Development of LLM and RAG

Yixin Hu

y.hu5@student.vu.nl

University of Amsterdam

Abstract

This study

Contents

1 Introduction

Large Language Models (LLMs) have revolutionized various applications across fields from academic achievements to industry scenarios. LLMs enable machines to understand, generate, and manipulate human language with unprecedented accuracy and fluency. The evolution of LLMs has been marked by significant milestones, from early statistical models to advanced neural networks like Transformers. Pure parameterized language models like LLMs store world knowledge from a large corpus in the parameters of the model, representing the model's understanding and generalization of the training data. However, LLMs still face several difficulties, such as factual hallucinations [1] and the lack of domain-specific expertise [2].

To address the limitation, language models can adopt semi parametric methods by integrating non parametric corpus databases with parametric models. Retrieval Augmented Generation (RAG) [3] empowers LLM models by incorporating external knowledge to enhance the relevance of the results and the availability in industrial scenarios. Especially for industries that need domain-specific expertise and to keep data confidential, embedding relevant data sources is crucial to let LLMs generate more accurate and reliable responses.

The reminder of this study is orgainized as follows: Section 2 introduces the background of LLM and RAG and other related techniques. Section 3 introduces current works of RAG. Section 3 we draw a brief conclusion by summarizing the main contributions. Finally, we discuss the remaining problems for future work.

2 Background

2.1 Large Language Models (LLMs)

LLMs are large-scale, pre-trained, language model based on artificial intelligence technology. With language modeling, machines can understand and communicate in human language. This research has been gained wide attention and can be regarded as 4 main phrases as in Figure 1. [4] We set the time mostly based on the publication date of the most representative research at each step, therefore the time period for each stage may not be entirely correct.

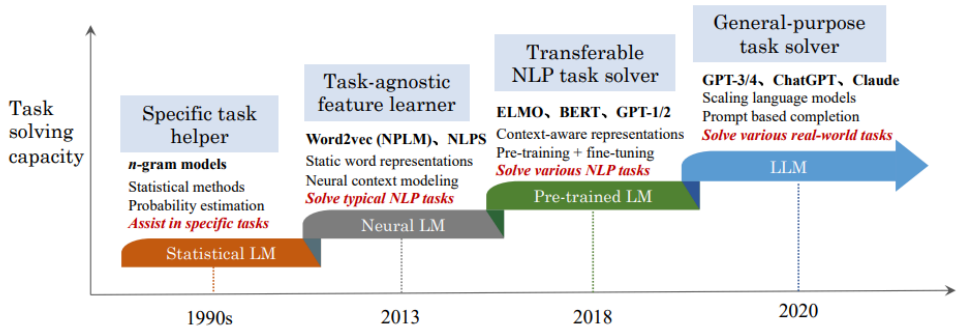

Figure 1: Language models (LM) path of evolution

- *Statistical language models (SLMs).* Originating in the 1990s, SLMs are primarily built on the principles of statistical. The dominating idea is based on Markov chain models, *e.g.*, predicting subsequent words based on recent contextual history. The SLMs with a fixed context length n are typically known as n-gram models [5]. It predicts next word based on the likelihood of occurrence of a sequence of words, and estimate the probability of text as the product of their word probabilities. SLMs have been widely used in many information retrieval (IR) and natural language processing (NLP) tasks. However, natural language is always sparse so n-gram models can not efficiently capture the pattern in diverse and various texts. N-gram models also face curse of dimensionality where estimation of high-order models becomes computationally infeasible due to the exponential growth in the number of possible word combinations.
- *Neural language models (NLM).* NLMs represent a more advanced phase in the development of language models, where neural networks such as multilayer perceptron (MLP) and recurrent neural networks (RNNs) [6] are employed. For the shortcomings in n-gram models, the work in [7] solve the curse of dimensionality by introducing distributed representation of words (commonly referred to as word embeddings). This technique fundamentally changed the way context is processed by aggregating features into distributed word vectors. Building on the idea of distributed representations, word2vec framework [8] was introduced specifically designed to efficiently compute word embeddings using neural networks. These researches first introduced

the use of language models for representation learning (beyond word sequence modeling) and had a huge impact for NLP field.

- *Pre-trained language models (PLM).* Pre-trained Language Models (PLM) have set a new benchmark in language processing with the advent of ELMo [9]. Different from word2vec where one word corresponds to one fixed vector, ELMo utilizes a bidirectional LSTM network to generate dynamic contextualized word representations. Furthermore, BERT was invented [10] with transformer [11]architecture, marking a significant evolution by enabling bidirectional training on extensive unlabeled datasets. These models serve as general-purpose semantic features that significantly boost the performance of NLP tasks. Following this study, virous of models like GPT-2 [12] and BART [13] emerged, establishing a robust "pre-training and fine-tuning" paradigm widely embraced in the field. It is often necessary to fine-tune the PLM to accommodate different downstream tasks.
- *Large language models (LLM).* In PLMs, scaling model size or data size usually improve model ability on downstream tasks following scaling law [14]. Even though just increasing model and data sizes and remaining similar structure, these models achieve unprecedented processing capabilities. They exhibit unique emergent abilities that surpass their predecessors in handling complex tasks, including effective few-shot learning. Consequently, the academic circles decided to name these large PLMs as "large language models". LLMs now specifically refers to these extensive models [15], [16], which continue to attract substantial research interest. Notably, applications like ChatGPT which showcases remarkable conversational skills with humans, highlighting the practical potential of LLMs. This surge in research and development reflects in the increasing volume of related academic publications post the release of such models.

In summary, Large Language Models (LLMs) derive from years of language modeling research than a brand new concept. Now LLMs show significant problem solving ability. These days, the AI community is being greatly impacted by LLMs. The problem solving ability and development of LLMs is revolutionizing

the fields of AI research. To a certain extent, LLMs can be used as general-purpose language task solvers in NLP, and the research paradigm is moving in favor of LLM utilization.

2.2 Retrieval Augmented Generation (RAG)

However, despite the success of LLMs, they still face challenges. The issues are, e.g., LLMs knowledge is usually out of date [17], sometimes generate hallucination content [1], also the lack of domain-specific expertise [2].

To address these limitations, The concept of Retrieval-Augmented Generation (RAG) [3] was proposed in 2020 and combines a pre-trained retriever for the first time with a pre-trained seq2seq model with end-to-end fine-tuning. It was after NLP moved into the era of LLMs in 2022 that RAG really emerged as a compelling solution, especially after ChatGPT was proposed the as shown in Figure 2 [18]. RAG effectively combines the parametric knowledge of LLMs with non parametric external knowledge bases, making it one of the most important methods for implementing large-scale language models.

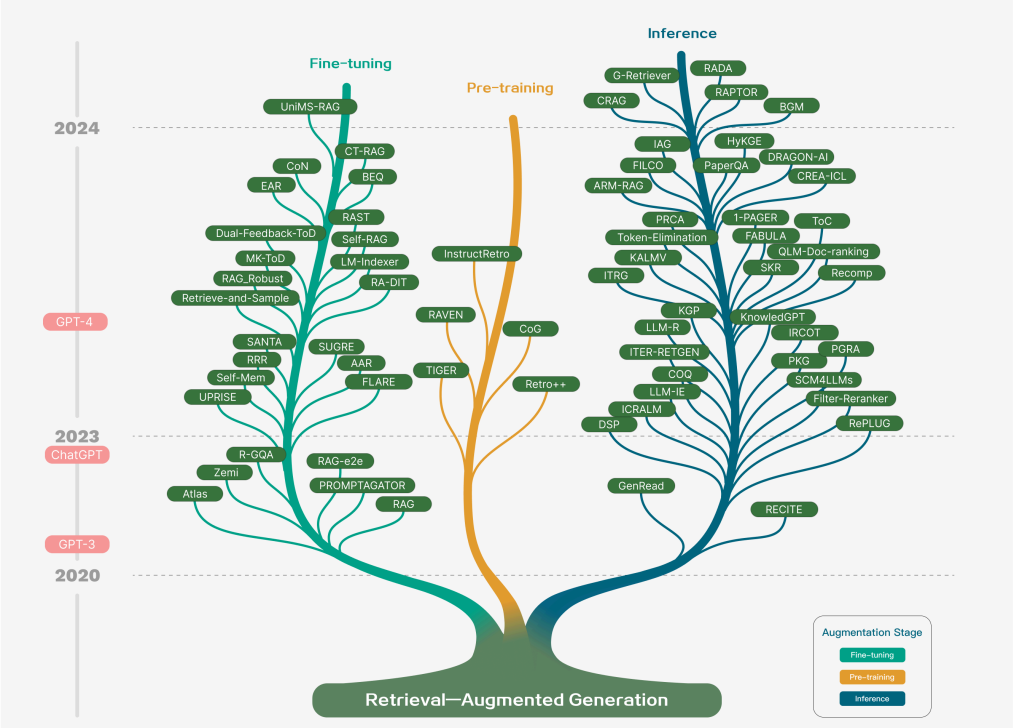

Figure 2: Technology tree of RAG research.

In common definition, RAG has two main phases: retrieval and generation. The retriever utilizes an embedding model to retrieve relevant documents based on

the problem. The retriever is the core component of the RAG framework, responsible for retrieving relevant information from an extensive knowledge base. The retriever analyzes the user's input query and retrieves the most relevant paragraphs. The generator then uses these retrieved contexts to generate answers based on the LLM. It is responsible for transforming the search results into natural and fluent text. Its input includes not only traditional contextual information, but also relevant text snippets obtained by the retriever. RAG method avoid retraining the entire large model for each specific task. On the contrary, they can attach a knowledge base to provide additional input information to the model and improve its response accuracy.

Because of the high cost of training high-performance large models, academia and industry have attempted to enhance model generation by incorporating RAG modules in the inference phase to integrate external knowledge in a more cost-effective way. RAG provides a more efficient solution for complex knowledge-intensive tasks in large models by optimizing key parts such as retrievers and generators. RAG improves the relevancy of the answers while decreasing the rate of mistakes in LLMs.

The scope of RAG search is also gradually expanding. Early RAG focused on open source, unstructured knowledge, e.g., Wikipedia. As the scope of search expands, structured, high-quality data can also be used as a knowledge source. Besides RAG development timeline, in Figure 2, the tree illustrates two more aspects of RAG development:

- Augmentation data: Data sources include unstructured data, structured data and LLM-generated content.
- Augmentation stage: Retrieval-Augmented Generation (RAG) can enhance performance at three stages: pre-training, fine-tuning, and inference, represented as three branches in the tree.

Pre-training involves initially training a model on a large-scale generalized dataset to learn a wide range of linguistic patterns and knowledge. The inception of RAG coincided with the rise of the Transformer architecture, aiming to combine broader knowledge with pre-training models for more robust representa-

tions. Inference occurs when a trained model generates predictions based on new input data. With the advent of LLMs, RAG research has focused on exploiting the powerful in-context learning (ICL) capabilities of LLMs to tackle knowledge-intensive tasks. By providing better information, RAG enhances LLMs' abilities to handle more complex tasks during inference. Fine-tuning is the further training of a pre-trained model for a specific task or domain to optimize its performance in that particular application. As LLMs have developed and found widespread use, domain-specific retrieval has improved accuracy and relevance by tailoring information to specific tasks.

Overall, RAG ensures that the generated content is not only contextually accurate but also deeply aligned with specialized knowledge, making it particularly suitable for industry scenario.

2.3 Prompt Engineering

Prompt engineering has been an important method for improving LLM performance. This method makes clear, task-specific clue for computers in natural language, and representative examples are chosen with care to be included in the prompt. Without changing the parameters, output of LLMs can improved by strategically designed prompts, enabling them to excel across diverse tasks and domains. It is important because of its ability to steer responses. Therefor LLMs can be more flexible and applicable in a wider range of industries. Next, we will provide a brief overview of prompt engineering techniques, spanning from basic to some latest advanced.

- Zero-Shot Prompting

No need for training on massive amount of data, LLMs can perform some zero-shot tasks [19]. In the prompt, there is no additional examples to guide the model. The model will only leverage its own knowledge to produce output based on the prompt.

- Few-Shot Prompting

Few-shot prompting enhances model understanding by providing a limited number of examples [20]. This technique is particularly effective for complex tasks, where even a small number of high-quality examples can significantly improve

performance. However, longer inputs lead to increased token consumption, and the impact of prompt selection and composition on the results is still not fully understood.

- Chain-of-Thought (CoT) Prompting

The two techniques mentioned above belong to In-Context Learning (ICL). The core idea of ICL is to learn from analogies, mimicking the human learning process. Initially, ICL is given some examples that form the context of the task. These examples are written in natural language templates. Then, ICL connects the query question (i.e., the input) with a contextual presentation (some relevant cases) to form the input with hints, and feeds it into the language model for prediction. However, this method has limitation such as unstable. To compensate these shortcomings, Chain-of-Thought (CoT) add thoughts in the middle, in each steps [21]. Unlike traditional ICL, which provides more input to generate output, CoT involves predicting the "thought process" (referred to as rationale in academic fields) along with the answer. These thought processes are used as hints to get better answers and do not need to be shown for actual use. Instead of rigidly providing sample questions and answers, intermediate reasoning sessions are given so that the model learns the logic of reasoning and thinking in the intermediate process. This method not only mimics human leaning process as learning from examples but more closely to the core of human intelligence, human cognitive processes.

However, high-quality examples which needed in CoT usually involves manual efforts. These could lead to suboptimal results. To mitigate this limitation, Auto-CoT [22] was introduced. As the name suggests, Auto-CoT automatically generates rationales by instructing LLMs with a "Let's think step-by-step" prompt. This automatic process may contain errors, so it is important to build diverse demonstrations. First, Auto-CoT divides questions into clusters. Then it selects a representative question from each cluster to generate its reasoning chain. The process is illustrated in Figure 3.

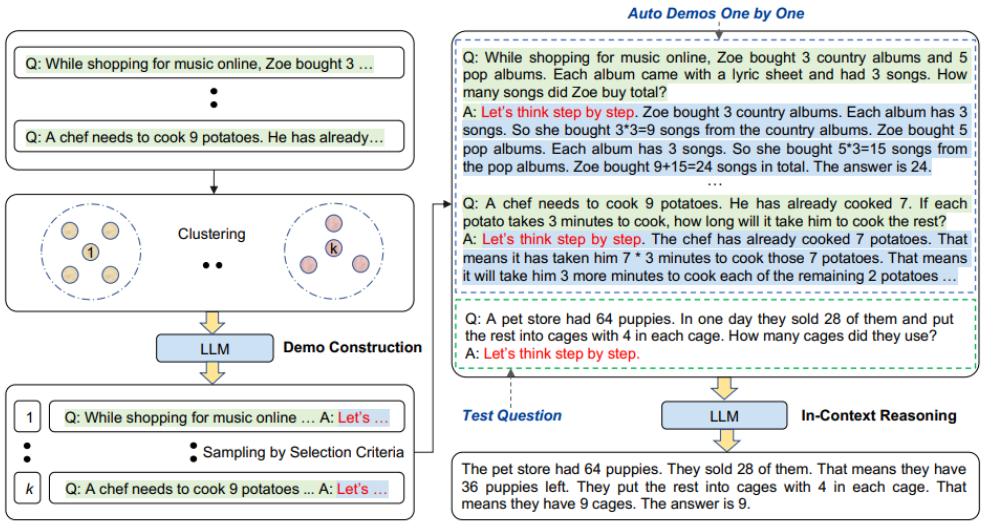

Figure 3: Overview of the Auto-CoT method

Auto-CoT has even better results than manual methods. One interpretation is that Auto-CoT does not apply a fixed template. Instead, each task generates its own set of examples. This is because the problem sets differ, leading to more various results.

- Self-Consistency

One of the more advanced techniques than CoT prompting is self-consistency [23]. First it uses a set of huamn set CoT examples as prompt as LLMs input. Then it samples a set of candidate outputs from the LLM to generate a set of different candidate inference paths. Eventually it selects the most consistent answer among multiple reasoning paths. By aggregating multiple responses to the same prompt, self-consistency ensures that the final answer to an input represents a consensus vote, which tends to be more reliable and accurate than simple CoT completions on their own. Even when regular CoT is ineffective, self-consistency still be able to improve results.

3 Current Work

This chapter is going to introduce three RAG frameworks and current study about RAG. The research paradigm of RAG has been evolving and can be summarized into three categories: Naive RAG, Advanced RAG, and Modular RAG. The emergence of Advanced RAG and Modular RAG is aimed at addressing the shortcomings of Naive RAG in use.

3.1 Framework

[18]

- Naive RAG

Naive RAG follows a traditional process that includes indexing, retrieval, and generation.

The indexing process is a crucial initial step in data preparation, which takes place offline and involves multiple stages. It starts with data indexing, where raw data is cleaned and extracted, and various file formats such as PDF, HTML, Word, and Markdown are converted into standardized plain text. In order to adapt to the contextual limitations of the language model, these texts are subsequently segmented into smaller, more manageable blocks, a process known as chunking. These blocks are then transformed into vector representations through an embedded model, which is chosen for its balance between inference efficiency and model size. This helps to compare similarities during the retrieval phase. Finally, create an index to store these text blocks and their vector embeddings as key value pairs, which allows for efficient and scalable search capabilities.

In retrieve process, after receiving the user query, the system uses the same encoding model as the indexing stage to convert the input into a vector representation. Then calculate the similarity score between the query vector and the vectorized blocks in the index corpus. The system prioritizes retrieving the top K blocks that are most similar to the query. These blocks are then used as the contextual basis for extensions to address user requests. The generated query and selected document are rendered as a prompt, and a large language model generates the answer.

The response method of the model may vary depending on task specific criteria, allowing it to either utilize its inherent parameter knowledge or limit its response to the information contained within the provided document. In the case of continuous dialogue, any existing dialogue history can be integrated into the prompt, enabling the model to effectively participate in multiple rounds of dialogue interaction.

Naive RAG faces significant challenges in three key areas: “retrieval,” “generation,” and “enhancement. The retrieval quality presents various challenges, including low accuracy, resulting in inconsistent blocks retrieved with the query, and possible issues such as hallucinations or airborne objects falling. Low recall rates can also occur, resulting in the inability to retrieve all relevant blocks, thereby hindering the ability of LLMs to build comprehensive responses. Outdated information further exacerbates the problem and may lead to inaccurate search results.

-Advanced RAG

Advanced RAG was developed to address the shortcomings of Naive RAG, implementing pre retrieval and post retrieval strategies.

Pre Retrieval Process: Optimizing Data Indexing: The goal of optimizing data indexing is to improve the quality of the indexed content. This involves five main strategies: enhancing data granularity, optimizing index structure, adding metadata, alignment optimization, and hybrid retrieval. Enhancing data granularity aims to improve the standardization, consistency, factual accuracy, and rich context of text, in order to enhance the performance of RAG systems. This includes removing irrelevant information, eliminating ambiguity between entities and terms, confirming factual accuracy, maintaining context, and updating outdated documents. Optimizing the index structure involves adjusting the size of blocks to capture relevant context, querying across multiple index paths, and utilizing node relationships in the graph structure to capture relevant context. Adding metadata information involves integrating referenced metadata (such as date and purpose) into blocks for filtering, and integrating reference metadata for chapters and subsections to improve retrieval efficiency. Alignment optimization solves alignment issues and differences between documents by introducing “hypothetical questions” in the documents. Hybrid retrieval refers to the fusion of retrieval through multiple recalls, such as the sparse retrieval method of BM25 and the dense retrieval method of deep learning models.

Post Retrieval Process: After retrieving valuable context from the database, the key is to merge it with the query as input to LLMs, while addressing the chal-

lenges posed by contextual window limitations. Simply presenting all relevant documents to LLM at once may exceed the contextual window limit, introduce noise, and hinder attention to key information. Additional processing is required on the retrieved content to address these issues. Rerank ◦ Re ranking the retrieved information and repositioning the most relevant content to the edge of the prompt is a key strategy. This concept has been implemented in frameworks such as LlamaIndex, LangChain, and HayStack. For example, diversity rerank prioritizes reordering based on document diversity, while LostInTTheMiddleRanker alternates between placing the best document at the beginning and end of the context window. In addition, methods such as coherenAI rerank, bge rerank, and LongLLMLingua recalculate the semantic similarity between relevant texts and queries, solving the challenge of interpreting semantic similarity in vector based simulation searches.

- Modular RAG

The modular RAG structure is different from the traditional plain RAG framework, providing greater flexibility and adaptability. The modular RAG paradigm is becoming increasingly common in the RAG field, allowing for serialized pipeline or end-to-end training through multiple modules.

The organizational structure of modular RAG is highly adaptable, allowing for the replacement or rearrangement of modules during the RAG process to adapt to specific problem contexts. For example, adding or replacing modules, adjusting the flow between modules and optimizing RAG assembly line

The optimization of the retrieval process aims to improve the efficiency and quality of information in the RAG system. The current research focuses on integrating diverse search techniques, refining retrieval steps, incorporating cognitive backtracking, implementing multifunctional query strategies, and utilizing embedded similarity. These efforts collectively aim to achieve a balance between retrieval efficiency and contextual information depth in RAG systems.

Assuming document embedding. HyDE is based on the belief that generated answers may be closer in the embedding space than direct queries. Use LLM and HyDE to create a hypothetical document (answer) for the query, embed

this document, and use result embedding to retrieve real documents that are similar to the hypothetical document. This method is not based on searching for embedding similarity through queries, but focuses on embedding similarity from one answer to another. However, it may not always produce ideal results, especially when the language model is unfamiliar with the topic, which may lead to more erroneous instances.

4 Future Work

Despite RAG technology has achieved significant progress, there are still some challenges that require further research.

- Context length

The effectiveness of RAG is limited by the contextual window size of large language models (LLMs). A balance window that is too short may lead to insufficient information, while a window that is too long may lead to information dilution. The trade-off is crucial. With continuous efforts to expand the LLM context window to almost infinite sizes, research on RAG adaptation to these changes has raised important questions [Xu et al., 2023c, Packer et al., 2023, Xiao et al., 2023].

- Robustness

The presence of noise or contradictory information during the retrieval process may have a negative impact on the output quality of RAG. This situation is vividly referred to as' misinformation may be worse than no information '. Improving the resistance of RAG to such adversarial or false inputs is gaining research momentum and has become a key performance indicator [Yu et al., 2023a, Glass et al., 2021, Baek et al., 2023].

- Actual use of RAG

The practicality and consistency with engineering requirements of RAG have promoted its adoption. However, improving retrieval efficiency, enhancing document recall in large knowledge bases, and ensuring data security - such as preventing LLMs from inadvertently disclosing document sources or metadata - are key engineering challenges that need to be addressed [Alon et al., 2022].

5 Conclusion

Our analysis divides the RAG framework into three developmental paradigms: primary, advanced, and modular RAG, each paradigm representing a gradual enhancement of its predecessor. The advanced RAG paradigm goes beyond basic methods by integrating complex architectural elements, including query rewriting, block reordering, and prompt summarization. These innovations have led to a more detailed and modular architecture, improving both the performance and interpretability of LLM. The integration of RAG with other AI methods such as fine-tuning and reinforcement learning has further expanded its capabilities. Although RAG technology has made significant progress, there are still many research opportunities to improve its robustness and ability to manage extended contexts.

Bibliography

[1] M. Cao, Y. Dong, J. Wu, and J. C. K. Cheung, "Factual Error Correction for Abstractive Summarization Models." 2021.

[2] X. Shen, Z. Chen, M. Backes, and Y. Zhang, "In ChatGPT We Trust? Measuring and Characterizing the Reliability of ChatGPT." 2023.

[3] P. Lewis *et al.*, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." 2021.

[4] W. X. Zhao *et al.*, "A Survey of Large Language Models." 2023.

[5] P. F. Brown, V. J. Della Pietra, P. V. Desouza, J. C. Lai, and R. L. Mercer, "Class-based n-gram models of natural language," *Computational linguistics*, vol. 18, no. 4, pp. 467–480, 1992.

[6] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model," 2010, pp. 1045–1048. doi: 10.21437/Interspeech.2010-343.

[7] Y. Bengio, R. Ducharme, and P. Vincent, "A Neural Probabilistic Language Model," 2000, pp. 932–938. doi: 10.1162/153244303322533223.

[8] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space." 2013.

[9] M. E. Peters *et al.*, "Deep contextualized word representations." 2018.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." 2019.

[11] A. Vaswani *et al.*, "Attention Is All You Need." 2023.

[12] A. Radford *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9–10, 2019.

[13] M. Lewis *et al.*, "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension." 2019.

[14] J. Kaplan *et al.*, "Scaling Laws for Neural Language Models." 2020.

[15] J. Hoffmann *et al.*, "Training Compute-Optimal Large Language Models." 2022.

[16] R. Taylor *et al.*, "Galactica: A Large Language Model for Science." 2022.

[17] H. He, H. Zhang, and D. Roth, "Rethinking with Retrieval: Faithful Large Language Model Inference." 2022.

[18] Y. Gao *et al.*, "Retrieval-Augmented Generation for Large Language Models: A Survey." 2024.

[19] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language Models are Unsupervised Multitask Learners," 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:160025533

[20] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha, "A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications." 2024.

[21] J. Wei *et al.*, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models." 2023.

[22] Z. Zhang, A. Zhang, M. Li, and A. Smola, "Automatic Chain of Thought Prompting in Large Language Models." 2022.

[23] X. Wang *et al.*, "Self-Consistency Improves Chain of Thought Reasoning in Language Models." 2023.