Less is More: Using Multiple LLMs for Applications with Lower Costs

Anonymous Authors[1]

Abstract

Large language models (LLMs) are increasingly used for querying purposes, but their associated costs vary significantly. This study investigates the pricing structures of popular LLM APIs, such as GPT-4, ChatGPT, and J1-Jumbo, revealing substantial fee differences. To mitigate the expense of using LLMs on extensive queries and text, we propose three strategies: prompt adaptation, LLM approximation, and LLM cascade. We present FrugalGPT, an adaptable LLM cascade that intelligently selects LLM combinations to reduce costs by up to 98% while matching or improving the accuracy of individual LLMs. This work establishes a foundation for sustainable and efficient LLM utilization, offering valuable insights and practical techniques for users.

1. Introduction

We are in the midst of an explosion of large language models (LLMs). The alluring possibilities of using LLMs for large-scale applications such as commerce, science, and finance have led a growing number of companies (OpenAI, AI21, CoHere, etc.) to offer LLMs as services.

While LLMs such as GPT-4 achieves unprecedented performance in tasks such as question answering, using them for high-throughput applications can be very expensive. For example, ChatGPT is estimated to cost over $700,000 per day to operate (Cosa), and using GPT-4 to support customer service can cost a small business over $21,000 a month (Cosb). In addition to the financial cost, using the largest LLMs encures substantial environmental and energy impact (BGMMS21; WRG+22), affecting the social welfare of current and future generations.

There are many LLMs now available via APIs and they charge heterogeneous prices. The cost of using a LLM API typically consists of three components: 1) prompt cost (proportional to the length of the prompt), 2) generation cost (proportional to the generation length), and 3) sometimes a fixed cost per query. We compared the cost associated with using 12 different commercial LLMs from mainstream providers including OpenAI, AI21, CoHere and Textsynth (Table 1). Their cost can differ by up to 2 orders of magnitudes: for example, the prompt cost for 10M tokens is $30 for OpenAI's GPT-4 but only $0.2 for GPT-J hosted by Textsyth. Given the heterogeneous cost and quality, how to leverage the full set of LLM options is a key challenge for pracitioners. Moreover, relying on one API provider is not reliable if that provider becomes unavailable, potentially due to spiking demand.

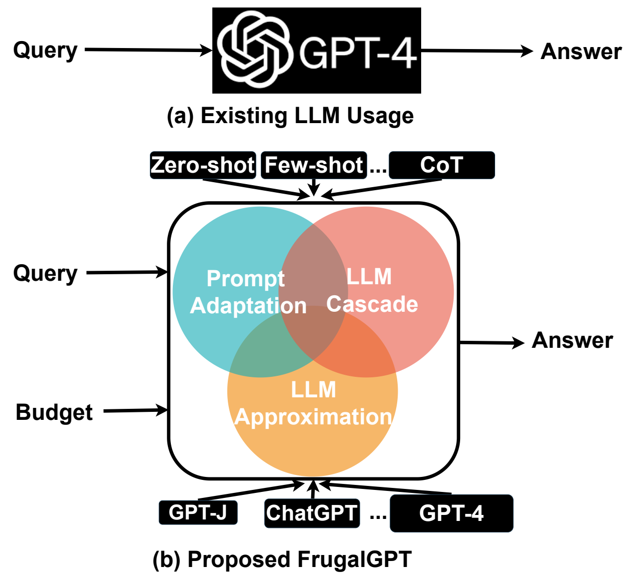

Figure 1. Our vision for reducing LLM cost while improving accuracy. (a) The standard usage sends queries to a single LLM (e.g. GPT-4), which can be expensive. (b) Our proposal is to use prompt adaption, LLM approximation and LLM cascade to reduce the inference cost. By optimizing over the selection of different LLM APIs (e.g., GPT-J and GPT-4) as well as prompting strategies (such as few-shot (LSZ+21) and chain-of-thought (CoT) (WWS+22)), we can achieve substantial efficiency gains.

Our contributions. We lay out our vision of a flexible framework that uses LLM APIs to process natural language

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

queries within a budget, termed FrugalGPT. The first half of this paper is structured as a position piece, where we discuss three main strategies for cost reduction: *prompt adaptation*, *LLM approximation*, and *LLM cascade* (Figure 1). The prompt adaptation explores how to identify effective prompts to save cost. LLM approximation aims to create simpler and cheaper LLMs to match a powerful LLM on specific tasks. LLM cascade focuses on how to adaptively choose which LLM APIs to use for different queries.

To illustrate the potential of these ideas, we implement and evaluate a simple version of FrugalGPT using LLM cascade in the second half. On each dataset and task, FrugalGPT learns to adaptively triage different queries in the dataset to different combinations of LLMs, including ChatGPT (Cha), GPT-3 (BMR+20) and GPT-4 (Ope23). Our experiments show that FrugalGPT saves up to 98% of the inference cost of the best individual LLM API while matching its performance on the downstream task. We believe this is only the tip of the iceberg and we hope FrugalGPT opens a new window toward reducing LLMs' inference cost and improving its performances.

Related Works. Prompt Engineering. Prompt engineering has emerged as a discipline for crafting prompts to enhance LLMs' performance across various applications. Recent developments include few-shot (BMR+20), chain-of-thought (WWS+22), knowledge enhancement (LLL+21; KSL+22), and numerous other prompting techniques (MDL+23; KTF+22; ZSH+22). Existing prompt engineering approaches often aim to provide detailed task explanations and in-context examples, resulting in long and expensive prompts.

System Optimization for LLMs. Numerous efforts have aimed to accelerate the training and inference time of modern deep learning models through system optimization (HMD15; Cas19; JZA19; RRWN11). Recent work focuses on post-training quantization (BHS+22; YLW+23; XLS+22), training pipeline parallelism (LZG+21), and hardware-aware pruning (KFA23) tailored for LLMs. System optimization requires modifications to LLMs' internal states (e.g., model weights), but many commercial LLM APIs do not release their models.

ML-as-a-Service. LLM APIs constitute a crucial component of the rapidly expanding machine-learning-as-a-service (MLaaS) industry. Recent studies have demonstrated the diversity of different ML APIs' predictions (BG18; KNL+20; CCZZ21) and proposed strategies for leveraging various classification ML APIs to improve performance (CZZ20; CZZ22). The outputs of LLM APIs encompass the entire natural language space, but existing work requires a fixed (and known) label set. Moreover, both prompt choices and LLM API selections significantly impact generative tasks' performance, resulting in a considerably larger optimization space than standard classification.

2. Scope and Problem Statement

Natural language query answering. In this paper, we concentrate on the standard natural language query answering task, where the objective is to answer a query q sampled from a natural language query distribution $\mathcal{Q}$. Various real-world natural language tasks, such as news classification, reading comprehension, and commonsense reasoning, can be formulated as query-answering problems.

LLM marketplace. We consider answering queries via the LLM market, which comprises K different LLM APIs, denoted by $\{f_i(\cdot)\}_{i=1}^K$. Each $f_i(\cdot) : \mathcal{P} \mapsto \mathcal{A}$ is a function that, given a prompt p from the prompt space $\mathcal{P}$, generates an answer from the answer distribution $\mathcal{A}$. Note that to use LLM APIs, one has to convert each query q to some prompt first. LLM APIs are associated with their own *cost*, typically consisting of three components: a portion proportional to the length of the prompt, a portion proportional to the length of the generated answer, and (sometimes) a fixed cost per query. Formally, given a prompt p, the cost of using the ith LLM API is denoted by $c_i(p) \triangleq \tilde{c}_{i,2}\|f_i(p)\| + \tilde{c}_{i,1}\|p\| + \tilde{c}_{i,0}$, where $\tilde{c}_{i,j}, j = 0, 1, 2$ are constants.

Problem statement: budget-aware LLM API usage. Our goal is *leveraging LLM APIs within a budget constraint*. Formally, this can be formulated as maximizing the overall task performance $\mathbb{E}_{(q,a)\in\mathcal{Q}\times\mathcal{A}}[r(a, \hat{a}(s, q))]$, while ensuring the average cost is bounded by a user-defined value b, i.e., $\mathbb{E}_{(q,a)\in\mathcal{Q}\times\mathcal{A}}[c(s, q)] \leq b$. Here, a denotes the correct answer to the query q, $\hat{a}(s, q)$ is the generated answer by some strategy s for query q, and $c(s, q)$ is the associated cost for processing query q using strategy s. The reward function $r(\cdot, \cdot)$ measures how closely the generated answer aligns with the correct one. It is crucial to note that the search space for the strategy is vast, encompassing factors such as which prompts to use, which LLM APIs to employ, and how to aggregate their responses.

3. How to Use LLMs Cheaply and Accurately

Now we present our vision on how to use LLM APIs within a budget. As shown in Figure 1 (b), we discuss three cost-reduction strategies: prompt adaptation, LLM approximation, and LLM cascade. We focus on one exemplar instance within each category, due to space limit.

Strategy 1: LLM cascade. The increasing availability of LLM APIs with heterogeneous performance and costs presents a unique opportunity for data-adaptive LLM selection. Appropriately selecting which LLMs to use can

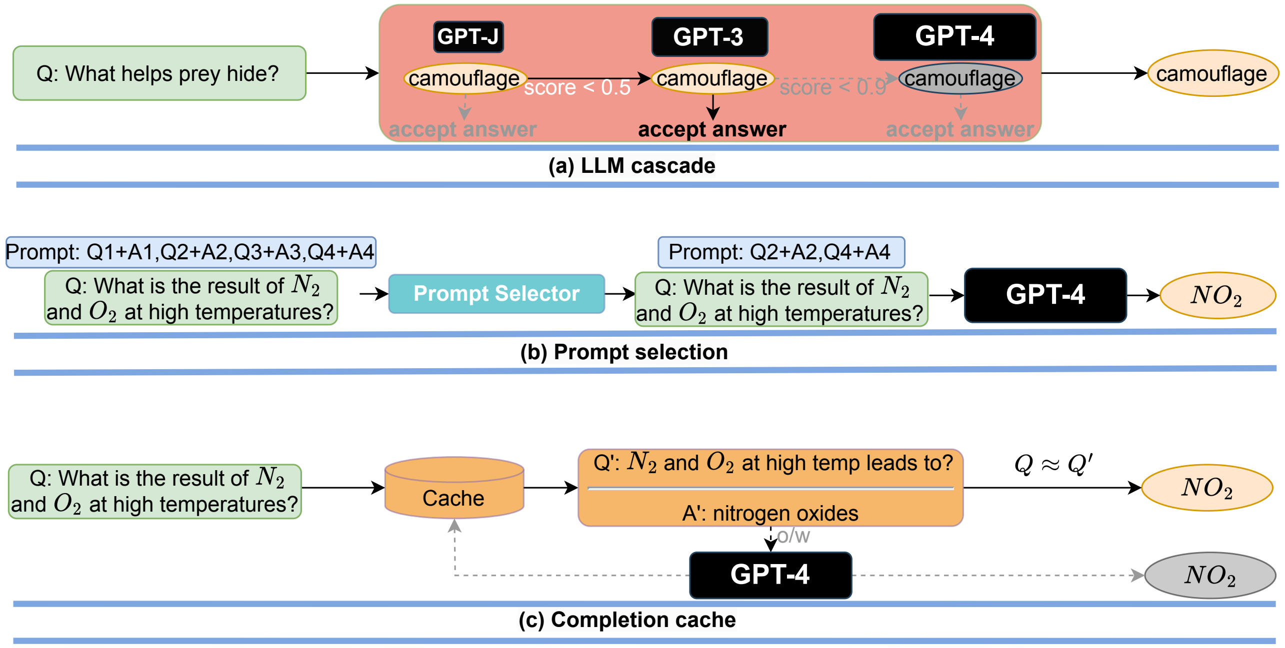

Figure 2. Illustrations of cost-saving strategies. (a) LLM cascade employs different LLM APIs for different queries. (b) Prompt selection uses a subset of in-context examples as the prompt to reduce the size of the prompt. (c) Completion cache stores and reuses an LLM API's response when a similar query is asked.

provide both cost reduction and performance improvements. *LLM cascade*, as illustrated in Figure 2 (a), is one such example. *LLM cascade* sends a query to a list of LLM APIs sequentially. If one LLM API's response is reliable, then its response is returned, and no further LLMs in the list are needed. The remaining LLM APIs are queried only if the previous APIs' generations are deemed insufficiently reliable. Query cost is significantly reduced if the first few APIs are relatively inexpensive and produce reliable generations. Below we propose a concrete cascade strategy, which we will experimentally test in the next section.

The key components of our LLM cascade consist of two elements: (i) a generation scoring function and (ii) an LLM router. The generation scoring function, denoted by $g_i(\cdot,\cdot) : \mathcal{Q} \times \mathcal{A} \mapsto [0,1]$, generates a reliability score given a query and an answer produced by the ith LLM API. The LLM router selects m LLM APIs to include in the list. Let $\boldsymbol{L} \in [K]^m$ denote the indexes of the m APIs selected by the router. Given a new query, it iteratively invokes the ith API in the list to obtain an answer $f_{L_i}(q)$. Then, it uses the scoring function to generate a score $g_i(q, f_{L_i}(q))$. It returns the generation if the score is higher than a threshold $\boldsymbol{\tau}_i$, and queries the next service otherwise.

The scoring function can be obtained by training a regression model that learns whether a generation is correct from the query and a generated answer. Learning the selected list $\boldsymbol{L}$ and the threshold vectors $\boldsymbol{\tau}$ can be modeled as a constraint optimization problem:

$$\max_{\boldsymbol{L},\boldsymbol{\tau}} \mathbb{E}\left[r(a, f_{L_z}(q))\right]$$

$$s.t.\ \mathbb{E}\left[\sum_{i=1}^{z} \tilde{c}_{L_i,2}\|f_{L_i}(q)\| + \tilde{c}_{L_i,1}\|q\| + \tilde{c}_{L_i,0}\right] \leq b,$$

$$z = \min_{i\in[K]:g_i(q,f_{L_i}(q))\geq \boldsymbol{\tau}_i} i$$

Here, z denotes the LLM API at which the router stops and returns the answer, the first constraint ensures the average cost is bounded by the budget, and the objective measures the quality of the generation $f_{L_z}(q)$ for a query q compared to the true answer a.

Strategy 2: Prompt adaptation. The cost of an LLM query increases linearly with the size of the prompt. Consequently, a logical approach to reduce the cost of using LLM APIs involves decreasing the prompt's size, a process we refer to as prompt adaptation. *Prompt selection* (as illustrated in Figure 2 (b)) is a natural example of prompt adaptation: rather than employing a prompt containing numerous examples that demonstrate how to perform a task, one can retain a small subset of examples in the prompt. This results in a smaller prompt and subsequently lower cost. An intriguing challenge of prompt selection lies in determining which examples to maintain for various queries without compromising task performance.

Strategy 3: LLM approximation. The concept of *LLM approximation* is quite simple: if an LLM API is too costly, one can approximate it using more affordable models or infrastructures. One example is the *completion cache*: as depicted in Figure 2 (c), the fundamental idea involves storing the response locally in a cache (e.g., a database) when submitting a query to an LLM API. To process a new query, we first verify if a similar query has been previously answered. If so, the response is retrieved from the cache. An LLM API is invoked only if no similar query is discovered in the cache. The completion cache provides substantial cost savings when similar queries are frequently posed. For instance, consider a search engine powered by an LLM API. If numerous users search for the same or similar keywords simultaneously, the completion cache facilitates answering all their queries by invoking the LLM only once.

Table 1. Summary of commercial LLM APIs. We use 14 LLM APIs from 6 providers. The cost was retrieved in March 2023. The cost can have three additive components: input (proportional to the number of input tokens), output (proportional to the number of generated tokens) and a fixed cost per request. IT and OT stand for input tokens and output tokens. The unit is 10M. The LLMs's costs can differ by up to 2 orders of magnitudes. For example, to process 10M input tokens, GPT-J from Textsynth costs only $0.2, but OpenAI's GPT-4 needs $30.

Provider	**API**	**Size/B**	**Cost (USD)**		
			IT	**OT**	**request**
OpenAI	GPT-Curie	6.7	2	2	0
	ChatGPT	NA	2	2	0
	GPT-3	175	20	20	0
	GPT-4	NA	30	60	0
AI21	J1-Large	7.5	0	30	0.0003
	J1-Grande	17	0	80	0.0008
	J1-Jumbo	178	0	250	0.005
Cohere	Xlarge	52	10	10	0
	Medium	6.1	10	10	0
Textsynth	GPT-J	6	0.2	5	0
	FAIRSEQ	13	0.6	15	0
	GPT-Neox	20	1.4	35	0
Databricks	Dolly	7	0.27	0.27	0
ForeFront	QA	16	5.8	5.8	0

4. LLM Cascade Reduces Cost

In this section, we present an empirical study on the FrugalGPT LLM cascade. Our goal is focused on demonstrating how much cost can be saved without hurting the accuracy.

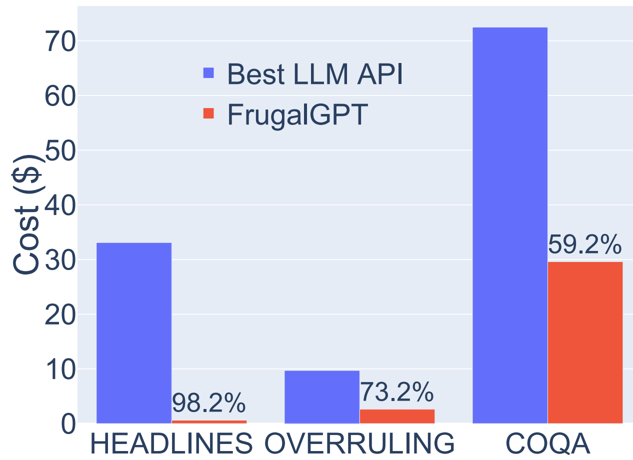

Figure 3. Cost savings achieved by FrugalGPT to match the best individual LLM APIs. Overall, FrugalGPT reduces the cost by from 50% to 90%.

Setups: LLM APIs, Tasks, Datasets, and FrugalGPT instances. We have selected 14 LLM APIs from 6 mainstream providers, namely, OpenAI (Ope), AI21 (AI2), CoHere (CoH), Textsynth (Tex), Databricks (Dol), and ForeFrontAI (FFA). The details are summarized in Table 1. FrugalGPT has been developed on top of these APIs and evaluated on three datasets, namely, HEADLINES (SK21), OVERRULING (ZGA^{+}21) and COQA (RCM19). We focus on the LLM cascade approach with a cascade length of 3, as this simplifies the optimization space and already demonstrates exciting results.

Cost Savings. Here, we focus on examie if FrugalGPT can reduce costs while maintaining accuracy and, if so, by how much. Figure 3 displays the overall cost savings of FrugalGPT, which range from 50% to 98%. This is feasible because FrugalGPT identifies the queries that can be accurately answered by smaller LLMs and, as a result, only invokes those cost-effective LLMs. Powerful but expensive LLMs, such as GPT-4, are utilized only for challenging queries detected by FrugalGPT.

5. Discussions and Future Prospects

The substantial cost of employing LLMs in real-world scenarios presents a considerable barrier to their widespread usage. In this paper, we outline and discuss practical strategies for reducing the inference cost of using LLM APIs. We also developed FrugalGPT to illustrate one of the cost-saving strategies, LLM cascade. Our empirical findings show that FrugalGPT can reduce costs by up to 98% while preserving the performance of cutting-edge LLMs.

References

[AI2] AI21 LLM API. https://www.ai21.com/. Accessed: 2023-03-31.

[BG18] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91. PMLR, 2018.

[BGMMS21] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623, 2021.

[BHS+22] Haoli Bai, Lu Hou, Lifeng Shang, Xin Jiang, Irwin King, and Michael R Lyu. Towards efficient post-training quantization of pre-trained language models. *Advances in Neural Information Processing Systems*, 35:1405–1418, 2022.

[BMR+20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[Cas19] Stephen Cass. Taking ai to the edge: Google's tpu now comes in a maker-friendly package. *IEEE Spectrum*, 56(5):16–17, 2019.

[CCZZ21] Lingjiao Chen, Tracy Cai, Matei Zaharia, and James Zou. Did the model change? efficiently assessing machine learning api shifts. *arXiv preprint arXiv:2107.14203*, 2021.

[Cha] ChatGPT Announcement. https://openai.com/blog/chatgpt. Accessed: 2023-03-31.

[CoH] CoHere LLM API. https://cohere.com/. Accessed: 2023-03-31.

[Cosa] Cost estimation of using GPT-3 for real applications. https://www.semianalysis.com/p/the-inference-cost-of-search-disruption. Accessed: 2023-03-31.

[Cosb] Cost estimation of using GPT-3 for real applications. https://neoteric.eu/blog/how-much-does-it-cost-to-use-gpt-models-g Accessed: 2023-03-31.

[CZZ20] Lingjiao Chen, Matei Zaharia, and James Y Zou. Frugalml: How to use ml prediction apis more accurately and cheaply. *Advances in neural information processing systems*, 33:10685–10696, 2020.

[CZZ22] Lingjiao Chen, Matei Zaharia, and James Zou. Efficient online ml api selection for multi-label classification tasks. In *International Conference on Machine Learning*, pages 3716–3746. PMLR, 2022.

[Dol] Dolly deployed on Databricks Model Serving. https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-inst Accessed: 2023-03-31.

[FFA] forefront AI LLM API. https://beta.forefront.ai/. Accessed: 2023-03-31.

[HMD15] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[JZA19] Zhihao Jia, Matei Zaharia, and Alex Aiken. Beyond data and model parallelism for deep neural networks. *Proceedings of Machine Learning and Systems*, 1:1–13, 2019.

[KFA23] Eldar Kurtic, Elias Frantar, and Dan Alistarh. Ziplm: Hardware-aware structured pruning of language models. *arXiv preprint arXiv:2302.04089*, 2023.

[KNL+20] Allison Koenecke, Andrew Nam, Emily Lake, Joe Nudell, Minnie Quartey, Zion Mengesha, Connor Toups, John R Rickford, Dan Jurafsky, and Sharad Goel. Racial disparities in automated speech recognition. *Proceedings of the National Academy of Sciences*, 117(14):7684–7689, 2020.

[KSL+22] Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp. *arXiv preprint arXiv:2212.14024*, 2022.

[KTF+22] Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*, 2022.

[LLL+21] Jiacheng Liu, Alisa Liu, Ximing Lu, Sean Welleck, Peter West, Ronan Le Bras, Yejin Choi, and Hannaneh Hajishirzi. Generated knowledge prompting for commonsense reasoning. *arXiv preprint arXiv:2110.08387*, 2021.

[LSZ+21] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*, 2021.

[LZG+21] Zhuohan Li, Siyuan Zhuang, Shiyuan Guo, Danyang Zhuo, Hao Zhang, Dawn Song, and Ion Stoica. Terapipe: Token-level pipeline parallelism for training large-scale language models. In *International Conference on Machine Learning*, pages 6543–6552. PMLR, 2021.

[MDL+23] Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, et al. Augmented language models: a survey. *arXiv preprint arXiv:2302.07842*, 2023.

[Ope] OpenAI LLM API. https://platform.openai.com/. Accessed: 2023-03-31.

[Ope23] OpenAI. Gpt-4 technical report. *arXiv preprint https://arxiv.org/abs/2303.08774*, 2023.

[RCM19] Siva Reddy, Danqi Chen, and Christopher D Manning. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2019.

[RRWN11] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. *Advances in neural information processing systems*, 24, 2011.

[SK21] Ankur Sinha and Tanmay Khandait. Impact of news on the commodity market: Dataset and results. In *Advances in Information and Communication: Proceedings of the 2021 Future of Information and Communication Conference (FICC), Volume 2*, pages 589–601. Springer, 2021.

[Tex] Textsynth LLM API. https://textsynth.com/. Accessed: 2023-03-31.

[WRG+22] Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, Jinshi Huang, Charles Bai, et al. Sustainable ai: Environmental implications, challenges and opportunities. *Proceedings of Machine Learning and Systems*, 4:795–813, 2022.

[WWS+22] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

[XLS+22] Guangxuan Xiao, Ji Lin, Mickael Seznec, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. *arXiv preprint arXiv:2211.10438*, 2022.

[YLW+23] Zhewei Yao, Cheng Li, Xiaoxia Wu, Stephen Youn, and Yuxiong He. A comprehensive study on post-training quantization for large language models. *arXiv preprint arXiv:2303.08302*, 2023.

[ZGA+21] Lucia Zheng, Neel Guha, Brandon R Anderson, Peter Henderson, and Daniel E Ho. When does pretraining help? assessing self-supervised learning for law and the casehold dataset of 53,000+ legal holdings. In *Proceedings of the eighteenth international conference on artificial intelligence and law*, pages 159–168, 2021.

[ZSH+22] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed Chi. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.