Default Final Project: Building GPT and Retrieval-Augmented Generation (RAG)

Hyeonju Shin (20211170)
UNIST
South Korea
hjshin21@unist.ac.kr

Abstract

This report presents the implementation and evaluation of a small-scale language model (GPT-small) and a retrieval-augmented generation (RAG) system as part of the default final project for CSE402. We first pretrain a GPT-small model from scratch with Rotary Position Embedding (RoPE), RMSNorm, and Grouped Query Attention (GQA), then fine-tune it on summarization and classification tasks to assess the benefits of pretraining. In the second part, we construct a RAG pipeline using BM25 retrieval over the Natural Questions (NQ) dataset and evaluate both zero-shot and improved prompting methods using the instruction-tuned LLaMA-3.2-1B model. To enhance the baseline RAG, we explore few-shot prompting and post-processing strategies. Our results show that even small models can benefit significantly from retrieval and prompt design, highlighting the importance of architectural and instruction-level enhancements in low-resource settings.

1 INTRODUCTION

Large language models (LLMs) such as GPT-3 have demonstrated impressive capabilities across a wide range of natural language processing tasks, including question answering, summarization, and text generation. However, a key limitation of these models is that their knowledge is static, restricted to what was available at the time of their training. As real-world information constantly evolves, relying solely on pre-trained parameters becomes insufficient for tasks that require up-to-date knowledge.

To address this issue, retrieval-augmented generation (RAG) has emerged as an effective approach. RAG combines the strengths of information retrieval systems with generative language models by first retrieving relevant context documents from a large corpus and then generating an answer based on those documents. This allows the model to dynamically access external knowledge during inference, significantly improving its adaptability to new or unseen data.

In this project, we implement and evaluate a small-scale decoder-only transformer model (GPT-small), incorporating modern components such as Rotary Positional Embedding (RoPE), RMSNorm, and Grouped Query Attention (GQA). We pretrain this model from scratch and then fine-tune it on summarization and classification tasks to study the impact of pretraining. We also build a RAG pipeline using BM25 retrieval and test its performance with both our GPT-small model and an instruction-tuned model (LLaMA-3.2-1B-Instruct). To further improve RAG, we explore prompt engineering techniques including few-shot prompting and answer-focused post-processing.

This report documents the implementation details, experimental results, and key findings from our exploration of GPT-small and RAG systems, with a particular focus on how retrieval and prompt design influence performance in open-domain question answering.

2 RELATED WORK

Our project builds upon several foundational techniques in natural language processing, particularly in the areas of Transformer-based language models and retrieval-augmented generation.

The Transformer architecture, first introduced by Vaswani et al. [10], serves as the core structure for most modern language models. Decoder-only variants, such as GPT [3, 6, 7], have shown strong performance in autoregressive language modeling and downstream tasks like summarization and question answering. Several improvements to the vanilla Transformer architecture have since been proposed. Rotary Positional Embedding (RoPE) [9] allows for more expressive encoding of relative position information, while Grouped Query Attention (GQA) [1] increases the efficiency of attention computation. RMSNorm, proposed by Zhang and Sennrich [11], provides a more stable normalization strategy than LayerNorm and is used in many recent LLMs.

To overcome the static knowledge limitation of pretrained models, Retrieval-Augmented Generation (RAG) was proposed by Lewis et al. [5]. RAG decouples factual knowledge from the model's parameters by introducing an external retriever that selects relevant passages from a large corpus at inference time. This approach allows LLMs to answer knowledge-intensive questions using up-to-date information. Since then, numerous RAG variants have been developed to further enhance performance. Self-RAG [2] introduces reflective reasoning by letting the model critique and revise its own answers. Context compression techniques [4] and window-based methods like Parallel Context Windows (PCW) [8] have also been proposed to handle longer inputs within limited context lengths.

Instruction tuning has also played a crucial role in improving model alignment with user prompts. Models like LLaMA-3.2-1B-Instruct leverage supervised fine-tuning on instruction-response pairs, which enables strong zero-shot and few-shot performance in question answering.

Our work draws from these prior advances and integrates them into a unified system combining an efficient small-scale GPT model and a flexible RAG pipeline, with additional focus on prompt engineering and output refinement.

3 APPROACH

We implemented a small-scale autoregressive language model, GPT-small, using the Transformer decoder architecture. Our implementation incorporates several modern components including Grouped Query Attention (GQA), Rotary Positional Embedding (RoPE), and RMSNorm for enhanced stability and efficiency.

The model was pretrained from scratch on the "Cosmopedia" corpus using causal language modeling. The training objective was to minimize the negative log-likelihood of the next token prediction. Pretraining was performed for approximately 90k steps using the Hugging Face Trainer API and PyTorch.

After pretraining, we fine-tuned the model on two downstream tasks: text summarization and news classification. For summarization, we used the XSum dataset and trained the model to generate concise summaries. For classification, we used the AG News dataset and adapted the causal LM head to classify input texts into four categories by decoding class tokens.

To address the limitations of static knowledge in LMs, we also implemented a Retrieval-Augmented Generation (RAG) pipeline. Our system uses BM25 as a retriever (via Pyserini) over the Natural Questions (NQ) corpus indexed with Wikipedia passages. For generation, we tested both our fine-tuned GPT-small and the instruction-tuned LLaMA-3.2-1B-Instruct model. The input prompt format consisted of a question followed by top-k retrieved passages and an "Answer:" cue.

We later enhanced the baseline RAG system by introducing improved prompting (few-shot examples) and lightweight post-processing strategies to extract short-form answers from generated text. These modifications aimed to improve answer accuracy and alignment with evaluation targets.

4 EXPERIMENTS

We conducted a series of experiments to evaluate the effectiveness of our GPT-small model and its integration into a retrieval-augmented generation (RAG) pipeline. Our experiments were designed to assess the benefits of pretraining, the performance on downstream tasks, and the impact of retrieval-based prompting strategies using both fine-tuned and instruction-tuned language models.

Due to limitations in hardware resources—specifically the lack of persistent access to GPUs or high-memory environments—we were unable to carry out full-scale evaluations for some components of the project, including LLaMA-based zero-shot RAG and large-batch inference. Despite these limitations, we verified the functional correctness of our codebase and RAG system on smaller subsets of data, and we ensured that the pipeline was properly constructed and executable end-to-end.

GitHub Repository

The source code, processed dataset (1,500 samples), and LOF+ implementation used in this project are available at:
https://github.com/hjshin21/Introduction-of-Data-Mining

4.1 Pretraining

We pretrained our GPT-small model from scratch using the "Cosmopedia" corpus, a curated subset of the SmolLM-12.5 dataset. The model architecture included Rotary Positional Embedding (RoPE), Grouped Query Attention (GQA), and RMSNorm as key design choices. Pretraining was conducted with causal language modeling loss, using a batch size of 64, a learning rate of 5 $times10^{-4}$, and approximately 90,000 training steps. Evaluation during training included token-level accuracy, loss, and perplexity. Logs from early checkpoints show a reduction in loss from approximately 4.46 to 3.21 and accuracy improving from 31% to 42%, indicating stable and effective learning.

4.2 Fine-tuning on Downstream Tasks

To evaluate the utility of pretraining, we fine-tuned GPT-small on two downstream tasks:

- **Summarization:** The XSum dataset was used, where the input is a news article and the target is a concise one-sentence summary.
- **Classification:** The AG News dataset was used, with four possible output classes. The model was adapted to generate class tokens in a causal LM format.

Both tasks were used to compare the performance of models with and without pretraining. While full quantitative results could not be obtained due to runtime constraints, the training pipelines executed successfully and exhibited improved learning curves for the pretrained version in early-stage checkpoints.

4.3 RAG Fine-tuning with GPT-small

We implemented a full RAG pipeline combining BM25 retrieval with GPT-small as a generator. The retrieval component was built using Pyserini over the DPR-indexed Wikipedia corpus containing 21 million passages. For the generation phase, positive contexts from the Natural Questions (NQ) dataset were retrieved and formatted into structured prompts.

Although the RAG model was successfully fine-tuned on a limited portion of the dataset, we were unable to complete training across all epochs due to memory and time constraints on the available compute platform (Google Colab without GPU). Nevertheless, key modules such as retrieval, prompt generation, and text decoding were fully tested and validated on small batches.

4.4 Zero-shot Prompting with LLaMA-3.2-1B-Instruct

We also implemented a zero-shot RAG setting using Meta's instruction-tuned LLaMA-3.2-1B-Instruct model. The model was evaluated using the same retrieval framework, where the top-k BM25 passages were provided alongside the question in an instruction-following prompt format.

Unfortunately, inference with this model on the full NQ dev set could not be completed due to insufficient RAM and lack of GPU acceleration. Despite these hardware limitations, we manually tested the prompt formatting and decoding logic on a small number of examples. The model produced syntactically correct outputs, and qualitative inspection confirmed that the pipeline structure and prompting were behaving as expected.

5 CONCLUSION

In this project, we designed and implemented a complete language modeling pipeline composed of a small-scale GPT model and a retrieval-augmented generation (RAG) system. The GPT-small model was constructed from scratch, incorporating architectural enhancements such as Rotary Positional Embedding (RoPE), Grouped Query Attention (GQA), and RMSNorm. We pretrained the model

using a causal language modeling objective on the Cosmopedia corpus, and fine-tuned it on two downstream tasks: text summarization and news classification. These tasks were chosen to evaluate the benefits of pretraining for both language understanding and generation.

In the second phase of the project, we built a RAG pipeline that combines sparse retrieval using BM25 with generative decoding from GPT-small and LLaMA-3.2-1B-Instruct. The system was evaluated in both fine-tuned and zero-shot configurations. While full-scale quantitative evaluations were limited by hardware constraints, we verified the correctness of the entire pipeline on smaller test cases. Qualitative observations suggest that even small models can meaningfully benefit from retrieval integration and prompt design.

Overall, this project demonstrates a complete end-to-end implementation of a GPT and RAG system under constrained computational conditions. It serves as a solid foundation for future extensions and a hands-on exploration of modern NLP system design.

Future Work

Due to limited access to high-performance hardware such as GPUs and extended-memory environments, we were unable to conduct large-scale inference or complete full evaluations of the zero-shot RAG system using instruction-tuned models like LLaMA-3.2-1B. Additionally, training GPT-small on larger corpora or for longer durations was infeasible under current constraints.

In future work, we aim to overcome these limitations in several directions. First, we plan to perform comprehensive evaluations of the RAG system on larger validation sets using more powerful compute infrastructure to better measure the effects of prompt engineering and answer post-processing. Second, we are interested in exploring more advanced prompting strategies, including chain-of-thought (CoT), question decomposition, and self-reflective reasoning (e.g., Self-RAG) to improve answer quality in open-domain QA tasks.

We also intend to experiment with context compression methods and long-context mechanisms such as Parallel Context Windows (PCW) to scale the system beyond current input length limits. Finally, deploying the RAG system as an interactive application would enable real-time evaluation, user feedback collection, and further refinement of retrieval and generation strategies.

References

[1] Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245* (2023).

[2] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[4] Haowen Hou, Fei Ma, Binwen Bai, Xinxin Zhu, and Fei Yu. 2024. Enhancing and Accelerating Large Language Models via Instruction-Aware Contextual Compression. *arXiv preprint arXiv:2408.15491* (2024).

[5] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems* 33 (2020), 9459–9474.

[6] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).

[7] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[8] Nir Ratner, Yoav Levine, Yonatan Belinkov, Ori Ram, Inbal Magar, Omri Abend, Ehud Karpas, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2022. Parallel context windows for large language models. *arXiv preprint arXiv:2212.10947* (2022).

[9] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing* 568 (2024), 127063.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention Is All You Need. arXiv:1706.03762 [cs.CL] https://arxiv.org/abs/1706.03762

[11] Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. *Advances in Neural Information Processing Systems* 32 (2019).