equation 1.

$$\mathbb{LP}_{LLM}(a_i|p) = \sum_{j=0}^{|a_i|} log\mathbb{P}_{LLM}(w_j|p, w_{<j}) \quad (1)$$

where $|a_i|$ is the length of the ith action, w_j is the jth token in a_i, and p is the prompt. So, for each iteration over the sum, a separate token position must be included in the input. This makes the number of input tokens on the order of $O(|\mathcal{A}| \times \max_{a_i \in \mathcal{A}} |a_i|)$, which in turn affects both the required number of forward passes and memory.

Instead, using Lookahead LLMs, the probability of each action is approximated with Equation 2:

$$\mathbb{P}_{LLM}(a_i|p) \approx \sum_{j=0}^{|a_i|} \mathbb{P}_{LA,j}(w_j|p) \quad (2)$$

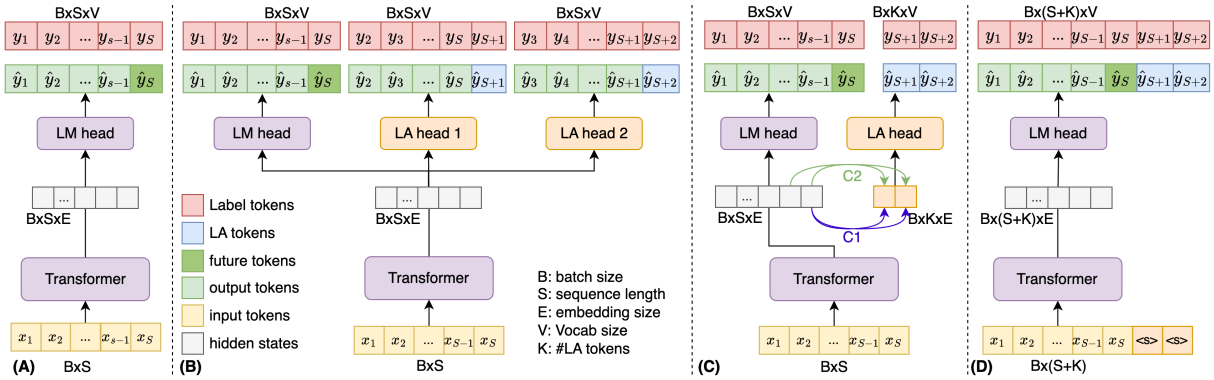

Figure 2: (A) a non-LA LLM. (B) The LA model for $K = 2$, the language modelling head is replicated twice. (C) LAA and $LAA2$ are similar, but the former uses K replicates of last hidden state (C1), while the latter uses the last K hidden states (C2). (D) LAE has no extra head but extends the input with K special tokens, thus outputs K extra tokens. Note that in all these figures, labels $y_i = x_{i+1}$ so y_S is the first token not present in the input.

where $\mathbb{P}_{LA,j}$ means the probability of jth next token given the prompt, e.g. $\mathbb{P}_{LA,0}$ is that of immediately next token, $\mathbb{P}_{LA,1}$ is that of the second next token, and so forth. Using this mechanism, the number of forward passes required to compute all $\mathbb{P}_{LA,j}$ is on the order of $O(\lceil \frac{\max_{a_i \in \mathcal{A}} |a_i|}{K} \rceil)$ and for the special case where K is greater than the maximum length of actions, i.e. $(K \geq \max_{a_i \in \mathcal{A}} |a_i|)$, a single forward pass would suffice, $O(1)$. Note that the log-likelihood is also omitted compared to Equation 1, GLAM uses it to avoid multiple normalizations, but this may overweight lower-probability actions. (see Appendix B for more details).

3.1 Lookahead LLMs

The main objective of the current research is to design the Lookahead feature with minimal complexity and overhead. To achieve this purpose, the LLM architecture is altered in four different ways, as illustrated in Figure 2.

1. In the simplest form, the language modelling head (LM in Figure 2) is repeated K times for each future position. The input to each head is the same as the original (Figure 2.B). The dataset is fetched in a way that the labels for each head are shifted right, thus the last position of each head is trained on, and will predict the ith next token. This model is named *LA* (LookAhead). Its main downside is that the LM head is typically large (depending on the vocabulary size, e.g. 30K) and, when replicated, the model size increases substantially. This is undesirable particularly because only the very last position of the output of each head is needed and the rest are discarded.

2. To address the aforementioned issue and to reduce model size and computational cost, the LM head is replicated only once and fed with a smaller input (Figure 2.C1). Assuming that the hidden states for the last token are informative enough to predict the next K tokens, it is replicated K times and used as input for the extra head. The output will then be a sequence of length K, each of which predicts one Lookahead token. This model is named *LAA* (LA with Additional head).

3. As another variation of the above model, it is possible to include the last K positions of hidden states as input to the new head. This is based on the assumption that the last K positions in the hidden state are more informative to predict the next K tokens. This model is named *LAA2* (Figure 2.C2).

4. The last model does not introduce extra heads, but extends the input with K additional positions, manipulated by special tokens, so that it outputs extra predictions. This is similar to (Xiao et al., 2024) but they have trained the model using knowledge distillation from a target model. In contrast, this variation simply fetches K extra tokens from the dataset as labels for the new positions and computes the loss as in the original LLM. This model is named *LAE* for Extended input (Figure 2.D).

4 Experimental Setup

To prototype the above architectures, nanoGPT[1] is chosen as the base model because it is easy to extend, with training data and algorithm ready to run.

[1] A LLM developed primarily for educational purposes, see https://github.com/karpathy/nanoGPT

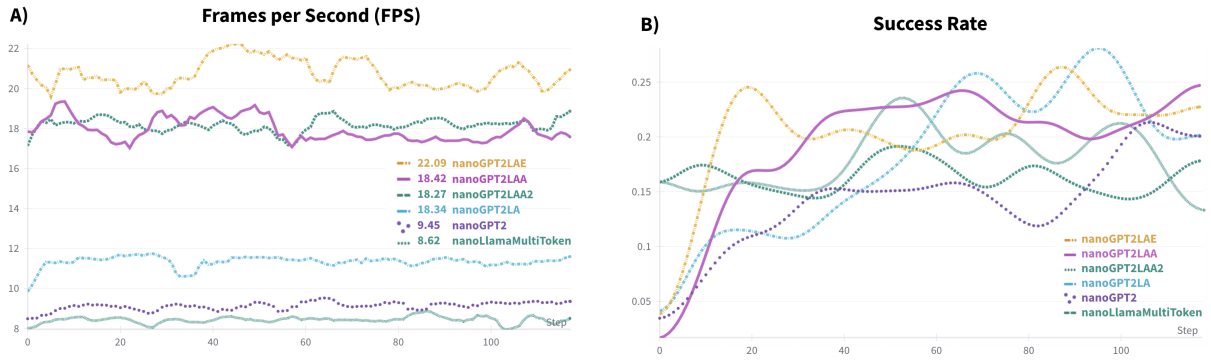

Figure 3: A) The speed of training models in GLAM, measured by FPS (frames per second) for a single run, the higher FPS means faster training. B) The success rate of the same models.

The original nanoGPT, together with four Lookahead models (explained in Section 3.1 and depicted in Figure 2) are pre-trained from scratch using the OpenWebText dataset (Peterson et al., 2019) on the GPT2 scale to fit within a limited budget. Also, as a state of the art, LlamaMultiToken (Gloeckle et al., 2024) is implemented on top of nanoGPT, hence the name *nanoLlamaMultiToken* and trained with the same scale and data as above. For clarity of presentation, K is set to 2 in Lookahead models. The technical details and results of the pretraining are reported in Appendix A.

The models were then deployed in the GLAM main experiment, after integrating lookahead functionality for ranking actions using a single forward pass. To explain it in more detail, the main GLAM experiment runs 32 instances of the BabyAI-Text game environment in parallel. At each step, six prompts are generated per game, one for each of the six actions, resulting in $32 \times 6 = 192$ prompts. For the LA models introduced earlier, this reduces to 32 prompts total, since they can predict up to $K + 1 = 3$ future tokens, and all BabyAI-Text actions are shorter than three tokens. Thus, a single prompt per game suffices.

Prompts are then batched and sent to the LLM; its output logits are used to compute action probabilities by looking up the relevant tokens and applying either Equation 1 or 2 for non-LA and LA models, respectively. An action is sampled from the resulting distribution and executed in each game. The rewards are then used to optimize the LLM and calculate success rates. The rest of the setup mirrors GLAM, except for batching parameters: a $batch\_size$ of 64 and $mini\_batch\_size$ of 16 were found to avoid out-of-memory errors in all experiments.

5 Results

The main metric for the speed of training is *FPS* (frames per second), which represents the number of steps per second the agent can perform in the game. As shown in Figure 3.A, it increases from 9 for non-LA model to a range of 11 to 20 for LA models, showing more than a 2x improvement. The LAE, LAA, and LAA2 models have gained better FPS compared to LA most probably because they have added less overhead to the number of parameters (see Table 2). This negative correlation between model size and FPS highlights the need for efficient models.

A notable observation is that the *nanoLlamaMultiToken* model performs worse than the non-LA model. This can potentially be explained by its architectural design, which introduces computational overhead. Specifically, the model splits the hidden states into multiple segments, feeds them to different layers, and then concatenates their outputs back into a single tensor. This split–recombine operation is executed at every iteration during the forward pass, thereby increasing the overall computational load. While theoretically plausible, further empirical investigation is required to validate this explanation.

Another metric is the *Success Rate* which represents the performance of the agent in the game. Figure 3.B does not show a significant change in this metric, demonstrating that the approach has not affected performance negatively. However, the LA models have achieved a better success rate compared to non-LA models. Considering both measures, the LAA model seems the best performing one, but this has to be further verified after instruction fine-tuning, and Reinforcement Learning from Human Feedback (RLHF).

6 Conclusion

Based on the analysis provided in Section 3, the performance gain is expected to be on the order of action space size (6x for the case of GLAM), however, the 2x speed up in the empirical results reinforces the importance of model size as a determining factor. Preliminary experimentation with Science World environments (Wang et al., 2022) that contain more actions further revealed the advantage of this approach. Even with a fixed-size action space, the improvement in running time provides the opportunity to run experiments for more steps, try larger LLMs, and employ parallel computation mechanisms. These results are limited by current GPU resources, but its advantages would be clearer with more powerful hardware.

Finally, the idea of using LA models for approximated ranking can be applied in other applications in which LLM are used not for generating a response, but for ranking a set of potential responses.

The project code has been made open source[2].

7 Future Works

The models in this study are decoder-only, but the same approach is implemented on encoder-decoder models like Flan-T5 in the Huggingface Transformers, with ongoing work to pre-train and deploy them in GLAM, both in the scale of nanoGPT as well as in the scale of T5-large. This then paves the way to perform a fair comparison between LA and non-LA LLMs in BabyAI-Text and games with larger action space.

Additionally, speculative decoding techniques could be applied to the proposed LA models to assess improvements in generation quality. Finally, the overall approach may also benefit other applications in which LLMs are used to rank responses rather than generate them.

Limitations

The success rate of models is currently low; however, it is worth considering that the original GLAM has also struggled with this metric and even with Google Flan-T5-Large (783M) it hardly achieved the top success rate of 1. Moreover, models presented in the current work are not fine-tuned on any instruction dataset or human preference feedback, and their knowledge is limited to just pre-training corpus. However, even without fine-tuning, the Lookahead models achieved a faster speed and an on-par success rate compared to non-LA models. It is planned to perform fine-tuning and study its effect as well.

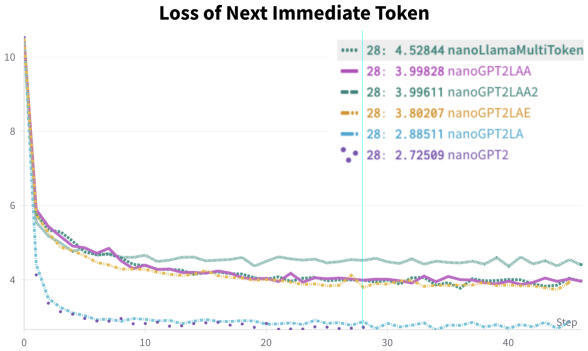

Figure 4: The loss of pertaining models reported only on the next immediate token after the prompt.

Predicting lookahead tokens imposes a negative impact on the quality of the next-immediate token compared to the same position predicted by a non-LA LLM. To confirm this intuition, the loss is tracked for each position individually during pre-training. The result is shown in Figure 4. As expected, all Lookahead models faced a higher loss, but the difference can be considered acceptable given the fact that generation is not the primary concern in GLAM design. Moreover, applying the verification phase (of the Guess-And-Verify paradigm) that is normally done in Speculative Decoding approaches might remedy this limitation.

The idea of this paper is examined in tiny-scale LLMs. On larger scales, though, the overhead on the number of parameters imposed by the first LA model is considerable, because it replicates the LM head, and that head is very large for fully-fledged LLMs. However, the other three proposed models are very efficient in this regard.

More broadly, although the aim of GLAM is language grounding in conversational interactions, the current work only proposes a novel way to boost training. However, this speed up has facilitated further investigations and experiments to measure the extent of impact on grounding as the ultimate goal. The work is in progress in this regard.

Most of the above limitations are primarily due to limited access to GPU infrastructures. The available resources were either 3xA40 40GB or 2xH100 PCIe 80GB each on a maximum of 2 days for a single run.

[2]For models based on nanoGPT see https://github.com/HRezaei/nanoGPT, for models based on T5 in Transformers, see https://huggingface.co/hrezaei/T5LA

References

Emily M. Bender and Alexander Koller. 2020. Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198, Online. Association for Computational Linguistics.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, et al. 2022. On the Opportunities and Risks of Foundation Models. *Preprint*, arXiv:2108.07258.

Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. 2023. Grounding Large Language Models in Interactive Environments with Online Reinforcement Learning. In *Proceedings of the 40th International Conference on Machine Learning*, pages 3676–3713. PMLR.

Joyce Y. Chai, Maya Cakmak, and Candace L. Sidner. 2019. Teaching Robots New Tasks through Natural Interaction. In Kevin A. Gluck and John E. Laird, editors, *Interactive Task Learning*, pages 127–146. The MIT Press.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Roziere, David Lopez-Paz, and Gabriel Synnaeve. 2024. Better & Faster Large Language Models via Multi-token Prediction. In *Forty-First International Conference on Machine Learning*.

Stevan Harnad. 2024. Language Writ Large: LLMs, ChatGPT, Grounding, Meaning and Understanding. *Preprint*, arXiv:2402.02243.

Sihao Hu, Tiansheng Huang, Fatih Ilhan, Selim Tekin, Gaowen Liu, Ramana Kompella, and Ling Liu. 2024. A Survey on Large Language Model-Based Game Agents. *Preprint*, arXiv:2404.02039.

Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander T. Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Michael Ahn, Omar Cortes, et al. 2023. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. In *Proceedings of The 6th Conference on Robot Learning*, pages 287–318. PMLR.

Taehyeon Kim, A. Suresh, K. Papineni, Michael Riley, Sanjiv Kumar, and Adrian Benton. 2024. Exploring and Improving Drafts in Blockwise Parallel Decoding.

Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Li Erran Li, Ruohan Zhang, Weiyu Liu, Percy Liang, Li Fei-Fei, Jiayuan Mao, and Jiajun Wu. 2024a. Embodied Agent Interface: Benchmarking LLMs for Embodied Decision Making. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Zhaowei Li, Qi Xu, Dong Zhang, Hang Song, YiQing Cai, Qi Qi, Ran Zhou, Junting Pan, Zefeng Li, Vu Tu, Zhida Huang, and Tao Wang. 2024b. GroundingGPT: Language Enhanced Multi-modal Grounding Model. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6657–6678, Bangkok, Thailand. Association for Computational Linguistics.

Jessy Lin, Yuqing Du, Olivia Watkins, Danijar Hafner, Pieter Abbeel, Dan Klein, and Anca Dragan. 2024. Learning to Model the World With Language. In *Forty-First International Conference on Machine Learning*.

Shuyuan Liu, Jiawei Chen, Shouwei Ruan, Hang Su, and Zhaoxia Yin. 2024. Exploring the Robustness of Decision-Level Through Adversarial Attacks on LLM-Based Embodied Models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, MM '24, pages 8120–8128, New York, NY, USA. Association for Computing Machinery.

Holger Lyre. 2024. "Understanding AI": Semantic Grounding in Large Language Models. *Preprint*, arXiv:2402.10992.

Ellie Pavlick. 2023. Symbols and grounding in large language models. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 381(2251):20220041.

Joshua Peterson, Stephan Meylan, and David Bourgin. 2019. Open clone of openai's unreleased webtext dataset scraper.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. ProphetNet: Predicting Future N-gram for Sequence-to-SequencePre-training. *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2401–2410.

Daniel Reich and Tanja Schultz. 2024. Uncovering the Full Potential of Visual Grounding Methods in VQA. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4406–4419, Bangkok, Thailand. Association for Computational Linguistics.

Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodola. 2023. Accelerating Transformer Inference for Translation via Parallel Decoding. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12336–12355, Toronto, Canada. Association for Computational Linguistics.

Omar Shaikh, Kristina Gligoric, Ashna Khetan, Matthias Gerstgrasser, Diyi Yang, and Dan Jurafsky. 2024. Grounding Gaps in Language Model Generations. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6279–6296, Mexico City, Mexico. Association for Computational Linguistics.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise Parallel Decoding for Deep Autoregressive Models. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. ScienceWorld: Is your Agent Smarter than a 5th Grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11279–11298, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. 2024. Unlocking Efficiency in Large Language Model Inference: A Comprehensive Survey of Speculative Decoding. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 7655–7671, Bangkok, Thailand. Association for Computational Linguistics.

Zilin Xiao, Hongming Zhang, Tao Ge, Siru Ouyang, Vicente Ordonez, and Dong Yu. 2024. ParallelSpec: Parallel Drafter for Efficient Speculative Decoding.

Shunyu Yao, Rohan Rao, Matthew Hausknecht, and Karthik Narasimhan. 2020. Keep CALM and Explore: Language Models for Action Generation in Text-based Games. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8736–8754, Online. Association for Computational Linguistics.

Wei Zhong and Manasa Bharadwaj. 2024. S3D: A Simple and Cost-Effective Self-Speculative Decoding Scheme for Low-Memory GPUs.

Appendix A Pretraining Models

To keep the comparison as fair as possible, the model configurations are kept the same, as listed in Table 1. Therefore, the discrepancy in the number of parameters, shown in Table 2, is mainly the result of different architectural designs. Regarding the training iterations, although nanoGPT's best results are reported after 600K iterations, taking nearly 4 days on a single 8xA100 40GB node [3], here the models are trained only for nearly 60K iterations during 2 days on a single 3xA40 40GB available node. This early stopping in pretraiing is decided to be performed, because the loss of all models remained almost constant after a while, indicating no further improvement, as reported in Figure 5.

[3] https://github.com/karpathy/nanoGPT

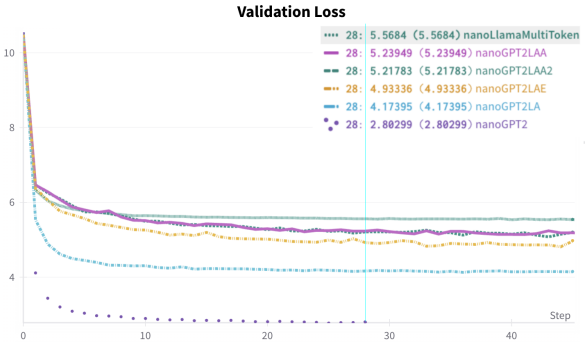

Figure 5: The loss of pertaining models on validation set has not changed significantly after 60K iterations.

Table 1: Configuration of all models.

Name	Value
Embedding size	768
# Heads	12
# Layers	12
Block size	1024
Batch size	12
Lookahead size	2
Data Type	bfloat16

Appendix B Action Selection Mechanism

As shown in Equation 1, GLAM used log probabilities to compute probability of actions and justified it in Section 3.2 of their paper with the intention *"to avoid multiple normalization operations"*. However, the multiple normalizations they were concerned about occur across different dimensions, and both are necessary. The first one (skipped by GLAM) is across tokens in the vocabulary. In more

Table 2: Size of models (number of parameters).

Name	Parameters (M)
nanoGPT2	110
nanoLlamaMultiToken	136
nanoGPT2LA	160
nanoGPT2LAA	135
nanoGPT2LAA2	135
nanoGPT2LAE	110

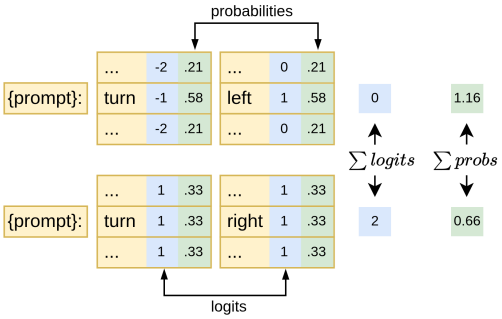

Figure 6: LLM prediction example (vocab size=3): "turn right" has lower probability, but GLAM wrongly selects it by comparing sum of log-likelihoods (2 > 0) instead of normalized logits (1.16 > 0.66).

details, for an action a_i the probability of its j-th token w_j after $p, w_0, w_1, ..., w_{j-1}$ is computed by:

$$\mathbb{P}_{LLM}(w_j|p, w_{<j}) = \frac{e^{\mathbb{LP}_{LLM}(w_j|p,w_{<j})}}{\sum_{k=0}^{|V|} e^{\mathbb{LP}_{LLM}(w_k|p,w_{<j})}} \tag{3}$$

where p is prompt, and $|V|$ stands for vocabulary size. For simplicity denote $\mathbb{P}_{LLM}(w_j|p, w_{<j})$ as $\mathbb{P}(j)$ then Equation 3 can be rewritten as:

$$\mathbb{P}(j) = \frac{e^{\mathbb{LP}(j)}}{\sum_{k=0}^{|V|} e^{\mathbb{LP}(k)}} \tag{4}$$

which means the probability of j-th token is equal to its log-likelihood normalized by sum of log-likelihood of all vocabulary tokens.

The second normalization however, is across actions in the game as formulated in the Equation 2 of the GLAM paper. The first one is needed, because without it, an action which is less likely to appear after prompt, might wrongly be selected, just because logits for the other actions neutralize each other, as shown in Figure 6.