Medical Question-Answering with Llama-2-7b: A Comparative Study
Ritik Panda (rp3111) under the guidance of Professor Sambit Sahu

Abstract

In this Project, we leverage the Llama-2-7b model to address medical question-answering. Employing the Retrieval Augmented Generation (RAG) methodology, we fuse the inherent knowledge of Llama-2-7b with a carefully curated medical knowledge base.We also explore the effect of using a fine-tuned Llama-2-7b model with our RAG approach and Finally, compare the usage of Llama-2-7B with and without fine-tuning. Our approach integrates both dense and sparse embeddings from MiniLM and Splade, respectively, enabling the generation of precise, relevant, and non-hallucinated responses. Using VertexAI and Pinecone, we establish a streamlined pipeline for embedding, retrieval, and the generation of insightful text.

1. System Overview

- In the GCP environment, the project's architecture is visually represented (Fig 1).
- During offline data ingestion, medical documents from the knowledge base are embedded and stored in the Pinecone vector database.
- User-submitted domain-specific medical knowledge queries prompt the system to embed the queries and retrieve semantically similar documents from the stored vector knowledge base.
- The retrieved documents are provided to the Llama-2–7b model as contextual input, feeding them into the prompt.
- Llama-2–7b generates a response, prioritizing efficiency and accuracy based on the contextual documents.
- The generated precise response is delivered back to the user.

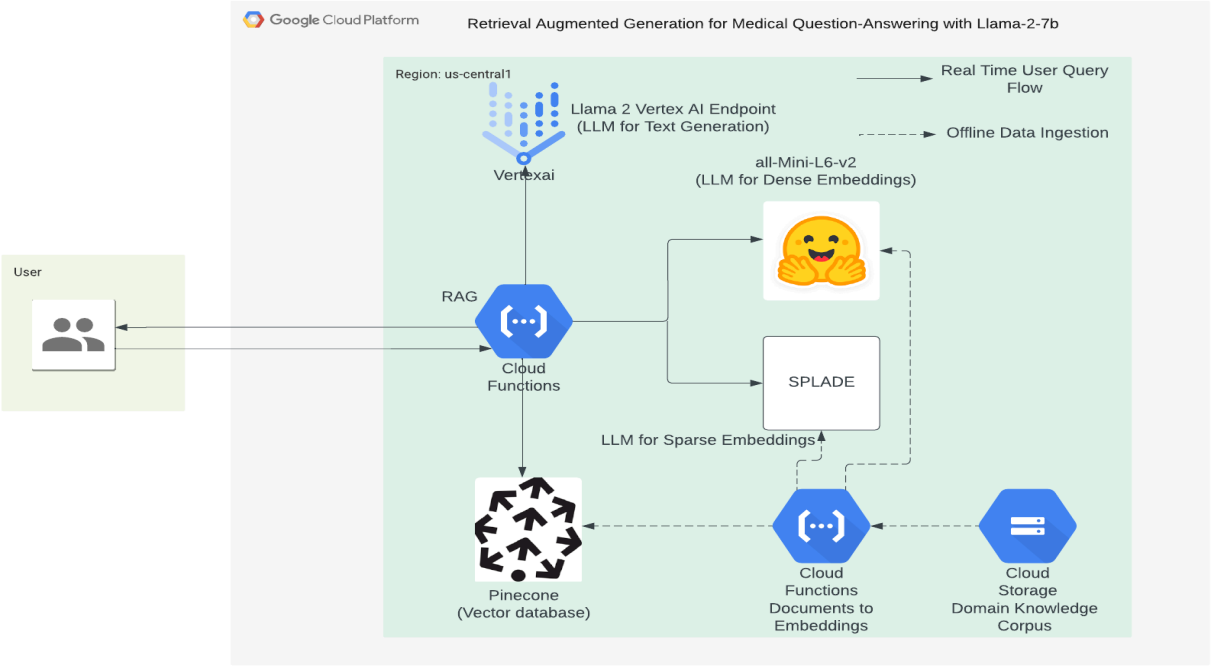


Figure 1. System Architecture

1.1 Dataset Preparation

PubMedQA [1] is a novel biomedical question answering dataset sourced from PubMed abstracts. It comprises 1,000 expert-annotated, 61.2k unlabeled, and 211.3k artificially generated QA instances. Tasks involve answering research questions with yes/no/maybe using abstracts. Instances include questions derived from article titles, abstract contexts, long answers (abstract conclusions), and corresponding yes/no/maybe answers. This dataset necessitates reasoning over biomedical research texts, emphasizing quantitative content comprehension.

We extract the context field from PubMed-QA dataset to form our domain knowledge corpus. An essential process is "chunking" the context into segments, aligning with a preset token length of 384. This step is crucial to meet the input size constraints of the model.

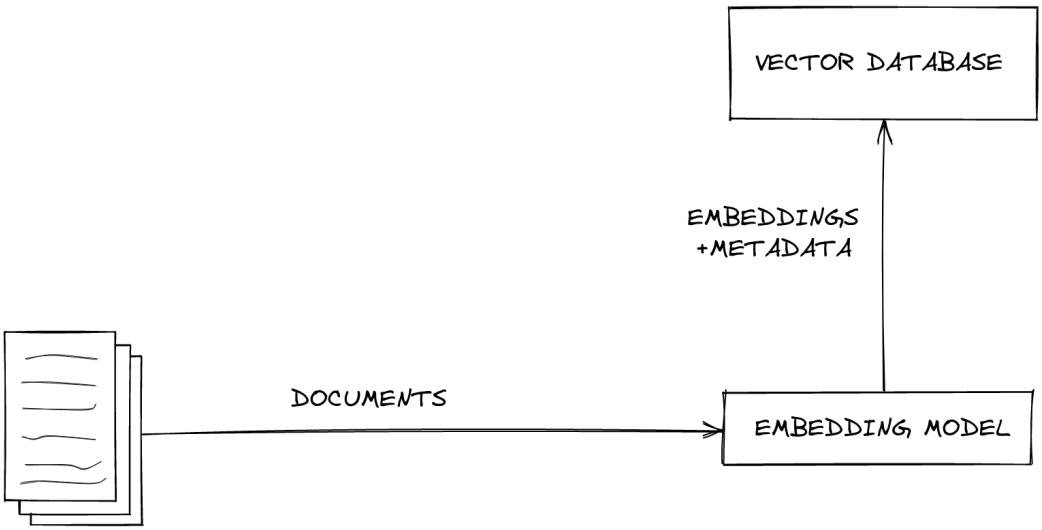


Figure 2. Creating a vector database from your domain-specific proprietary data using an embedding model.

1.2 Llama-2-7b

Llama 2, created by Meta, consists of a suite of sophisticated generative text models which have parameters spanning from 7 to 70 billions. These models employ an optimized transformer architecture [3] and incorporate techniques like supervised fine-tuning and reinforcement learning, closely mimicking human dialogue preferences.

The Transformer is a new type of network that simplifies how computers understand and generate sequences of information. Unlike older models, it uses attention mechanisms without the need for complex structures. In translation tasks, it performs better than existing models, achieving high scores with faster training times.

From [3]

Specifically, Llama-2-7b, with its 7 billion parameters, functions as a potent generator in this project. It adeptly synthesizes medical information retrieved from the knowledge base, showcasing its pivotal role in elevating the quality of AI-driven responses to medical queries.

1.3 Dense Embeddings - all-MiniLM-L6-v2 model

Dense embeddings, like those from the sentence-transformers/all-MiniLM-L6-v2 model [4], are numerical vectors representing discrete objects, such as words, in a continuous mathematical space. These fixed-size vectors convert textual information into a format suitable for machine learning models. The chosen all-MiniLM-L6-v2 model efficiently maps sentences and paragraphs to a 384-dimensional dense vector space, allowing machine learning models to comprehend and leverage semantic relationships between words and phrases.

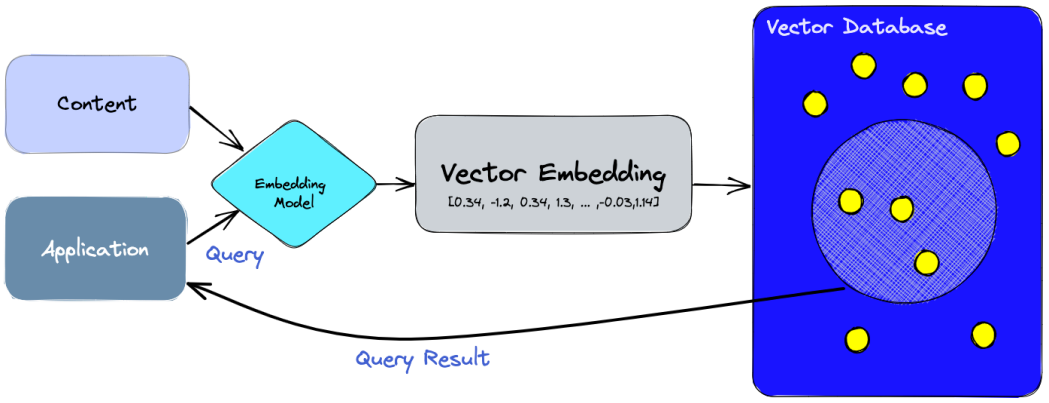
1.4 Sparse Embeddings - SPLADE

Sparse embeddings, unlike dense embeddings, use vectors with mostly zero values to represent words or phrases, providing a high-dimensional representation of linguistic data. In Retrieval Augmented Generation (RAG), sparse embeddings excel at handling large-scale databases using inverted indexes for swift and precise retrieval of relevant information. Our approach, SPLADE [5], harnesses sparse embeddings by learning query/document sparse expansion through the BERT MLM head and sparse regularization. This ensures efficient computational use, maintains explicit lexical matching, and enhances interpretability. Sparse representations often show better generalization, particularly on out-of-domain data, boosting the model's robustness and versatility across various applications.

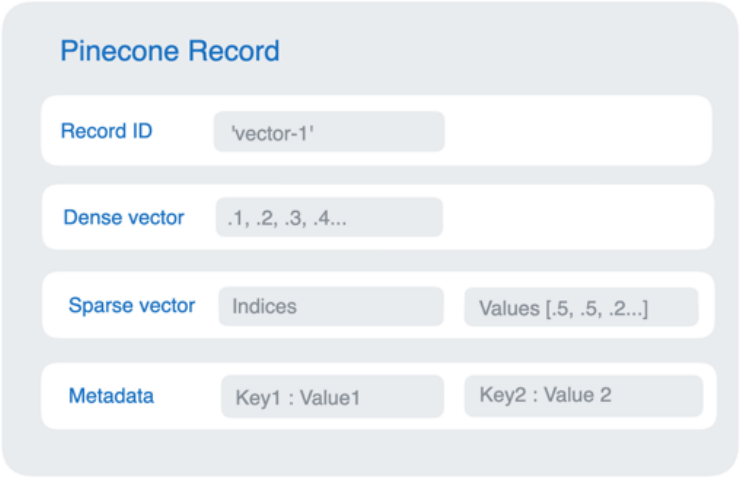
1.5 Vector Database

Vector databases, such as Pinecone, excel in storing and querying high-dimensional vectors, mathematical representations of objects in multi-dimensional space. Traditional scalar-based databases struggle with the complexity and scale of such data, hindering real-time analysis and insights extraction. Unlike vector indexes like FAISS, vector databases integrate

traditional database features with optimized vector index performance. Pinecone, a fully managed vector database, enhances the vector search process. These databases employ specialized organizing systems like tree-based structures (e.g., k-d trees) or hashing techniques (e.g., locality-sensitive hashing) for efficient searching of similar vectors. They prioritize quick retrieval based on vector similarity or distance, often including metadata for related information.



From [6]

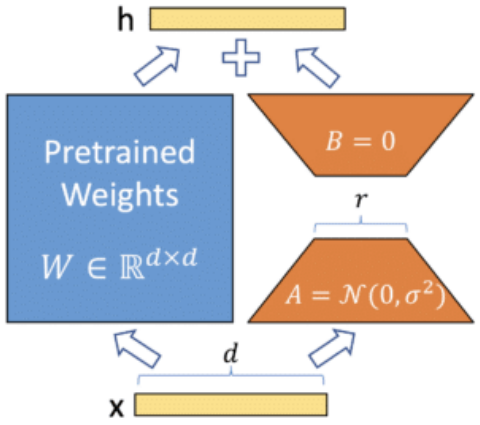


Pinecone indexes store records with vector data

2. Fine-Tuning Llama with QLoRA

LoRA, or Low Rank Adaptation, is a game-changer in making large language models accessible. Unlike before, where such models were deemed exclusive, LoRA has democratically transformed their use. By reducing trainable parameters up to 10,000 times through low-rank adaptation, it offers:

- A threefold decrease in GPU requirements, a significant bottleneck.
- Comparable or superior performance without fine-tuning the entire model.



Traditional fine-tuning required a matrix of the same size, but LoRA employs rank factorization, splitting it into smaller matrices. This innovative approach drastically cuts down the number of parameters, making large language models more feasible and efficient.

2.1 Model and Tokenizer Setup for Llama 2:

In the setup process for the Llama 2 model and its enhanced version, llama-2-7b-mlabonne-enhanced, specific configurations are applied. The tokenizer is loaded and fine-tuned for compatibility with half-precision floating-point numbers (fp16), optimizing memory consumption and training speed.

To ensure efficient text data preparation, the pre-trained Llama 2 model is loaded with quantization configurations, enabling a reduction in model size and faster inference. The use of 4-bit quantization, specifically 'nf4' type, is implemented through the BitsAndBytesConfig, aiming to experiment with different quantization types for potential performance variations. Additionally, caching is disabled, and a pretraining temperature parameter is set.

2.2 Dataset Preparation:

Templates, namely prompt_template and answer_template, are created to structure instructions, responses, and the entire text.

```
# creating templates
prompt_template = """Below is an instruction that describes a task.
Write a response that appropriately completes the request.

### Instruction:

{instruction}

### Response:\n"""

answer_template = """{response}"""
```

A function, _add_text, is defined to add keys for prompt, answer, and text to each record in the dataset. This function extracts the instruction ("question") and response ("long_answer") from each record, ensuring their existence, and combines them using the templates. The

dataset undergoes mapping using the _add_text function to include the newly added keys. The resulting dataset is then ready for use in a question-answering task.

An example is illustrated

```
Below is an instruction that describes a task. Write a response that appropriately completes the request.\n\n### Instruction:\n\nDo mitochondria play a role in remodelling lace plant leaves during programmed cell death?\n\n### Response:\nResults depicted mitochondrial dynamics in vivo as PCD progresses within the lace plant, and highlight the correlation of this organelle with other organelles during developmental PCD. To the best of our knowledge, this is the first report of mitochondria and chloroplasts moving on transvacuolar strands to form a ring structure surrounding the nucleus during developmental PCD. Also, for the first time, we have shown the feasibility for the use of CsA in a whole plant system. Overall, our findings implicate the mitochondria as playing a critical and early role in developmentally regulated PCD in the lace plant.
```

2.3 Quantization Configurations for LoRA Training:

In the LoRA (Low-Rank Adaptation) training process, specific parameters play crucial roles. Here's a simplified breakdown of each of the LoRA-Specific Parameters:

1. Dropout Rate (lora_dropout): This parameter defines the probability of setting each neuron's output to zero during training. It acts as a preventive measure against overfitting.
2. Rank (r): Rank quantifies how the original weight matrices are decomposed into smaller matrices, reducing computational demands and memory usage. Lower ranks enhance model speed but might impact performance. QLoRA requires a rank of 64, deviating from the original LoRA paper's recommendation of starting with a rank of 8.
3. lora_alpha: This parameter governs the scaling of the low-rank approximation. It strikes a balance between the influence of the original model and the low-rank approximation. Higher values amplify the approximation's impact in fine-tuning, influencing both performance and computational cost.

Step	Training Loss
25	2.118400
50	1.394900
75	1.466800
100	1.237200
125	1.494500
150	1.244800
175	1.498800
200	1.265600
225	1.460300
250	1.269500

```python
# LoRA Config
peft_parameters = LoraConfig(
    lora_alpha=16,
    lora_dropout=0.1,
    r=8,
    bias="none",
    task_type="CAUSAL_LM"
)

# Training Params
train_params = TrainingArguments(
    output_dir="./results_modified",
    num_train_epochs=1,
    per_device_train_batch_size=4,
    gradient_accumulation_steps=1,
    optim="paged_adamw_32bit",
    save_steps=25,
    logging_steps=25,
    learning_rate=2e-4,
    weight_decay=0.001,
    fp16=False,
    bf16=False,
    max_grad_norm=0.3,
    max_steps=-1,
    warmup_ratio=0.03,
    group_by_length=True,
    lr_scheduler_type="constant",
    report_to="tensorboard"
)

# Trainer
fine_tuning = SFTTrainer(
    model=base_model,
    train_dataset=small_dataset,
    peft_config=peft_parameters,
    dataset_text_field="text",
    tokenizer=llama_tokenizer,
    args=train_params
)

# Training
fine_tuning.train()

# Save Model
fine_tuning.model.save_pretrained(refined_model)
```

3. Results

To test the RAG / LLM Models we create a test dataset consisting of 15 questions along with the ground truth answers for each of them. These questions are created using the PubMed data.

We make use of the ragas framework [link] to create an evaluation pipeline and choose 3 evaluation metrics as part of our pipeline:

Faithfulness - This measures the factual consistency of the generated answer against the given context. It is calculated from answer and retrieved context. The answer is scaled to (0,1) range. Higher the better.

$$\text{Faithfulness score} = \frac{|\text{Number of claims in the generated answer that can be inferred from given context}|}{|\text{Total number of claims in the generated answer}|}$$

Answer Relevance - Focuses on assessing how pertinent the generated answer is to the given prompt. A lower score is assigned to answers that are incomplete or contain redundant information. This metric is computed using the question and the answer, with values ranging between 0 and 1, where higher scores indicate better relevancy. This metric does not check for factuality.

Answer Correctness - The assessment of Answer Correctness involves gauging the accuracy of the generated answer when compared to the ground truth. This evaluation relies on the ground truth and the answer, with scores ranging from 0 to 1. A higher score indicates a closer alignment between the generated answer and the ground truth, signifying better correctness. Answer correctness encompasses two critical aspects: semantic similarity between the generated answer and the ground truth, as well as factual similarity. These aspects are combined using a weighted scheme to formulate the answer correctness score.

Model → Metric ↓	**Llama-2-7b (Baseline)**	**RAG (with Llama-2-7b)**	**Fine-tuned Llama-2-7b**	**RAG (with Fine-tuned Llama-2-7b)**
Faithfulness	-	**0.6778**	-	**0.6756**
Answer Relevance	**0.9428**	**0.9052**	**0.9540**	**0.9134**
Answer Correctness	**0.4168**	**0.5810**	**0.4576**	**0.6756**

4. Conclusion

LoRA fine-tuning may improve the models' contextual understanding, enabling them to better interpret and respond to queries that involve nuanced information. When fine-tuned with LoRA, the RAG model adapts to specific tasks or domains, leveraging both the general language understanding encoded in LoRA and task-specific nuances learned during fine-tuning.

In summary, RAG with LoRA is a powerful combination that leverages both the strengths of retrieval mechanisms and the pre-trained language understanding capabilities of LoRA, resulting in a model that excels in tasks requiring a balance of information retrieval and generation. The RAG model seamlessly integrates information retrieval and text generation. It retrieves relevant information from the knowledge and generates responses based on the retrieved information, ensuring that the generated output is contextually informed.

References

[1] Jin, Qiao, et al. "Pubmedqa: A dataset for biomedical research question answering." *arXiv preprint arXiv:1909.06146* (2019).

[2] Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

[3] “Introducing Llama2: The next generation of our open source large language model”, *Meta*, https://ai.meta.com/llama/ Accessed 10 Dec. 2023.

[4] Reimers, Nils and Iryna Gurevych. “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.” *Conference on Empirical Methods in Natural Language Processing* (2019).

[5] Formal, Thibault, et al. "SPLADE v2: Sparse lexical and expansion model for information retrieval." *arXiv preprint arXiv:2109.10086* (2021).

[6] Schwaber-Cohen, Roie. “What Is a Vector Database & How Does It Work? Use Cases + Examples.” *Pinecone*, https://www.pinecone.io/learn/vector-database/ Accessed 10 Dec. 2023.