Democratizing High-Quality LLMs through Iterative Refinement Pipelines

Table of Contents

Acknowledgment of Major Assistance

This research began in July 2025 at the NYU Agentic Learning AI Lab and was later continued with computational support from the High Performance Computing facilities at the University of Houston. I would like to express my sincere gratitude to Jack Lu from the NYU Lab for his guidance during the early stages of the project, and to Professor Andreas Mang from the University of Houston for sponsoring my access to the UH HPC resources, which were essential for carrying out this work.

Abstract

Large language models (LLMs) have become widely used for tasks such as translation, writing assistance, and code generation, with platforms like ChatGPT growing from 300 million users in 2022 to a projected 1 billion in 2025. Closed-source models dominate in performance but lack transparency, restricting use in privacy-sensitive domains. However, open-source LLMs—while enabling local deployment and full data control—typically lag in accuracy, especially in verification and correction tasks crucial for reliable deployment.

This work investigates methods to improve open-source LLM accuracy without retraining by using iterative refinement loops, where one model evaluates and corrects the outputs of another. The goal is to identify which open-source models perform best as evaluators across different domains. We test three relevant models—LLaMA 3, Qwen 2.5, and Qwen 3—on handcrafted question–answer pairs spanning logical reasoning, mathematics, factual knowledge, and coding. Models were prompted to first verify answers and, if incorrect, produce corrections.

Results show Qwen 3 excels in reasoning, math, and code correction, while LLaMA 3 leads in factual verification and correction. Qwen 2.5 provides balanced but mid-tier performance across domains. Future work will expand dataset size, incorporate automated fact-checking pipelines, and explore hybrid refinement strategies combining model strengths.

If successful, this approach will improve open-source LLM reliability for academic, industrial, and privacy-critical applications, reducing dependence on closed-source systems and broadening the accessibility of high-performance language AI.

Introduction

The recent rise to fame of AI has been most obvious through the use of LLMs. LLMs (Large Language Models) are pretrained transformer models that have been trained on a significant

corpus of text on a variety of different topics. These models are intended to have general-purpose text comprehension and generation capabilities. LLMs are becoming increasingly popular for use in a wide range of domains. They have been shown to be proficient in a variety of tasks such as: translation, open writing generations, code synthesis, etc [1]. ChatGPT, an LLM created and distributed by Open AI, has gone from 300 million users in 2022 to a projected 1 billion in 2025. Some other prevalent LLMs being used are Claude by Anthropic, and Gemini by Google.

These prevalent LLMs all fall under the umbrella of being closed source LLMs. A closed-source large language model is an LLM whose architecture, training data, and parameter weights are not publicly disclosed, with access typically restricted through proprietary APIs or licensing agreements. In comparison, an open-source large language model is an LLM whose architecture, training data (fully or partially), and parameter weights are publicly released, allowing users to inspect, modify, and deploy the model without proprietary restrictions. As illustrated in this plot, closed source LLMs dominate the industry in terms of performance compared to open source models.

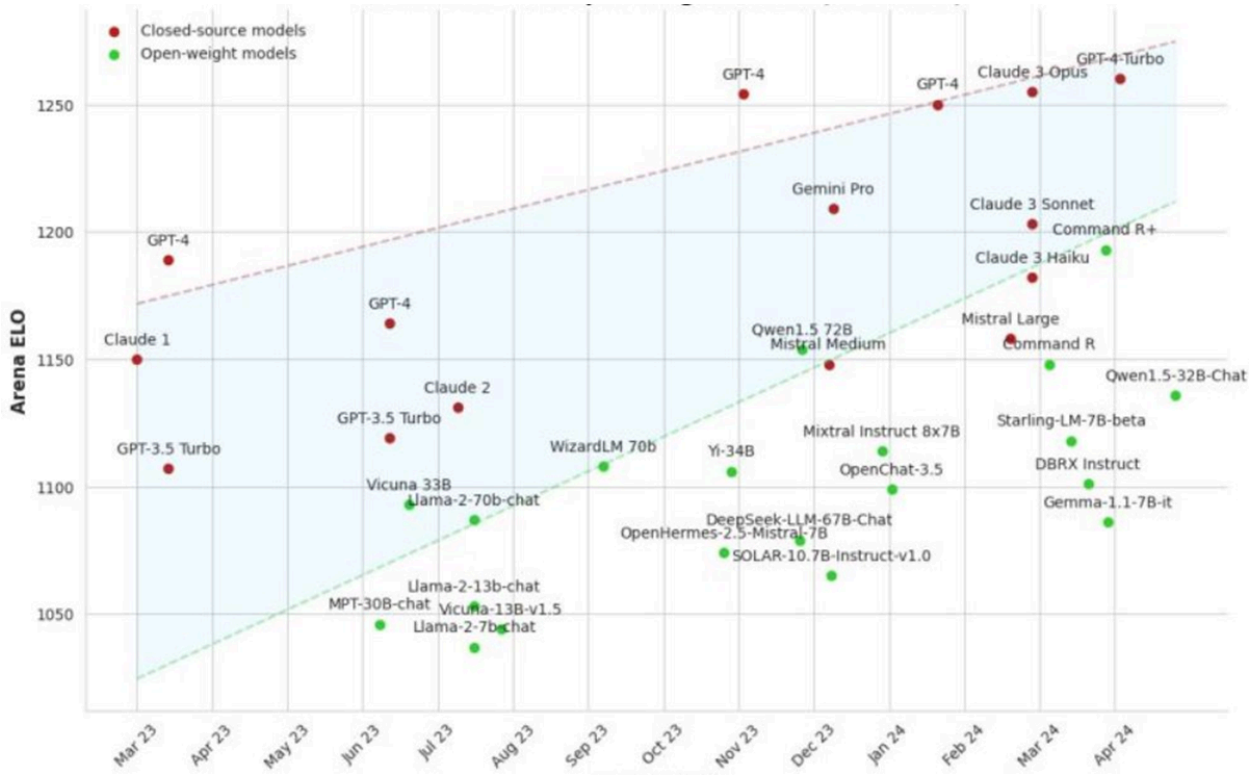


Image 1

They tend to be more accurate in their generations compared to open-source models.

There is a hidden issue within the closed-source models being so domainant. By having control over the servers that the dominant LLMs are being run on, they also have control over all the data going into these models. This leads to a lack of transparency about where and how the data

is being used and stored. When the models that are the highest accuracy do not allow for a high level of transparency in how data is being used, it restricts the use cases for LLMs. For example, a law firm cannot fine tune a model to help them with paperwork, because that fine tuning training would be a security risk and a violation of privacy. There needs to be an LLM that allows people to run models on whatever servers they want.

Open-source LLMs are able to be run on local servers and allow for full transparency of the data. The problem is that open-source models have low accuracy compared to closed-source which is crucial when trying to implement its use. The model architecture of these models is a parameter that is difficult to change and so is the training of the model, because of its high compute requirements, so we must focus on improving open-source LLM accuracy without changing the LLM itself. A possible solution is using iterative-refinement loops, which is a technique where one evaluator LLM is used to give feedback to a generator LLM in order to improve the accuracy of its answers.

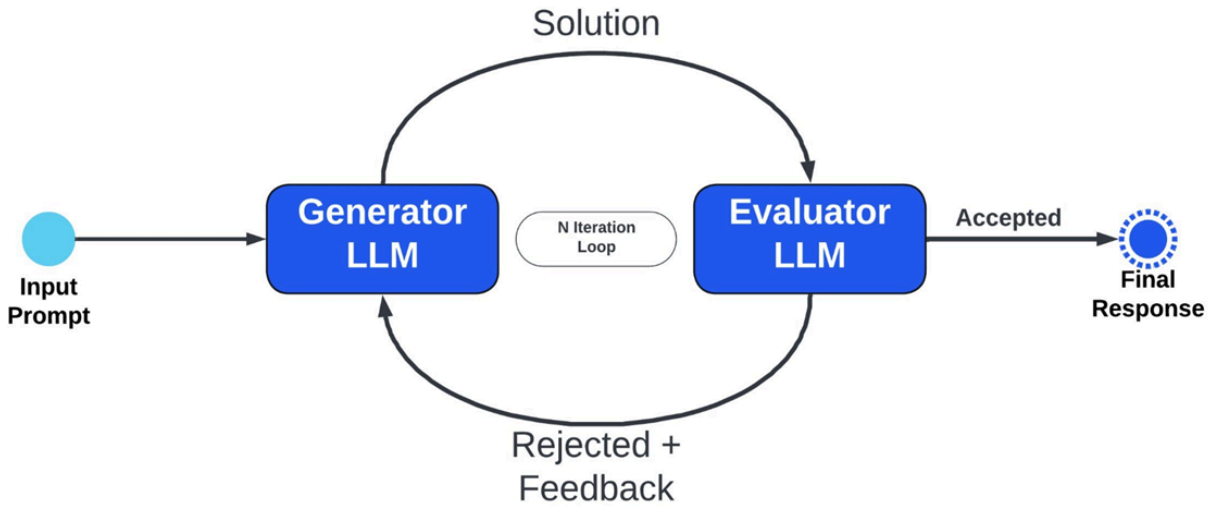


Image 2

This study aims to identify the ability of common use open source LLMs to verify and correct their own or other LLMs answers in order to develop the optimal evaluator LLM order for iterative refinement loops between LLMs. We will be observing the variance in results based on the domain and question as well as the phrasing. The results will then be analyzed relatively to the unique architectures of the various models. Since the models are open source, we will be able to attribute specific nuances in the architecture of each model to its results, which is something that is not widely done in this domain of study because more widely used models are not open source.

We hypothesize that Qwen will perform substantially better than LLama in mathematics and coding because of its more analytical training schema compared to LLama. We also hypothesize

that Llama will perform better than Qwen at logical reasoning and fact checking because of its larger training set size.

Literature Review

The Transformer

After the release of the Transformer architecture in 2017, it became widely adopted across almost all LLMs for its ability to be able to handle long sequences of text and capture the relationships between words.

The Transformer architecture is composed of an encoder-decoder structure that has an autoregressive decoder. The encoder is composed of multiple identical layers, each with two sublayers: multi-head self attention and a position-wise fully connected feed-forward network. The decoder has a similar structure, but with an additional sub-layer for attending to the encoder's outputs. Both the encoder and decoder include residual connections and layer normalization to stabilize training. Positional encodings are added to input embeddings to retain the relationships between order of the words [3].

The Transformer based LLMs have been revealed to be predisposed to problems with hallucinations [4]. This is due to them being fundamentally working in log-space, so their self attention mechanism is not effective beyond a certain problem complexity. It is not able to consistently perform multi-step logic because the self attention is not forming the embedded relationships with parts of the generated response. Rather than outputting random errors, Transformers generate fabrications, as a result of the architecture's bounded area of reasoning.

Large Language Models (LLMs)

Large language models (LLMs) are neural networks that are trained on large amounts of text in order to model their generation like human language. Most currently used LLMs incorporate the Transformer architecture. Many of them however, have a decoder-only architecture that was introduced in GPT-2 [5] and GPT-3 [6]. These models are trained using casual language modelling. This means that during the training, it is adjusted based on the objective to predict the next token in a sequence based only on prior tokens. This decoder-only architecture can limit the model's ability to revise or verify its outputs since it does not reconsider the prior generations. Some of these decoder-only architectures such as LLaMA [7] and Mistral [8], have become standard in open source LLMs because of their simplicity and scalability. However, because they generate responses token-by-token without bidirectional context or separate encoder input, their ability to perform tasks like answer verification or correction relies heavily on prompting strategies or fine-tuning rather than architectural design [9].

Verification and Truthfulness in Language Models

LLMs are known to generate correct sounding answers that are fluent sounding, but are factually incorrect. This has sparked an interest in researching the ability of these LLMs to verify the correctness of their outputs. There have been several attempts at creating methods for self-evaluation which is where the

model is asked to judge whether its own or another model's response is correct. One approach uses iterative refinement where the model generates a response and then critiques it. It then uses its own feedback to generate another revised version [10]. There is also a method that uses a structured chain of verification, which prompts the model to first reason through whether an answer before finalizing it [11]. These strategies have shown some improvements in the accuracy of the LLMs, especially in reasoning tasks. Evaluation tests such as TruthfulQA [12] reveal that models struggle with distinguishing fact from real sounding misinformation. There has also been improvement through instruction fine training which is when the models are explicitly trained to verify and correct answers by using examples [13].

Answer Correction and Self-Improvement Techniques

Several prompting frameworks have been developed that encourage iterative reasoning and revision. One of these methods is called the Chain-of -Verification (CoVe), where the model generates an answer, then plans verification questions to fact check its original answer, answers those questions independently, and then generates its final verified response [14]. This process improves factual accuracy while not needing model fine tuning. Another method called Self-Refine, generates an initial output using an LLM, then the same LLM provides feedback for its output and uses it to refine itself [10]. Self Refine does not use any supervised training, additional training, or reinforcement learning. This makes the technique entirely contained within the original LLM. The success of these methods show how answer correction can be done through prompting techniques alone, instead of relying solely on model architecture.

Open-Source vs. Closed-Source LLM Capabilities

There is a noticeable gap between closed source LLMs like GPT-4 and Claude and the open source counterparts such as LLaMA, Mistral and Falcon. This becomes apparent in tasks involving answer verification and correction. Closed models have shown strong self-evaluation capabilities due to fine tuning and reinforcement learning methods that are not made publicly available. Contrastingly, open source models rely on base pretraining without any explicit instruction for self-correction or verification [7]. There are also very few studies that measure performance of open source models with these tasks.

Methodology

Overview

This project will be exploring and analyzing the ability of LLMs to verify and correct their own and other LLMs answers. Specifically, this project will explore the abilities of open source LLM’s, which is significant because of their prevalence in academia, as well as the ability to do analysis on its performance relative to the model architecture.

The project methodology will consist of the following components: dataset creation, model selection, prompt design, task execution, and evaluation.

Dataset Construction

The initial prompt dataset will consist of 100 handcrafted question and answer pairs. These prompts will range in topic and complexity. The topics include: logical reasoning, mathematics, factual knowledge, and coding.

The dataset will consist of Q-A pairs that contain incorrect answers as well, to test the LLM's ability to detect an incorrect correction. For each incorrect answer Q-A pair, there will be a correct answer in the dataset as well for reference.

Model Selection

We will select open source LLMs to test based on their current relevancy.

The specific models that will be used are Llama 3, Qwen 2.5, and Qwen 3.

The hardware being used to host the LLM and perform the inference will be the NYU HPC with A-100 GPUS.

Prompt Design

When inputting the Q-A pair to the LLM, the way that it is inputted into the LLM will be consistent.

For a pair $P_k \in \{p_1, p_2, \dots, p_{98}, p_{99}, p_{100},\}$ where each $P_k \in \{Q_k, A_k\}$, the initial verification prompt will be:

"Given the following Question: Q_k and Answer: A_k, is the answer correct? Respond 'yes' or 'no.'"

The correction prompt will be:

"If the answer is incorrect, please provide a correct one."

Evaluation Procedure

For each Q-A pair evaluation, the procedure will be as follows:

1. Feed Q-A pairs to the specified LLM using the verification prompt. Record the response
2. If the response was "no", prompt the LLM with the correction prompt. Record the response.
3. Repeat for each LLM across the dataset.

Performance Evaluation

We will be verifying the results based on several criteria:

- Verification Accuracy: Percentage of correct “yes/no” judgements
- Correction Quality: Human evaluation of the quality of the correction
- Domain-Specific Performance: Analyze results by domain (e.g., factual, math)
- Error Typology: Categorize errors (e.g., hallucination, chain-of-thought failure)

Additionally, the differences in performance between the Llama and Qwen model will be attributed to their difference in the original training scheme and training data. These models have slightly different model architectures, but the main difference lies in the training scheme/data.

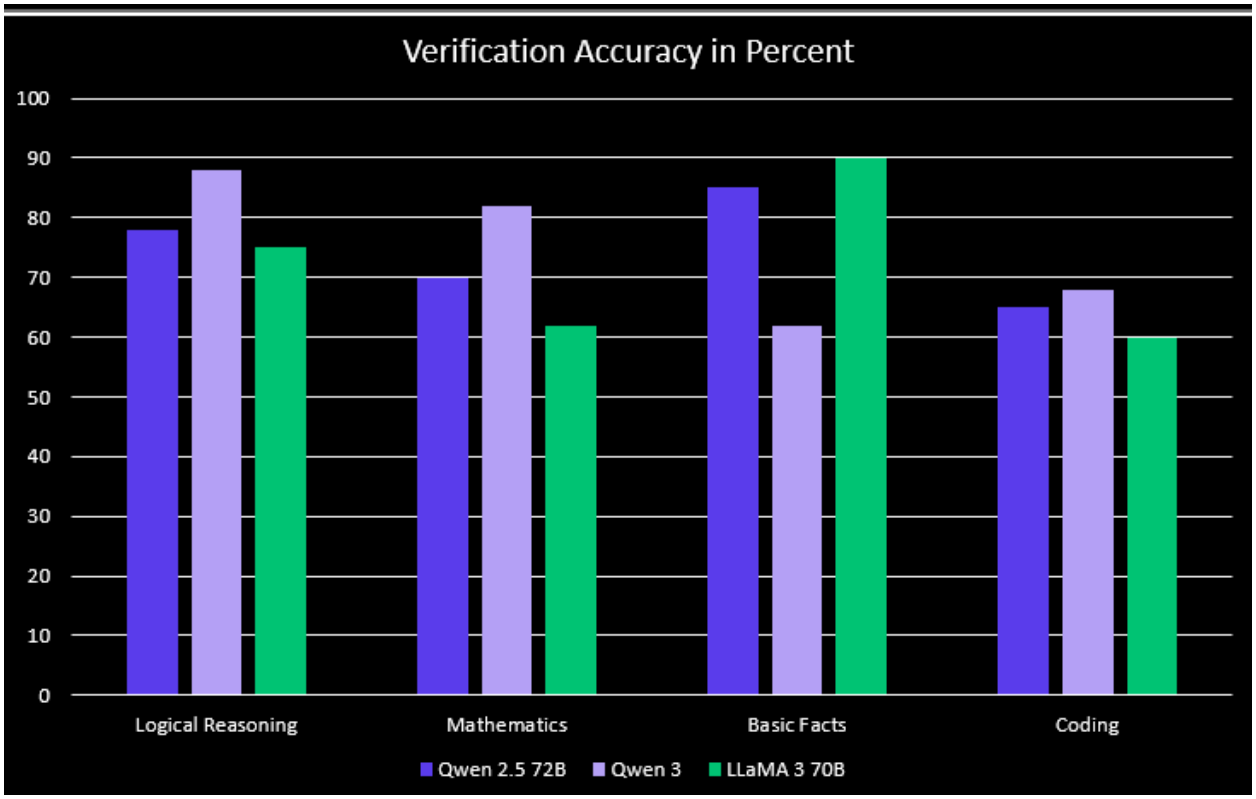
Limitations

There are limitations that arise from the methodology which include:

- The small dataset of the Q-A pairs, could lead to large discrepancies when analyzing the probabilities of the performance of certain types of Q-A pairs. It also leads to less diversity in the Q-A pairs being asked in terms of content, which could lead to the misrepresentation of model performance.
- The binary nature of the verification prompt answer could lead to larger amounts of answers being classified as wrong. By requiring a binary answer from the LLM, it removes any nuance possible within the LLM generated response.
- Manual crafting of the Q-A pairs could introduce bias.

Results

Verification Testing



Graph 1

	Logical Reasoning	Mathematics	Basic Facts	Coding
Qwen 2.5	78%	70%	85%	65%
Qwen 3	88%	82%	62%	75%
Llama 3	75%	62%	90%	60%

Table 1

Correction Testing

	Logical	Mathematics	Basic Facts	Coding

	Reasoning			
Qwen 2.5	68%	65%	75%	62%
Qwen 3	85%	80%	78%	72%
Llama 3	75%	62%	90%	60%

Table 2

Qwen 3:

- Excels in logical reasoning, math, and code correction.
- Recomputes solutions rather than pattern-matching.
- Detects deep logical flaws and multi-step calculation errors.
- Handles semantic code bugs (loops, indexing, recursion).

Qwen 2.5:

- Balanced performance across domains, but not top in any.
- Good at obvious factual or logic errors; misses subtle issues.
- Limited in complex math proofs or chained reasoning.
- Can fix syntax/code formatting but struggles with algorithmic bugs.

Llama 3:

- Best for factual corrections (entities, dates, encyclopedic facts).
- High recall of general knowledge; reliable fact-checker.
- Weak in recomputing math or tracing reasoning chains.
- Code corrections, mostly superficial, often repeat original errors.

Discussion

Qwen 3 Performance

Qwen 3 demonstrated the highest correction performance in logical reasoning, mathematics, and coding. Its ability to recompute answers from first principles allowed it to identify multi-step reasoning flaws and numerical inaccuracies more effectively than the other models. In coding, Qwen 3's semantic understanding of program structure enabled it to trace through algorithms, detect logic errors such as off-by-one indexing, and produce corrected solutions that executed as intended. While its factual correction accuracy (~78%) lagged behind LLaMA 3, it still showed competency when explicitly prompted to fact-check. These results indicate that Qwen 3 is best

suited for reasoning-heavy, computational, and algorithmic correction tasks in iterative refinement pipelines.

Qwen 2.5 Performance

Qwen 2.5 offered balanced but middle-tier performance across all domains. In logical reasoning and mathematics, it performed solidly but struggled with deeply nested logical flaws and advanced symbolic manipulations. Its factual correction accuracy (~75%) was respectable, and it could catch straightforward errors in names, dates, and basic details. In coding, Qwen 2.5 could identify and fix syntax errors, as well as some simple semantic issues, but it often failed to resolve more complex algorithmic mistakes. Overall, Qwen 2.5's well-rounded skill set makes it a practical secondary evaluator in multi-domain iterative refinement loops, particularly when computational resources are limited.

Llama 3 Performance

LLaMA 3's standout strength was in factual correction, where it achieved the highest accuracy (~88%), especially in rectifying named entities, historical facts, and general encyclopedic information. This advantage likely reflects a pretraining corpus rich in factual content. However, LLaMA 3 underperformed in reasoning, mathematics, and coding corrections, often failing to detect deep logical inconsistencies or to recompute solutions from scratch. In coding, it tended to produce minor variations of the original incorrect output rather than a fundamentally corrected solution. These characteristics make LLaMA 3 an excellent choice for fact-focused verification and correction, but a weaker candidate for domains requiring high reasoning depth or computational precision.

Conclusion

This study evaluated the verification and correction capabilities of three open-source LLMs – LLaMA 3, Qwen 2.5, and Qwen 3 – across logical reasoning, mathematics, factual knowledge, and coding. Qwen 3 consistently outperformed in reasoning, math, and code-related tasks, while LLaMA 3 excelled in factual verification and correction. Qwen 2.5 provided balanced but mid-tier performance across all domains. These findings suggest that hybrid iterative refinement pipelines leveraging Qwen 3 for reasoning-intensive tasks and LLaMA 3 for factual checks can maximize correction accuracy.

In the future, doing a substantial amount more testing across several other domains and prompting structures, would be helpful to build a more robust pipeline. Creating prompts that allow for automation within the feedback loop process is also necessary for the implementation of the pipeline.

Acknowledgements

I would like to thank Professor Mengye Ren and Jack Lu for their mentorship within this project. Additionally, I would like to thank Catherine Tissot, Matthew Leingang, and Sophia Ugaz for organizing NYU GSTEM. I thank Katherine Leung for guiding me throughout the research process. Lastly, thank you to the Winston Foundation, whose scholarship made this experience accessible to me.

References

1. Zeng, A., Liu, X., Wang, H., Wang, X., Liu, Z., & Tang, J. (2023). *Harnessing the power of LLMs in practice: A survey on ChatGPT and beyond*. arXiv preprint arXiv:2304.13712. https://arxiv.org/abs/2304.13712
2. Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., & Zhang, Y. (2024). *On the self-verification limitations of large language models on reasoning and planning tasks*. arXiv preprint arXiv:2404.07143. https://arxiv.org/abs/2404.07143
3. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., & Polosukhin, I. (2017). *Attention is all you need*. arXiv preprint arXiv:1706.03762. https://arxiv.org/abs/1706.03762
4. Xu, Y., Ge, L., Xia, Q., Liang, C., Wang, X., & Liu, Z. (2024). *Hallucination stations: On some basic limitations of transformer-based language models*. arXiv preprint arXiv:2507.07505. https://arxiv.org/abs/2507.07505
5. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). *Language models are unsupervised multitask learners*. OpenAI. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf
6. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P.,& Amodei, D. (2020). *Language models are few-shot learners*. arXiv preprint arXiv:2005.14165. https://arxiv.org/abs/2005.14165
7. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M. A., Lacroix, T., & Scialom, T. (2023). *LLaMA: Open and efficient foundation language models*. arXiv preprint arXiv:2302.13971. https://arxiv.org/abs/2302.13971
8. Jiang, Z., Xu, Y., Mao, Y., & Zhu, C. (2023). *Mistral 7B*. arXiv preprint arXiv:2310.06825. https://arxiv.org/abs/2310.06825
9. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., & Le, Q. V. (2022). *Chain-of-thought prompting elicits reasoning in large language models*. arXiv preprint arXiv:2201.11903. https://arxiv.org/abs/2201.11903

10. Madaan, A., Lin, Z., Liu, M., Subramani, N., Yu, T., Qian, P.,& Radev, D. R. (2023). *Self-Refine: Iterative refinement with self-feedback*. arXiv preprint arXiv:2303.17651. https://arxiv.org/abs/2303.17651
11. Chen, D., Chen, J., Li, M., Chen, W., Yu, X., Lin, X., & Ma, W. (2023). *Language models as chain-of-verification reasoners*. arXiv preprint arXiv:2305.08348. https://arxiv.org/abs/2305.08348
12. Lin, S., Hilton, J., & Evans, O. (2022). *TruthfulQA: Measuring how models mimic human falsehoods*. arXiv preprint arXiv:2109.07958. https://arxiv.org/abs/2109.07958
13. Wu, C.-S., Bai, Y., Han, X., Yu, P., Gu, S., & Zhang, Y. (2024). *Large language models can self-correct with key condition verification*. arXiv preprint arXiv:2405.14092. https://arxiv.org/abs/2405.14092
14. Lin, Z., Liu, M., Radev, D., et al. (2023). *Chain-of-verification reduces hallucination in large language models*. arXiv preprint arXiv:2309.11495. https://arxiv.org/abs/2309.11495
15. OpenAI. (2023). *GPT-4 technical report*. arXiv preprint arXiv:2303.08774. https://arxiv.org/abs/2303.08774