The Costly Dilemma: Are Large Language Models the Pay-Day Loans of Machine Learning?

Abi Aryan, Aakash Kumar Nain, Andy McMahon, Lucas Augusto Meyer, Harpreet Singh Sahota

(In Review at EMNLP 2023)

Abstract

When deploying machine learning models in production for any product/application, there are three properties commonly that are commonly desired. First, the models to be generalizable, second they should be evaluable and finally the deployment to be cost-optimal. In this paper we propose that these three objectives (i.e. generalization, evaluation and cost-optimality) are relatively orthogonal and that for large language models, despite their performance over conventional NLP models, enterprises need to carefully assess all the three factors mentioned before making substantial investments in this technology. In this paper, we propose a framework for generalization, evaluation and cost-modeling specifically tailored to large language models, offering insights into the intricacies of development, deployment and management for these large language models.

---

1 Introduction

According to two separate Gartner reports [1][2], 85% of AI and machine learning projects fail to deliver, with only 53% of projects finally making it from prototypes to production. The four key reasons for this mentioned in a subsequent study by Gartner [3] were - 1) a lack of business-use case clarity, 2) inadequate skills within the team for end-to-end deployments, 3) neglecting organizational change, and 4) failure to experiment. While proprietary LLMs like Chat-GPT have made it incredibly easy to quickly deploy these models in production via just an API call. However, there are still three core challenges that require careful considerate analysis before making any strategic business changes. Firstly, the build vs buy hypothesis and which model we should use for our particular use-case. Secondly, how do the costs for LLM development, deployment and management scale once the business use-case has been established. And thirdly, given all the evaluation frameworks and leaderboards out there, how should the engineering teams evaluate these models?

We assert that these challenges will only be exacerbated as organizations adopt and adapt LLMs for two reasons - compliance risk [4] due to lack of clear evaluation metrics, and hidden costs associated with the model deployments. In this paper, we will explore these issues and posit some of the known and some not-so-known challenges.

We posit that given these challenges can quickly escalate, leading to significant financial burdens and potential operational risks. Thus, making it more important than ever to set the expectations right if the projects are going to be considered “successful”.

The first part of the paper introduces the question of generalized v/s domain-specific large language models, the second part talks about the goes into the visible as well as hidden short and long-term costs of deploying

these models in production and the third part about the LLMOps production pipeline, pitfalls and model evaluation.

Some of the related research that explores this question through the lens of making extensive investments generating intangible results for a considerable time for substantial returns down the line has been addressed in a paper by E. Brynjolfsson et al. [5] Another related research conducted by Rosa et. al [6] analyzes one-time vs recurrent costs in favor of cost-benefit analysis for multi-lingual methods.

2 The GCE Trifecta

In the realm of project management, the GCE trifecta, consisting of generalization, cost optimization, and evaluation, plays a pivotal role in determining project success. Each component - generalization, cost, and evaluation - carries its own significance, yet they collectively contribute to achieving project objectives.

Firstly, generalization stands as a fundamental pillar of LLM project success. It encompasses the ability of a large language model project to deliver its intended outcomes across a broad range of contexts and situations. Generalization enables scalability, adaptability, and the potential for replication, allowing projects to tackle complex challenges while remaining applicable to diverse scenarios.

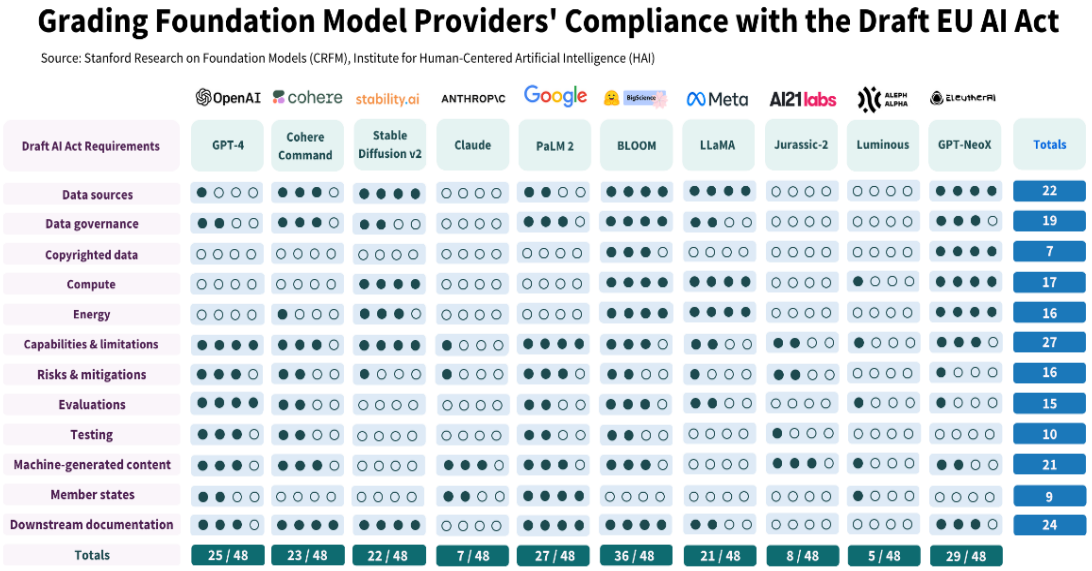
Secondly, cost optimization serves as a critical factor in the ability of an organization to harness the potential of large language models. Cost-benefit analysis enables organizations to achieve their goals within budgetary constraints, maximizing value and gaining a return on investment.

Lastly, evaluation acts as the cornerstone of machine learning project success providing a systematic and objective assessment of the model performance. By employing rigorous evaluation methodologies, product teams can ensure accountability, transparency, and the ability to do continuous experimentation and improvement. Effective evaluation techniques enable stakeholders to gauge the impact of the LLM and make data-driven decisions for future endeavors.

Despite their inherent interdependencies, we propose that generalization, cost optimization, and evaluation are relatively orthogonal in the context of project-success. By recognizing the challenges for each of these objectives and employing tailored strategies for your specific use-case, organizations can strike a harmonious balance, leading to holistic project success. Through this research, we aim to shed light on the unique dynamics of the GCE trifecta and provide insights that would be helpful across the organization and different stakeholders within the team.

2.1 Generalization

Broadly speaking, there are two different interest-groups amongst enterprises working on large language models. The first are the Foundation Model (also now starting to be referred to as Base Model) providers that make it easy for anyone to choose and access any pre-trained large language model using a cloud-based or self-hosted infrastructure. Depending on the provider, these models may be open-source or proprietary, based on the release strategy [7] of the provider. Amongst this category are companies like OpenAI, Cohere, Google, Microsoft, Anthropic, Nvidia, Mosaic, Hugging Face etc. While the out-of-the-box direct use of a foundational model may be the quickest way to deploy a LLM-based product or application. However, it may not add much substantial value depending on the use-case. Most of the enterprises would instead benefit from domain-specific knowledge injection to improve task-specific performance of the models.



Stanford HAI evaluation of foundation model providers for their compliance with proposed EU law on AI.

While LLMs have seen adoption for several NLP applications across a wide-range of industries including coding assistants, copywriting, language prompted visual design, drug discovery, legal reviews etc, however an extensive review of all existing applications across industries is missing. That said, several economists have done extensive reviews on the potential impact of GPTs on the labor market [12][13]14]. LLMs are as of now more generally used for generative use-cases than discriminative use-cases. As of writing, we have seen four popular use-case specific explorations for LLM-based applications. Firstly, for knowledge retrieval [8], second for recommender systems [9], thirdly cross-lingual translation [10] and finally as autonomous agents or AI Agents [11].

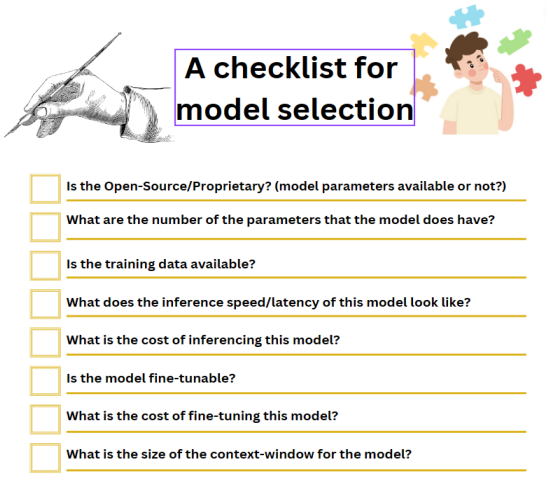
However, using these models out of the box exposes the organizations to several unexpected risks including compliance risk, prompt drifts, security risks, poor performance etc, Thus, we strongly believe that there will be more companies fine-tuning their models on their industry-specific data, if not building them in-house depending on the business use-case maturity and skill levels within the organizations for end-to-end development and deployments.

The second interest-group are the developers purely interested in integrating LLMs into their own products and services. These include Plugins, AI Agents as well as products that use LLMs for NLP applications, for tasks like podcast summarization, copywriting etc. Depending on the purpose, both groups have different challenges when it comes to which model to consume or develop. This can be challenging as at the time of writing there are no substantive studies that compare which provider would be better for which particular use-case. However, one of the strong factors that could guide the choices of different enterprises may be the regulatory authorities. (eg. EU-AI Act [15]) that can limit the availability of a certain model or provider by the region depending on the compliant-levels. (see chart above).

Deciding which provider and foundational model would be a better choice depends on several factors including number of parameters, size of context window, training

type, inference speed, cost, fine-tunability as well as data security.

While there are subjective quality-measures for all models, the extensive model quality depends on your particular domain as well as the sophistication of application using a LLM. For example, if the goal is to do knowledge retrieval on unstructured data, GPT-4 may be an excellent choice however when doing knowledge retrieval on structured data, a model with a larger context window may be the most optimal choice.



Although, one of the key limitations with building using proprietary models is there is a lack of best-practices and information on long-term-support (LTS), prompt-drift on updates etc. This lack of clarity can lead to operational risks around reliability, explainability and predictability when moving into production.

2.2 Cost Optimality

One question that concerns every organization today is the cost of deploying LLMs.

Building an in-house LLM and maintaining it in production is no easy feat. It requires a significant investment in infrastructure, data collection, and hiring skilled personnel. Most importantly, this will not be a one-time investment. As you scale LLMs, the cost to maintain deployed LLMs increases.

In contrast, vendor-based LLMs like GPT-4 seem a great choice, given they operate on a pay-as-you-go model, reducing upfront costs and allowing for better cost control, making them an attractive choice.

Table 1. Build vs Buy Hypothesis

Attribute	Build	Buy
Predictable Workload	Native LLMs require significant upfront capital investment for hardware, software, networking equipment, and facilities. Ongoing costs include maintenance, staff, and energy consumption.	Vendor-Based LLMs typically operate on a pay-as-you-go model, reducing upfront costs and allowing for better cost control. They provide almost limitless scalability, enabling organizations to easily expand or contract their resources according to demand
Hardware Costs	Upfront and maintenance	Nominal
Security	More secure	Depends on the provider's infrastructure
Compliance	High Compliance	Check Chart (on Pg.2)
Latency	Lower Latency	Higher Latency

While such quick conclusions make the build vs buy hypothesis make the vendor-based models seem like an obvious choice, however, the day-to-day experience of product development and scaling contrasts this common reasoning.

Deploying an LLM is very different from deploying any other machine learning model because the cost in the case of LLMs is

two-fold: infra-model related cost, and the hidden cost.

Although we can significantly reduce our costs with these vendor-based models, they come with their own sets of challenges. Organizations may be reticent when it comes to sharing sensitive data with any API-based LLM vendor. Organizations will also require that vendor based models are compliant with their own organizational data policies. This can be hard to ensure if data and implementation details are not shared freely.

Given this, there is the risk of lock in as switching to a new model or vendor will incur new operational costs as these due diligence and compliance exercises are completed. In short, security, compliance, and latency become major concerns when choosing a vendor-based model.

Another dimension to consider when working with LLMs is the question of prompt engineering v/s fine-tuning. This choice depends on several factors, and has some associated consequences on the overall cost. An important consideration for this question is the length of the context window of the model and understanding how the overall cost is related to it. Although transformer models show exceptional scaling capabilities, one computational bottleneck that remains an open-challenge is the processing of long sequences. The complexity of the naive attention mechanism grows quadratically in terms of both the compute and the memory [17].

With the latest Anthropic model, we have a context window of 100K tokens that translates to roughly 75,000 words. Such a lengthy context window opens up opportunities for accomplishing tasks that were almost impossible to achieve in the past. For example, you can input an entire book into the model and dynamically query the content just from that provided context. This is a qualitative step up compared to the capabilities of some earlier LLMs.

With the larger context windows, you can retain more in-context information and the model can handle more complex and longer inputs. However, one of the challenges of large context windows is that the costs increase almost quadratically as the number of tokens are increased and can also affect the inference latency due to the slow-down of model computations. For example, Anthropic latest model response time on a 100K context window is roughly 22 seconds. Also, most use cases don't require such a large context window. It is also not so easy to write and modify lengthy prompts.

Smaller context windows allow for smaller input lengths thus requiring clear, concise, and clever prompts to obtain a desirable output. One of the advantages of short prompts is that they are easy to write and modify compared to the lengthier prompts. The overall latency is low, the chain of thoughts becomes easy and they also enable faster iteration. You can also leverage parallel context windows for many use cases to achieve acceptable performance on a task [18] without fine-tuning or using an expensive model with a bigger context window.

While it seems like there is an obvious upside to using smaller context windows and investing your time in prompt engineering, the iterative costs can certainly sneak up on you. To obtain a similar result from an LLM, you may be required to write multiple prompts and make multiple calls to the model. With multiple prompts and calls, it quickly adds to your overall inference cost of the model.

Another disadvantage of shorter prompts is that it makes it hard to decide when to go with fine-tuning instead of prompting. Most of the time prompting can take you far, but it may require you to run several trials before you decide to fine-tune, either with an explicit

reward function or Reinforcement Learning from Human Feedback (RLHF).

Fine-tuning is substantially more expensive than prompting and not always the right approach depending on the complexity of the conditional. There may exist valid inferences that satisfy the conditional argument, however sampling them can be incredibly hard if we don't already know the factorization ahead of time. Generally speaking, prompt engineering works well for embeddings, fine-tuning may produce better results for categorization and filtering however there is no conclusive work that compares the generalization for both the options.

Another factor to keep in mind is the scaling laws. It has been proven that language models get predictably better as the number of parameters, amount of compute used, and dataset size increases [16][19]. Query caching can significantly reduce costs for LLM inferencing, but the exact amount of cost reduction depends on various factors, such as the frequency of repeated queries, performance of the underlying language model, cache hit ratio, and the overall architecture of the LLM inferencing system. [20].

While the above discussion focused on the infrastructure and the modeling-related costs, LLMs also have associated hidden costs.

The first key cost being the cost of prompt drift. LLMs do offer very little reproducibility even with the same prompts between different versions of the LLMs as you update to a faster, better, distilled LLM. Second, with traditional machine learning models, it is common to hire annotators to annotate datasets. Hiring annotators is cheap, and it takes a very short amount of time to train them for the defined annotation task. Validating the annotations done by the annotators is easy, and we can automate the validation process to a large extent.

However, the same process becomes very complex in the case of LLMs due to the need for domain-specific knowledge to create good prompts. Either the team writes and validates all the prompts every time, or you hire a prompt engineer. Hiring prompt engineers is expensive. On top of that, it creates an indirect dependency on the prompt engineer within the team if they choose to leave. Retraining new prompt engineers for your tasks is time-consuming, and expensive (remember the cost related to the API calls?). Even if you hire a prompt engineer, there is no way to automate the validation of prompts.

Depending on whether you choose to call the model from the front-end or the back-end, and fine-tune vs prompt engineer, it would result in costs that can vary across a wide scale. LLMs are still relatively new in the machine learning world, which means that there are unknowns associated with using them in production. Some typical risks associated with machine learning models when used in production are:

- Compliance and regulatory risk: This refers to the risk of breaking rules set by governing bodies of various flavors, be they government themselves, regulators or other institutions with powers to enforce compliance with set rules. In this scenario organizations can face potential large fines or other punitive measures. For example the upcoming EU AI Act, which is undergoing final review by European lawmakers at the time of writing, could mean fines of 10 million euros or 2% of global profits (whichever is higher) on organizations that breach these rules [21].

- Reputational risk: A system may not necessarily breach legal or regulatory guidelines but it may still behave outside the expected norms for interaction with a variety of stakeholders. Some examples could be a banking customer being faced with derogatory remarks, a hospital patient being blamed for their illness, or a customer being given suggestions that conform to racial or gender

biases [22]. These scenarios can then lead to huge reputational damage for organizations and for the concept of AI and ML systems as a whole and lead to losses in income and future revenue.

- Operational risk: Many organizations now use data and machine learning to inform operational decisions. If an LLM generates inconsistent output or leads to an erroneous operational decision this could incur large costs as well. For example, if an LLM based chatbot was being used by a senior executive in an organization to help make an investment decision, hallucinated facts could lead to a large amount of that investment being wasted in a low growth area. Similarly incorrect information may lead to erroneous decisions around technology adoption, system design, logistical operations, administration execution that could lead to very costly outcomes.

Table 2. Costs Associated with LLM Applications

Upfront Costs	Hidden Costs
Context Window	Model Drift, Prompt Drift
Prompting/Fine-Tuning	Hardware Costs
Data	Compliance
Infrastructure	People
Scalability	Reliability

2.3 Evaluation

With the rise of Large Language Models (LLMs) the question of what best evaluation practice looks like must be revisited as some assumptions usually employed for ML evaluations no longer hold and need augmented. Large language models are often trained on massive amounts of data and require more than a few million parameters further limiting their reproducibility as well as interpretability.

This gets even more tricky as more and more companies are moving to closed-version of the models keeping the model parameters as well as information on RLHF and red-teaming the models through adversarial examples private thus making it close to impossible to fully evaluate these models.

In the past, transformer based language models were typically evaluated using perplexity, the BLEU score and Human Evaluations.[23] However, these metrics have been criticized for being too simplistic and not taking into account much of the nuance of human language. This is counteracted somewhat when using techniques based on human evaluation, however this can also be the most time-consuming and expensive approach, with particular challenges around scaling to large input and outputs, as is the case with LLMs.

There are several general benchmarks available for LLMs, namely OpenAI Evals, HELM, Evals-Harness, etc. however elo-based systems [25] are quickly gaining popularity over community-based leaderboards [24] v/s vendor-based evals (Nemo Guardrails, Aviary, etc). While the above-mentioned generalized benchmarks are helpful for some contexts, most organizations need domain-specific benchmarks that are specific to the company and their business use-case.

We break it down into five concerns that need to be addressed to develop a comprehensive evaluation framework.

2.3.1 What does “performance” mean for an LLM application?

Since LLMs do not have clear objective functions, thus it is hard to conventionally evaluate them using the conventional ML metrics. Thus, performance comes down to a combination of several factors-

1. Accuracy
2. Inference Speed
3. Latency

	HELM / lm-evaluation-harness	OpenAI/eval	Alpaca Evaluation	Vicuna Evaluation	Chatbot Arena
Question Source	Academic datasets	Mixed	Self-instruct evaluation set	GPT-4 generated	User prompts
Evaluator	Program	Program/Model	Human	GPT-4	User
Metrics	Basic metrics	Basic metrics	Win rate	Win rate	Elo ratings

LMSys.org - Comparison between different evaluation methods

2.3.2 How do we create stable experimental setups for evaluating LLM applications?
While having a test set you test/benchmark against is incredibly important. However, the setup comes down to -

1. Data Sampling in Test vs Evals
2. Logging Prompts and Inferences
3. Checkpointing the models.

2.3.3 What benchmarks do we have or do we need to create to enable consistency of approach?
The generalized benchmarks depending on the use-case do allow for grounding. However, organizations still standard proxies like accuracy and other metrics against your own test dataset and/or public benchmarks.

2.3.4 For third party hosted models, what assurances can we give ourselves as downstream consumers through validation?
Delegating the model development and maintenance to vendors like OpenAI, Anthropic etc does allow one to have significant assurances when it comes to model staling, latency, scalability and easy deployment.

2.3.5 How do we evaluate and monitor the accuracy of our LLM-based solutions during development and post-deployment?
For applications, public benchmarks are not useful because it's not measured on the data distribution you care about (data your users give). So building elo-based benchmarks for your data can be an important step in the right direction.

3. Conclusion

The integration of Language Models (LLMs) into applications brings forth numerous benefits, but it also introduces the concept of technical debt. This debt can manifest as potential risks or hidden costs that may arise in the future. However, it is important to emphasize that LLMs remain highly valuable despite these considerations, and technical debt itself is not inherently negative. Just as individuals make informed decisions regarding financial debt and actively manage it, a similar approach must be adopted when dealing with technical debt in LLM-based solutions.

Choosing an appropriate level of technical debt becomes crucial in LLM integration. This involves carefully evaluating the trade-offs between short-term gains and long-term consequences. LLMs offer immediate advantages such as enhanced natural language processing capabilities and improved user experiences. However, hasty implementation or overreliance on LLMs without addressing potential technical debt can lead to challenges down the line.

Managing technical debt in LLM-based solutions requires a proactive and strategic approach. Just as financial debt requires diligent monitoring and repayment plans, technical debt should be assessed, documented, and accounted for. Organizations must invest resources in identifying areas where technical debt may accumulate, such as code complexity, potential performance

bottlenecks, or lack of maintainability. By acknowledging these risks, teams can make informed decisions, and allocate resources accordingly.

References:

1. Ga*rtner: Fueling the future of business*. (2019, January 3). Gartner. https://blogs.gartner.com/andrew_white/2019/01/03/our-top-data-and-analytics-predicts-for-2019/
2. Gartner identifies the top strategic technology trends for 2021. (n.d.). Gartner. https://www.gartner.com/en/newsroom/press-releases/2020-10-19-gartner-identifies-the-top-strategic-technology-trends-for-2021
3. Gartner says nearly half of CIOs are planning to deploy artificial intelligence. (n.d.). Gartner. https://www.gartner.com/en/newsroom/press-releases/2018-02-13-gartner-says-nearly-half-of-cios-are-planning-to-deploy-artificial-intelligence
4. *Managing the risk of large language models like ChatGPT*. (2023, May 10). Governance, risk, and compliance advisory in financial services. https://www.acaglobal.com/insights/managing-risk-large-language-models-chatgpt
5. Brynjolfsson, E., Rock, D., & Syverson, C. (2021). The productivity J-curve: How intangibles complement general purpose technologies. *American Economic Journal: Macroeconomics*, *13*(1), 333-72.
6. Rosa, G. M., Bonifacio, L. H., de Souza, L. R., Lotufo, R., & Nogueira, R. (2021). A cost-benefit analysis of cross-lingual transfer methods. *arXiv preprint arXiv:2105.06813*.
7. Solaiman, I., Brundage, M., Clark, J., Askell, A., Herbert-Voss, A., Wu, J., ... & Wang, J. (2019). Release strategies and the social impacts of language models. *arXiv preprint arXiv:1908.09203*.
8. Peng, B., Galley, M., He, P., Cheng, H., Xie, Y., Hu, Y., ... & Gao, J. (2023). Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813*.
9. Lin, J., Dai, X., Xi, Y., Liu, W., Chen, B., Li, X., ... & Zhang, W. (2023). How Can Recommender Systems Benefit from Large Language Models: A Survey. *arXiv preprint arXiv:2306.05817*.
10. Whitehouse, C., Choudhury, M., & Aji, A. F. (2023). LLM-powered Data Augmentation for Enhanced Crosslingual Performance. *arXiv preprint arXiv:2305.14288*.
11. Talebirad, Y., & Nadiri, A. (2023). Multi-Agent Collaboration: Harnessing the Power of Intelligent LLM Agents. *arXiv preprint arXiv:2306.03314*.
12. Eloundou, T., Manning, S., Mishkin, P., & Rock, D. (2023). Gpts are gpts: An early look at the labor market impact potential of large language models. arXiv preprint arXiv:2303.10130.
13. Chen, L., Chen, X., Wu, S., Yang, Y., Chang, M., & Zhu, H. (2023). The future of chatgpt-enabled labor market: A preliminary study. arXiv preprint arXiv:2304.09823.
14. Lou, B., Sun, H., & Sun, T. (2023). Gpts and labor markets in the developing economy: Evidence from china. Available at SSRN 4426461.
15. The Artificial Intelligence Act (2023, June 14). https://artificialintelligenceact.eu/
16. Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., ... & Sifre, L. (2022). Training compute-optimal large language models. arXiv preprint arXiv:2203.15556.
17. Dong, Y., Cordonnier, J. B., & Loukas, A. (2021, July). Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In International Conference on Machine Learning (pp. 2793-2803). PMLR.
18. Ratner, N., Levine, Y., Belinkov, Y., Ram, O., Abend, O., Karpas, E., ... & Shoham, Y. (2022). Parallel Context Windows Improve In-Context Learning of Large Language Models. arXiv preprint arXiv:2212.10947.
19. Anil, R., Dai, A. M., Firat, O., Johnson, M., Lepikhin, D., Passos, A., ... & Wu, Y. (2023). Palm 2 technical report. arXiv preprint arXiv:2305.10403.
20. Chen, L., Zaharia, M., & Zou, J. (2023). FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. arXiv preprint arXiv:2305.05176.
21. *52021pc0206*. EUR-Lex — Access to European Union law — https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%3A52021PC0206
22. Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021, March). On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?🦜. In Proceedings of the 2021 ACM conference on fairness, accountability, and transparency (pp. 610-623).
23. Aryan, A. McMahon, A. (2023, June 14). *Decoding the puzzle: Unraveling the evaluation and interpretation conundrum for LLMs*.

Medium. https://medium.com/the-llmops-brief/decoding-the-puzzle-unraveling-the-evaluation-and-interpretation-conundrum-for-llms-9b57a384df08

24. *AI2 Leaderboard*. https://leaderboard.allenai.org/natural-instructions/submissions/public

25. *Chatbot arena: Benchmarking LLMs in the wild with Elo ratings*. (n.d.). LMSYS Org. https://lmsys.org/blog/2023-05-03-arena/