Speculative Decoding with Quantized Drafters

Chenhui Zhang
Institute for Data, Systems, and Society
Massachusetts Institute of Technology
chenhui5@mit.edu

Haoxuan Li
EECS
Massachusetts Institute of Technology
haoxuanl@mit.edu

Abstract

The tremendous scaling effect of modern large language models (LLMs) has unlocked a variety of capabilities ranging from following complex human instructions to performing complicated reasoning. Despite the success in terms of *capabilities*, the ever-increasing model sizes also pose significant challenges in both training and inference time, leading to decreased *accessibity* and *affordability*. In particular, the memory-bound nature of the sequential decoding process of autoregressive language models poses a significant challenge to the efficient scaling of their inference stacks. To address this problem, Speculative Decoding, a block-wise decoding strategy, utilizes another small draft model to generate a small batch of tokens for the large model to process in batch through rejection sampling. However, finding and serving a separate small draft model remains challenging as a draft model that is less capable can lead to a high rejection rate of the draft on challenging texts, diminishing the benefits of Speculative Decoding. On the other hand, model compression methods, especially low-bit quantization, can vastly reduce the storage and inference overhead of large models with minimal loss of knowledge and language modeling abilities. In this project, we aim to explore the possibility of using a quantized LLM as a draft model for Speculative Decoding. This setting is especially appealing because a more capable draft can produce better draft tokens with a higher acceptance rate. However, our experiments show that although a large quantized draft model can have a higher token acceptance rate, the advantage of latency gained from better draft tokens can be outweighed by the increased latency from the larger model, despite the quantization.

1 Introduction

Recent breakthroughs in large language models (LLMs) have enabled a wide range of applications, exemplified by the tremendous success of instruction-following chatbots such as GPT-4 and Llama-2-Chat [3, 23, 28]. As researchers and practitioners started to discover the trend of scaling up LLM pretraining [14, 16, 15, 2, 6], in terms of the number of parameters and the number of tokens in the corpus, we have seen a myriad of models with ever-growing sizes being released, either open or closed. The rapid development of scaling LLMs clearly brought about a variety of capabilities, as reflected in the rapid progress in a variety of benchmarks such as HELM, BigBench, and Huggingface OpenLLM leaderboard [20, 26, 22].

However, such incredible scaling also leads to unique challenges from a system point of view. How to efficiently train, fine-tune, and serve LLMs poses a significant entry barrier to capable LLMs due to the training and inference efficiency that arises from scaling. In order to tackle the efficiency challenges to achieve better training and inference performance, researchers and practitioners have tried to tackle this problem from multiple perspectives, including model compression [21, 10], approximate attention [27], memory efficient exact attention [8, 7, 25], architectural changes [12], and decoding optimization [19, 9, 17, 4, 13, 11].

Preprint. Work In Progress.

More recently, transformer inference optimization, especially block-wise decoding algorithms [19, 4, 13], have gained traction in the community due to their abilities to further increase hardware utilization during the decoding process by breaking up the sequential nature of autoregressive decoding. On high-level, block-wise decoding, pioneered by Speculative Decoding [19], tries to decode multiple tokens at a time to increase the overall hardware utilization. Intuitively, Speculative Decoding employs a smaller scratch model to propose several draft tokens and verify them together with the large model by using a procedure similar to rejection sampling. Although this approach sounds appealing at first glance, using a small draft model still poses several challenges: small draft models are much weaker than the large model, so the rejection rate may be high in certain domains challenging to the small model; the involvement of the small draft model brings about additional considerations on the trade-off between decoding accuracy and the extra overhead; it involves in heterogeneous model access by introducing another model.

In this project, we aim to examine the first two challenges of speculative decoding discussed above from the angle of model compression. We aim to ask: *could a quantized large model serve as a capable yet efficient drafter? Could the better draft lead to more acceptance so the number of queries to the large model is reduced?* In order to answer this question, we conduct a series of benchmarks of the Speculative Decoding performance on a variety of base models and quantized draft models under both HPC environments with tensor parallel and laptops with M3 chips. However, despite the initial hypothesis, we discover that the overhead of using a more capable large draft model, despite having a variety of quantization techniques, is no better than a small yet much less capable draft model, suggesting that the increased token acceptance is outweighed by the overhead of decoding a large draft model with quantization.

We summarize our contributions as the following:

- In this project, we probe the possibility of increasing the inference efficiency by using a capable (quantized) draft model to increase the acceptance rate.
- We test our hypothesis under a variety of use cases, including HPC environments and M3 laptops.

2 Transformer Inference Optimization

Auto-regressive LLMs pose challenges in inference due to the disjoint nature of training and inference, resulting in different optimization recipes. During auto-regressive decoding, a prefilling stage will first fill the KV Cache by ingesting the user prompt and decoding the first token. This stage is largely compute-bound since we can compute the K and V matrices for the prompt tokens in parallel. The decoding stage, however, is memory-bound since we can only decode one token at a time when using the vanilla auto-regressive decoding. Based on these two challenges, researchers and practitioners have tried different techniques to increase the parallelism of the second decoding stage, increase the memory efficiency of attention operations, and optimize the KV cache mechanism under limited GPU memory and requirements for latency and throughput. Overall, we can classify the inference optimization techniques into the following categories.

2.1 Architecture & Modeling Optimization

Widely adopted by open-source LLMs like Llama-2-70b, Group-Query Attention (GQA) [1], proposes a generalization of multi-query attention which partitions query heads into G groups, with each group sharing a single key head and value head. This adaption reduces the memory bandwidth requirements of transferring Ks and Vs from HBM to SRAM. In addition, more importantly, FlashAttention and its variants [8, 7] utilize tiling and kernel fusion to reduce the kernel calls and thus the need for slow memory transfers between HBM and SRAM.

In addition to the change to optimizing the attention operations, traditional approaches to model compression, especially quantization, also apply to optimizing LLM inference, especially on edge devices. Most importantly, quantization methods like AWQ [21], GPTQ [10], and QuIP [5] push the limits of quantizing LLM weights and optimization to a new level.

2.2 Decoding Optimization

In order to increase the parallelism during the auto-regressive decoding process, a variety of block-wise decoding strategies have been proposed and are constantly evolving even after the proposal of this project. Most prominently, Speculative Decoding [19] utilizes a small draft model to decode for a couple of steps further and have the large model verify the guess. Taking a step further, Medusa [4] tries to remove the need for a separate draft model by training separate lookahead decoding heads on top of the original LLM. Meanwhile, [13] utilizes a retrieval database to generate draft tokens instead of having a smaller LLM to generate the draft tokens. Finally, LookAhead Decoding [11] performs parallel decoding using Jacobi iterations for future tokens while verifying promising n-grams.

In addition to block-wise decoding, Flash Decoding [17] adds the keys/values sequence length as a new parallelization dimension on top of the original Flash Attention algorithm to better suit LLM inference.

2.3 System-Level Optimization

A variety of other methods also try to tackle transformer inference from a system perspective. For example, Pope et al. proposes a multi-dimensional sharding technique optimized for TPU platforms to achieve a new Pareto frontier on the latency and model FLOPS utilization (MFU) in distributed inference. In addition, vLLM [18] proposed PagedAttention to reduce the waste of the KV cache memory and enable cross-query cache sharing during LLM serving.

3 Method

In order to explore the influence of quantized draft models on the effectiveness of Speculative Decoding, we mainly consider GPTQ quantization and the original formulation of Speculative Decoding.

In order to use Speculative Decoding to better parallelize the decoding of a larger model, we select a draft model with much less overhead than the large base model. The only hard requirement is to make sure that they share the same tokenizer so that we can meaningfully compare the logits of the two models.

During the speculative decoding process, we generate a specific number of candidate new tokens, usually three, with the small model by running three forward passes of the small model. Then, we combine the proposed draft tokens with the older tokens to feed into the larger model for a single forward pass, which returns the corresponding probabilities for all input tokens. Then, we decode the last number of tokens from the draft step using greedy decoding. Finally, we compare the decoded tokens from the large model with the candidate new tokens. If there is a match, we accept the proposed tokens until the first mismatch occurs, at which point we add the token from the large base model to the decoding result.

4 Experiments

In order to answer our research questions about the effectiveness of using a large quantized model of the draft model and the related influence of decoding latency, we design a series of experiments to benchmark the performance of Speculative Decoding under different quantization methods. We mainly contrast quantized large Llama models such as Llama-2-13b and Llama-2-7b with small models like Tiny Llama-1.1b.

In Table 1, we compare the decoding latency of Llama-2-13b with a Llama-2-13b quantized draft model and a TinyLlama 1.1b model. We compare their relative latency increase over baseline without any speculative decoding. Overall, we find that contrary to our hypothesis, the TinyLlama model draft model, despite being less capable, achieves better metrics in terms of tokens/second.

In Table 2, we also test Llama-2-70b models with two A100 GPUs and tensor parallelism. We consider the quantized Llama-2-7b model and the unquantized TinyLlama model as the draft models. Overall, similar to the prior finding, a quantized large model did not improve the performance despite being more capable. This indicates that the extra overhead from running a large model, despite being quantized, outweighs the performance gain from proposing better draft tokens.

In and , we evaluate speculative decoding on the Apple M3 device, with the target model also being quantized. We find that running another large draft model significantly degrades the inference performance, indicating the need for smaller draft models.

Table 1: Speculative Decoding with 1xA100 80G

Target Model	Draft Model	Quantization	Token / s	Relative Improvement
Llama 2 13B	N/A	N/A	59.02	N/A
	TinyLlama 1.1B	N/A	105.42	78.61%
	TinyLlama 1.1B	int8	109.22	85.05%
	TinyLlama 1.1B	int4	111.09	**88.24%**
Llama 2 13B	Llama 2 13B	int8	63.80	8.09%
	Llama 2 13B	int4	76.73	30.06%

Table 2: Speculative Decoding with 2xA100 80G and Tensor Parallelism

Target Model	Draft Model	Token/s	Improvement
Llama 2 70B	N/A	21.74	0%
	TinyLlama 1.1B	43.49	100.04%
	TinyLlama 1.1B Int8 GPTQ	39.01	79.43%
	Llama 2 7B Int8 GPTQ	38.56	77.36%
	Llama 2 7B Int4 GPTQ	37.55	72.72%

Table 3: Speculative Decoding (7b) with M3Max

Target Model	Draft Model	Token/s	Improvement
Llama 2 7B Int8	N/A	3.26	0%
	TinyLlama 1.1B	4.84	**48.47%**
	TinyLlama 1.1B Int8 GPTQ	4.23	29.75%
	Llama 2 7B Int8 GPTQ	N/A	N/A
	Llama 2 7B Int4 GPTQ	2.69	-17.49%

Table 4: Speculative Decoding (13b) with M3Max

Target Model	Draft Model	Token/s	Improvement
Llama 2 13B Int8	N/A	1.20	0%
	TinyLlama 1.1B	2.90	**141.66%**
	TinyLlama 1.1B Int8 GPTQ	2.54	111.66%
	Llama 2 7B Int8 GPTQ	1.32	10%
	Llama 2 7B Int4 GPTQ	1.10	-8%

5 Conclusion & Future Work

In conclusion, by probing the latency behavior of Speculative Decoding, we find that in both HPC and laptop environments, smaller, unquantized draft models work best. In HPC environments With the same target model, the latency of speculative decoding with a large quantized draft model is no better than the 1-B draft model. In the desktop environment, the effectiveness of SD is multidimensional,

especially on edge hardware. We attribute this to the fact that laptops and edge hardware are often both memory-and-compute-bound. In addition, quantization disproportionately reduce memory size, memory bandwidth, and computation, making it difficult to detect the bottleneck.

References

[1] J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai. GQA: training generalized multi-query transformer models from multi-head checkpoints. In H. Bouamor, J. Pino, and K. Bali, editors, Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023, pages 4895–4901. Association for Computational Linguistics, 2023. URL https://aclanthology.org/2023.emnlp-main.298.

[2] Y. Bahri, E. Dyer, J. Kaplan, J. Lee, and U. Sharma. Explaining neural scaling laws. arXiv preprint arXiv: 2102.06701, 2021.

[3] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. arXiv preprint arXiv:2303.12712, 2023.

[4] T. Cai, Y. Li, Z. Geng, H. Peng, and T. Dao. Medusa: Simple framework for accelerating llm generation with multiple decoding heads, 2023.

[5] J. Chee, Y. Cai, V. Kuleshov, and C. D. Sa. Quip: 2-bit quantization of large language models with guarantees. arXiv preprint arXiv: 2307.13304, 2023.

[6] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, et al. Scaling instruction-finetuned language models. arXiv preprint arXiv:2210.11416, 2022.

[7] T. Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. arXiv preprint arXiv: 2307.08691, 2023.

[8] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. Advances in Neural Information Processing Systems, 35: 16344–16359, 2022.

[9] T. Dao, D. Haziza, F. Massa, and G. Sizov. Flash-decoding for long-context inference, 2023.

[10] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. arXiv preprint arXiv:2210.17323, 2022.

[11] Y. Fu, P. Bailis, I. Stoica, and H. Zhang. Breaking the sequential dependency of llm inference using lookahead decoding, November 2023. URL https://lmsys.org/blog/2023-11-21-lookahead-decoding/.

[12] A. Gu and T. Dao. Mamba: Linear-time sequence modeling with selective state spaces. arXiv preprint arXiv: 2312.00752, 2023.

[13] Z. He, Z. Zhong, T. Cai, J. D. Lee, and D. He. Rest: Retrieval-based speculative decoding. arXiv preprint arXiv:2311.08252, 2023.

[14] T. Henighan, J. Kaplan, M. Katz, M. Chen, C. Hesse, J. Jackson, H. Jun, T. B. Brown, P. Dhariwal, S. Gray, et al. Scaling laws for autoregressive generative modeling. arXiv preprint arXiv:2010.14701, 2020.

[15] D. Hernandez, J. Kaplan, T. Henighan, and S. McCandlish. Scaling laws for transfer. arXiv preprint arXiv: 2102.01293, 2021.

[16] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, et al. Training compute-optimal large language models. arXiv preprint arXiv:2203.15556, 2022.

[17] K. Hong, G. Dai, J. Xu, Q. Mao, X. Li, J. Liu, K. Chen, H. Dong, and Y. Wang. Flashdecoding++: Faster large language model inference on gpus. arXiv preprint arXiv:2311.01282, 2023.

[18] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica. Efficient memory management for large language model serving with pagedattention. Symposium on Operating Systems Principles, 2023. doi: 10.48550/arXiv.2309.06180.

[19] Y. Leviathan, M. Kalman, and Y. Matias. Fast inference from transformers via speculative decoding. In International Conference on Machine Learning, pages 19274–19286. PMLR, 2023.

[20] P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar, et al. Holistic evaluation of language models. arXiv preprint arXiv:2211.09110, 2022.

[21] J. Lin, J. Tang, H. Tang, S. Yang, X. Dang, and S. Han. Awq: Activation-aware weight quantization for llm compression and acceleration. arXiv preprint arXiv:2306.00978, 2023.

[22] Open LLM Leaderboard. Huggingface.

[23] OpenAI. Gpt-4 technical report. ArXiv, abs/2303.08774, 2023. URL `https://api.semanticscholar.org/CorpusID:257532815`.

[24] R. Pope, S. Douglas, A. Chowdhery, J. Devlin, J. Bradbury, J. Heek, K. Xiao, S. Agrawal, and J. Dean. Efficiently scaling transformer inference. Proceedings of Machine Learning and Systems, 5, 2023.

[25] L. Qian, X. Han, F. Wang, H. Liu, H. Dong, Z. Li, H. Wei, Z. Lin, and C.-B. Jin. Xformer: Fast and accurate monocular 3d body capture. International Joint Conference on Artificial Intelligence, 2023. doi: 10.48550/arXiv.2305.11101.

[26] A. Srivastava, A. Rastogi, A. Rao, A. A. M. Shoeb, A. Abid, A. Fisch, A. R. Brown, A. Santoro, A. Gupta, A. Garriga-Alonso, A. Kluska, A. Lewkowycz, A. Agarwal, A. Power, A. Ray, A. Warstadt, A. W. Kocurek, A. Safaya, A. Tazarv, A. Xiang, A. Parrish, A. Nie, A. Hussain, A. Askell, A. Dsouza, A. Slone, A. Rahane, A. S. Iyer, A. Andreassen, A. Madotto, A. Santilli, A. Stuhlmüller, A. Dai, A. La, A. Lampinen, A. Zou, A. Jiang, A. Chen, A. Vuong, A. Gupta, A. Gottardi, A. Norelli, A. Venkatesh, A. Gholamidavoodi, A. Tabassum, A. Menezes, A. Kirubarajan, A. Mullokandov, A. Sabharwal, A. Herrick, A. Efrat, A. Erdem, A. Karakaş, B. R. Roberts, B. S. Loe, B. Zoph, B. Bojanowski, B. Özyurt, B. Hedayatnia, B. Neyshabur, B. Inden, B. Stein, B. Ekmekci, B. Y. Lin, B. Howald, B. Orinion, C. Diao, C. Dour, C. Stinson, C. Argueta, C. F. Ramírez, C. Singh, C. Rathkopf, C. Meng, C. Baral, C. Wu, C. Callison-Burch, C. Waites, C. Voigt, C. D. Manning, C. Potts, C. Ramirez, C. E. Rivera, C. Siro, C. Raffel, C. Ashcraft, C. Garbacea, D. Sileo, D. Garrette, D. Hendrycks, D. Kilman, D. Roth, D. Freeman, D. Khashabi, D. Levy, D. M. González, D. Perszyk, D. Hernandez, D. Chen, D. Ippolito, D. Gilboa, D. Dohan, D. Drakard, D. Jurgens, D. Datta, D. Ganguli, D. Emelin, D. Kleyko, D. Yuret, D. Chen, D. Tam, D. Hupkes, D. Misra, D. Buzan, D. C. Mollo, D. Yang, D.-H. Lee, D. Schrader, E. Shutova, E. D. Cubuk, E. Segal, E. Hagerman, E. Barnes, E. Donoway, E. Pavlick, E. Rodola, E. Lam, E. Chu, E. Tang, E. Erdem, E. Chang, E. A. Chi, E. Dyer, E. Jerzak, E. Kim, E. E. Manyasi, E. Zheltonozhskii, F. Xia, F. Siar, F. Martínez-Plumed, F. Happé, F. Chollet, F. Rong, G. Mishra, G. I. Winata, G. de Melo, G. Kruszewski, G. Parascandolo, G. Mariani, G. Wang, G. Jaimovitch-López, G. Betz, G. Gur-Ari, H. Galijasevic, H. Kim, H. Rashkin, H. Hajishirzi, H. Mehta, H. Bogar, H. Shevlin, H. Schütze, H. Yakura, H. Zhang, H. M. Wong, I. Ng, I. Noble, J. Jumelet, J. Geissinger, J. Kernion, J. Hilton, J. Lee, J. F. Fisac, J. B. Simon, J. Koppel, J. Zheng, J. Zou, J. Kocoń, J. Thompson, J. Wingfield, J. Kaplan, J. Radom, J. Sohl-Dickstein, J. Phang, J. Wei, J. Yosinski, J. Novikova, J. Bosscher, J. Marsh, J. Kim, J. Taal, J. Engel, J. Alabi, J. Xu, J. Song, J. Tang, J. Waweru, J. Burden, J. Miller, J. U. Balis, J. Batchelder, J. Berant, J. Frohberg, J. Rozen, J. Hernandez-Orallo, J. Boudeman, J. Guerr, J. Jones, J. B. Tenenbaum, J. S. Rule, J. Chua, K. Kanclerz, K. Livescu, K. Krauth, K. Gopalakrishnan, K. Ignatyeva, K. Markert, K. D. Dhole, K. Gimpel, K. Omondi, K. Mathewson, K. Chiafullo, K. Shkaruta, K. Shridhar, K. McDonell, K. Richardson, L. Reynolds, L. Gao, L. Zhang, L. Dugan, L. Qin, L. Contreras-Ochando, L.-P. Morency, L. Moschella, L. Lam, L. Noble, L. Schmidt, L. He, L. O. Colón, L. Metz, L. K. Şenel, M. Bosma, M. Sap, M. ter Hoeve, M. Farooqi, M. Faruqui, M. Mazeika, M. Baturan, M. Marelli, M. Maru, M. J. R. Quintana, M. Tolkiehn, M. Giulianelli, M. Lewis, M. Potthast, M. L. Leavitt, M. Hagen, M. Schubert, M. O. Baitemirova, M. Arnaud, M. McElrath, M. A. Yee, M. Cohen, M. Gu, M. Ivanitskiy, M. Starritt, M. Strube, M. Swędrowski, M. Bevilacqua, M. Yasunaga, M. Kale, M. Cain, M. Xu, M. Suzgun, M. Walker, M. Tiwari, M. Bansal, M. Aminnaseri, M. Geva, M. Gheini, M. V. T, N. Peng, N. A. Chi, N. Lee, N. G.-A. Krakover, N. Cameron, N. Roberts, N. Doiron, N. Martinez, N. Nangia, N. Deckers, N. Muennighoff, N. S. Keskar, N. S. Iyer, N. Constant, N. Fiedel, N. Wen, O. Zhang, O. Agha, O. Elbaghdadi, O. Levy, O. Evans, P. A. M. Casares, P. Doshi, P. Fung, P. P. Liang, P. Vicol, P. Alipoormolabashi, P. Liao, P. Liang, P. Chang,

P. Eckersley, P. M. Htut, P. Hwang, P. Miłkowski, P. Patil, P. Pezeshkpour, P. Oli, Q. Mei, Q. Lyu, Q. Chen, R. Banjade, R. E. Rudolph, R. Gabriel, R. Habacker, R. Risco, R. Millière, R. Garg, R. Barnes, R. A. Saurous, R. Arakawa, R. Raymaekers, R. Frank, R. Sikand, R. Novak, R. Sitelew, R. LeBras, R. Liu, R. Jacobs, R. Zhang, R. Salakhutdinov, R. Chi, R. Lee, R. Stovall, R. Teehan, R. Yang, S. Singh, S. M. Mohammad, S. Anand, S. Dillavou, S. Shleifer, S. Wiseman, S. Gruetter, S. R. Bowman, S. S. Schoenholz, S. Han, S. Kwatra, S. A. Rous, S. Ghazarian, S. Ghosh, S. Casey, S. Bischoff, S. Gehrmann, S. Schuster, S. Sadeghi, S. Hamdan, S. Zhou, S. Srivastava, S. Shi, S. Singh, S. Asaadi, S. S. Gu, S. Pachchigar, S. Toshniwal, S. Upadhyay, Shyamolima, Debnath, S. Shakeri, S. Thormeyer, S. Melzi, S. Reddy, S. P. Makini, S.-H. Lee, S. Torene, S. Hatwar, S. Dehaene, S. Divic, S. Ermon, S. Biderman, S. Lin, S. Prasad, S. T. Piantadosi, S. M. Shieber, S. Misherghi, S. Kiritchenko, S. Mishra, T. Linzen, T. Schuster, T. Li, T. Yu, T. Ali, T. Hashimoto, T.-L. Wu, T. Desbordes, T. Rothschild, T. Phan, T. Wang, T. Nkinyili, T. Schick, T. Kornev, T. Tunduny, T. Gerstenberg, T. Chang, T. Neeraj, T. Khot, T. Shultz, U. Shaham, V. Misra, V. Demberg, V. Nyamai, V. Raunak, V. Ramasesh, V. U. Prabhu, V. Padmakumar, V. Srikumar, W. Fedus, W. Saunders, W. Zhang, W. Vossen, X. Ren, X. Tong, X. Zhao, X. Wu, X. Shen, Y. Yaghoobzadeh, Y. Lakretz, Y. Song, Y. Bahri, Y. Choi, Y. Yang, Y. Hao, Y. Chen, Y. Belinkov, Y. Hou, Y. Hou, Y. Bai, Z. Seid, Z. Zhao, Z. Wang, Z. J. Wang, Z. Wang, and Z. Wu. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. arXiv preprint arXiv: 2206.04615, 2022.

[27] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler. Efficient transformers: A survey. ACM Computing Surveys, 2020. doi: 10.1145/3530811.

[28] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288, 2023.