RAG Based QA for Low Resource Languages

Berhanu Bogale

berhanubogale0101@gmail.com

Bahir Dar university Institute of Technology

Tesfa Tegegne

Bahir Dar university Institute of Technology

Solomon Teferra

Addis Ababa University

Gebeyehu Belay

Bahir Dar university Institute of Technology

Research Article

Keywords: LLM, Question Answering, RAG, LORA

Posted Date: November 11th, 2024

DOI: https://doi.org/10.21203/rs.3.rs-5360450/v1

Additional Declarations: No competing interests reported.

RAG Based QA for Low Resource Languages

Berhanu Bogale[1,2], Tesfa Tegegne[2], Solomon Abera[3], and Gebeyehu Belay[2]

[1]Departement of computer science, BIT,Bahir Dar university
[2]Department of Computer science,Gafat institute of technology,Debere Tabore university
[3]Department of computer science,addis abeba university

July 3, 2024

Abstract

Abstract—Question Answering (QA) has been an important research direction in Natural Language Processing (NLP) and artificial intelligence. The majority of current large language models (LLMs) that have been fine-tuned concentrate on improving performance on different NLP tasks including question answering with new dataset since the current model does not give accurate results. To adapt the LLM to a specific domain the fine-tuned method have a great impact. However, fine-tuning have a limitation of labeled data. To address such problem we use RAG with LLMs on question answering NLP tasks using different documents. We use Samuael/llama-2-7b-tebot-amharic a fine-tuned LLM for Question answering tasks including RAG techniques. We use publicly available autoregressive language models Samuael/llama-2-7b-tebot-amharic from hugging face as a base model. We use RAG because it helps to augment the knowledge of different documents such as text, doc, xml, html, pdf with large language model. We also fine-tune with LORA method on Amharic (AmharicInstructiondataset) dataset from the hugging face having a collection of more than 100000 records in different domains. Fine-tuning in AI is the process of adjusting the weights and parameters of a pre-trained model on new data to improve its performance on a specific task[2]. Experimental results on 50 test sets for named entity recognition, question answering tasks achieves superior performance compared to general LLMs. We termed a fine-tuned version of Samuael/llama-2-7b-tebot-amharic as llama-2-AmLLM that is optimized for question answering. After fine-tuning the model achieve a BLEU score of 0.4432 on the given test set, significantly exceeding previous state of the art for this task.

Keywords: LLM,Question Answering,RAG,LORA

---

1 Introduction

Transformer is a deep learning model based on an attention mechanism for processing sequence data that can effectively solve complex natural language processing problems. This model was first proposed in 2017 [6], and replaced the traditional recurrent neural network architecture [10] in machine translation tasks as the state-of-the-art model at that time. Due to its suitability for parallel computing and the complexity of the model itself, Transformer outperforms the previously popular recurrent neural networks in terms of accuracy and performance. The Transformer architecture consists primarily of two modules, an Encoder and a Decoder, as well as the attention mechanism within these modules. Now,LLMs are a type of artificial intelli-

gence system that can process and generate natural language texts. They are trained on massive amounts of text data, such as books, articles, web pages, social media posts, and more, using deep neural networks[9]. In 2023, Large Language Models (LLMs) like GPT-4 have become integral to various industries, with companies adopting models such as ChatGPT, Claude, and Cohere to power their applications. Now, businesses are increasingly fine-tuning these foundation models to ensure accuracy and task-specific adaptability. LLMs can learn the patterns and structures of natural language from the data, and use them to perform various tasks, such as answering questions, summarizing texts, translating languages, and writing essays[13].

Despite their remarkable capabilities, LLMs are coming with different limitations. Concerns regarding potential biases embedded within training data, factual inaccuracies, and the absence of ethical consideration. Also, they lack explainability and transparency like other deep learning models. This makes it nearly impossible to understand how they generate responses and how the LLM works. Some of the limitation of LLMs include presenting false information when they do not have the answer,sometimes it contains fictitious information[4] presenting out-of-date or generic information, and creating a response from non-authoritative sources in train data.The following figure shows how the system prompts when the low resourced language is prompted to llama2 model. Amharic is the low resourced and the second most widely-spoken Semitic language (Ethio-Semitic language) after Arabic[12] . currently, It is the working language of the FDRE(Federal Democratic Republic of Ethiopia) . It is also the working language of many regional states of Ethiopia like Amhara, Addis Abeba,South Nations and Nationalities, Benishangul Gumuz, and Gambella. The lan- guage has a considerable number of speakers in all regional states of the country[12]. It is very morphologically-rich language that has its own characterizing like phonetic, phonological, and morphological properties[12]. It is a low-resource language without well-developed natural language processing applications and resources such as question answering, named entity , word disambiguation ,document summarization etc.

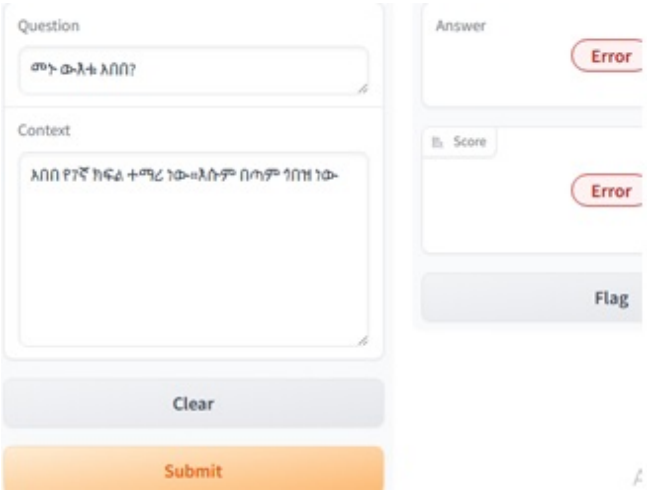

Figure 1: Error message when the context vocabulary is new

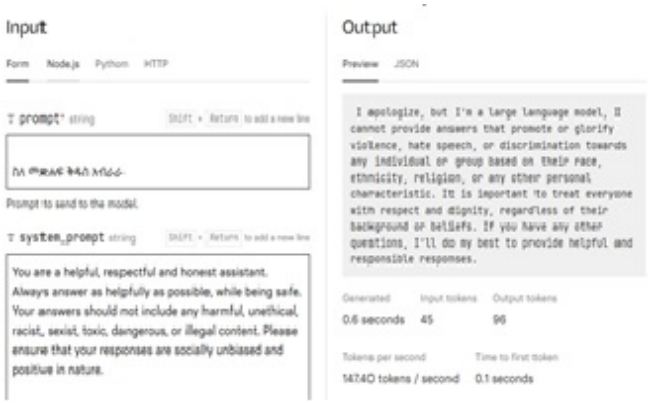

Figure 2: Error results when the Amharic query is given to llama2

As shown above the system responded apologize message in the first figure and the error message.in the second figure. In order to overcome these obstacles, RAG reroutes the LLM to obtain essential data from reliable, pre-selected knowledge sources. In this manner, LLM can make advantage of the context provided by the acquired data to produce a response that is more accurate, relevant, and helpful in different settings. By disclosing the sources of the collected data, RAG applications may increase user transparency by illuminating the process by which the LLM produces its results[3]. Some other techniques include fine tuning to solve the limitations of LLM. Fine-tuning in large language models (LLMs) involves re-training pre-trained models on specific datasets, allowing the model to adapt to the specific context of your domain. This process can help you create highly accurate language models, tailored to your specific business use cases. Fine-tuning involves updating the weights of a pre-trained language model

on a new task and dataset. Fine-tuning is most effective when we have a small dataset and the pre-trained model is already trained on a similar task or domain[7]. Here are some important tools and techniques for fine-tuning Large Language Models (LLMs):-

- **Hugging Face Library:-** provides pre-trained models and utilities for fine-tuning them on your specific task
- **DeepSpeed:-** It can accelerate fine-tuning, especially for large language models (LLM).
- **PyTorch:-** PyTorch is a widely used open-source machine learning library. we use PyTorch to fine tune a large language model like GPT.
- **Databricks:-** A platform that provides cloud-based big data processing using Apache Spark. It can be used to fine tune large language models.
- **Simform's Guide:** Simform provides a comprehensive guide on fine-tuning large language models, covering fundamentals, training data methodologies, strategies, and best practices.

We clearly show the process of fine-tuning Samuael/llama-2-7b-tebot-amharic with LoRA for question Answering tasks. Samuael/llama-2-7b-tebot-amharic is a LlaMA fine-tuned model with Amharic vocabulary. Llama is a family of large language models a collection of pre-trained and fine-tuned large language models (LLMs) ranging from 7 billion to 70 billion parameters[11]. Trained on a massive 2 trillion tokens with an increased context length of 4K and the model can grasp and generate extensive content. Llama 2 models have consistently outperformed their predecessors. Why llama2? Because it is openly accessed in contrast to pretrained LLMs such as BLOOM[5], LLaMa-1[11], and Falcon[1] that match the performance of closed pretrained competitors like GPT-3[13] and Chinchilla[8], but none of these models are suitable substitutes for low resource langauges. These closed product LLMs are heavily fine-tuned to align with human preferences, which greatly enhances their usability and safety.

The remainder of this paper describes our objective (Section 2), fine-tuning methodology (Section 3), Experimental setups (Section 4), relevant related work (Section 5), and conclusions (Section 6).

2 OBJECTIVE OF THE STUDY

- Apply LORA to fine tune the LLM model.
- Efficiently fine-tune for Llama-v2-7b type mode on a Single GPU.
- Compute the answer(s) to a given question from a large set of documents using RAG

3 Contribution of the study

- review the existing LLMs
- we analyze the use of large language models in a curated Amharic datasets.
- we fine-tune Amharic Question Answering LLM using LORA.
- we prepare RAG with LLM Amharic Auestion Answering

4 Statements of the problem

The availability of different pre-trained large language models has enabled the quick development of deep learning components for downstream applications. However, even if texts are abundant for low-resource languages, there are very few large language models and resources available online. Some of the publicly available pre-trained models are usually built as a multilingual version of semantic models that will not fit well with the need for low-resource languages. We fine tune with LORA for efficient memory and computing consumption and use RAG models for Amharic Question Answering. After we investigate the publicly available pre-trained large language models, we fine-tune the pre-trained models and use RAG with LLM for better response from different documents.

5 methodology

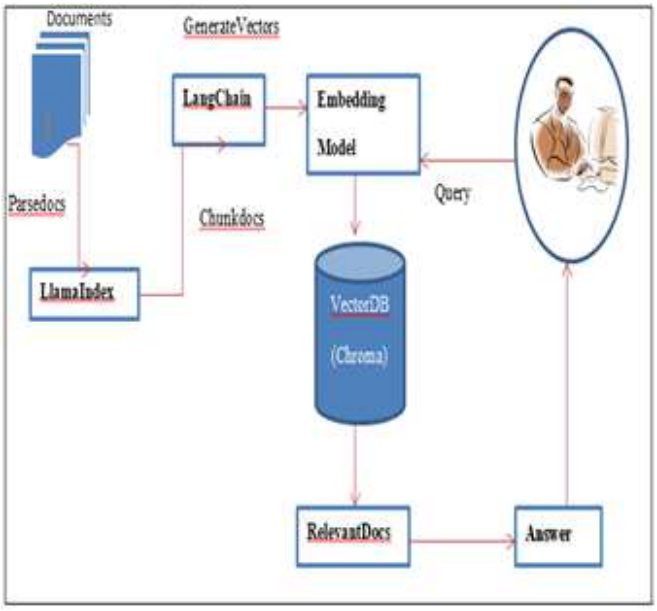

Figure 3: Architechure of RAG+LLM+QA with chroma vector database

There are significant benefits to using a pre-trained model. There are techniques used to adapt a new language to large language model. In this paper, we use the two most common adaptation techniques. These are fine tuning a large language model using a parameter efficient fine-tuning method specifically LORA. The technique reduces computation costs, your carbon footprint, and allows you to use the models without having to train one from scratch. Secondly, we use RAG with LLM for question answering system for augmenting additional knowledge from different documents such as pdf, text, html and etc. Retrieval-Augmented Generation (RAG) is a new approach that leverages Large Language Models (LLMs) to automate knowledge search, synthesis, extraction, and planning from unstructured data sources. The pre-train and fine-tune paradigm replaced fully supervised learning as the primary learning strategy for NLP models between 2017 and 2019. In this paradigm, the likelihood of observed text data is predicted by pre-training a model with a fixed architecture as a language model[2]. These language models can be trained on huge datasets because a lot of raw text data is needed to train them. Subsequently, language models are capable of acquiring strong, universal aspects of the language they represent. The above PLM will then adjust to various downstream tasks by adding more parameters and fine-tuning them using task-specific goal functions. Research then turned its attention to the establishment of training objectives for pre-training and fine-tuning, or objective engineering[2]. This paper uses Jupyter notebooks which is a perfect tool for learning how to work with LLM systems because it guides in an interactive environment which is a great mechanism to better understand the errors occurred.

As a first step, the input text is typically chunked into smaller pieces in the document ingestion stage. A basic pre-processing step in RAG is converting these chunks into embeddings using a specific embedding model. A typical sequence window for an embedding model is 512 tokens, which limits a practical target for chunk size. Once the documents are chunked and encoded into embedding's, a similarity search using the query embedding's can be performed to build the context for generating the answer. We use the following RAG components

- **Data ingestion:** At the first stage, data is ingested from various sources, such as text documents, websites, or databases. This data can be in a raw or preprocessed format.
- **Document processing:** This include tasks like tokenization, stemming, and removing stop words.
- **Embedding Data:** The basic preprocessing step in RAG is converting these chunks into embedding's using a specific embedding model. Each piece of data is converted into a numerical representation called an embedding. This embedding captures semantic information about the data, making it easier for the LLM to understand and process.
- **Vector Database:** The embedding's are stored in a vector database, which allows

for efficient retrieval based on similarity metrics. This database enables quick access to relevant data points during the generation process.

- **Retrieval** and Prompting: During the generation process, the LLM can retrieve relevant data points from the vector database based on the context of the current input. This retrieval mechanism helps the LLM provide more accurate and contextually relevant outputs.

Some of the libraries used in our experiment are the following

- **It provides a standard interface for chains, lots of integrations with other tools**
- **OpenAI**Used for embedding
- **python-dotenv**Store the API key in an environmental variable
- **Chroma**used to store embedding vectors

6 Experimental setups

We used the llama-2-7b-amharic 16k context window version. ChatGPT 3.5 can use a maximum of 32 thousand tokens. When we set the chunk size to 2000, we can provide up to 15 chunks of information. We used OpenAI embeddings and the temperature as 0.5. Memory vector store is used for now. We use chunk Size 2000, chunk overlap 100. We give the top 5 chunks as a result of the search. We use Google Colab Notebook which is the preferred environment for our experiment since it provides a Python environment and free GPU access. we install all the required Python libraries and modules. They help us with training efficiency (accelerate), allow for low-rank adaptations (peft), facilitate quantized training (bitsandbytes), and give access to pre-trained models and tools (transformers, trl). The pre-trained Llama 2 model is loaded with our quantization configurations. We then disable caching and set a pretraining temperature parameter. o reduce the model size and increase inference speed, we use a 4-bit quantization provided by the BitsAndBytesConfig. Quantizing the model means representing its weights in a way that uses less memory. We specify the pre-trained Llama 2 model and prepare for its enhanced version, llama-2-7b-ethiogezamh.. The main hyper parameters for the fine tuning process are set as follows: Batch size per GPU of 12, learning rate of 0.0002, warmup ratio of 0.1, max length of 1024, lora rank of 64, lora alpha of 16, and lora dropout of 0.05.

7 Relevant related works

Recently, RAG has also been widely applied in LLMs to provide additional knowledge and reduce model hallucination[3]. They overcome challenges in LLMs effectively using the retrieved information. They propose an information refinement training method named INFO1-RAG that optimizes LLMs for RAG in an unsupervised manner. They perform extensive experiments on zero-shot prediction of 11 datasets in diverse tasks including Question Answering, Slot-Filling, Language Modeling, Dialogue, and Code Generation. In their experiment they show that INFO-RAG improves the performance of LLaMA2 by an average of 9.39[8]uses DPR as a dense retrieval approach that enhances the retrieval component of RAG models by encoding passages and queries into dense vectors, allowing for efficient and effective retrieval of relevant information. [1] propose a method REALM which integrates retrieval into the pre-training of a language model, which improves the model's ability to incorporate external knowledge and generate accurate answers in a QA setting. A new family of Llama2 having more data cleaning, updated data mixes, trained on 40% more total tokens, doubled the context length, and used grouped-query attention (GQA) to improve inference scalability for their larger models is created by [9]. They adopt most of the pretraining setting and model architecture from Llama1.The primary architectural differences from Llama 1 include increased context length and grouped-query attention (GQA).

8 Conclusion

We fine-tune the Samuaelllama-2-7b-tebot-amharic mode as llama-2-AmLLM model on a T4 GPU using a parameter efficient technique called LORA. It is impossible to fine-tune

LLMs on consumer hardware due to inadequate VRAMs and computing. However, in this tutorial, we will overcome these memory and computing challenges and train our model using a free version of Google Colab Notebook. The Google Colab T4 GPU has a limited 16 GB of VRAM. That is barely enough to store the model weights, which means full fine-tuning is not possible, and we used parameter-efficient fine-tuning techniques like LoRA and take the base model Samuaelllama-2-7b-tebot-amharic model and EthioNLP/AmharicInstructiondataset from hugging face dataset. We have used the hugging face Amharic instruction dataset and libraries' such as transformers, accelerate, trl, peft and bitsandbytes. The experiment dramatically improves accessibility and usability for real-world applications. For RAG experiment, we used a model named Samuaelllama-2-7b-tebot-amharic from hugging face. When we set the chunk size to 1000 and chunk overlap of 20. We used OpenAI embeddings and the temperature as 0.5. Langchain toolkit for RAG. Faiss database vector store was used for now. We give the top 5 chunks as a result of the search. IN this study we use OpenAI embedding which is better than OLLama and other Embedining.

9 Future tasks

we recommend researchers to add prompt engineering to our language models.Prompt learning substitutes pre-trained cues and predictions for the pre-trained and fine-tuning procedure. Under this paradigm, the downstream task is to redefine the downstream task using text prompts so that it more closely resembles the tasks completed during the first LM training, rather than adapting the pre-trained LM to the downstream task through objective engineering.

References

[1] Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. The falcon series of open language models, 2023.

[2] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235, 2023.

[3] Aman Gupta, Anup Shirgaonkar, Angels de Luis Balaguer, Bruno Silva, Daniel Holstein, Dawei Li, Jennifer Marsman, Leonardo O Nunes, Mahsa Rouzbahman, Morris Sharp, et al. Rag vs fine-tuning: Pipelines, tradeoffs, and a case study on agriculture. *arXiv preprint arXiv:2401.08406*, 2024.

[4] Laura Illia, Elanor Colleoni, and Stelios Zyglidopoulos. Ethical implications of text generation in the age of artificial intelligence. *Business Ethics, the Environment & Responsibility*, 32(1):201–210, 2023.

[5] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. 2023.

[6] Zhaoyang Niu, Guoqiang Zhong, and Hui Yu. A review on the attention mechanism of deep learning. *Neurocomputing*, 452:48–62, 2021.

[7] Ade Ramdan, Ana Heryana, Andria Arisal, R. Budiarianto S. Kusumo, and Hilman F. Pardede. Transfer learning and fine-tuning for deep learning-based tea diseases detection on small datasets. In *2020 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET)*, pages 206–211, 2020.

[8] Nikhil Sardana and Jonathan Frankle. Beyond chinchilla-optimal: Accounting for inference in language model scaling laws, 2023.

[9] Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. Galactica: A large language model for science, 2022.

[10] Ah Chung Tsoi. Recurrent neural network architectures: an overview. *International School on Neural Networks, Initiated by IIASS and EMFCSC*, pages 1–26, 1997.

[11] Qianqian Xie, Qingyu Chen, Aokun Chen, Cheng Peng, Yan Hu, Fongci Lin, Xueqing Peng, Jimin Huang, Jeffrey Zhang, Vipina Keloth, Xinyu Zhou, Huan He, Lucila Ohno-Machado, Yonghui Wu, Hua Xu, and Jiang Bian. Me llama: Foundation large language models for medical applications, 2024.

[12] Seid Muhie Yimam, Abinew Ali Ayele, Gopalakrishnan Venkatesh, Ibrahim Gashaw, and Chris Biemann. Introducing various semantic models for amharic: Experimentation and evaluation with multiple tasks and datasets. *Future Internet*, 13(11):275, 2021.

[13] Ceng Zhang, Junxin Chen, Jiatong Li, Yanhong Peng, and Zebing Mao. Large language models for human–robot interaction: A review. *Biomimetic Intelligence and Robotics*, 3(4):100131, 2023.