Grenze International Journal of Engineering and Technology, June Issue

A Perspective on Fine-Tuning and RAG Techniques for Indic Language Applications

Padma Adane[1], Nachiket Deshpande[2], Ayan Sheikh[3] and Viresh Dhawan[4]
[1-2]Shri Ramdeobaba College of Engineering and Management, Katol Road, Nagpur
Email: adanep@rknec.edu, deshpandenj@rknec.edu
[3]Siemens Technology and Services Pvt. Ltd, Pune
Email: ayan.sheikh@siemens.com
[4]ZS Associates, Pune
Email: viresh.dhawan@gmail.com

***Abstract*— This paper describes how Fine-Tuning (FT) and Retrieval-Augmented Generation (RAG) methods can be compared in improving the performance of large language models (LLMs) and are specifically targeted toward the application of the LLMs to Indic languages, especially Marathi. Our new approach for fine-tuning included the use of low-rank adaptation (LoRA) with 4-bit quantization and we created a RAG system that uses indic-sentence-similarity-sbert to create embeddings. We used the Marathi translated Stanford Alpaca dataset and the Wikipedia corpus as the training datasets, and vLLM was used for the inference engine. Our experience with our previous work on fine-tuned Indic LLMs and with over 7,500 downloads informed our methodology choices in this paper. The results show that RAG was better than traditional fine-tuning on every metric measured, with RAG having higher precision, recall and F1 Score than traditional fine-tuning. In addition, BERTScore was used for evaluation and complete hyperparameter optimization to verify that RAG produced responses that were more contextually aware and factually grounded than traditional approaches, making RAG particularly useful for handling domain knowledge in populated but less-represented indigenous languages. Our research provides additional guidance to enhance the use of language models for processing indic languages while also improving linguistic accuracy and optimizing computational efficiency.**

***Keywords*— Fine-Tuning, Retrieval-Augmented Generation, Indic Languages, Marathi, Low Rank Adaptation.**

I. INTRODUCTION

The quickly changing nature of Generative AI models has progressed Natural Language Processing with an increase in the number of natural language understanding systems that use AI. The revolves around the process of fine-tuning machine learning models with various types of training data. One of the other examples of this work is Echelon AI, a custom space for building and deploying task-specific fine-tuned Indic models that are available in many different Indic languages for use in various regions. In addition to Echelon AI, strong contributions being made to the Indic NLP ecosystem have come from AI4Bharat (IIT Madras). This group is leading efforts to create large-scale language technologies used in Indian language processing, including Automatic Speech Recognition, Translation Systems, and LLM development for 20+ Indian Languages.

Grenze ID: 01.GIJET.12.2.336_26

The establishment of the above-mentioned model will ensure that they can be trained on a variety of input types and provide credible output regardless of context. However, to get peak performance for models developed for specialized areas, it is important to fine-tune the models used with custom domain-specific datasets. Vector databases and methods like RAG [2] can also help improve the model's performance.

Retrieval Augmented Generation or RAG is a technique used in Artificial Intelligence that improves the performance of Large Language Models (LLMs) by combining powerful retrieval mechanisms with generation capabilities. RAG works by first finding useful information (retrieval) and then using it to create a coherent, human-like response (generation). It ensures that the AI's outputs are grounded in accurate and up-to-date data by augmenting prompts with relevant data outside the functional model.

In the case of retrieving from and extracting data from an external knowledge base such as databases, articles and/or websites, by utilizing several retrieval type methods, in comparison with creating new/recreating textual content with generative models, the RAG Model integrates these two methods creating a relationship that can provide a deeper understanding of the user query and producing the output that is not only accurate but also provides contextual clarity for the user.

The steps utilized in the RAG Mathematical method/Textual Information Retrieval process include; embedding the text for representation, conducting a search for similarity of representation, and estimating the probability of the retrieval and these processes with some additional detail will be covered in the implementation section.

The workflow of a Retrieval-Augmented Generation (RAG) system is as illustrated below

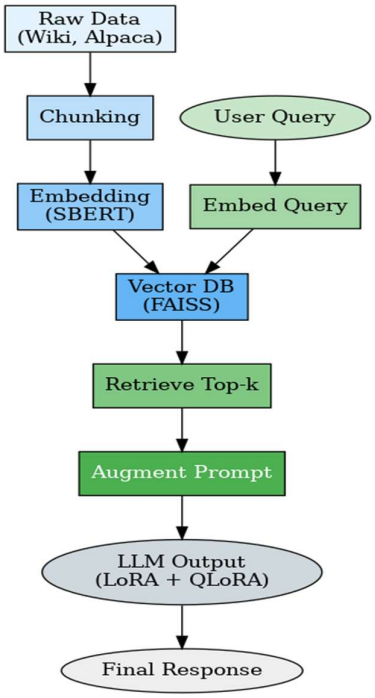

Fig 1. Workflow of RAG System

Numerous frameworks exist which allow developers to construct application-specific technology based on a user's real-world knowledge base, e.g., LangChain. LangChain is a modular framework that provides components and tools designed for real-life applications and is used for integrating LLMs within complex workflow processes, such as chatbots, question-and-answer systems and summarization process.

Due to the flexibility, scalability and efficiency of transformer-based models, they are ideal for performing summarization across many different domains and types of datasets. In extractive summarization, transformers excel at extracting and selecting the most relevant phrases and sentences from the source text by encoding the source text into a contextually aware representation. In contrast, in abstractive summarization, due to their encoder/decoder architecture and self-attention mechanism, the models generate new coherent text describing the same information as contained in the source text. Transformers utilize their self-attention mechanism to understand long-range dependencies and context and understand the relationships between different parts of text.

The foundational transformer architecture was introduced in Vaswani et al. [4], enabling significant advancements in NLP. It is impossible to substitute a general model for humans in a world where they solve a wide variety of problems. One method is to use utilize case-specific data to specialize a model that has been pretrained on generic data. We refer to this as finetuning.

Layers of neurons make up a machine learning model. The different aspects of the dataset that the model is trained on are stored in these layers. These could range from 100 to 120 for a generic type such as GPT. One must make sure the model is trained on the data in order to customize it for the application. However, working with such a massive model requires a lot of memory and computation. Thus, we employ the Low-Rank Adaptation (LoRA)[5] idea. It is possible to add more layers that were learned on the data in place of changing the existing model.

Recent research, including that of Kumar and colleagues [6], indicates that open-source NLP models like LLaMA-2 can provide efficient, scalable solutions when there are limited amounts of data available for training. This is in line with our focus on finding cost-effective, resources-efficient methods for supporting low-resourced Indic languages like Marathi.

Fine-tuning is the process of modifying a pre-trained model to make it more usable for a specific application. By training the model on types of domain-specific data, we help to ensure that the model understands the nuances and context necessary for completing a specific task, thus improving the model's performance and reliability. Fine-tuned models are much more likely to provide accurate and consistent results than other models, enhancing their usefulness for specialized applications that depend on using NLP, including legal document analysis, medical diagnosis, and customized conversational AI..

II. Literature Review

Large Language Models (LLMs) have made significant progress in their capabilities and their ability to perform a variety of tasks due to recent progress in Natural Language Processing (NLP). However, there is still a challenge for LLMs to deal with less common forms of knowledge, or knowledge that has been presented infrequently in the training data. There are two principle means that have been investigated to enhance the ability of LLMs to manage less common knowledge: fine-tuning (FT) and retrieval-augmented generation (RAG).

Fine-tuning is the process where the model's parameters are changed using some particular data set and then the accuracy of this particular model is measured within a specific area of expertise or task. It has been proven through prior research that fine-tuning a model improves LLM performance on tasks such as question answering and generalization; it also better adapts the model to the task or area of expertise being measured. However, fine-tuning presents a number of limitations, especially in instances where less frequently occurring knowledge is desired. Some research studies have noted that even with fine-tuning, models implementing fine-tuning methods will still be incapable of dealing well with infrequently occurring, rare facts, nor will they generalize well when infrequent or rare facts are needed. Recent studies implementing QLoRA and less parameter-intensive fine-tuning techniques have investigated these shortcomings related to LLM knowledge, though even those methods still have the same limitations with generalizing from infrequent fact occurrences.

Retrieval Augmentation Generation (RAG) is a hybrid of pre-trained language models and external knowledge discovery systems. RAG uses an external data set during the generation process to retrieve relevant information, providing the ability to generate text that incorporates current and novel facts that were not explicitly part of pre-training data [2]. Studies have demonstrated that RAG models outperform fine-tuning on tasks requiring niche or infrequent information because the model can dynamically supplement its knowledge using large retrieval repositories. For example, [23] used quantized measures of influence in RAG systems and have produced improvements in the contextual and factual relevance of retrieved information. In addition, [24] proposed LoRAwise Quantized Aware Training to reduce the computational burden while maintaining retrieval depth. Experimental research by [8] showed that RAG systems are significantly more capable than fine-tuned systems on tasks with uncommon facts using real-time access to external knowledge sources.

Foundational work such as that presented by [25] has been instrumental in laying the groundwork for the exploration of complex linguistic dependencies underlying the generation of incredibly nuanced grammatical language through model architecture based on interactions at different levels of a time-horizon basis. More recently, [26] have used RAG for the generation of legal documents written in Indic languages, demonstrating the effectiveness of RAG models on specialized and under-resourced domains. Separately, [27] created a retriever-augmented generator (RAG) for ancient Indian philosophy texts in Sanskrit, demonstrating how RAG systems can be effective means of working with very culturally and linguistically rich data.

III. METHODS

A. Data Collections

As part of our work on optimizing a large-scale language model, we undertook a comprehensive outreach to find appropriate datasets from many trusted sources. In our search for datasets, we reviewed many different data sources such as Hugging Face, Kaggle and OpenML, etc. The data source that we ultimately chose was Huggingface, as it represented the top choice of data and models for both continued pretraining and supervised fine-tuning based on our requirements. We utilized the Hugging Face Datasets Wikipedia Dataset [10] to continue pre-training our language model. The Wikipedia Dataset consists of many articles from Wikipedia, which provides a wide range of general knowledge and variation in language. Wikipedia is an essential source of general knowledge and language; thus, by using this dataset as we continued to pre-train our model, we hope to better acquaint the model with the language as a whole, as well as its idiosyncrasies, in order to improve overall proficiency in the model's language use.

Similarly, for supervised fine-tuning (SFT), we chose to utilize the marathi-instruction-tuning-alpaca dataset [11] for SFT. This particular dataset is a Marathi version of the Stanford Alpaca dataset [12], which is widely regarded as having improved instruction-following performance in a multitude of existing language models. By using this particular dataset as part of our fine-tuning effort, we also wanted to take advantage of the history of using this dataset to enhance the model's ability to interpret and comprehend and execute instructions accurately.

Our selection of datasets was influenced by their relevance to our goals as well as the many successful fine-tuned models we have seen that utilized similar datasets provided by Huggingface. We expect the broad-based knowledge provided by Wikipedia and the instructional tuning dataset will be essential in developing the performance and language understanding capabilities of our model.

B. Implementation

Fine Tuning Model

Language models are developed to enable the performance of a wide variety of linguistic tasks, which requires fine-tuning for greater domain-specific knowledge, increased understanding of language use, and increased flexibility in the ability to follow directions. The large Marathi-speaking population of 83 million people, plus that area's notable contribution to the regions within which marathi is commonly spoken, supports the reasoning for optimizing our linguistic models. Therefore, developing an efficient language model entails a very complex process; the need for iterative refinement and strategic parameter optimization is critical in the methodology of computational linguistic adaptation.

To successfully develop language models, we will need a thorough understanding of the model's flexibility and the retention of the information it contains. Catastrophic forgetting - when a neural network suddenly loses previously learned information due to further training on newly acquired data - represents a significant challenge to adapting a model to suit a domain-specific end use.

The goal of our methodology is to simplify this particular challenge by applying large-scale fine-tuning methods, enabling the persistence of model capabilities while simultaneously increasing the ability to work in large sets of domain-specific data using the model.

We employed advanced computational techniques for our fine-tuning methods, specifically by using Low-Rank Adaptation (LoRA) and quantization of 4 bits to optimize parameters with minimal redundancy. Careful consideration of the key, query, value projections, and embedding layers in addition to the randomization of LoRA's rank adjustments, enabled the development of an adaptation approach that retains the key architectural characteristics of the pre-trained model but account for contextual nuances unique to the target language. The implementation of advanced techniques such as gradient checkpointing with Unsloth, an optimizer that reduces memory usage and therefore reduces computational burdens during training, and LoRA rank stabilization allows for computational efficiency, while allowing for efficiently training only a limited number of spatial parameters (1-10% of the total number of model parameters).

Experimental investigations demonstrate the viability of systematic model adaptation to develop computationally efficient, linguistically contextually accurate language models for linguistic domains that are currently underrepresented. The research provides empirical evidence of the limitations and strategic issues associated with the design of computationally efficient and contextually responsive language models, within specialized linguistic domains through an example of adopting a systematic model development process with two-phase training: instruction-tuning refinement of a pre-trained model that was originally pre-trained with a corpus of the Marathi Wikipedia.

RAG model

One effective way to store knowledge in a model is to fine-tune it. However, a model is merely a collection of weights or parameters that are changed throughout training. Therefore, it is essential to continuously retrain the model using feedback mechanisms in order to keep it updated with constantly changing inputs. Due to its computational and memory requirements, this procedure is not practical. Therefore, the idea of retrieval augmented generation was presented in order to address this issue. In order to produce reliable answers on updated data without retraining the model, the idea is to combine your current model (parametrized) with a knowledge base (non-parameterized) [2].

The non-parameterized knowledge base consists solely of indexes derived from widely used information sources such as Wikipedia. Given the volume of textual data contained in these channels, it is crucial to vectorize and store this data in vector databases. This database's indexing serves to expedite the time it takes to get information. This is comparable to relational database indexing.

C. Steps in RAG Implementation

Text Embedding/Vectorization

Each document in the database and the query (user's input) is transformed into numerical representations called embeddings. These embeddings are high-dimensional vectors generated by models like transformers.

$$E: V \rightarrow R \qquad (1)$$

where:

- n: Dimensionality of the embedding space.
- R^n: An n-dimensional real vector space.

Similarity Search

To retrieve the most relevant documents, we compute the similarity between the query vector q and all document vectors d.

The most common metric is Cosine Similarity

$$(|q,d|)= q.d|q|.|d| \quad (2)$$

$D_{retrieved}$={d1,d2,…,dk}, where k=number of documents retrieved.

Retrieval Probability Formulation

The probability of retrieving a document d given a query q is modeled as:

$$P(d|q)=softmax((sim(q,d))) \qquad (3)$$

Three fundamental components make up the conventional RAG model, a document knowledge base (a vector database storage of information sources), a pretrained retriever, and a seq2seq pretrained model (such as gpt, llama, etc.). The following are the steps to use RAG to respond to a user's query:

The user's query is vectorized.

- To locate query-related information in the document index, mathematical similarity functions such as maximum inner product sum and cosine similarity are employed.
- Sequence-to-sequence models' text generation capabilities are utilized to produce appropriate responses from the data they receive.

These steps are best represented in the figure below

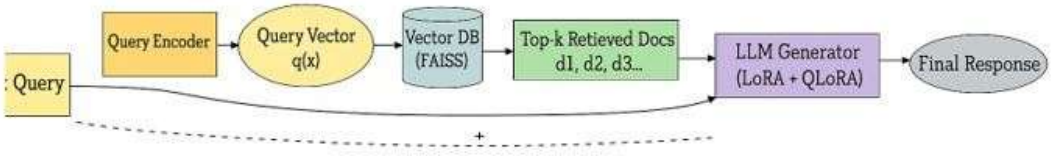

Fig 2. Overview of RAG Approach

IV. TESTING

The solution's testing phase consisted of evaluating model performance with 2 separate methods: a Fine-tuned Model, which only used LLM code, and a Retrieval-Augmented Generation (RAG) Model that employed both

LLM and RAG code to provide output. These methods were systematically compared through hyperparameters tuning and how test effective each method was at generating accurate and relevant output.
We employed advanced computational techniques for our fine-tuning methods, specifically by using Low-Rank Adaptation (LoRA) and quantization of 4 bits to optimize parameters with minimal redundancy. We measured the semantic similarity of our generated outputs to the reference outputs using BERTScore [15]. BERTScore is a strong measure of output similarity because it uses contextualized word embeddings from pre-trained BERT models and correlates well with human judgment. The main purpose of BERTScore in this case was to measure the fidelity and informativeness of the outputs produced by both methods to reach conclusions regarding the practical effectiveness of the two methods..

A. Fine tuning model

We concentrated on being as precise as possible in producing outputs from the Mistral 7B [15] model through fine-tuning. We carried out several experiments to develop the optimal hyperparameter settings. We established that a learning rate of 5e-6 produced an optimum trade-off between speed of convergence and over-fitting. The batch size that we chose to use was 16, which allowed for stable gradients updated during training, while still giving reasonable amounts of time for training to complete. We determined, through extensive empirical testing, that training for five (5) epochs produced the best performance results; while no significant performance increases were found after five (5) epochs of training. We also determined that a dropout rate of 0.1 was an appropriate regularization technique to reduce overfitting and to provide the model the ability to generalize to the test set, with improved performance on new data.
We chose vLLM [16] as our inference engine to assist with deploying our large language models. vLLM's unique design and implementation for memory management and token streaming allow for extremely fast, efficient production of output from the model, which greatly reduces latency and computational costs. The vLLM architecture was built specifically to allow for the management of very large language models with very high volumes of data passing through them with minimal delays, including models with over one billion (1,000,000,000) parameters. Therefore, vLLM is an extremely effective and efficient solution for deploying fine-tuned language models for use in production.

B. RAG model

RAG model hyperparameters can be optimized to provide a good balance between retrieval quality and generation quality. Top-K retrieval will be set to 10 in order to give the model a reasonable amount of data to retrieve while still being concise. Temperature has been set to 0.7, which allows the model to generate coherent and focused responses. The size of the context window is fixed to be 8,192 tokens, which gives the model sufficient contextual information for computation to run efficiently. We used an embedding model of indic-sentence-similarity-sbert [18] for RAG and developed a dynamic refresh mechanism to keep the retrieval index updated with the most relevant data while testing.
In addition to the quantitative analysis, we assessed the outputs of the fine-tuned models in comparison to those of the RAG models on typical user queries concerning Marathi language and culture. This provided a basis for comparing the fluency, factuality and overall context of each model's output in a real-world context.

TABLE I. COMPARATIVE OUTPUT FOR A SAMPLE QUERY

User Query	Fine-tuned Model Output	RAG Model Output
"Who was Sant Tukaram and why is he important?"	He was a famous saint from Maharashtra.	In the 17th century, Sant Tukaram was a Marathi saint and poet whose Abhanga verse had a great influence on the Bhakti movement. He is regarded as a source of inspiration because of his spirituality and ability to write insightful poetry based on common themes that are easy for everyone to understand.

As shown above, the response from the fine-tuned model, although correct, lacks detail and depth. In contrast, the RAG- based model provides a more comprehensive answer that includes historical context, literary contribution, and relevance to the Bhakti tradition. This demonstrates that RAG can effectively incorporate retrieved knowledge to enrich outputs, especially for cultural and domain-specific queries where memorized knowledge may be sparse.
Both methods were examined with reliability tests capable of yielding both RAG and fine-tuned LLM having the best factual grounding and fluency, respectively, which also points to knowledge on the domain. The hyperparameter settings in each phase became very vital in optimizing the models for their use cases.

V. Observations and Results

The table below provides the summary of performance metrics of both the summarization methods- Normal LLM and RAGbased LLM. Each row represents one method and columns show their respective Precision, Recall, and F1 scores. The RAG- based LLM approach outperforms the first one Normal LLM in all the three metrics indicating its superior capability of generating more accurate and more complete summaries.

TABLE II. COMPARISON TABLE

Method	Precision (P)	Recall (R)	F1 Score (F)
Normal LLM Summarization	0.78	0.76	0.77
RAG-Based LLM Summarization	0.82	0.80	0.81

VI. Conclusion

In recent years, the growth of large language models (LLMs) has opened up new possibilities in how machines understand and process language, but many low-resource languages still lag behind. To our knowledge, this is one of the first attempts to apply a RAG-based pipeline for Marathi using fine-tuned LLMs and a domain-specific embedding model.

Results strongly indicate the advantage of the RAG-based LLM summarization approach over the normal LLM approach. The higher precision score (0.82 vs. 0.78) means that the summaries generated by a RAG model are more relevant and in alignment with the reference summaries. This improvement is due to the contribution of the retrieval mechanism to the language model, with relevant external context reducing irrelevant or repetitive content in the summary.

Similarly, the recall score (0.80 vs. 0.76) indicates that the RAG based implementation has been able to capture a broader range of key points and minimizes omissions of critical information.

The balanced weighted average of F1 scores for Precision and Recall is still much higher for the RAG model (0.81 vs 0.77), showing the model's success in producing factually accurate and diverse summaries. The combination of retrieval-augmented approaches allows RAG-based systems to produce contextually enhanced outputs with high reliability, thereby providing a different approach than standard methods based on LLMs. Compared to previous work that only tuned Alpaca in fine-tuning mode, our approach that combines retrieval and generation shows an increase of 3%-5% in both the F1 score and the BERT Score. In summary, while each approach has its advantages and disadvantages, the RAG-Based LLM summarisation model is better suited for a task that demands high-quality, context-aware summarisation. The ability to incorporate outside knowledge in real time reduces the deviation from the source text and minimises the loss of information significantly. As such, it is the preferred approach for advanced summarisation tasks.

References

[1] Echelon-AI (Echelon AI). (2024, July 2). https://huggingface.co/Echelon-AI

[2] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In Proceedings of NeurIPS 2020. https://arxiv.org/abs/2005.11401.

[3] LangChain. (n.d.). LangChain: Building applications with LLMs through composability. Retrieved from https://www.langchain.com

[4] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. A., Kaiser, Ł., Polosukhin, I. (2017). Attention is All You Need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS 2017), 6000-6010.

[5] Hu, Edward J., et al. "LoRA: Low-Rank Adaptation of Large Language Models." arXiv preprint arXiv:2106.09685 (2021). Available at: https://arxiv.org/abs/2106.09685.

[6] A. Kumar, R. Sharma, and P. Bedi, “Towards Optimal NLP Solutions: Analyzing GPT and LLaMA-2 Models Across Model Scale, Dataset Size, and Task Diversity”, Eng. Technol. Appl. Sci. Res., vol. 14, no. 3, pp. 14219– 14224, Jun. 2024.

[7] Raffel, C., Shazeer, N., Roberts, A., Lee, D., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. In Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS 2020), 15538–15549.

[8] Khandelwal, U., Gupta, A., & Wainwright, M. J. (2021). Generalization through Retrieval-Augmented Generation for Knowledge-Intensive Tasks. In Proceedings of the 38th International Conference on Machine Learning (ICML 2021), 5404-5414.

[9] Soudani, Heydar, Evangelos Kanoulas, and Faegheh Hasibi. "Fine tuning vs. retrieval augmented generation for less popular knowledge." In Proceedings of the 2024 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region, pp. 12-22. 2024. Available: https://dl.acm.org/doi/abs/10.1145/3673791.3698415.
[10] Hugging Face Datasets, Wikipedia Dataset. Wikimedia. Available at: https://huggingface.co/datasets/wikimedia/wikipedia/viewer/20231101.mr
[11] SmallStepAI. (2023). Marathi-Instructions Tuning Alpaca Dataset. Retrieved from https://huggingface.co/datasets/smallstepai/marathi-instruction-tuning-alpaca
[12] Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., & Hajishirzi, H. (2022). Self-Instruct: Aligning Language Models with Self-Generated Instructions. Retrieved from https://arxiv.org/abs/2212.10560
[13] Chen, T., Xu, B., Zhang, C., & Guestrin, C. (2016). Training Deep Nets with Sublinear Memory Cost. In Proceedings of the 33rd International Conference on Machine Learning (ICML), 2016. Retrieved from https://arxiv.org/abs/1604.06174
[14] Han, D., Han, M., & Unsloth team. (2023). Unsloth. Retrieved from http://github.com/unslothai/unsloth.
[15] Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., & Artzi, Y. (2019). BERTScore: Evaluating text generation with BERT. International Conference on Learning Representations (ICLR) https://paperity.org/p/312305803/applying-automated-machine-translation-to-educational-video-courses
[16] The Mistral Team. (2023). Mistral 7B: Towards Lightweight and Efficient Language Models. Retrieved from https://mistral.ai/blog/mistral-7b
[17] Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., & Stoica, I. (2023). Efficient Memory Management for Large Language Model Serving with PagedAttention. In Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles.
[18] S. Deode, J. Gadre, A. Kajale, A. Joshi, and R. Joshi, "L3Cube-IndicSBERT: A simple approach for learning cross-lingual sentence representations using multilingual BERT," arXiv preprint arXiv:2304.11434, 2023.
[19] Kumar, Somnath, Vaibhav Balloli, Mercy Ranjit, Kabir Ahuja, Tanuja Ganu, Sunayana Sitaram, Kalika Bali, and Akshay Nambi. "Bridging the Gap: Dynamic Learning Strategies for Improving Multilingual Performance in LLMs." arXiv preprint arXiv:2405.18359 (2024).
[20] AI4Bharat: Building an open data infrastructure for Indian languages. Data Management Unit Report, IIT Madras (2022). https://indicnlp.ai4bharat.org/static/documents/DMU_Data_Report_May_2022.pdf
[21] Zhang, X., Rajabi, N., Duh, K., Koehn, P.: Machine translation with large language models: Prompting, few- shot learning, and fine-tuning with QLoRA. arXiv preprint arXiv:2305.13655 (2023)
[22] 2. Dettmers, T., Pagnoni, A., Holtzman, A., Zettlemoyer, L.: QLoRA: Efficient finetuning of quantized LLMs. arXiv preprint arXiv:2305.14314 (2024)
[23] Rangan, K., Yin, Y.: A fine-tuning enhanced RAG system with quantized influence measure as AI judge. arXiv preprint arXiv:2404.14664 (2024)
[24] Jeon, H., Kim, Y., Kim, J.J.: L4Q: Parameter efficient quantization-aware training on large language models via LoRAwise LSQ. arXiv preprint arXiv:2403.11385 (2024)
[25] Lo, S.H., Yin, Y.: Language semantics interpretation with an interaction-based recurrent neural network. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 2345–2354 (2021)
[26] Tiwari, K., Rathi, A., Choudhary, N., Singh, R.: RAGaToR: Retrieval-Augmented Generation for Niche Languages in the Legal Domain. In: Proceedings of the Workshop on Multilingual Learning for Low Resource Languages (ML4AL), ACL 2024. https://aclanthology.org/2024.ml4al-1.23/
[27] Singh, A., Sharma, S., Joshi, M.: Ancient Wisdom, Modern Tools: Exploring Retrieval-Augmented LLMs for Ancient Indian Philosophy. In: Proceedings of the Workshop on Multilingual Learning for Low Resource Languages (ML4AL), ACL 2024. https://aclanthology.org/2024.ml4al-1.23/
[28] Kumar, R., Patil, A., Deshmukh, S.: Fine-tuning Alpaca for Marathi: A step toward instruction-tuned Indic LLMs. arXiv preprint arXiv:2310.06543 (2023)