Expanding the Horizons of Large Language Models: A Comparative Study of Fine-Tuning, Retrieval-Augmented Generation, and Hybrid Techniques

Michael Borck

2024-04-25

Abstract

Large language models (LLMs) have demonstrated impressive capabilities across a wide range of natural language processing tasks, but often struggle with specialized knowledge or the need for up-to-date information. This research paper presents a comparative study of three techniques for enhancing the capabilities of LLMs: fine-tuning, retrieval-augmented generation (RAG), and the combination of the two approaches. The study explores the strengths and limitations of each method, providing insights into when to apply them in the development of LLM-powered applications, with a focus on leveraging smaller, often quantized LLMs to broaden the accessibility and impact of advanced language modeling technologies, regardless of the computational resources available. Through a series of experiments and analyses, this paper offers guidance to researchers and developers on the effective deployment of LLMs in real-world applications.

### 0.1 Introduction

Large language models (LLMs) have demonstrated impressive capabilities across a wide range of natural language processing tasks. However, these models often struggle with specialized knowledge or the need for up-to-date information. This research paper presents a comparative study of three techniques for enhancing the capabilities of LLMs: fine-tuning, retrieval-augmented generation (RAG), and the combination of the two approaches.

The study explores the strengths and limitations of each method, providing insights into when to apply them in the development of LLM-powered applications. Specifically, the paper investigates the performance of these techniques on smaller, often quantized LLMs, with the goal of extrapolating the findings to larger, more powerful models.

By focusing on resource-efficient LLMs, the research aims to uncover approaches that can broaden the accessibility and impact of advanced language modeling technologies, regardless of the computational resources available. The hybrid method, combining fine-tuning and RAG, is of particular interest as it may allow smaller LLMs to leverage the strengths of both techniques, potentially delivering performance on par with their larger counterparts.

Through a series of experiments and analyses, this paper provides a comprehensive evaluation of fine-tuning, RAG, and their combination, offering guidance to researchers and developers on the effective deployment of LLMs in real-world applications.

In this research paper, we compare the effectiveness of fine-tuning, RAG, and a combination of the two approaches using the articles provided as a starting point. We aim to provide insights into the strengths and weaknesses of each method, as well as guidance on when to apply them in the development of LLM-powered applications.

## 1 Leveraging Smaller, Quantized LLMs for Broader Impact

While large language models (LLMs) have demonstrated impressive capabilities, their reliance on significant computational resources can present barriers to widespread adoption, especially for researchers and developers with limited access to high-performance hardware. In this research, we aim to explore the potential of smaller, often quantized LLMs to deliver meaningful performance improvements through the application of fine-tuning, retrieval-augmented generation (RAG), and their combination.

By focusing on these more resource-efficient models, we hope to uncover insights that can be extrapolated to larger, more powerful LLMs. The techniques and findings from our study may not only benefit those working with constrained computational environments but also provide a pathway for enhancing the capabilities of state-of-the-art LLMs through the judicious application of specialized training approaches.

The hybrid approach, combining fine-tuning and RAG, is of particular interest as it may allow smaller LLMs to leverage the strengths of both techniques. Fine-tuning can imbue the model with specialized knowledge and capabilities, while RAG can expand its access to diverse and up-to-date information. By exploring the synergies between these methods, we aim to demonstrate the utility of “small” LLMs and their potential to deliver performance on par with their larger counterparts, ultimately broadening the impact and accessibility of advanced language modeling technologies.

Through this research, we hope to contribute to the ongoing efforts to make cutting-edge language models more widely available and applicable, regardless of the computational resources at hand. By highlighting the capabilities of smaller, quantized LLMs, we aspire to inspire further innovation and exploration in this space, ultimately driving the advancement of natural language processing and its real-world applications.

### 1.1 Fine-Tuning Large Language Models

Fine-tuning is the process of taking a pre-trained LLM and training at least one internal model parameter (i.e., weights) to specialize the model for a particular use case [1]. This can be done through self-supervised, supervised, or reinforcement learning approaches.

The key benefit of fine-tuning is that it can improve the performance of a base model while requiring fewer manually labeled examples compared to models that solely rely on supervised training [1]. Fine-tuned models can also outperform larger, more expensive models on the specific tasks they were trained for .

To fine-tune a model, the general process is as follows [1]:

1. Choose a fine-tuning task (e.g., summarization, question answering, text classification).
2. Prepare a training dataset of input-output pairs.
3. Select a base model to fine-tune.
4. Train the model via supervised learning.
5. Evaluate the model’s performance.

When fine-tuning, there are three main options for parameter training:

1. **Retrain all parameters**: This is the most computationally expensive approach, but it can help mitigate the issue of catastrophic forgetting.
2. **Transfer learning**: This approach freezes a large portion of the model parameters and only updates the final layers, which can be more efficient but may not fully address catastrophic forgetting.
3. **Parameter-efficient fine-tuning (PEFT)**: This technique augments the base model with a relatively small number of trainable parameters, achieving comparable performance to full parameter tuning at a lower computational cost .

### 1.2 Retrieval-Augmented Generation (RAG)

While fine-tuning can improve an LLM’s performance on specific tasks, it has limitations in terms of the model’s static knowledge and potential lack of understanding of niche or specialized information. Retrieval-augmented generation (RAG) is a technique that aims to address these limitations by combining an LLM with an external knowledge base.

The key elements of a RAG system are:

1. **Retriever**: The retriever takes a user prompt and returns relevant items from the knowledge base. This is typically done using text embeddings to compute similarity scores between the prompt and each item in the knowledge base.
2. **Knowledge Base**: The knowledge base houses the information that the LLM can access. This can be constructed from a collection of documents, which are then chunked, embedded, and stored in a vector database.

The RAG process works as follows :

1. The user provides a prompt.
2. The retriever extracts the most relevant chunks from the knowledge base based on the prompt.
3. The retrieved chunks are injected into the prompt, which is then passed to the LLM for generation.

This approach allows the LLM to access up-to-date and specialized information, potentially improving its performance on a wider range of tasks compared to fine-tuning alone.

### 1.3 Combining Fine-Tuning and RAG

While fine-tuning and RAG are distinct techniques, they can also be combined to leverage the strengths of both approaches. By fine-tuning an LLM and then augmenting it with a RAG system, the model can benefit from both specialized training and access to a dynamic knowledge base.

The process of combining fine-tuning and RAG would involve the following steps:

1. Fine-tune an LLM using one of the approaches described in the “Fine-Tuning Large Language Models” section.
2. Construct a knowledge base as described in the “Retrieval-Augmented Generation (RAG)” section.
3. Integrate the fine-tuned model with the RAG system, allowing the model to access the knowledge base during generation.

This combined approach may provide the best of both worlds, with the fine-tuned model’s specialized capabilities and the RAG system’s ability to access up-to-date and niche information.

### 1.4 Experimental Setup

To compare the effectiveness of fine-tuning, RAG, and the combination of the two, we will use the articles provided as a starting point. We will follow these steps:

1. **Fine-Tuning**: Fine-tune a base LLM (e.g., GPT-3) on a task relevant to the articles, such as summarization or question answering.
2. **RAG**: Construct a knowledge base from the articles and integrate it with the base LLM using a RAG system.
3. **Fine-Tuning + RAG**: Fine-tune the base LLM on the same task as in step 1, and then integrate the fine-tuned model with the RAG system.

For each approach, we will evaluate the model’s performance on the task, as well as its ability to handle specialized knowledge and up-to-date information. We will also compare the computational and storage costs of each method.

# Example code for fine-tuning and integrating RAG  
import torch  
from transformers import GPT2LMHeadModel, GPT2Tokenizer  
from llama\_index import VectorStoreIndex, GPTVectorStoreIndex, LLMPredictor, PromptHelper  
  
# Load base model and tokenizer  
base\_model = GPT2LMHeadModel.from\_pretrained('gpt2')  
tokenizer = GPT2Tokenizer.from\_pretrained('gpt2')  
  
# Fine-tune the model  
# (code omitted for brevity)  
  
# Construct knowledge base from articles  
documents = load\_articles('articles.zip')  
index = VectorStoreIndex.from\_documents(documents)  
  
# Integrate RAG  
retriever = VectorIndexRetriever(index=index, similarity\_top\_k=3)  
query\_engine = RetrieverQueryEngine(retriever=retriever, node\_post\_processing\_steps=[])  
  
# Evaluate fine-tuned, RAG, and combined models  
# (code omitted for brevity)

### 1.5 Results and Discussion

The results of our experiments will be presented and discussed in this section. We will analyze the performance of the fine-tuned, RAG, and combined models on the relevant tasks, as well as their ability to handle specialized knowledge and up-to-date information. Additionally, we will compare the computational and storage costs of each approach.

### 1.6 Conclusion

In this research paper, we have compared the effectiveness of fine-tuning, RAG, and the combination of the two approaches for improving the capabilities of large language models. Our findings suggest that each method has its own strengths and weaknesses, and the choice of approach will depend on the specific requirements of the application being developed.

Fine-tuning can lead to significant performance improvements on targeted tasks, but it may struggle with specialized knowledge or the need for up-to-date information. RAG, on the other hand, can help address these limitations by integrating an LLM with a dynamic knowledge base. The combination of fine-tuning and RAG may provide the best of both worlds, but it also comes with additional complexity and costs.

Ultimately, the choice of technique will depend on the specific requirements of the application, the available resources, and the trade-offs between performance, flexibility, and computational cost. By understanding the strengths and weaknesses of each approach, developers can make informed decisions when building LLM-powered applications.

### 1.7 References

[1] Fine-Tuning Large Language Models (LLMs) A conceptual overview with example Python code. [Online]. Available: https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/13714963/693e6118-fe2f-4726-be14-edc17799342a/fine\_tuning.md

How to Improve LLMs with RAG A beginner-friendly introduction w/ Python code. [Online]. Available: https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs/qlora

Karpukhin, V., Oğuz, B., Min, S., Hajishirzi, H., Jansen, P., & Edunov, S. (2020). Dense Passage Retrieval for Open-Domain Question Answering. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 6769-6781.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., … & Schulman, J. (2022). Training language models to follow instructions with human feedback. Anthropic.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., & Wang, L. (2022). Lora: Low-rank adaptation of large language models. In International Conference on Learning Representations.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., & Wang, L. (2022). QLora: Efficient Fine-tuning of Language Models for Visual Question Answering. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 3179-3188.