COSC 6368 - Artificial Intelligence
Project Report

BUDGETGPT: EMBRACING SMART COST MANAGEMENT FOR LEANER AI SOLUTIONS

November 18, 2023

Group 2
Sanjith Balachandran
Yang Lu
Bader Ala Salem
Department of Computer Science
College of Natural Sciences and Mathematics
University of Houston
sbalachandran2@uh.edu, ylu17@central.uh.edu,
basalem@uh.edu

1 Abstract

The rapid evolution of large language models (LLMs) and their growing demand have highlighted the need for cost-effective query strategies in the marketplace. This project builds on FrugalGPT, an API specifically developed to facilitate accurate queries to LLMs within a budget constraint. Our main objective is to enhance the score model integral to the LLM cascade strategy, leading to the development of a novel LLM API called BudgetGPT. By incorporating the fundamental tenets of FrugalGPT, BudgetGPT presents an efficient solution aimed at diminishing inference costs by minimizing LLM API calls in situations where the likelihood of obtaining an acceptable answer is low. This optimization is achieved while preserving the accuracy of the output. Our findings, through the utilization of our scoring model, have revealed the effectiveness of skipping certain API calls, leading to substantial cost reduction. This work also demonstrates the potential to address cost management challenges in the dynamic landscape of the LLM marketplace.

2 Introduction

This project is built upon the research paper titled *FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance*. The paper introduces FrugalGPT, a Large Language Model (LLM) API designed to enable users to make accurate queries to LLMs within a constrained budget. The authors propose three query strategies: 1) prompt adaptations, 2) LLM approximation, and 3) LLM cascade. These strategies aim to enhance accuracy while reducing the inference cost. In their experiments, the data demonstrates that FrugalGPT can achieve up to 98% cost savings or improve accuracy by 4% with the same cost benchmark compared to GPT-4 [1]. Our goal is to enhance the score model which the LLM cascade strategy relies on. By developing our novel score model, we introduce a new LLM API, BudgetGPT. Building upon FrugalGPT, BudgetGPT presents a creative solution to effectively decrease inference costs, while also preserving output accuracy.

3 Literature Review

3.1 Large Language Model Landscape Survey

ChatGPT, the large language model developed by OpenAI, has achieved the fastest-growing user base in the history of web applications [2]. According to a Reuters report on Feb. 1, 2023, within just two months of its launch in November 2022, ChatGPT was able to amass 100 million monthly active users in January 2023 [3]. The exponential growth shows the tremendous advancements in Natural Language Processing (NLP) and understanding in large language models. Models like OpenAI's GPT-3, with 175 billion parameters, showcases the recent trend of LLMs increasing in size and complexity. Figure 1 shows the LLM model's size has surged in recent years.

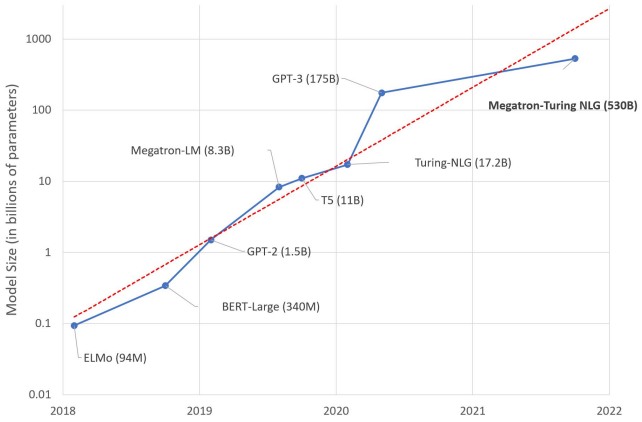

Figure 1: Large Language Model Size[4]

In the year 2023, the race to develop the most exceptional LLM has intensified, with large corporations, small startups, and the open-source community all vying to create cutting-edge models [5]. As a result, there has been an influx of hundreds of newly released LLMs. As per industry analysis, NLP market is projected to escalate from $11 billion in 2020, to over $35 billion in 2026 [6]. It is worth noting some of the most popular LLMs and their significance. Here, we only list some of the LLMs used in the FrugalGPT paper, such as:

- **GPT-4** GPT-4 is the champion of LLMs available in 2023. Besides the remarkable capabilities with complex reasoning understanding, advanced coding, proficiency in various academic exams, and human-level performance skills, GPT-4 scores approximately 80% in factual evaluations across various categories, and OpenAI has invested considerable effort in aligning the model with human values through Reinforcement Learning from Human Feedback (RLHF) and adversarial testing with domain experts. GPT-4 has been trained with over 1 trillion parameters and a maximum context length of 32,768 tokens [5].

- **GPT-3.5** We can say GPT-3.5 is the lite version of GPT-4. It inherits from GPT-4 with less expertise in specific domain. This makes GPT-3.5 very agile and generate output in seconds. In the HumanEval benchmark, the GPT-3.5 model achieved a score of 48.1%, while the GPT-4 model scored an impressive 67%, the highest among general-purpose large language models. It has been trained on 175 billion parameters.

- **Cohere** Cohere is an AI startup founded by former Google employee. It distinguishes itself by catering specifically to enterprises and addressing generative AI use cases for corporations. The range of their models varies from small ones, with 6 billion parameters, to large models trained on 52 billion parameters [7].

- **TextSynth** Textsynth is a web service that provides access to Large Language Models (LLMs) for text completion, translation, and image generation. Textsynth uses custom inference code to get a faster inference (hence lower costs) on standard GPUs and CPUs. Textsynth supports many Transformer variants (GPT-J, GPT-NeoX, GPT-Neo, OPT, Fairseq GPT, M2M100, CodeGen, GPT2) and Stable Diffusion. The limitations of Textsynth are: rate-limited (user can only make limited queries per hour), inaccurate (it is a relatively small LLM in the current market), as well as not being versatile (it is not well trained) [8].

- **AI21** AI21 Labs is a Tel Aviv-based company specializing in Natural Language Processing (NLP), which develops AI systems that can understand and generate natural language. AI21 Labs' most popular product is Jurassic-1, a Large Language Model (LLM) with 137 billion parameters. Jurassic-1 is trained on a massive dataset of text and code, and it can be used for a variety of tasks, including text generation, translation, QA, and summarization [9].

Choosing the right LLM can significantly impact cost reduction and ensure reliable performance. In order to make wise decisions, we have to study on the pricing models of the current LLM marketplace. Here are some of the most common pricing models on LLM marketplace [10]:

- **Pay-per-token**: This is the most common pricing model. Users pay for each token they use.

- **Pay-per-prompt**: This pricing model is similar to pay-per-token, but users pay for each prompt they use.

- **Subscription**: This pricing model is less common, but it is becoming more popular. Users pay a monthly or annual subscription fee to access an LLM.

In addition to these pricing models, some LLM marketplaces also offer discounts for bulk purchases or for long-term subscriptions. We did a further inspection on the pricing model of the LLMs listed above, and here are the details below:

- **GPT-4** It is currently available through the ChatGPT Plus subscription service, which costs $20 per month. Users are charged a rate of $0.03 per 1,000 prompt tokens and $0.06 per 1,000 completion tokens. The pricing model for GPT-4 is based on the number of tokens used.

- **GPT-3.5** It is also token-based, and charges with different tiers [11].

- **Textsynth** Textsynth uses the pay-per-token pricing model. It also offers a free tier that allow users to use up to 500 tokens per hour. After that, user will be charged at regular rates. The price for GPT-J is $0.02 per million input tokens, and $5.00 per million generated token [8].

- **AI21** AI21 uses the pay-per-token pricing model for Jurassic-1 LLM. The prices differ, based on its three different qualities [9].

3.2 FrugalGPT Review

In the FrugalGPT paper, the authors propose three strategies to query LLMs with a limited budget. They are 1) prompt adaption, 2) LLM approximation, and 3) LLM cascade.

3.2.1 Prompt adaption

The prompt adaption is based on prompt engineering techniques, such as few-shot [12], chain-of-thought [13], knowledge enhancement [14], and more. The primary goal is to decrease the prompt size and subsequently reduce the cost, while also improving the performance of LLMs.

3.2.2 LLM approximation

The fundamental idea of LLM approximation is to store the response locally in a cache. When processing a new query, the system first checks for similar queries in the cache. If a match is found, the response is directly retrieved from the cache. Within this strategy, the authors also introduce another method called model fine-tuning, which utilizes a large LLM to train and improve the performance of a smaller AI model.

3.2.3 LLM cascade

This strategy is particularly intriguing. It involves sending a query to a sequence of LLM APIs, starting from the most affordable one. If the output score meets the threshold, the response is returned; otherwise, the next LLM is queried. If the output is still not satisfactory, the last and most expensive LLM (GPT-4) is consulted. The crucial components in the LLM cascade are the score function and the selected list of LLMs. The score function is a simple regression model trained by DistilBERT [15], a pre-trained smaller general-purpose language representation model. DistilBERT can be fine-tuned to achieve good performance across a wide range of tasks, similar to its larger counterparts. The score function determines the quality of the output. The threshold is constrained, and the process of determining the selected list L and the threshold vector τ can be formulated as a constraint optimization problem. Figure 2 illustrates the high level architecture of score function in FrugalGPT LLM cascade strategy. The authors use three datasets to train the score function:

- Headlines: using financial news title to determine the gold price trend.
- Overruling: using legal document dataset to determine a giving sentence is an overruling.

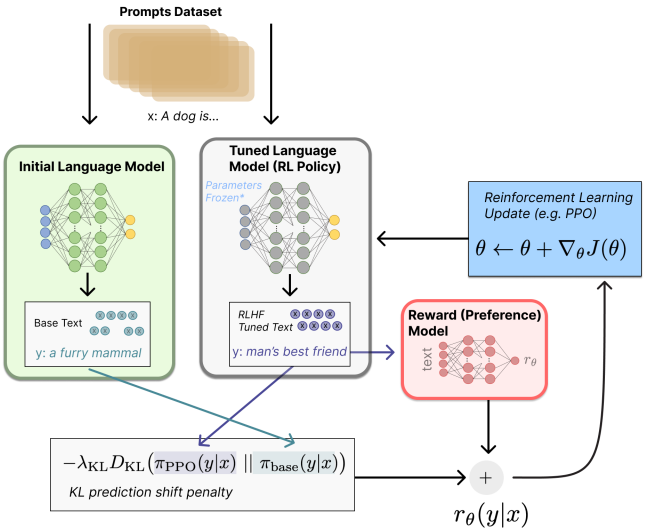

Figure 2: Using reinforcement learning to optimize the LLM with the reward Model [16]

- COQA: a reading comprehensive dataset.

Based on these datasets experiment, the authors observe the similarity among a pair of LLMs. Figure 3 provides a summary of the similarity between GPT-4 and other LLMs. It illustrates that GPT-3 exhibits the highest similarity to GPT-4, while Textsynth GPT-J stands as the least similar model in comparison to GPT-4. The remaining LLMs are in between. Based on this comparison, we can infer that Textsynth GPT-J tends to provide the least reliable answers among all the LLMs, while also being the lowest cost option.

4 Our Approach

The fundamental concept behind FrugalGPT is to prioritize initially querying the cheapest LLM. While this strategy is effective for simple queries, where small LLMs can produce satisfactory output, it becomes problematic when faced with challenging queries beyond the capabilities of small LLMs. In such cases, FrugalGPT continues to query from the small LLMs, resulting in unnecessary expenses. These expenses can include charges for querying the LLMs, internet cost, and caching the results. The total cost of such queries can be represented as:

$$TotalCost = \sum_{i=1}^{m} C_i{}^{1}$$

[1] m is the depth of LLM cascade to answer a query

	GPT-4 Headlines	GPT-4 Overruling	GPT-4 COQA	Total	
GPT-C	21	25	18	64	11.33%
ChatGPT	7	5	22	34	6.02%
GPT-3	6	2	9	17	3.01%
J1-L	17	26	16	59	10.44%
J1-G	12	29	13	54	9.56%
J1	7	22	13	42	7.43%
CoHere	12	9	14	35	6.19%
FA-Q	18	26	17	61	10.80%
GPT-J	14	46	19	79	13.98%
FSQ	18	34	16	68	12.04%
GPT-Neo	9	26	17	52	9.20%
Total				565	100.00%

Figure 3: Similarity of LLMs with GPT-4

This means the user may end up wasting money on small LLMs that cannot adequately handle the difficult queries.

We came up with a new strategy to prevent wasting money on small LLMs for domain-demanding queries. With this approach, we employ a new score model to predict the likelihood of receiving a correct answer from an LLM before actually querying it. If the possibility is below a threshold, we bypass that LLM and move on to the next one for evaluation. This way, we can effectively save both money and time by avoiding the inefficient LLMs.

5 Project Implementation

5.1 BudgetGPT Architecture

In the context of FrugalGPT, "Quality" represents a numeric measure of the exact match between the Large Language Model (LLM) answer and the true answer. This quality metric serves as the target variable for our scoring model. To train the scoring model, we utilize DistilBERT and the training dataset from FrugalGPT, which comprises 5,000 queries, along with their corresponding true answers.

During training, we split the training data into two parts: 70% is used to train the score model, and the remaining 30% is utilized as test data. There is a distinction between our score model and the FrugalGPT score model. Unlike the FrugalGPT score model, which uses both the query and the LLM answer as features, our score model only utilizes the query as a feature.

Once the score models are trained, it should be able to predict the possibility of each LLM's capability to render the correct answer for a given query. After the training of the

score model, we use the test data split from the training data to generate a score for each LLM score model. Then we used these scored data to find an optimal scoring threshold for each LLM. Figure 4 shows the diagram of our scoring model process.

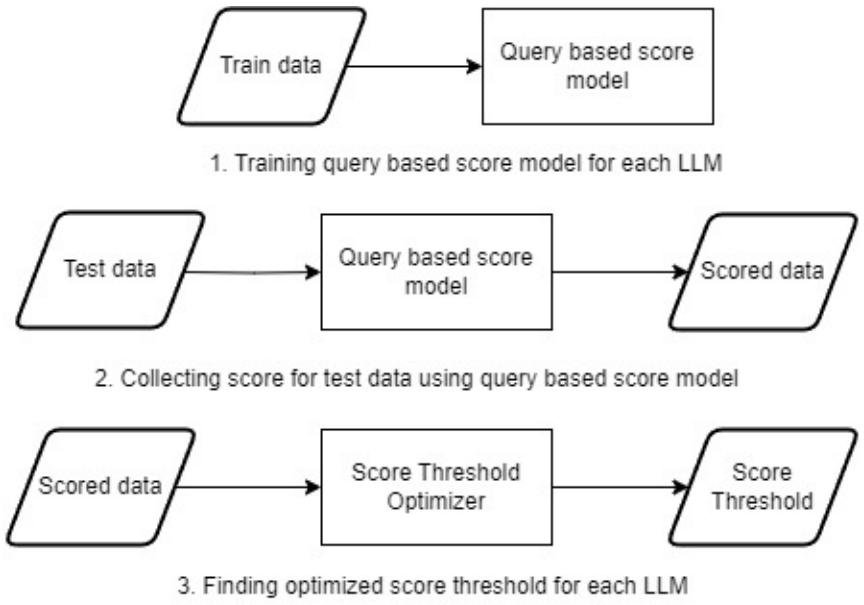

Figure 4: Our Scoring Model Process

Our query-based score evaluation is done as a pre-condition to FrugalGPT's score model. If the evaluation was successful for a query, then the LLM API will be called to get the LLM answer, and FrugalGPT's scoring will be evaluated. Figure 5 illustrates the changes made to FrugalGPT's architecture.

We used the test data from FrugalGPT to test the performance of our model. The expected cost should be lower than FrugalGPT.

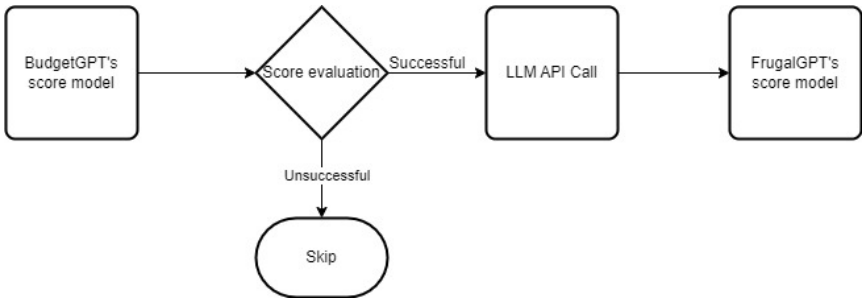

Figure 5: BudgetGPT Architecture

5.2 Query Score Threshold Optimization Algorithm

We implemented the Query Score Threshold Optimization Algorithm to find an optimal threshold to predict an acceptable LLM answer using our score model.

Algorithm 1 Query Score Threshold Optimization

procedure THRESHOLDOPTIMIZATION(data: DataFrame containing the dataset)
  $X, y \leftarrow$ SplitFeaturesAndLabels($data$, features = ["$score\_int$"], label = "$quality\_boolean$")
  $X\_train, X\_test, y\_train, y\_test \leftarrow$ TrainTestSplit(X, y, test_size = 0.3)
  $X\_train\_resampled, y\_train\_resampled \leftarrow$ ApplySMOTE(X_train, y_train)
  models $\leftarrow$ DefineModels()
  desired_precision $\leftarrow$ 0.95
  best_threshold $\leftarrow$ 0.5
  for model, param_dist **in** models **do**
    model $\leftarrow$ HyperparameterTuning(model, param_dist, $X\_train\_resampled, y\_train\_resampled$)
    $y\_pred\_proba \leftarrow$ model.PredictProbability(X_test)
    for threshold **in** ThresholdRange() **do**
      $y\_pred\_binary \leftarrow$ model.ApplyThreshold(y_pred_proba, threshold)
      precision $\leftarrow$ CalculatePrecision(y_test, y_pred_binary)
      if precision $\geq$ desired_precision **then**
        best_model $\leftarrow$ model
        best_threshold $\leftarrow$ threshold
        best_precision $\leftarrow$ precision
        break
      end if
    end for
  end for
  return best_threshold
end procedure

The score data that we have does contain some imbalanced data. So before building the models, we address the class imbalance issue by applying Synthetic Minority Over-sampling Technique (SMOTE) to oversample the minority class (True values). SMOTE generates synthetic samples to balance the class distribution by interpolating new instances based on existing minority class instances.

We used several models to evaluate, each with their own respective hyperparameter ranges for hyperparameter tuning, using RandomizedSearchCV. The models considered are Logistic Regression, Random Forest, Gradient Boosting, and Support Vector Machine (SVM).

The selected threshold is the one that achieves the desired precision on the test set. We

used 0.95 as the desired precision. In case the desired precision is not achieved, the selected threshold will be defaulted to 0.5.

6 Testing and Experiments

Based on our test using the same test dataset (test.csv) as FrugalGPT, we performed queries and found that 35 rows were skipped from the first LLM in the LLM cascade chain. Using the spreadsheet generated from BudgetGPT, we analyzed the results and identified 35 specific rows where we successfully avoided querying the small LLMs, after estimating their probability of generating correct answers. This strategic approach allowed us to avoid wasting money on these low-quality LLMs and optimize our query decisions effectively. Figure 6 showcases the observed difference in query size and the corresponding trend of additional costs incurred by FrugalGPT.

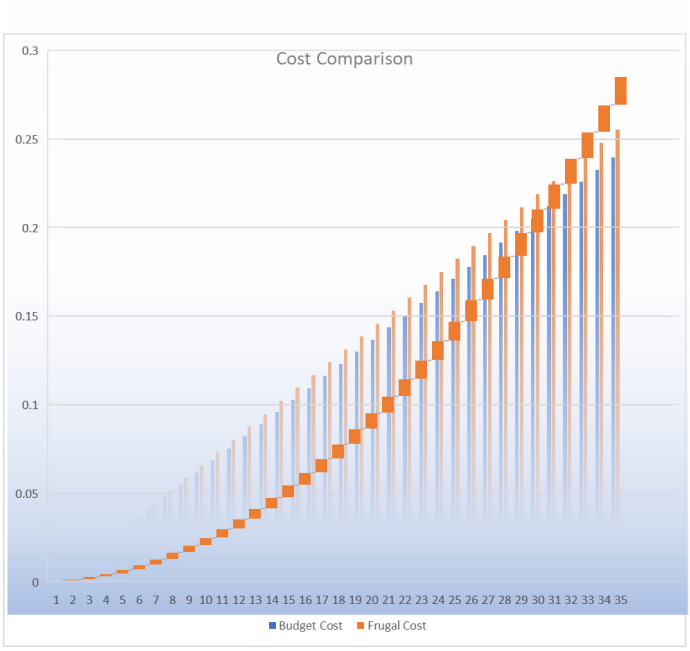

Figure 6: BudgetGPT cost vs. FrugalGPT cost

As the query size increases, we can observe the trend of extra costs from FrugalGPT. While the 35 cases out of 5,000 samples may not seem statistically significant, they serve as evidence validating the BudgetGPT concept's effectiveness and potential for cost savings.

7 Conclusion

The increasing popularity of LLMs has highlighted the importance of query strategies and cost management. As more LLMs launch into the marketplace, consumers require effective strategies to handle inference costs. BudgetGPT has demonstrated advancements in cost-saving over FrugalGPT. Our experiments have successfully validated the concept, showing its potential for significant money savings. While BudgetGPT has room for improvement, further fine-tuning with diverse training datasets can enhance accuracy and cost efficiency. Considering our limited funding and time constraints, we believe our current version serves as a promising starting point.

We also noticed the fundamental problem with FrugalGPT API. The assumption made in the FrugalGPT paper that LLM pricing models will remain unchanged in the future is not realistic. In a competitive marketplace, LLM providers may adopt various pricing strategies to acquire market shares. Some providers, like Textsynth, may subsidize their revenue through VC investment, offering lower prices to attract users and dominate the market. However, once they establish a strong position, they might significantly alter their pricing structure.

Furthermore, large LLMs like OpenAI GPT-4 could also adopt dynamic pricing models. In their quest for market share, they may charge different prices depending on the complexity of the query. For easy queries, the cost might be significantly reduced, while for challenging queries requiring specific domain knowledge, they may charge higher prices, possibly two or more times the standard rate.

This dynamic nature of LLM pricing models can introduce uncertainty and complexity for cost management and strategy optimization. It emphasizes the need for continuous monitoring and adaptation to ensure cost-effectiveness in utilizing LLMs, particularly when relying on cost-saving approaches like FrugalGPT.

As LLM providers continue flocking into the LLM marketplace, introducing a query auction model could revolutionize the way consumers interact with LLMs. In this model, queries would be submitted to the marketplace and auctioned off to different LLM providers. The provider offering the most accurate and cost-efficient response would win the auction, benefiting the consumer.

Such a marketplace would introduce an intriguing research topic surrounding query strategies and how they can optimize consumer benefits. The query strategy's role in navigating this dynamic marketplace, selecting the most suitable LLM provider for each query based on factors like accuracy, cost, and other criteria, could lead to significant cost savings and improved performance for consumers.

The concept of a query auction pricing model opens up exciting avenues for research and development in the LLM field, providing the potential to transform the way LLM services are accessed and utilized by users.

We provide a repository including an implementation of the algorithm: `https://github.com/sanjithuom/BudgetGPT/tree/main`

References

[1] L. Chen, M. A. Zaharia, and J. Y. Zou, "Frugalgpt: How to use large language models while reducing cost and improving performance," 2023.

[2] C. Gordon, "Chatgpt is the fastest growing app in the history of web applications." [Online]. Available: https://www.forbes.com/sites/cindygordon/2023/02/02/chatgpt-is-the-fastest-growing-ap-in-the-history-of-web-applications/?sh=86f18a5678ce

[3] K. Hu, "Chatgpt sets record for fastest-growing user base - analyst note." [Online]. Available: https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/

[4] J. Simon, "Large language models: A new moore's law?" [Online]. Available: https://huggingface.co/blog/large-language-models

[5] A. Sha, "12 best large language models (llms) in 2023." [Online]. Available: https://beebom.com/best-large-language-models-llms/

[6] S. Siddiqui, "Breaking new ground: Whatâs on the horizon for large language models (llms)?" [Online]. Available: https://tblocks.com/breaking-new-ground-whats-on-the-horizon-for-large-language-models-llms/

[7] [Online]. Available: https://cohere.com/

[8] [Online]. Available: https://textsynth.com/pricing.html

[9] [Online]. Available: https://www.ai21.com/studio/pricing

[10] Truefundry. [Online]. Available: https://blog.truefoundry.com/economics-of-large-language-models/

[11] [Online]. Available: https://openai.com/pricing

[12] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[13] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," 2023.

[14] J. Liu, A. Liu, X. Lu, S. Welleck, P. West, R. L. Bras, Y. Choi, and H. Hajishirzi, "Generated knowledge prompting for commonsense reasoning," 2022.

[15] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," 2020.

[16] N. Lambert, L. Castricator, L. von Werra, and A. Havrilla, "Illustrating reinforcement learning from human feedback (rlhf)." [Online]. Available: https://huggingface.co/blog/rlhf