computers

Review

Exploring Data Analysis Methods in Generative Models: From Fine-Tuning to RAG Implementation

Bogdan Mihai Guțu and Nirvana Popescu *

Computer Science and Engineering Department, National University of Science and Technology Politehnica Bucharest, 060042 Bucharest, Romania; bogdan_mihai.gutu@stud.acs.upb.ro
* Correspondence: nirvana.popescu@upb.ro

Abstract: The exponential growth in data from technological advancements has created opportunities across fields like healthcare, finance, and social media, but sensitive data raise security and privacy challenges. Generative models offer solutions by modeling complex data and generating synthetic data, making them useful for the analysis of large private datasets. This article is a review of data analysis techniques based on generative models, with a focus on large language models (LLMs). It covers the strengths, limitations, and applications of methods like the fine-tuning of LLMs and retrieval-augmented generation (RAG). This study consolidates, analyzes, and interprets the findings from the literature to provide a coherent overview of the current research landscape on this topic, aiming to guide effective, privacy-conscious data analysis and exploring future improvements, especially for low-resource languages.

Keywords: generative models; fine-tune; RAG; embeddings; question–answer system

1. Introduction

In recent years, the proliferation of data has reached unprecedented levels, driven by advancements in technology and the digital transformation of various sectors. Large corpora of data, encompassing diverse domains such as healthcare, finance, and social media, present immense opportunities to derive insights and drive innovation. However, the private nature of many of these data poses significant challenges, particularly in ensuring data security and privacy while harnessing their full potential [1]. This necessitates the development and application of sophisticated data analysis techniques that can operate efficiently and effectively within these constraints. Generative models, especially when enhanced with retrieval-augmented generation (RAG), have emerged as powerful tools in the realm of data analysis, offering unique capabilities in modeling complex data distributions and generating synthetic data [1,2]. Unlike traditional analytical models, generative models do not merely focus on predictive accuracy but strive to understand and replicate the underlying data distribution. This intrinsic quality makes them particularly valuable in analyzing large, private data corpora, where data modeling can achieve remarkable results and the generation of synthetic data can facilitate research and development without compromising privacy [3].

This paper presents a detailed review of data analysis methodologies leveraging generative models, with the objective of synthesizing, analyzing, and interpreting the literature findings to provide a cohesive overview of the current research landscape. The primary aim is to address the research questions outlined in Section 2.1 by assessing the most effective approaches to analyzing large, private datasets through the application of large language models and retrieval-augmented generation (RAG) techniques. This review specifically investigates strategies for the optimization and fine-tuning of generative models concerning efficiency, accuracy, and applicability; methods for the retrieval of pertinent data across varied tasks; and the implementation of RAG as part of comprehensive data

check for updates

Citation: Guțu, B.M.; Popescu, N. Exploring Data Analysis Methods in Generative Models: From Fine-Tuning to RAG Implementation. *Computers* **2024**, *13*, 327. https://doi.org/10.3390/computers13120327

Academic Editors: Paolo Bellavista and Ananda Maiti

Received: 27 September 2024
Revised: 17 November 2024
Accepted: 2 December 2024
Published: 5 December 2024

analysis. Through an in-depth examination of these methodologies, this paper seeks to offer insights into the practical application of these techniques for the extraction of valuable information from private data collections.

The subsequent sections of this study are organized as follows. In Section 2, we delineate the research methodology employed in the systematic literature review (SLR). Section 3 elucidates the processes involved in the fine-tuning of a large language model, the techniques utilized to query relevant data within databases, and strategies for the implementation of retrieval-augmented generation. This section includes a comprehensive inventory of the methods, techniques, and strategies identified, accompanied by succinct descriptions. Additionally, it presents the metrics applicable for the evaluation of the performance of large language models and those employed in question-answering (QA) tasks. Section 4 is structured into two main parts. The first part presents the answers to the research questions derived from the collected data. The second part outlines potential future work in this field, informed by the findings of this study.

2. Research Methodology

This section outlines the systematic literature review (SLR) protocol employed to locate, collect, and evaluate the state-of-the-art techniques under study. The SLR process is divided into four phases: the formulation of research questions, the development of a research strategy, the establishment of article selection criteria, and the analysis of the research results.

2.1. Research Questions

The goal of the systematic literature review (SLR) is to address the research questions by identifying all relevant research findings from prior studies. The research questions are organized into five sub-questions:

1. Which techniques are used to fine-tune a large language model (LLM)?
2. Does retrieval-augmented generation enhance the performance of a large language model (LLM)?
3. Which methods are used to search for relevant texts in response to human queries?
4. What are the best methods and metrics for the evaluation of the results of a large language model (LLM) or a document search?
5. Are Llama models capable of performing text understanding and information extraction tasks for retrieval-augmented generation systems?

To gather relevant studies, we searched several databases, including Elsevier, IEEE Xplorer, and Google Scholar. Studies that focused primarily on addressing one or more of our research questions were included in our review.

2.2. Research Strategy

We conducted a systematic literature review, collecting numerous studies published over the past five years (2019–2024) and relevant to our research topic from databases such as Elsevier, IEEE Xplorer, and Google Scholar. Using combinations of keywords and terms like "Large Language Model" AND "fine-tune", "Retrieval-Augmented Generation" OR "RAG", "Text similarity" AND "Question Answering", "Text Embeddings multilingual model" OR "Text Embeddings Romanian model", and "LLama" AND "Retrieval-Augmented Generation", we identified approximately 3473 articles. The results of these queries can be found in Table 1.

To refine our collection, we removed duplicates across the databases and discarded articles with fewer than four pages, leaving us with 3148 articles. Further examination based on the titles and abstracts, and the application of new keywords, including "Retrieval-Augmented Generation", "Large Language Models", "Semantic Similarity", "Question Answering Systems", and "Evaluation Metrics", led to the exclusion of 3041 articles. This process resulted in 107 remaining articles, of which 81 were excluded after a thorough reading based on specific exclusion criteria focused on the use of large language models for the extraction of data or methods used to search relevant text for questions. In the end, 26 studies met our research criteria and were selected for in-depth analysis. The SLR technique allowed us to identify the most relevant articles from extensive databases, as shown in Figure 1. The final set of articles was not only the result of automated selection based on keyword combinations but also addressed our research questions, as discussed in Section 2.3.

Table 1. Number of results for each database search.

Query	Elsevier	Google Scholar	IEEE	Total
Large Language Model AND fine-tune	225	453	919	1597
Retrieval-Augmented Generation OR RAG	351	330	117	798
Text similarity AND Question Answering	57	173	386	616
Text Embeddings multilingual model OR text embeddings romanian model	33	183	109	325
LLama AND Retrieval-Augmented Generation	1	81	56	138
Total	667	1220	1587	3474

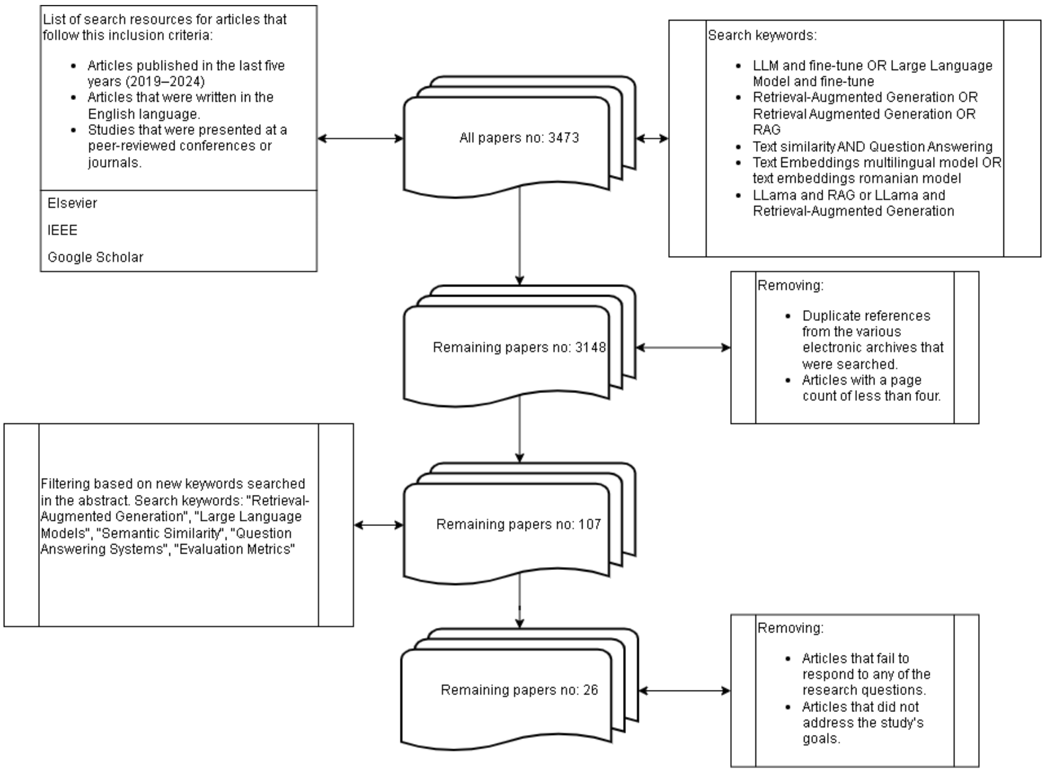
List of search resources for articles that follow this inclusion criteria:
Articles published in the last five years (2019–2024)
Articles that were written in the English language.
Studies that were presented at a peer-reviewed conferences or journals.
Elsevier
IEEE
Google Scholar
All papers no: 3473
Search keywords:
LLM and fine-tune OR Large Language Model and fine-tune
Retrieval-Augmented Generation OR Retrieval Augmented Generation OR RAG
Text similarity AND Question Answering
Text Embeddings multilingual model OR text embeddings romanian model
LLama and RAG or LLama and Retrieval-Augmented Generation
Remaining papers no: 3148
Removing:
Duplicate references from the various electronic archives that were searched.
Articles with a page count of less than four.
Filtering based on new keywords searched in the abstract. Search keywords: "Retrieval-Augmented Generation", "Large Language Models", "Semantic Similarity", "Question Answering Systems", "Evaluation Metrics"
Remaining papers no: 107
Remaining papers no: 26
Removing:
Articles that fail to respond to any of the research questions.
Articles that did not address the study's goals.

Figure 1. Diagram illustrating the article selection procedure.

2.3. Criteria for Article Selection

The following criteria were established for the selection of articles:

1. Articles utilizing the latest techniques to analyze text data;
2. Articles written in English;
3. Articles published within the last five years (2019–2024);
4. Studies presented in peer-reviewed conferences or journals.

After defining the inclusion criteria, the following exclusion criteria were determined:

1. Duplicate references from the various electronic archives searched;
2. Articles with fewer than four pages;
3. Articles that did not address any of the research questions;
4. Articles written in languages other than English;

5. Articles that did not meet the study's objectives.

2.4. Research Results

After reviewing the scientific databases and pinpointing the relevant research findings, the most pertinent articles aligning with the research objectives were identified. These articles primarily explored various techniques for text data analysis, with a particular emphasis on generative models and text searching in question-answering tasks. Each selected article was meticulously examined, and the extracted findings were analyzed and evaluated. This process aimed to summarize the current research landscape, highlight the most effective techniques, and suggest potential areas for future investigation.

3. Results of Systematic Review

This section presents the results derived from the background literature. It is organized into three parts: the first part addresses the methods of fine-tuning a large language model, the second part explores techniques used to search for relevant data for users' questions in databases, and the third part focuses on strategies for the implementation of retrieval-augmented generation. Each part aligns with the primary objectives of the systematic review.

3.1. Methods of Fine-Tuning a Large Language Model

A large language model is characterized by its extensive number of parameters, making it computationally challenging to pre-train or fine-tune the entire model to meet specific user requirements. Therefore, it is crucial to explore fine-tuning methods that are efficient in terms of time, computational power, and results.

The methods for the fine-tuning of models, as identified in the research articles reviewed in Section 2, are summarized in Table 2. This table provides the name of each method, the type of optimization that it represents, and a brief description.

For each identified method, we provide a detailed explanation, along with an analysis of their respective advantages and disadvantages.

LoRA, or Low-Rank Adaptation, introduced in reference [4] , is an efficient reparameterization technique that optimizes the memory and computation usage by introducing low-rank decompositions into large pre-trained weight matrices. As shown in Figure 2, LoRA operates by adding two smaller trainable matrices, $W_{\text{up}} \in \mathbb{R}^{d\times r}$ and $W_{\text{down}} \in \mathbb{R}^{r\times k}$, in parallel, to the original pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$. The rank r is chosen such that $r \ll \min(d,k)$, ensuring that the number of additional parameters introduced is minimal relative to the size of W_0.

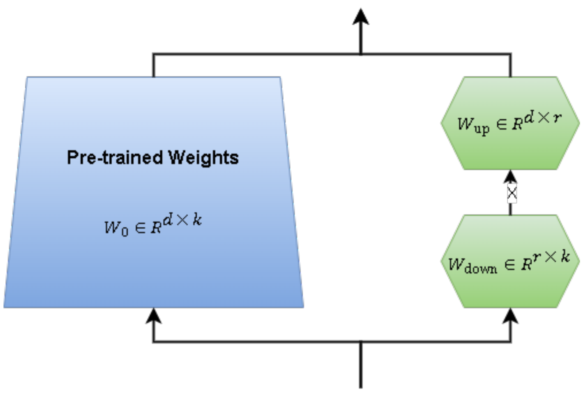
Pre-trained Weights
$W_0 \in R^{d\times k}$
$W_{\text{up}} \in R^{d\times r}$
$\times$
$W_{\text{down}} \in R^{r\times k}$

Figure 2. LoRA algorithm representation. Blue indicates frozen; green indicates trainable.

For a given input h_{in}, the output of the original weight matrix alone would be

$$h_{\text{out}} = W_0 h_{\text{in}} \tag{1}$$

However, LoRA modifies this by incorporating an incremental update, ΔW, which captures task-specific information:

$$h_{\text{out}} = W_0 h_{\text{in}} + \frac{\alpha}{r}\Delta W h_{\text{in}} = W_0 h_{\text{in}} + \frac{\alpha}{r} W_{\text{up}} W_{\text{down}} h_{\text{in}} \tag{2}$$

where α is a scaling factor that controls the contribution of the adaptation matrices. This approach allows LoRA to adapt the model to specific tasks without altering the pre-trained W_0, preserving the underlying knowledge in the original weights.

The training process for LoRA begins with W_{down} initialized from a random Gaussian distribution and W_{up} initialized to zero, ensuring that the initial value of ΔW is zero, meaning no modification to W_0 at the start. Over the course of training, the updates to W_{up} and W_{down} enable the model to learn task-specific adjustments, which can then be integrated with the pre-trained model's parameters.

Table 2. Methods of fine-tuning an LLM.

Method	Type	Description
LoRA	Parameter-Efficient Fine-Tuning	Reduces the number of parameters by decomposing weight matrices into low-rank products
BitFit	Parameter-Efficient Fine-Tuning	Only fine-tunes the bias terms in the model
Adapters	Parameter-Efficient Fine-Tuning	Introduces additional small neural network layers that can be fine-tuned, while keeping the main model fixed
Prefix-Tuning	Parameter-Efficient Fine-Tuning	Prepends a trainable continuous vector to the input
Prompt-Tuning	Parameter-Efficient Fine-Tuning	Optimizes a small set of input tokens (prompts) while keeping the model parameters fixed
QLoRA	Memory-Efficient Fine-Tuning	Combines quantization with low-rank adaptation to reduce memory usage
LOMO	Memory-Efficient Fine-Tuning	Fuses gradient computation and parameter updates into one step to minimize memory usage during backpropagation
Delta-Tuning	Parameter-Efficient Fine-Tuning	Restricts tuning to a low-dimensional manifold acting as optimal controllers, guiding model behavior
Diff Pruning	Parameter-Efficient Fine-Tuning	Introduces sparsity in the parameter updates to reduce the number of parameters that need to be fine-tuned
LoftQ	Memory-Efficient Fine-Tuning	Combines quantization with low-rank adaptation techniques to improve memory efficiency
AdapterDrop	Memory-Efficient Fine-Tuning	Reduces the number of adapters used during inference to save memory without sacrificing performance
AutoPEFT	Parameter-Efficient Fine-Tuning	Automates the configuration of PEFT methods to find the most efficient setup for a given task
Few-Shot Learning	Data Augmentation-Based Fine-Tuning	Uses training data to auto-complete the prompt with examples, aiding the model in understanding the task better
Zero-Shot Learning	Data Augmentation-Based Fine-Tuning	Incorporates external data into the prompt, enabling the model to interpret and answer queries with reduced hallucinations
LoRA-Adapter Fusion	Parameter-Efficient Fine-Tuning	Fine-tunes multiple independent LoRA adapters on distinct datasets and then fuses them with learnable weights to enhance model versatility and performance
Traditional Fine-Tuning	Full Model Fine-Tuning	Updates all parameters of the model based on the new task-specific data

One of the primary advantages of LoRA is the drastic reduction in the number of parameters that need to be fine-tuned, allowing for efficient memory usage and reduced computational demands. This is particularly beneficial for large-scale models with billions of parameters, as it enables fine-tuning without a substantial increase in the model size or inference cost. Selecting an optimal rank r is crucial in balancing model flexibility and efficiency, and it may require tuning depending on the complexity of the task [4–7].

BitFit, on the other hand, fine-tunes only the bias terms in the model, while keeping the rest of the model parameters fixed. This method is extremely parameter-efficient as it updates only a small subset of the model's parameters, significantly reducing the computational cost and memory usage. However, BitFit may not capture all task-specific nuances due to its limited parameter updates, and, while it works well on deep neural networks, it often fails with large language models [4,5].

Adapters introduce small neural network layers within the model that can be fine-tuned for specific tasks. These layers are added after each layer of the pre-trained model, with only these new layers being updated during fine-tuning. This approach allows for modular updates specific to different tasks, while keeping the main model parameters fixed, thereby preserving the model's general knowledge. However, this method adds some computational overhead during inference due to the additional layers [4,5].

Prefix-Tuning involves prepending a trainable continuous vector to the input embeddings. This vector is optimized while the rest of the model remains unchanged, making it an efficient method as it requires the optimization of only a small set of parameters. While this method is effective in modifying model behavior with minimal parameter changes, the added prefix can increase the input sequence length, slightly affecting the processing time [4,5,8].

Prompt-Tuning optimizes a small set of input tokens (prompts) while keeping the model parameters fixed. These prompts guide the model towards better performance on specific tasks without altering the model architecture, making it very lightweight in terms of additional parameters. However, the effectiveness of this method heavily depends on the quality and design of the prompts [4,5].

QLoRA combines quantization with low-rank adaptation to reduce the memory usage. It employs 4-bit quantization for weight matrices while applying low-rank adaptation techniques, greatly reducing the memory requirements and enabling the fine-tuning of very large models on limited hardware. Although QLoRA maintains performance close to that of full-precision models, managing the quantization errors is crucial to avoid performance degradations [4,5,8].

LOMO, or Low Memory Overhead, fuses gradient computation and parameter updates into one step to minimize the memory usage during backpropagation. This method is advantageous as it reduces the memory footprint during training, enabling efficient fine-tuning on hardware with limited memory capacities. However, the fused operations can complicate its implementation and debugging [4,5].

Delta-Tuning restricts tuning to a low-dimensional manifold, acting as an optimal controller to guide model behavior with minimal parameter updates. This method is highly efficient in terms of the number of parameters that need updating and maintains a high level of control over the model's behavior. However, this restriction to a low-dimensional space may limit the model's ability to fully adapt to complex tasks [4,5].

Diff Pruning introduces sparsity in the parameter updates, reducing the number of parameters that need to be fine-tuned. This method can lead to more efficient models if the sparsity is managed well, as it reduces the number of parameters to update, saving computational resources. However, if not applied carefully, pruning can lead to the loss of important information [4,5].

LoftQ combines quantization with low-rank adaptation techniques to improve the memory efficiency. This approach balances memory efficiency with fine-tuning performance and enhances generalization for downstream tasks. However, it requires the careful handling of quantization errors [4,5].

AdapterDrop reduces the number of adapters used during inference to save memory without sacrificing the performance. This technique is advantageous as it saves memory during inference while maintaining the performance by using only the necessary adapters. However, dropped adapters may sometimes contain useful task-specific information [4,5].

AutoPEFT automates the configuration of parameter-efficient fine-tuning methods to find the most efficient setup for a given task. It automates the selection and configuration process, saving time and effort, and can find optimal configurations that may not be obvious. However, the automation process itself may require additional computational resources and time [4,5].

In one study [9], an alternative approach to the fine-tuning of a large language model (LLM) is explored, utilizing the OpenAI API to enhance the performance in educational tasks with the GPT-3.5-turbo model. This technique is distinct as it does not modify the model's weights or architecture. Instead, it employs training data to auto-complete the inference prompt with examples that aid the model in comprehending the task and generating more accurate responses. This approach is termed "few-shot learning". According to [10], few-shot learning demonstrates the potential to train models effectively with limited labeled data, leveraging examples embedded within the prompt to adapt to the task requirements. A related technique, "zero-shot learning", is detailed in reference [3]. This method leverages external data to augment the model's knowledge by incorporating these data into the prompt, allowing the model to interpret the text and generate accurate answers to user queries. This approach reduces the likelihood of hallucinations and is beneficial for the integration of newer data. Both few-shot and zero-shot learning methods have been demonstrated to improve the response quality of large language models.

Multiple LoRA-Adapter Fusion, as utilized in reference [6], represents an innovative approach in the fine-tuning of large language models (LLMs). This technique involves independently fine-tuning multiple LoRA adapters on distinct datasets, subsequently fusing these adapters using learnable weights. This method capitalizes on the specific strengths of each adapter, trained on diverse data, to ensure that the resultant model is versatile and performs optimally across various tasks. A notable advantage of this approach is its mitigation of the performance degradation typically associated with the direct fusion of different datasets. By employing learnable fusion weights, the contribution of each adapter is optimized, enhancing the overall model performance. However, this method's complexity necessitates careful optimization to achieve the best results.

Traditional fine-tuning [11] entails updating all parameters of a pre-trained LLM to tailor it to a specific downstream task. This process typically begins with a model trained on a broad corpus, which is then fine-tuned by adjusting the entire set of model parameters using a task-specific dataset. The primary advantage of traditional fine-tuning is its flexibility and capacity to fully adapt the model to new tasks, capturing intricate task-specific nuances while leveraging the comprehensive pre-trained knowledge of the model. However, this process is computationally intensive, requiring substantial memory and processing power, particularly as the model size increases. Additionally, it can be time-consuming and carries a risk of overfitting if not properly managed, given that the entire model is adjusted for potentially smaller, more specific datasets.

In reference [11], a comparative analysis is presented between various fine-tuning methodologies, including traditional fine-tuning without prompts, hard prompting with unfrozen LLMs, soft prompting with unfrozen LLMs, and soft prompting with frozen LLMs. Traditional fine-tuning involves updating all model parameters, ensuring thorough adaptation to new tasks but at a high computational cost. Hard prompting with unfrozen LLMs integrates explicit textual prompts and updates all parameters, incorporating clear task-specific instructions directly into the input, yet still demands significant computational resources. Soft prompting, identified as prefix-tuning in Table 1, with unfrozen LLMs, employs continuous, trainable vectors added to the input sequence, facilitating adaptation with fewer parameter changes, thus balancing efficiency and performance. Notably, the cited study highlights that soft prompting with frozen LLMs, where only the prompt embeddings are updated while the model parameters remain fixed, significantly reduces the computational costs and simplifies the training process. This method not only enhances the efficiency but also improves few-shot learning and cross-institution's generalizability, making it a viable strategy for the deployment of LLMs in clinical settings where multiple tasks must be managed efficiently by a single model.

In terms of applicability and trade-offs, LoRA offers a highly efficient approach for scenarios where hardware constraints exist, such as edge devices or environments with limited computational resources. Its ability to maintain the integrity of the original model's parameters makes it ideal for use cases requiring model reuse across different tasks. However, tuning the low-rank matrices to achieve optimal performance can be a delicate and time-consuming process, as it depends on the selection of appropriate ranks for the matrices.

Similarly, QLoRA is beneficial when memory efficiency is paramount, as it combines quantization with low-rank adaptation to enable fine-tuning on resource-limited hardware. This makes QLoRA particularly applicable in settings where the memory bandwidth is restricted, such as mobile devices. However, the potential performance degradation due to quantization errors remains a trade-off that requires careful mitigation to maintain accuracy close to that of full-precision models.

Adapters and Prefix-Tuning are especially suitable for multi-task environments, as they allow modular updates tailored to specific tasks while preserving the model's core knowledge. Adapters add only a minor computational overhead, but this can accumulate when handling many tasks, leading to longer inference times. Prefix-Tuning, while lightweight and parameter-efficient, can introduce additional sequence lengths, which may impact the processing time in latency-sensitive applications.

AdapterDrop is advantageous in applications where memory efficiency during inference is critical, as it selectively reduces the number of active adapters. This method strikes a balance between memory usage and task performance; however, it may inadvertently omit valuable task-specific information when certain adapters are excluded.

LoRA-Adapter Fusion provides enhanced adaptability by fusing multiple task-specific adapters, making it particularly useful for comprehensive models applied across diverse domains. Nevertheless, this technique requires substantial optimization to manage the increased complexity, as the fused adapters must be weighted carefully to avoid diminishing returns from suboptimal fusion configurations.

3.1.1. Computational Efficiency of Methods for Fine-Tuning of Large Language Models

One of the primary considerations for any fine-tuning method is its computational efficiency, particularly when working with large models like GPT or LLaMA. Among the methods analyzed, LoRA (Low-Rank Adaptation) and its variant, QLoRA, stand out for their exceptional efficiency. By focusing on low-rank updates to certain layers, these methods significantly reduce the memory usage and computational requirements while maintaining model performance [4,8]. BitFit, which fine-tunes only the bias terms, is also extremely lightweight, making it ideal for low-resource settings [4]. However, traditional fine-tuning, which updates all model parameters, is resource-intensive and often requires parallel processing on multiple GPUs, making it less suitable for applications where the computational resources are limited [4].

Techniques like adapters and Prefix-Tuning offer a balanced trade-off between resource consumption and performance, introducing additional parameters while only training these, thus reducing the need to update the entire model [5]. AdapterDrop, a variation of the adapter method, improves the efficiency by activating the adapters only when necessary, thereby dynamically adjusting the resource usage [6]. On the other hand, LoRA-Adapter Fusion, which combines the strengths of both LoRA and adapters, demonstrates moderate efficiency by enabling multi-domain fine-tuning without requiring the full resources of traditional methods [6].

Delta-Tuning and Diff Pruning are noteworthy for their ability to reduce the computational overhead by selectively updating only the most important model parameters or pruning non-essential ones. These methods strike a balance between efficiency and performance [4]. In contrast, few-shot learning and zero-shot learning offer the highest levels of computational efficiency since they require minimal or no training data for new tasks, leveraging the model's pre-existing capabilities [5].

Based on the findings from this systematic literature review, we conducted a comparative analysis of the efficiency levels of various fine-tuning methods for a large language model (LLM), with a focus on the computational power required for each method. The methods were categorized into five efficiency levels: low, low to moderate, moderate, high, and very high.These results, which are our conclusions based on information extracted from the articles selected through the SLR, are summarized and presented in Figure 3.

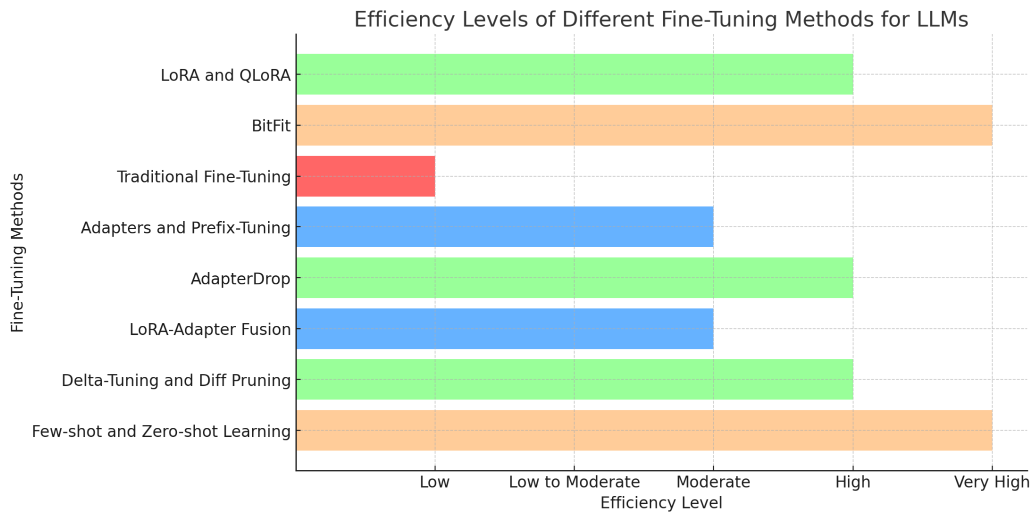

Figure 3. Comparison of fine-tuning efficiency across different methods for large language models (LLMs).

For a more comprehensive comparison, Table 3 presents the details regarding the efficiency of each fine-tuning method.

Table 3. Computational efficiency of methods for fine-tuning of large language models.

Method	Efficiency	Description
LoRA (Low-Rank Adaptation) and QLoRA	High	Focus on low-rank updates to certain layers, significantly reducing memory usage and computational requirements while maintaining model performance.
BitFit	Very High	Fine-tunes only the bias terms, making it extremely lightweight and ideal for low-resource settings.
Traditional Fine-Tuning	Low	Updates all model parameters, requiring significant computational resources and often parallel processing on multiple GPUs.
Adapters and Prefix-Tuning	Moderate	Introduce additional parameters but only train these, reducing the need to update the entire model, offering a balanced trade-off between resource consumption and performance.
AdapterDrop	Moderate to High	A variation of the adapter method that activates adapters only when necessary, dynamically adjusting the resource usage for improved efficiency.
LoRA-Adapter Fusion	Moderate	Combines the strengths of LoRA and adapters, enabling multi-domain fine-tuning with moderate efficiency, without requiring full resources of traditional methods.
Delta-Tuning and Diff Pruning	High	Reduce computational overhead by selectively updating or pruning only the most important model parameters, balancing efficiency and performance.
Few-Shot Learning and Zero-Shot Learning	Very High	Offer the highest computational efficiency by requiring minimal or no training data for new tasks, leveraging the model's pre-existing capabilities.

3.1.2. Accuracy of Methods for Fine-Tuning of Large Language Models

When it comes to accuracy, traditional fine-tuning generally offers the highest performance, as it updates all model parameters, ensuring complete adaptation to the target task [4]. However, methods like LoRA, adapters, and Prefix-Tuning can achieve comparable levels of accuracy, particularly when fine-tuned for specific domains. LoRA maintains high accuracy across various NLP tasks, especially in domain-specific settings [11], while adapters and LoRA-Adapter Fusion allow for strong performance in multi-domain applications [6].

QLoRA, which combines quantization with LoRA, is particularly effective in memory-intensive tasks like medical summarization, maintaining high accuracy while significantly reducing the memory usage [8]. On the other hand, BitFit and LOMO might sacrifice some accuracy due to their minimalist approach, but they still perform well in low-resource environments [4]. Prompt-Tuning, while efficient, tends to show slightly lower accuracy in complex tasks due to its reliance on modifying input prompts rather than the model parameters [5].

Methods like few-shot learning and zero-shot learning exhibit varying levels of accuracy depending on the similarity between the pre-trained model's capabilities and the new task. While these methods perform remarkably well in structured tasks, they may struggle in highly specialized domains where more fine-grained adaptation is needed [9]. Delta-Tuning and Diff Pruning also offer solid accuracy levels, particularly in tasks that require specialized adaptations without the need for full model retraining [4].

Building on the insights gained from this systematic literature review, we carried out an extensive comparative analysis focusing on the accuracy levels of the different fine-tuning methods applied to a large language model (LLM). The analysis classified these methods into five distinct accuracy categories: low, low to moderate, moderate, high, and very high. A summary of these results, which are our conclusions based on information extracted from the articles selected through the systematic literature review, is illustrated in Figure 4, providing a clear visual representation of how each method performs across these categories.

For a more detailed comparison, Table 4 provides the accuracy specifics for each fine-tuning method.

Table 4. Accuracy of methods for fine-tuning of large language models.

Method	Accuracy	Description
Traditional Fine-Tuning	Very High	Updates all model parameters, ensuring complete adaptation to the target task.
LoRA	High	Comparable to traditional fine-tuning for specific domains, especially in domain-specific settings.
Adapters	High	Often used with LoRA for strong performance across multiple domains.
LoRA-Adapter Fusion	High	Suitable for multi-domain applications.
QLoRA	High	Effective in memory-intensive tasks like medical summarization; reduces memory usage while maintaining accuracy.
BitFit	Moderate	Minimalist approach; may sacrifice some accuracy but performs well in low-resource environments.
LOMO	Moderate	Similar to BitFit, may trade some accuracy for reduced complexity.
Prompt-Tuning	Low to Moderate	Modifies input prompts rather than model parameters, leading to efficiency but lower accuracy in complex tasks.
Few-Shot Learning	High	Performs well in structured tasks and is effective depending on task similarity; may face challenges in highly specialized domains.
Zero-Shot Learning	High	Similar to few-shot learning; accuracy depends on similarity between pre-trained model capabilities and new task requirements.
Delta-Tuning	High	Effective for tasks that require specialized adaptations without full model retraining.
Diff Pruning	High	Similar to Delta-Tuning; avoids full retraining while offering good accuracy.

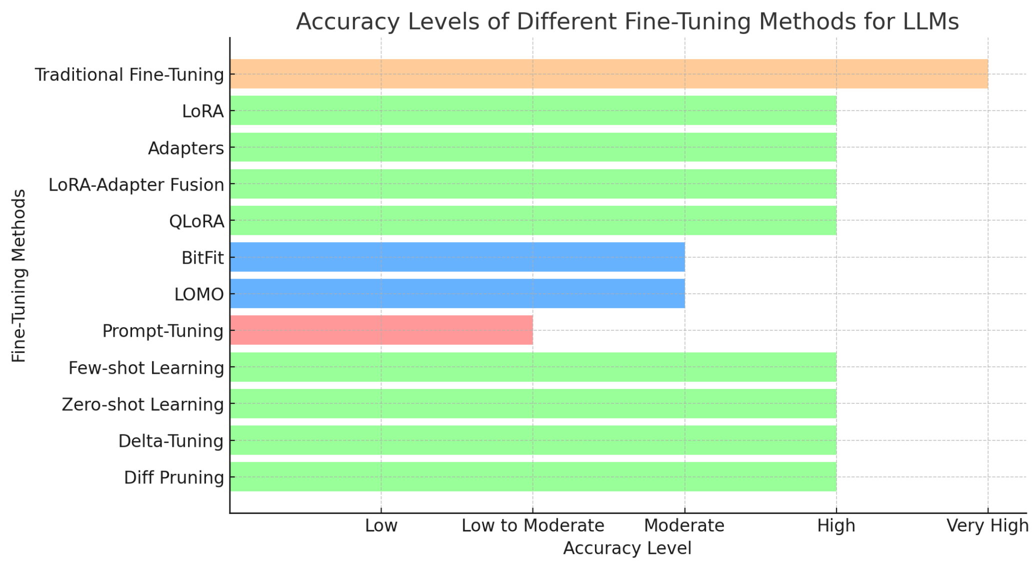

Figure 4. Comparison of fine-tuning accuracy across different methods for large language models (LLMs).

3.1.3. Applicability of Methods for Fine-Tuning of Large Language Models

The applicability of fine-tuning methods depends largely on the target task and the availability of resources. LoRA and adapters are widely applicable across various NLP tasks, making them go-to options for both academic research and industry use cases. LoRA, with its efficiency and strong performance, is ideal for applications where the computational resources are constrained but high accuracy is still required [11]. Adapters, and particularly LoRA-Adapter Fusion, are suitable for multi-domain tasks, allowing for flexible fine-tuning across different applications without the need to retrain the entire model [6].

In environments with stringent resource limitations, BitFit and LOMO are highly applicable, offering lightweight fine-tuning solutions that can be deployed on low-end hardware [4]. Prompt-Tuning, due to its focus on modifying input prompts, is particularly useful in zero-shot or rapid deployment scenarios where retraining the model is not feasible [11]. Few-shot learning and zero-shot learning are applicable in cases where there are limited or no training data available, making them ideal for generalized tasks or domains with few labeled data [5].

More specialized methods like QLoRA, Delta-Tuning, and Diff Pruning are highly applicable in domains such as healthcare or finance, where resource efficiency must be balanced with high accuracy and privacy protection [8]. QLoRA, in particular, is well suited for applications involving large-scale datasets and sensitive information, as it minimizes the memory usage while maintaining the performance [8]. Diff Pruning, which focuses on reducing the model size without sacrificing accuracy, is ideal for deployment on edge devices with limited computational power [4].

The choice of fine-tuning method depends heavily on the specific requirements of the task, the available computational resources, and the desired level of accuracy. Methods like LoRA, adapters, and LoRA-Adapter Fusion offer strong performance across a wide range of tasks, while BitFit, LOMO, and Prompt-Tuning provide efficient alternatives for low-resource environments. Few-shot and zero-shot learning are invaluable when task-specific data are scarce, and QLoRA and Diff Pruning cater to specialized, resource-constrained applications. Each method presents a unique balance of computational efficiency, accuracy, and applicability, making it essential to choose the one that aligns best with the specific needs of the project.

Drawing on the insights obtained from this systematic literature review, we performed an in-depth comparative analysis that examined the applicability levels of the various fine-tuning methods utilized for a large language model (LLM). In this analysis, each method was categorized into five specific applicability levels: low, low to moderate, moderate, high, and very high. The findings from this assessment, derived from our conclusions based on information extracted from the articles selected through the SLR, are summarized in Figure 5, which offers a comprehensive visual overview of how each fine-tuning method aligns across these categories. This detailed comparison sheds light on the relative applicability and suitability of the different techniques evaluated.

To offer a more detailed comparison, Table 5 outlines the specifics of the applicability of each fine-tuning method.

Table 5. Applicability of fine-tuning methods for large language models.

Method	Applicability	Comments
LoRA	High	Ideal for applications with limited computational resources but requiring high accuracy.
Adapters	High	Suitable for multi-domain tasks; allows flexible fine-tuning across different applications without retraining the entire model.
LoRA-Adapter Fusion	Very High	Combines strengths of LoRA and adapters; applicable for multi-domain tasks with a focus on resource efficiency.
BitFit	High	Highly applicable in resource-constrained environments; suitable for deployment on low-end hardware.
LOMO	High	Lightweight and resource-efficient, ideal for low-resource environments.
Prompt-Tuning	Moderate	Useful for zero-shot or rapid deployment scenarios where retraining is not feasible.
Few-Shot Learning	Moderate	Applicable when limited training data are available, suitable for generalized tasks or domains with scarce labeled data.
Zero-Shot Learning	Low to Moderate	Useful in scenarios with no training data, applicable for generalized tasks or domains with few labeled data.
QLoRA	Very High	Specialized for domains like healthcare or finance; minimizes memory usage while handling large datasets and sensitive information effectively.
Delta-Tuning	High	Applicable for specialized domains requiring a balance of resource efficiency, accuracy, and privacy.
Diff Pruning	Very High	Ideal for edge device deployment, focuses on reducing model size without sacrificing accuracy.

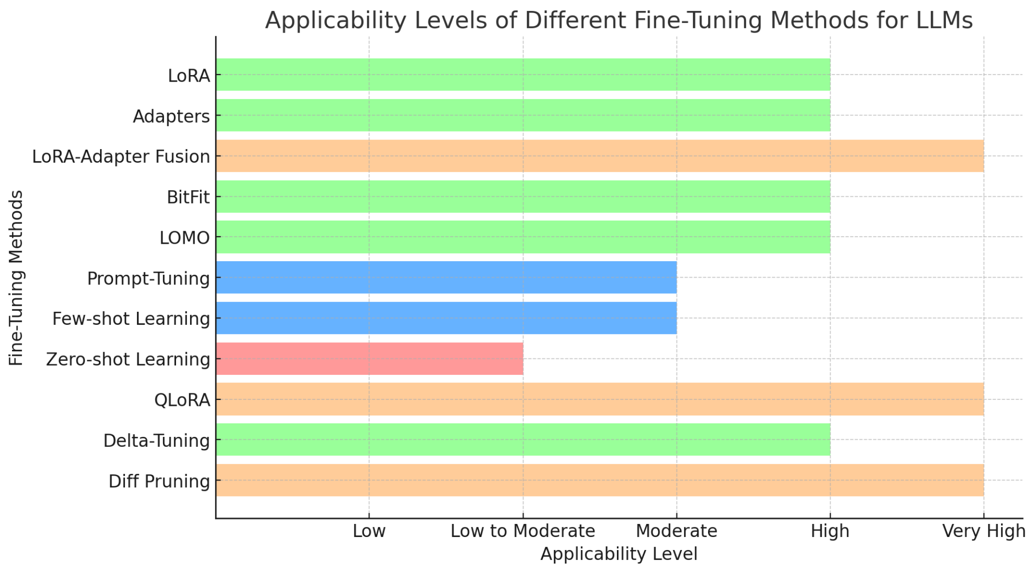

Figure 5. Comparison of fine-tuning applicability across different methods for large language models (LLMs).

3.2. Techniques for Searching of Relevant Data for User Questions in Databases

In the field of automatic question-answering systems, finding relevant data to address user questions is a critical task. This section delves into the various methodologies employed to search for relevant data within databases, ensuring precise and contextually appropriate responses. The methods highlighted here stem from comprehensive literature reviews, advanced retrieval techniques, and the innovative integration of machine learning models.

Initial QA systems heavily relied on pattern matching using regular expressions and context-free grammars (CFG). These methods involved hard-coding complex logical rules to identify specific sequences of characters or parts of speech within documents. Despite enabling basic QA capabilities, these techniques were rigid and limited in capturing the semantic nuances of natural language, restricting their application to simple keyword searches and scripted responses [12].

As the field evolved, statistical methods emerged, representing words as numerical vectors and reducing documents to fixed-length lists of numbers. These methods involved preprocessing text by removing stop-words and stemming and then constructing matrices, such as document–term matrices. Statistical operations computed the distances and similarities between vectors, allowing more sophisticated QA by comparing vector representations of text. However, these methods still struggled to capture deeper contextual meanings and relationships within the text [12].

Building on the foundational tasks of data identification, diverse methodologies have been developed for QA systems.

- Text-Based QA Systems: These systems utilize unstructured documents, such as Wikipedia entries, through processes like question analysis, passage retrieval, and answer extraction. Question analysis dissects the input to understand its intent using morphological, syntactical, and semantic analysis. Passage retrieval searches for relevant documents, extracting and ranking excerpts based on their potential to answer the query. Finally, answer extraction generates and validates the answer from the retrieved passages [13].
- Knowledge-Based QA Systems: These systems leverage structured data from knowledge bases. They employ information retrieval to sort candidate answers and semantic parsing to convert sentences into executable semantic representations. Neural networks enhance the parsing capabilities, making these systems more robust in handling complex queries [13].
- Hybrid QA Systems: Combining structured and unstructured data, hybrid systems integrate text-based and knowledge-based approaches. They employ question analysis to reduce the search space by understanding the question’s context and intent, utilizing techniques like information retrieval and semantic parsing to enhance the efficiency and accuracy [13].

Transformer-based models have significantly advanced QA systems. BERT (Bidirectional Encoder Representations from Transformers) and DistilBERT exemplify this progress. BERT is pre-trained on extensive text corpora using a masked language model objective and fine-tuned for tasks like question answering. It identifies the start and end positions of answers within passages, providing a robust mechanism for context understanding. DistilBERT, a compact and faster variant of BERT, achieves similar performance with reduced computational requirements. The CO-SE pipeline, presented in [14], leverages Transformer-based models, consisting of two main components: the retriever and the reader. The retriever employs the TF-IDF (Term Frequency-Inverse Document Frequency) model to fetch relevant documents. These documents are then passed to the reader, which uses the fine-tuned DistilBERT model to extract precise answers by understanding the context within the text, efficiently addressing specific queries like those related to COVID-19 [14,15].

Contextual embeddings, such as ELMo, BERT, and GPT, also enhance QA systems. ELMo uses a bidirectional LSTM model to derive word embeddings based on entire input sentences, capturing deeper, context-dependent word meanings. BERT employs a bidirectional Transformer encoder to predict masked words, integrating both left and right contexts, making it highly effective for QA. GPT combines unsupervised pre-training with supervised fine-tuning, learning long-range dependencies and acquiring substantial world knowledge, thus improving the QA performance [12,15].

An alternative approach to measuring textual similarity involves integrating two methods, word embedding and TF-IDF weighting, as described in reference [16]. This approach leverages models such as Word2Vec, GloVe, and FastText to generate word embeddings, which capture the semantic relationships between words. Concurrently, the TF-IDF model is employed to assess the significance of the words within the text. By combining these models, the method calculates word embeddings for each word in a document and then determines the document's overall embedding by computing a weighted average of these word embeddings, with weights derived from the TF-IDF scores. This integrated approach enables the retrieval of relevant information from extensive databases by calculating the cosine similarity between a user's query and each document, thus facilitating efficient and accurate data retrieval.

A solution for low-resource languages consists of cross-lingual models like XLM-R and Unicoder, which leverage multilingual data to optimize QA tasks across different languages. XLM-R enhances the performance through cross-lingual transfer, while Unicoder employs various pre-training tasks to become a robust language-independent encoder for QA [12,15]. The idea of using bilingual models was also presented in reference [17] as a solution for low-resource languages.

In some special cases, an approach like a template-based method can be used to address the limitations inherent in more complex question-answering systems. A template-based approach employs predefined templates to generate responses to specific types of queries. This method involves creating a set of patterns or templates that can be matched against the input query to produce a structured response. The templates are designed based on the anticipated structure of the questions, allowing the system to quickly identify the relevant template and fill in the necessary information from the provided data. This approach is particularly effective in handling straightforward, factual queries where the answer can be directly extracted from the available data, without the need for extensive reasoning or inference. However, template-based systems may struggle with more complex queries that require deeper understanding and contextual analysis, as they rely heavily on predefined templates and may not adapt well to variations in question phrasing or unexpected query types [15].

The hybrid approach outlined in [18] utilizes MultiWordNet to identify synonyms and hypernyms, along with a Word2Vec model trained on domain-specific data to rank and filter these expanded terms based on their semantic similarity to the query. This multi-step pipeline ensures that queries are enhanced with contextually relevant terms, thereby improving the precision and recall of relevant sentence retrieval. By contextualizing terms within the document collection, this approach addresses the limitations of previous methods, capturing both the semantic and syntactic nuances, which is essential for accurate QA. The integration of lexical resources and word embeddings represents a significant advancement in searching for relevant data in automatic question-answering systems, demonstrating improved performance in closed-domain QA tasks.

A technique to enhance the accuracy of a system that retrieves pertinent information for user queries from extensive databases is discussed in reference [19]. Large databases present certain challenges, such as prolonged search times and the presence of similar data that may confuse the model. This paper introduces the method of clustering questions, where a user's query is categorized into one of these clusters. After identifying the relevant cluster, the system conducts a search within this cluster to locate the most suitable answer to the user's query. This process entails comparing the user's query with the questions and answers within the cluster using predefined similarity measures. This comprehensive approach enables question-answering (QA) systems to address a wide range of question types and contexts, ultimately enhancing the user experience by delivering more precise and relevant answers.

To thoroughly evaluate the performance of question-answering (QA) systems, several well-established metrics are employed, each providing unique insights into different aspects of accuracy and effectiveness. Firstly, accuracy is a straightforward metric often used for datasets containing single-answer questions. It measures the proportion of correctly answered questions by dividing the number of correct answers by the total number of questions answered. This metric is particularly useful for simple, fact-based questions where there is only one correct answer. However, it may not fully capture the nuances of QA systems designed to handle more complex or multi-answer queries.

For cases where multiple correct answers are possible, the F-score is more appropriate as it combines both precision and recall into a harmonic mean. Precision represents the fraction of correctly retrieved answers out of all answers retrieved, while recall indicates the fraction of correctly retrieved answers out of all relevant answers. By calculating the F-score, which gives equal weight to precision and recall, we obtain a balanced view of the system's ability to generate and retrieve correct answers. This is particularly useful in multi-answer scenarios, where both the completeness and correctness of the answers are crucial for a meaningful evaluation.

Another critical metric is the mean reciprocal rank (MRR), which assesses the QA system's capability to rank correct answers highly. The MRR is calculated by taking the mean of the reciprocal ranks of the first correct answer across all questions, providing an average measure of how well the system prioritizes the most relevant answers. This metric is especially valuable in scenarios where users expect the most accurate answer to appear first or near the top of the list.

Mean Reciprocal Rank (MRR): This metric evaluates the rank of the first relevant answer for a set of questions. It is calculated as the average reciprocal rank across all questions and is defined as

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i} \tag{3}$$

Here, $|Q|$ is the total number of questions, and rank_i represents the rank position of the first correct answer for question i. A higher MRR value indicates that relevant answers are ranked higher on average.

Mean Average Precision (MAP): This metric extends the evaluation by considering the precision of relevant answers at multiple recall levels for each question. It averages the average precision (AP) across all questions and is defined as

$$\text{MAP} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \text{AP}_i \tag{4}$$

Here, AP_i is the average precision for question i, calculated as

$$\text{AP}_i = \frac{1}{|R_i|} \sum_{j=1}^{n} P(j) \cdot \text{rel}(j) \tag{5}$$

In this equation, $|R_i|$ is the total number of relevant answers for question i, $P(j)$ represents the precision at position j, and $\text{rel}(j)$ is a binary indicator that is 1 if the answer at position j is relevant and 0 otherwise. The MAP provides a comprehensive view of a QA system's performance by considering the ranked list of possible answers.

Here, $|R_i|$ is the number of relevant documents for question i, $P(j)$ is the precision at position j, and $\text{rel}(j)$ is a binary function that is 1 if the document at position j is relevant and 0 otherwise. By employing these metrics together, researchers can gain a multidimensional understanding of the strengths and weaknesses of QA systems, enabling them to make targeted improvements [13].

This overview highlights the diverse and evolving methodologies in automatic QA systems, showcasing the continuous advancements aimed at improving the accuracy and efficiency in answering user queries.

3.3. Strategies for Implementation of Retrieval-Augmented Generation

Retrieval-augmented generation (RAG) improves the responses of large language models (LLMs) by leveraging external, authoritative knowledge bases, rather than depending solely on potentially outdated training data or the model's internal knowledge. This method addresses the critical issues of accuracy and timeliness in LLM outputs. By incorporating RAG, the problem of hallucination in LLMs is significantly reduced, leading to responses that are relevant, up-to-date, and verifiable [1]. Furthermore, the lack of knowledge about private data often leads to hallucinations, but, by using RAG, the model can access these data and minimize such inaccuracies. This enhances user trust and provides developers with a cost-effective means to boost the reliability and usefulness of LLMs across various applications. For instance, RAG enables advanced chatbot functionalities by combining techniques like LangChain and performance-optimized LLM fusion to ensure effective and precise responses. It employs web scraping capabilities to integrate extensive information, improving the contextual relevance and informativeness of responses. When fine-tuned with efficiency-driven strategies like LoRA and QLoRA, RAG can provide a seamless blend of external knowledge retrieval and computational efficiency, empowering applications such as custom chatbot development in medical and other specialized domains [10]. RAG implementation can be divided into strategies applied before generation, during generation, after generation, and end-to-end [1].

For the before generation strategies, we found that it involves augmenting the LLM before the text generation process begins.

- LLM-Augmenter [1,20] is a system that augments a black-box LLM with plug-and-play (PnP) modules. The system retrieves evidence from external knowledge bases and forms evidence chains. This evidence is then used to prompt the LLM, which generates a response grounded in the retrieved evidence. The response is iteratively verified and refined to ensure accuracy.
- FreshPrompt [1,21] is a method to address the static nature of most LLMs by incorporating dynamic, up-to-date information through a search engine. This approach enhances the model's ability to adapt to evolving knowledge, improving the correctness of the responses in fast-changing scenarios.
- Context curation [22] involves adjustments to the retrieved content before inputting it to the LLM, which include reranking and context selection/compression, to highlight the most pertinent results and reduce noise.

The during generation strategy focuses on integrating knowledge retrieval during the text generation process, ensuring that each generated sentence is supported by relevant information.

- Knowledge retrieval [1,23] is a method where potential hallucinations are detected and corrected during sentence generation. By utilizing logit output values, the system identifies inaccuracies and retrieves supporting knowledge to correct them, thus reducing the likelihood of hallucinations.
- The Decompose-and-Query (D&Q) framework [1,24] addresses challenges in question answering by guiding models to use external knowledge and constrains their reasoning to reliable information. This framework significantly improves the robustness and accuracy of LLMs in generating answers.
- The EVER framework [1,25] introduces a real-time verification and rectification framework that detects and corrects hallucinations during the generation process. This stepwise approach ensures that the generated content is trustworthy and factually accurate.

- Adaptive retrieval [22] involves techniques like Self-RAG, which allow the model to determine optimal moments and content for retrieval, enhancing its efficiency and relevance.
- Recursive retrieval [22] involves iteratively refining the search queries based on the previous results to improve the depth and relevance of information. This is particularly useful for complex search scenarios.

Post-generation or after generation techniques involve refining the generated output by cross-verifying it with external sources.

- RARR (Retrofit Attribution using Research and Revision) [1,26] represents a system that retrofits attributions by conducting research and revising the generated content based on the retrieved evidence. This post hoc verification ensures that the final output aligns with verifiable facts.
- High-entropy word spotting and replacement [1,27] involves detecting high-entropy words, which are more prone to hallucinations, and replacing them with lower-entropy alternatives from reliable models. This method effectively reduces hallucinations by focusing on the most problematic words in the generated text.

The end-to-end strategy of RAG [1,2] integrates a pre-trained sequence-to-sequence (seq2seq) Transformer with a dense vector index of Wikipedia, accessed through the Dense Passage Retriever (DPR). This combination allows the model to generate outputs based on both the input query and relevant documents retrieved by the DPR. By training the generator and retriever jointly, the model achieves enhanced performance in knowledge-intensive tasks, demonstrating the efficacy of combining parametric and non-parametric memory in LLMs.

The Naive RAG paradigm, one of the earliest methodologies in the field, is characterized by a basic "Retrieve–Read" framework, consisting of indexing, retrieval, and generation phases [22]. Initially, raw data are indexed by cleaning and transforming diverse document formats into a consistent vectorized form, which allows efficient similarity-based retrieval. Upon receiving a user query, the model retrieves relevant content chunks and synthesizes a response through a frozen language model. However, Naive RAG often suffers from limitations in its retrieval precision and generation coherence, which may lead to irrelevant or hallucinated outputs. Figure 6 (left) illustrates the simple linear process of Naive RAG, showcasing its foundational approach.

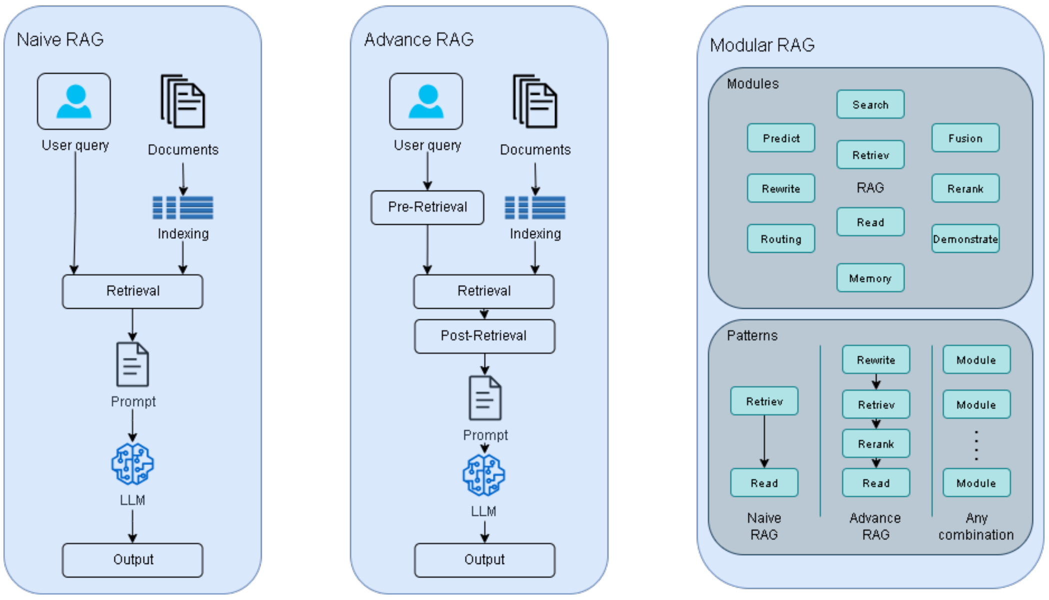

Figure 6. Visual representations of Naive RAG, Advanced RAG, and Modular RAG.

Advanced RAG builds upon the Naive RAG paradigm by introducing optimizations in the pre- and post-retrieval processes. In the pre-retrieval phase, techniques like query expansion and rewriting are employed to enhance the query clarity, while post-retrieval processes such as re-ranking and summary fusion are used to refine the retrieved context [22]. These enhancements address some of the precision and relevance issues seen in Naive RAG, making the Advanced RAG framework more robust. As depicted in Figure 6 (middle), Advanced RAG maintains a sequential structure but incorporates additional steps for higher retrieval accuracy and content integration.

Modular RAG represents a further advancement by enabling flexible module configurations that can be adapted to various tasks. Unlike Naive and Advanced RAG, which follow a rigid retrieval-to-generation flow, Modular RAG allows for the dynamic reconfiguration of components, such as adding dedicated search, fusion, and memory modules [22]. This flexibility supports both sequential and iterative retrieval processes, enhancing the framework's adaptability across diverse applications. As seen in Figure 6 (right), Modular RAG's architecture supports complex interactions between modules, facilitating enhanced retrieval performance and task-specific adjustments.

Each RAG paradigm illustrates the progressive evolution of retrieval-augmented generation systems, from the simplicity of Naive RAG to the adaptable, task-oriented structure of Modular RAG. These developments underscore the importance of modular design and strategic optimization in addressing retrieval challenges and achieving more contextually relevant outputs in natural language generation.

The evolution of retrieval-augmented generation (RAG) paradigms—from Naive to Advanced and Modular—highlights the significant advancements in integrating external knowledge with language models, each with distinct advantages and limitations [22]. The Naive RAG framework, characterized by its simple "Retrieve–Read" process, excels in its straightforward implementation and accessibility for foundational use cases. However, it often suffers from low retrieval precision and generation coherence, which can lead to irrelevant or hallucinated outputs [22]. Advanced RAG addresses these shortcomings through optimizations in the pre-retrieval and post-retrieval stages, such as query expansion and re-ranking, enhancing the retrieval accuracy and relevance. Despite these improvements, its sequential framework remains rigid, limiting its adaptability in dynamic scenarios [22]. Modular RAG, the most flexible of the paradigms, introduces configurable components like dedicated search and memory modules, enabling task-specific and iterative retrieval processes. This adaptability makes it suitable for diverse applications but also increases the architectural complexity and resource requirements [22]. Collectively, these paradigms illustrate the trade-offs between simplicity, robustness, and flexibility in RAG systems [22].

In our research, we also found a case study in clinical informatics [3], which demonstrated the practical application and benefits of RAG by employing the Llama 2 model with zero-shot prompting and integrating RAG to summarize and extract key clinical information from electronic health records (EHRs) related to malnutrition management. The incorporation of RAG significantly improved the model's performance by providing access to an extended knowledge base, enhancing the accuracy and relevance of the generated summaries and extracted information. This exemplifies a "before generation" strategy, where information was extracted at an initial phase and subsequently used in the prompt of the LLM, enabling it to utilize new data, combined with the multi-tasking power of LLMs. By integrating RAG, the model was able to access more detailed and accurate information, leading to improved performance in tasks such as the structured summarization of clinical notes and the extraction of malnutrition risk factors. This integration mitigated the hallucination problem commonly observed in LLMs, demonstrating the value of RAG in enhancing the reliability of generative models in clinical settings.

In reference [28], a method was discovered that utilized customer reviews and Q&A pairs to produce contextually relevant answers during response generation. This approach involved calculating embeddings for all e-commerce data collected from the internet and saving them. By using the cosine similarity between the user's question embeddings and the stored embeddings, the most relevant review and Q&A pair were identified and used as input for the model. This dynamic retrieval and incorporation of pertinent user-generated content improves the relevance and accuracy of the responses, especially in e-commerce and customer service settings. Consequently, this strategy ensures that the generated text is not only precise but also closely aligns with users' expectations and real-world feedback.

In reference [29], the implementation of retrieval-augmented generation (RAG) within the domain of healthcare administrative task automation was examined. This investigation introduced a multi-agent framework that incorporated RAG to manage and optimize administrative functions, including patient registration, medical billing, and appointment scheduling. This system employed RAG to retrieve pertinent patient information from external databases and seamlessly integrate it into the large language model's processing pipeline, ensuring that the generated responses were both accurate and contextually relevant. The multi-agent architecture facilitated efficient task decomposition and execution, utilizing RAG to verify and refine the data throughout various stages of the workflow. The RAG system operated using data embeddings, extracted from PDF files and databases, which were segmented into fixed-size chunks. The system then calculated embeddings for user queries to extract the most relevant information. This methodology not only alleviated the administrative workload on healthcare professionals but also improved the overall efficiency and accuracy of administrative processes, demonstrating a comprehensive end-to-end strategy for the application of RAG in a complex, real-world scenario.

Some more strategies can be found in reference [30], where the authors presented the development of a retrieval-augmented generation (RAG) medical assistant for infectious diseases. This incorporated three distinct retrieval techniques: Naive RAG, auto-merging retriever-based RAG, and ensemble retriever-based RAG. The Naive RAG approach utilized a direct retrieval process that synthesized documents based on user queries, although it may not always yield the most precise information. The auto-merging retriever-based RAG method enhanced the accuracy by dividing longer documents into smaller sections, enabling more precise retrieval and improving the relevance of the synthesized responses. The ensemble retriever-based RAG further refined the retrieval process by combining multiple algorithms, employing the Reciprocal Rank Fusion (RRF) algorithm to aggregate the rankings and ensure that the final results were highly relevant. This comprehensive approach significantly bolstered the chatbot's capability to provide accurate, contextually pertinent answers from a graph database-structured knowledge graph. Notably, the ensemble retriever demonstrated exceptional performance in terms of answer accuracy and contextual relevance, underscoring RAG's potential to enhance the reliability and effectiveness of large language models (LLMs) in healthcare by grounding the responses in the most pertinent and up-to-date information available.

In Figure 7, the process of answering complex user queries using RAG is illustrated. The figure showcases a workflow where a user query triggers the retrieval of relevant documents from a database, which are then processed by a large language model to generate a factually accurate and contextually rich response. This example emphasizes the advantages of RAG by integrating real-time data retrieval with LLM capabilities, ensuring that the generated outputs are informed by up-to-date and contextually relevant information. Unlike traditional approaches that rely solely on static, pre-trained knowledge, RAG dynamically incorporates external knowledge, enhancing the precision and relevance of the answers. This approach is particularly advantageous in addressing queries that require current and accurate data, such as understanding the lifespans of electric vehicle batteries, as shown in the example provided. A similar example based on the Public Sector Decarbonization Scheme can be found in [31], underscoring the effectiveness of RAG in real-world applications.

In order to address the privacy concerns associated with sensitive data in a retrieval-augmented generation (RAG) system, a user rule-based database can be utilized to ensure that only accredited individuals have access. This approach involves setting access control rules that define permissions based on user roles and accreditation, which the system checks before granting data access. By leveraging role-based access control and authentication protocols, the system restricts data retrieval to authorized users only. This method not only protects sensitive information but also supports logging and auditing to ensure compliance with privacy regulations.

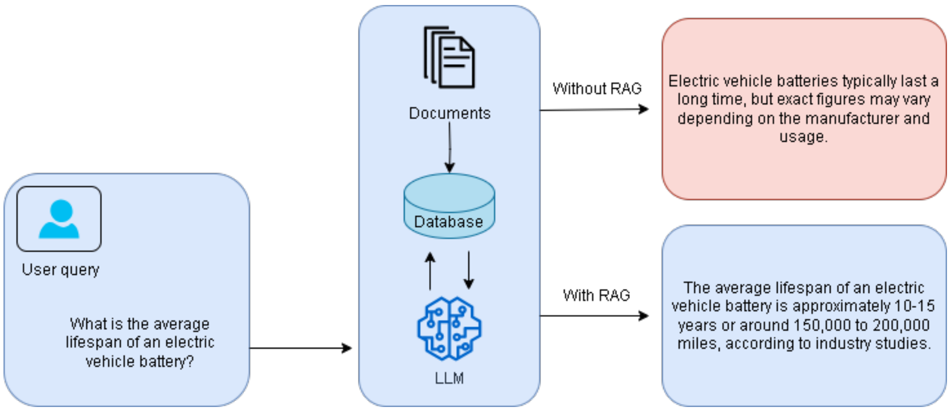
Documents
Database
LLM
User query
What is the average lifespan of an electric vehicle battery?
Without RAG
With RAG
Electric vehicle batteries typically last a long time, but exact figures may vary depending on the manufacturer and usage.
The average lifespan of an electric vehicle battery is approximately 10-15 years or around 150,000 to 200,000 miles, according to industry studies.

Figure 7. The benefits of retrieval-augmented generation.

To address the empirical evidence supporting the effectiveness of retrieval-augmented generation (RAG) in enhancing accuracy and reducing hallucination in large language models (LLMs), various studies provide concrete examples across different domains. For instance, the application of RAG in clinical settings demonstrated improved summarization and extraction accuracy when dealing with electronic health records (EHRs), as [3] reported that integrating RAG with a generative model increased the summarization accuracy by 6%, achieving an impressive 99.25% overall accuracy, while also reducing hallucinations in extracting critical information from EHRs. Additionally, RAG has the capability to diminish hallucinations by leveraging external, authoritative knowledge bases, which allows models to generate responses based on current and factual data [22]. This approach has also been effective in tasks requiring precise numerical extraction, as illustrated in [32] via document-based question answering, which showed that RAG improved the accuracy in handling exact answer selection and complex numerical data. By drawing on external databases, RAG has proven instrumental in enhancing the reliability of LLMs, particularly in domains where outdated or unverified information could compromise the performance [1].

After extracting the strategies for the implementation of a retrieval-augmented generation (RAG) system, which were identified using the systematic literature review (SLR) method, it was essential to establish a methodology for the evaluation of the interpretative capacity of large language models (LLMs) in processing retrieved data. As highlighted in reference [32], several evaluation tasks have been identified to assess this interpretative ability. These tasks involve various question types, such as single-choice, multiple-choice, single-choice (numbers), yes–no questions, and number extraction. Each of these question formats is designed to challenge the model's capacity to select or generate accurate answers based on the retrieved content. For evaluation purposes, a dataset in English, tailored to advanced models like GPT-3 and GPT-4, was employed. Using predefined answers, the accuracy of the models was measured, allowing for a comparative analysis of their performance based on their ability to interpret and process the data retrieved.

In addition to the accuracy-focused metrics, reference [22] introduces a set of quality scores, including context relevance, answer faithfulness, and answer relevance. These scores are crucial in assessing the efficiency of LLMs in handling retrieved data, as they measure the model's ability to deal with noise, filter out irrelevant data, synthesize information from multiple documents, and identify counterfactuals. For performance assessment, task-specific metrics such as the exact match (EM), $F1$ score, BLEU, and ROUGE are commonly applied to evaluate an LLM's ability to produce responses that are both accurate and coherent. These metrics each contribute unique insights into the model's performance. The $F1$ score, for example, measures the balance between precision (the proportion of relevant items among the retrieved items) and recall (the proportion of relevant items that were retrieved), which is especially useful in evaluating whether the model's responses are both relevant and complete in covering the expected information. BLEU (Bilingual Evaluation Understudy) compares the overlap of n-grams between the model-generated response and a known result, providing a quantitative measure of how similar the response is to an expected output. ROUGE (Recall-Oriented Understudy for Gisting Evaluation), on the other hand, assesses the quality of the generated response by evaluating recall-based overlaps of n-grams, which helps to determine how much of the reference text the response captures.

The $F1$ score, BLEU score, and ROUGE score are defined as follows.

$F1$ Score: The $F1$ score is a statistical measure that combines precision and recall into a single metric. It is particularly useful when the balance between precision (how many predicted positives are actual positives) and recall (how many actual positives are correctly identified) is critical. The $F1$ score is the harmonic mean of the precision and recall and is defined as

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6}$$

In this formula, a higher $F1$ score indicates better performance in balancing precision and recall.

BLEU (Bilingual Evaluation Understudy): BLEU is a metric designed to evaluate the quality of the text generated by a model by comparing it to a reference text. It focuses on the precision of n-grams (sequences of n words), penalizing outputs that are too short using a brevity penalty (BP). To evaluate BLEU with multiple n-grams, the formula is expressed as

$$\text{BLEU} = \text{BP} \times \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{7}$$

Here, p_n represents the precision of n-grams, w_n indicates the weight assigned to each n-gram level (e.g., unigrams, bigrams, etc.), and BP is the brevity penalty used to discourage excessively short generated outputs. BLEU captures the precision of n-grams while accounting for the length of the generated text.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation): ROUGE is a family of metrics that measure the quality of generated text by assessing the overlap of n-grams between the generated and reference texts. ROUGE-N, one of the commonly used variants, is defined as

$$\text{ROUGE-N} = \frac{\sum_{\text{ngram} \in \text{Reference}} \text{Count}_{\text{match}}(\text{ngram})}{\sum_{\text{ngram} \in \text{Reference}} \text{Count}(\text{ngram})} \tag{8}$$

In this formula, N refers to the length of the n-grams being evaluated. ROUGE-N specifically focuses on recall, measuring how many n-grams from the reference text are present in the generated text.

Equations (6)–(8) offer detailed insights into the mathematical foundations of these widely used evaluation metrics, each tailored to specific evaluation scenarios in text generation tasks.

Further elaboration of the evaluation techniques is provided in reference [33], which discusses model scoring systems like GPT-Eval and LLM-Mini-CEX. These LLM-based scoring systems are specially trained to assess the responses generated by other models, adding an extra layer of validation and enabling a deeper understanding of the model performance across multiple dimensions.

4. Discussion and Future Directions

4.1. Answers to Research Questions

The data were extracted from the studied articles as they were explained and clarified in Section 3.

Q1: Which techniques are used to fine-tune a large language model (LLM)? Fine-tuning an LLM traditionally involves adjusting its weights via backpropagation during the training process, which, although effective, requires substantial computational resources due to the model's size [11]. This study explores alternative fine-tuning techniques to enhance the model quality without extensive computational costs. These techniques fall into three main categories: (1) methods that modify only a subset of the model's parameters, such as LoRA (Low-Rank Adaptation) [4–7], BitFit [4,5], and adapters [4,5]; (2) approaches that integrate additional neural network layers (e.g., adapters) while keeping the original model's weights frozen; and (3) strategies that enhance the input data without altering the model's weights, such as few-shot and zero-shot learning [3,9]. Each of these techniques has demonstrated task-specific performance improvements. A key advantage of these fine-tuning methods is their combinability, allowing for the development of state-of-the-art models for specific tasks with minimal resource expenditure.

Q2: Does retrieval-augmented generation enhance the performance of a large language model (LLM)? Our research identified multiple studies demonstrating the advantages of retrieval-augmented generation (RAG) for specific tasks [1,3,20,22,29,30]. RAG methods enhance LLMs' performance by retrieving relevant data from diverse databases in response to user input and incorporating these data into the model's output generation. While LLMs are powerful tools, they are limited by their inability to stay updated with the latest information and by challenges when handling sensitive or classified data. Although RAG implementations improve LLMs' performance significantly, they are task-specific and do not apply to all LLM tasks. Implementing a RAG method can enhance the response accuracy and reduce the occurrence of hallucinations by dynamically retrieving and incorporating up-to-date information [1,32]. Strategies like embedding-based similarity scoring and adaptive retrieval techniques support LLMs in providing contextually accurate outputs, ensuring that the responses align with the current knowledge and user expectations [18,22].

Q3: Which methods are used to search for relevant texts in response to human queries? Our analysis reveals a range of methods for the identification of relevant data. Traditional approaches, such as statistical data extraction from texts and pattern matching, have evolved with advanced techniques like Transformer models (e.g., BERT and DistilBERT) that create vector spaces for more sophisticated text comparison [12–15]. Studies show that Transformer models outperform statistical methods by capturing textual nuances and supporting tasks like text-based and knowledge-based question answering (QA) [13]. Additionally, hybrid approaches combining TF-IDF for word importance calculation with word embeddings to create document vectors have been effective [16]. However, Transformer-based models like BERT and GPT have shown superior results in retrieving relevant texts for natural language queries by capturing the semantic and contextual relationships within data [12].

Q4: What are the best methods and metrics for the evaluation of the results of a large language model (LLM) or a document search? Our research identifies various metrics used to assess LLM and document retrieval performance. For the retrieval of relevant data from a document dataset in response to a user query, traditional metrics like the accuracy, precision, recall, and $F1$ score are commonly used [13]. However, for real-world scenarios with multiple potential answers, metrics such as the mean reciprocal rank (MRR), average precision, and mean average precision (MAP) are more suitable for the assessment of the ranking ability [13,33]. Evaluating an LLM is complex due to its multitask nature and language-format outputs. For tasks like text understanding, presenting data as a quiz provides an exact accuracy score, while, for information extraction, the BLEU, ROUGE, and $F1$-score metrics offer effective assessments [22,32]. These metrics, including the exact match (EM) for tasks requiring factual accuracy, help to provide a comprehensive evaluation of LLMs' performance [33].

Q5: Are Llama models capable of performing text understanding and information extraction tasks for retrieval-augmented generation systems? Multiple studies support the Llama model's effectiveness in performing text understanding and information extraction within RAG systems [3,28–30]. Practical applications show that Llama models can fulfill these tasks, but implementing them requires the careful consideration of factors like language and data complexity [3,30]. Integration into RAG systems necessitates planning for scalability, computational resources, and large dataset handling, ensuring that the model can maintain accuracy across applications [29].

The conclusions derived from addressing the scientific questions are summarized in Table 6.

Table 6. Summary of main ideas extracted from the answers to the questions and their references.

References	Main Idea
[11]	Fine-tuning an LLM by adjusting all weights requires substantial computational resources.
[4–7]	LoRA (Low-Rank Adaptation) allows fine-tuning by adjusting only a subset of parameters.
[4,5]	BitFit and adapters are alternative parameter-efficient fine-tuning techniques.
[3,9]	Few-shot and zero-shot learning enhance input data without changing model weights.
[1,3,20,22,29,30]	RAG enhances LLM performance by retrieving relevant external data.
[1,32]	RAG reduces hallucinations and improves response accuracy by incorporating current information.
[18,22]	Embedding-based similarity scoring and adaptive retrieval provide contextually accurate outputs in LLMs.
[12–15]	Transformer models like BERT and DistilBERT are advanced methods for retrieval of relevant data using vector spaces.
[16]	Hybrid methods combining TF-IDF and embeddings create effective document vectors for relevance.
[13,33]	Traditional and advanced metrics (accuracy, precision, recall, MRR, MAP) evaluate LLM retrieval performance.
[22,32,33]	BLEU, ROUGE, and exact match (EM) are valuable for factual accuracy in language-format outputs.
[3,28–30]	Llama models are effective for text understanding and information extraction in RAG systems.

In practical applications, fine-tuning large language models (LLMs) at scale presents significant computational challenges, especially for models with billions of parameters that require substantial memory and processing resources [4]. Such demands can result in prohibitive costs, particularly in resource-constrained environments [8]. To address these issues, pruning techniques are often employed to reduce the LLM's size by removing less critical parameters, enhancing the computational efficiency with minimal performance loss [4].

Quantization is another optimization approach, reducing the weight precision from 32-bit to 8-bit integers or lower and decreasing the memory and computational needs while maintaining acceptable accuracy levels. This technique has been especially useful in deploying LLMs on edge devices or in cloud environments with limited computational power [8]. Parameter-efficient fine-tuning, which modifies only a subset of parameters, also offers a cost-effective alternative to full retraining [11].

Another challenge is the high energy consumption associated with training and inference, particularly in large-scale applications. Efficient hardware utilization through distributed training and model parallelism can help to mitigate these costs, although it requires strategic infrastructure planning [4]. Model distillation, namely training a smaller model to replicate the behavior of a larger one, offers additional optimization. Although computationally intensive, this process can reduce the resource demands while maintaining the performance.

Overall, achieving an optimal balance in terms of model size, accuracy, and efficiency remains a critical goal in LLM development and deployment, with ongoing research needed to refine these approaches [1].

4.2. Future Directions

The implementation of a retrieval-augmented generation (RAG) system is often both time-intensive and computationally demanding, yet its benefits are substantial. Future work will focus on enhancing these systems and adapting them to low-resource languages, such as Romanian. This research will include a comparative analysis of open-source models for text understanding and information extraction tasks in Romanian, experiments in fine-tuning these models, a comparison of open-source models for feature extraction, and the fine-tuning of such models. Additionally, algorithms for data preprocessing within a RAG system will be evaluated, culminating in a comprehensive project to implement a RAG system. To address the computational costs, we also propose to explore strategies like model pruning and quantization to optimize the performance.

The current study has advanced our understanding of the most effective techniques, which will inform our future research. Our findings have identified the most common methods for the fine-tuning of models, the retrieval of relevant text from databases, and the implementation of RAG systems, alongside their advantages and disadvantages, thereby facilitating the continuation of our research endeavors.

5. Conclusions

In this critical review, we have systematically examined data analysis techniques based on generative models, with a particular emphasis on large private data sets. Our study delved into various aspects of these techniques, such as enhancing large language models (LLMs), methods of searching for key information, and approaches to retrieval-augmented generation (RAG).

This research was conducted in two main parts. The first part focused on the methodologies used to identify the most relevant papers addressing our research questions. This phase was successful, answering all posed questions and uncovering additional relevant information about the field. The second part provided a deeper understanding of the domain, facilitating future research and experiments.

Our primary objective was to answer the initial research questions posed at the start of our work, and we achieved this goal. We identified several methods to reduce the computational demands involved in fine-tuning large language models. These methods operate at different levels of the model (memory, a subset of parameters, or even the input data), allowing for simultaneous use to achieve better results. Additionally, our research revealed that some limitations of LLMs could be addressed by implementing a RAG system, which enhances the model's capabilities by providing access to new or private data and reduces the risk of hallucination.

Regarding data selection for LLM augmentation, we found that representing a document in multiple vector spaces and calculating the distance between these vectors and user input vectors is an effective method. To achieve optimal results, it is essential to conduct experiments that rank models within a specific domain. We identified specific metrics for embeddings (vectors), such as the mean reciprocal rate, which measures the ability to rank information by semantic relevance; for the LLM model, we noted metrics like the accuracy, ROUGE score, BLEU score, and $F1$ score for the evaluation of the extraction of information from text using natural language queries.

Finally, we explored the potential of the open-source LLaMA model as a starting point for our research. While promising, this model presents challenges, particularly in adapting it to the Romanian language.

Author Contributions: Conceptualization, B.M.G. and N.P.; methodology, B.M.G.; validation, N.P.; formal analysis, B.M.G. and N.P.; investigation, B.M.G.; resources, B.M.G. and N.P.; data curation, B.M.G.; writing—original draft preparation, B.M.G. and N.P.; writing—review and editing, B.M.G.; visualization, B.M.G.; supervision, B.M.G. and N.P.; project administration, B.M.G. and N.P.; funding acquisition, N.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No new data were generated in the course of this study.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

LLM	Large Language Model
LLaMA	Large Language Model Meta AI
SLR	Systematic Literature Review
RAG	Retrieval-Augmented Generation
QA	Question Answering
CFG	Context-Free Grammar
TF-IDF	Term Frequency-Inverse Document Frequency
BERT	Bidirectional Encoder Representations from Transformers
DPR	Dense Passage Retriever
LSTM	Long Short-Term Memory
GPT	Generative Pre-Trained Transformer
ELMo	Embeddings from Language Models
XLM-R	Cross-Lingual Language Model—RoBERTa
PEFT	Parameter-Efficient Fine-Tuning
LoRA	Low-Rank Adaptation
LoftQ	Low Memory Quantization and Fusion Technique
BLEU	Bilingual Evaluation Understudy
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
MAP	Mean Average Precision
MRR	Mean Reciprocal Rank
EM	Exact Match
CO-SE	COVID-19 Search Engine

References

1. Tonmoy, S.M.T.I.; Zaman, S.M.M.; Jain, V.; Rani, A.; Rawte, V.; Chadha, A.; Das, A. A Comprehensive Survey of Hallucination Mitigation Techniques in Large Language Models. *arXiv* **2024**, arXiv:2401.01313v3.
2. Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.T.; Rocktäschel, T.; et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *arXiv* **2021**, arXiv:2005.11401v4.
3. Alkhalaf, M.; Yu, P.; Yin, M.; Deng, C. Applying Generative AI with Retrieval Augmented Generation to Summarize and Extract Key Clinical Information from Electronic Health Records. *J. Biomed. Inform.* **2024**, *156*, 104662. [CrossRef] [PubMed]
4. Han, Z.; Gao, C.; Liu, J.; Zhang, J.; Zhang, S.Q. Parameter-Efficient Fine-Tuning for Large Models: A Comprehensive Survey. *arXiv* **2024**, arXiv:2403.14608v5.
5. Zhang, S.; Dong, L.; Li, X.; Zhang, S.; Sun, X.; Wang, S.; Li, J.; Hu, R.; Zhang, T.; Wu, F.; et al. Instruction Tuning for Large Language Models: A Survey. *arXiv* **2024**, arXiv:2308.10792v5.
6. Gao, D.; Ma, Y.; Liu, S.; Song, M.; Jin, L.; Jiang, W.; Wang, X.; Ning, W.; Yu, S.; Xuan, Q.; et al. FashionGPT: LLM Instruction Fine-Tuning with Multiple LoRA-Adapter Fusion. *Knowl.-Based Syst.* **2024**, *299*, 112043. [CrossRef]
7. Guthrie, E.; Levy, D.; del Carmen, G. The Operating and Anesthetic Reference Assistant (OARA): A Fine-Tuned Large Language Model for Resident Teaching. *Am. J. Surg.* **2024**, *234*, 28–34. [CrossRef] [PubMed]
8. Goswami, J.; Prajapati, K.K.; Saha, A.; Saha, A.K. Parameter-Efficient Fine-Tuning Large Language Model Approach for Hospital Discharge Paper Summarization. *Appl. Soft Comput.* **2024**, *157*, 111531. [CrossRef]
9. Latif, E.; Zhai, X. Fine-Tuning ChatGPT for Automatic Scoring. *Comput. Educ. Artif. Intell.* **2024**, *6*, 100210. [CrossRef]
10. Vidivelli, S.; Ramachandran, M.; Dharunbalaji, A. Efficiency-Driven Custom Chatbot Development: Unleashing LangChain, RAG, and Performance-Optimized LLM Fusion. *Comput. Model. Eng. Sci. (CMC)* **2024**, *80*, 2424–2442. [CrossRef]
11. Peng, C.; Yang, X.; Smith, K.E.; Yu, Z.; Chen, A.; Bian, J.; Wu, Y. Model Tuning or Prompt Tuning? A Study of Large Language Models for Clinical Concept and Relation Extraction. *J. Biomed. Inform.* **2024**, *153*, 104630. [CrossRef] [PubMed]
12. Patil, R.; Boit, S.; Gudivada, V.; Nandigam, J. A Survey of Text Representation and Embedding Techniques in NLP. *IEEE Access* **2023**, *11*, 36120–36146. [CrossRef]
13. Caballero, M. A Brief Survey of Question Answering Systems. *Int. J. Artif. Intell. Appl.* **2021**, *12*, 5. [CrossRef]
14. Raza, S. A COVID-19 Search Engine (CO-SE) with Transformer-based Architecture. *Health* **2022**, *2*, 100068. [CrossRef]
15. Saeed, N.; Ashraf, H.; Jhanjhi, N.Z. Deep Learning Based Question Answering System (Survey). *Preprints* **2023**, 20231217390. [CrossRef]
16. Seegmiller, B.; Papanikolaou, D.; Schmidt, L.D.W. Measuring Document Similarity with Weighted Averages of Word Embeddings. *Explor. Econ. Hist.* **2023**, *87*, 101494. [CrossRef]
17. Wijayanti, R.; Khodra, M.L.; Surendro, K.; Widyantoro, D.H. Learning Bilingual Word Embedding for Automatic Text Summarization in Low Resource Language. *J. King Saud Univ. Comput. Inf. Sci.* **2023**, *35*, 224–235. [CrossRef]

18. Esposito, M.; Damiano, E.; Minutolo, A.; De Pietro, G.; Fujita, H. Hybrid Query Expansion Using Lexical Resources and Word Embeddings for Sentence Retrieval in Question Answering. *Inf. Sci.* **2020**, *514*, 88–105. [CrossRef]
19. AlMahmoud, R.H.; Alian, M. The Effect of Clustering Algorithms on Question Answering. *Expert Syst. Appl.* **2024**, *243*, 122959. [CrossRef]
20. Li, M.; Peng, B.; Galley, M.; Gao, J.; Zhang, Z. Self-Checker: Plug-and-Play Modules for Fact-Checking with Large Language Models. *arXiv* **2024**, arXiv:2305.14623.
21. Vu, T.; Iyyer, M.; Wang, X.; Constant, N.; Wei, J.; Wei, J.; Tar, C.; Sung, Y.H.; Zhou, D.; Le, Q.; et al. FreshLLMs: Refreshing Large Language Models with Search Engine Augmentation. *arXiv* **2023**, arXiv:2310.03214.
22. Gao, Y.; Xiong, Y.; Gao, X.; Jia, K.; Pan, J.; Bi, Y.; Dai, Y.; Sun, J.; Wang, M.; Wang, H. Retrieval-Augmented Generation for Large Language Models: A Survey. *arXiv* **2024**, arXiv:2312.10997v5.
23. Varshney, N.; Yao, W.; Zhang, H.; Chen, J.; Yu, D. A Stitch in Time Saves Nine: Detecting and Mitigating Hallucinations of LLMs by Validating Low-Confidence Generation. *arXiv* **2023**, arXiv:2307.03987.
24. Cao, H.; An, Z.; Feng, J.; Xu, K.; Chen, L.; Zhao, D. A Step Closer to Comprehensive Answers: Constrained Multi-Stage Question Decomposition with Large Language Models. *arXiv* **2023**, arXiv:2311.07491.
25. Kang, H.; Ni, J.; Yao, H. Ever: Mitigating Hallucination in Large Language Models through Real-Time Verification and Rectification. *arXiv* **2024**, arXiv:2311.09114.
26. Gao, L.; Dai, Z.; Pasupat, P.; Chen, A.; Chaganty, A.T.; Fan, Y.; Zhao, V.Y.; Lao, N.; Lee, H.; Juan, D.C.; et al. RARR: Researching and Revising What Language Models Say, Using Language Models. *arXiv* **2023**, arXiv:2210.08726.
27. Rawte, V.; Chakraborty, S.; Pathak, A.; Sarkar, A.; Tonmoy, S.M.T.I.; Chadha, A.; Sheth, A.P.; Das, A. The Troubling Emergence of Hallucination in Large Language Models—An Extensive Definition, Quantification, and Prescriptive Remediations. *arXiv* **2023**, arXiv:2310.04988.
28. Kim, E.; Yoon, H.; Lee, J.; Kim, M. Accurate and Prompt Answering Framework Based on Customer Reviews and Question-Answer Pairs. *Expert Syst. Appl.* **2022**, *203*, 117405. [CrossRef]
29. Gebreab, S.A.; Salah, K.; Jayaraman, R.; ur Rehman, M.H.; Ellaham, S. LLM-Based Framework for Administrative Task Automation in Healthcare. In Proceedings of the 2024 International Symposium on Digital Forensics and Security (ISDFS), San Antonio, TX, USA, 29–30 April 2024. [CrossRef]
30. Kirubakaran, S.; Kathrine, J.W.; Kanaga, G.M.; Raja, M.; Singh, R.G.; Yuvaraajan, E. A RAG-Based Medical Assistant Especially for Infectious Diseases. In Proceedings of the 2024 International Conference on Innovations in Information Technology (ICICT), Lalitpur, Nepal, 24–26 April 2024. [CrossRef]
31. Arslan, M.; Mahdjoubi, L.; Munawar, S. Driving Sustainable Energy Transitions with a Multi-Source RAG-LLM System. *Energy Build.* **2024**, *324*, 114827. [CrossRef]
32. Rasool, Z.; Kurniawan, S.; Balugo, S.; Barnett, S.; Vasa, R.; Chesser, C.; Hampstead, B.M.; Belleville, S.; Mouzakis, K.; Bahar-Fuchs, A. Evaluating LLMs on Document-Based QA: Exact Answer Selection and Numerical Extraction Using CogTale Dataset. *Nat. Lang. Process. J.* **2024**, *8*, 100083. [CrossRef]
33. Yuan, M.; Bao, P.; Yuan, J.; Shen, Y.; Chen, Z.; Xie, Y.; Zhao, J.; Lih, Q.; Chen, Y.; Zhang, L.; et al. Large Language Models Illuminate a Progressive Pathway to Artificial Intelligent Healthcare Assistant. *Med. Phys.* **2024**, *1*, 100030. [CrossRef]