IJMSIT

International Journal of Multidisciplinary Studies and Innovative Technologies

e-ISSN: 2602-4888
dergipark.org.tr/en/pub/ijmsit
Research Article

2025, 9 (2), 249-261
DOI: 10.36287/ijmsit.9.2.10
Received: October 15, 2025; Accepted: November 26, 2025

An Integrative Approach to LLM Literature with the Combination of QLoRa, SFT and Agentic RAG

Aslı Güngör [1], Büşra Nur Emir [2], Sedanur Yılmaz [3], Melike Akdağ [4], Ali Berkol [5*]

[1]Department of Computer Engineering, TED University, Ankara, Turkey (asli.gungor@tedu.edu.tr) (ORCID:0009-0009-2916-2488)
[2]Department of Computer Engineering, TED University, Ankara, Turkey (bnur.emir@tedu.edu.tr) (ORCID:0009-0009-8019-9999)
[3]Department of Computer Engineering, TED University, Ankara, Türkiye (sedanur.yilmaz@tedu.edu.tr)(ORCID: 0009-0008-2218-9031)
[4]BİTES Defence and Aerospace Technologies, Ankara, Türkiye (melike.colak@bites.com.tr) (ORCID: 0000-0002-7779-4756)
[5]BİTES Defence and Aerospace Technologies, Ankara, Türkiye (ali.berkol@bites.com.tr) (ORCID: 0000-0002-3056-1226)*

***Abstract* –** This study offers a solution for document-based question-answering systems for both mobile and web-based applications. This solution combines the fine-tuning of the transformer architecture found suitable for the problem using the appropriate dataset and the agentic Retrieval-Augmented Generation (RAG) methodology. This allows the system to handle not only document-based questions but also non-document questions through the web search agent. A separate agent structure was also incorporated into the solution to facilitate communication with the model in various languages. In the first phase, the Llama 3.1–8B Instruct model was quantized using the quantized Low-Rank Adaptation (QLoRa) method using a dataset with a context-question-answer structure and trained with Supervised Fine-Tuning (SFT). To overcome common problems encountered in the classical RAG architecture, such as hallucination existence, inaccurate document analysis, and missing answers due to insufficient context, agents such as web search, language translation, and techniques like document ranking, and hallucination checking were included, and the agentic RAG architecture was proposed. This system provides a dynamic structure, where user questions and answers are systematically orchestrated. The model's performance has been tested using metrics such as Exact Match, ROUGE-L, BLEU, and F1, and performance improvements have been observed. The test results demonstrate that the modular system achieved through agent integration significantly improves contextual accuracy.

Keywords *– natural language processing, large language models, fine-tune, supervised fine-tuning, quantized low-rank adaptation, agentic retrieval augmented generation, multi-step reasoning, context-based models*

Citation: Gungor, A. et.al., (2025). An Integrative Approach to LLM Literature with the Combination of QLoRa, SFT and Agentic RAG. International Journal of Multidisciplinary Studies and Innovative Technologies, 9(2): 249-261.

I. INTRODUCTION

Large Language Models (LLMs) have been led to a significant paradigm shift in the field of Natural Language Processing (NLP) in recent years. These models, thanks to their structures with billions of parameters, show extraordinary success in a wide variety of text understanding and generation tasks. Nevertheless, training such models on general datasets is subject to limitations in terms of lack of domain-specific knowledge and context sensitivity. Therefore, in order to achieve higher accuracy in domain-focused question-answering systems, pre-trained models need to be retrained with specific data.

Against this background, the IntelliMate project is proposed in this study aims to develop a language model that can be provided more accurate and contextual answers to user queries. The project is used the Llama-3.1 model by Meta [1] and this model was retrained on the Natural Questions (NQ) dataset consisting of real user queries using the SFT method. This dataset consists of real questions that users ask the Google search engine and the long and short answers to these questions provided by Wikipedia [2].

In the literature, two main approaches are generally at the forefront in adapting LLMs for specific tasks: Instruction Fine-Tuning (IFT) and SFT. IFT provides guidance to the model by giving it a task description [3], but in order for this method to be effective, cautiously created and formatted instructions are needed. This makes the preprocessing process very costly in terms of both time and quality, particularly, in complex datasets such as NQ. Therefore, the direct SFT method was preferred in the study. SFT permits for direct optimization of the model's ability [4] to generate responses using labeled data and can produce more efficient results in Llama-based models.

Due to the size of the model, 4-bit quantization technique was used in order to avoid exceeding GPU memory limits and optimize training time, and the model was made lighter with the QLoRA method. This technique significantly reduces memory usage without sacrificing much of the model's accuracy [5]. Additionally, the training parameters were cautiously adjusted, for example, the training batch per device value was kept low while the training was balanced with gradient accumulation steps. Various difficulties have been encountered in the training process. For instance, GPU selection, pad token incompatibility errors were systematically addressed and each was resolved, and the training was completed successfully.

The information retrieval mechanism [6] of this system is based on the Agentic RAG structure. Unlike classical RAG systems, Agentic RAG organizes capabilities such as multi-step questioning, task planning and response verification within the framework of an autonomous agent. This structure facilitates information gathering by the model in a sequential manner rather than all at once, enabling it to incorporate external knowledge at each step. IntelliMate analyzes user queries, plans multi-step actions, optimizes information retrieval, and produces coherent, context-aware responses using the Agentic RAG structure. In this way, not only accuracy but also explainability and modularity was significantly improved.

As a result, this study was examined the adaptability of LLMs to domain-focused question-answering systems through SFT and the Agentic RAG architecture. The contributions to literature are as follows.

1. To conduct with the aim of applying SFT using a combination of 4-bit quantization (NF4), double quantization and LoRa (QLoRA) for the Llama 3.1 8B-Instruct model, contributes as one of the first studies in the literature that adopts this approach.
2. To apply SFT with the NQ dataset in order to run the Llama 3.1 8B-Instruct model in the desired direction; the official version known in the literature has not been previously subjected to SFT with the NQ dataset.
3. To develop an agentic RAG application specific to the domain, components were integrated such as text embedding, retrieval, multi-agent routing, accelerated generation, web search and language detection/translation into a pipeline adapted to the specific application domain; this approach fills a substantial gap in the literature.

II. Literature Review

This section analyzes the studies and dataset in the literature to identify aspects that have not been addressed so far and justifies the original contribution of our research.

A. Dataset

The model that could be appropriate for the question-answering structure of the project and could increase the accuracy rate and the most compatible dataset and version for the SFT process were selected.

a) Confirmation for Selecting the Natural Questions (NQ) Dataset

The NQ dataset [7], introduced by Tom Kwiatkowski et al., [2] serves as a benchmark for question answering research. It contains real user queries and their corresponding Wikipedia articles, which include context, long answers, and short answers. According to the research, annotator is presented with a question along with a Wikipedia page from the top 5 search results and annotates a long answer (typically a paragraph) and a short answer (one or more entities) if present on the page, or marks null if no long/short answer is present. The public release consists of 307,373 training examples with single annotations; 7,830 examples with 5-way annotations for development data; and a further 7,842 examples with 5-way annotated sequestered as test data. While short answers are taken from the long answer span and usually consist of either text spans or boolean responses (i.e., yes/no), long answers are marked as appropriate HTML sections, such as a paragraph, table, or list. The dataset simulates situations [2] in which there is no answer available because some examples do not contain valid responses, in addition to being used to assess model performance in question-answering systems. The dataset is also described analysis of 25-way annotations on 302 examples, giving insights into human variability on the annotation task. For all these reasons, NQ is the ideal selection for reliable fine-tuning and offering a practical benchmark.

The Natural Questions Filtered dataset is available on Kaggle [7] and offers a cleaned subset of the original data. The dataset consists of individual entries, each containing the columns: question, long answers, and short answers. According to its Kaggle description, this subset, which has been pre-filtered and includes both long and short answers, has roughly 65,013 long answer examples and 57,257 short answer examples. Containing approximately 86,212 unique questions in total, it provides a considered collection of high-quality, annotated examples extracted from the original NQ dataset. This version [8] contains less noise compared to the original NQ dataset. Moreover, in terms of training data size [2], [9], NQ (~307k examples) is more comprehensive than SQuAD (~108k examples). Therefore, using the filtered version provides a cleaner and more homogeneous dataset. This dataset was also mentioned in the literature with different versions in several studies.

The NQ Open dataset in the TensorFlow Datasets library is an example of these studies. This dataset [8] contains only the short-answer format. There are 87,925 examples in the "train" section and 3,610 in the "validation" section. NQ-Open offers only clear and short answers by applying a natural data filter, which improves the data quality during the fine-tuning process. Another example is the DistilBERT model, which can be accessed through the Hugging Face library. DistilBERT-based model by datarpit [10] has been fine-tuned on Natural Questions. This is an example of a powerful application that shows that the dataset is widely used within QA models. A related study is the methodological comparisons conducted by Lee et al. [11]. The NQ-Open derivative has been defined as a standard benchmark for "open-domain QA". The work of Lee et al. supports the use of this subset, emphasizing the importance of data in terms of quality and performance.

B. Fine-Tuning Methods

Fine-tuning is applied to pre-trained large language models (LLMs) in order to adapt them to specific tasks. One of these techniques, full fine-tuning, requires updating all model parameters; however, it was not preferred in this study due to its high computational cost and difficulties in implementing it on large models [12]. Instead, SFT method was used, which allows model outputs to be aligned with user objectives [13], [14]. Furthermore, in order to make the process more efficient, Parameter-Efficient Fine Tuning (PEFT) approaches, in which only a limited number of parameters are updated, were utilized [15]. The LoRA architecture, widely used in the literature, was taken as a reference in this context, and the QLoRA method was preferred in practice due to its low memory consumption and high efficiency [16]. In this context, the customization efforts of the Meta Llama 3.1 8B Instruct model, which forms the basis of our project, have an significant role in the literature. For example, a study conducted by FoundationAI [44] took the base Llama 3.1 8B model and underwent post-training to improve its conversational and instruction-following capabilities in a specialized field such as cybersecurity. Works of this kind contribute a strong motivation and reference point for our current study in terms of demonstrating that SFT plays a critical role in successfully aligning models such as Llama 3.1 8B to specific industry needs.

a) Supervised Fine-Tuning

The SFT process enables human-centered training by incorporating user intent into the model training process [13]. In this context, SFT plays a mediating role by providing greater control and scrutiny to the learning process. Furthermore, SFT eliminates the need to retrain models from scratch, reducing both cost and computational burden. By integrating SFT into workflows, model outputs can be audited and documented at every stage. Thus, by leveraging the existing capabilities of large pre-trained models (e.g., Llama, GPT-3, etc.), new subtasks can be learned more efficiently [14].

b) Parameter-Efficient Approaches

PEFT is a frequently preferred method for customizing large language models for specific tasks. While traditional fine-tuning methods may require updating all of the model's parameters, PEFT requires training only a small subset of parameters, reducing computational cost and providing comparable, or even higher, performance in some cases [15]. Various architectures have been developed within PEFT. These include sub-techniques such as prompt-based tuning, reparameterization-based methods, serial adapters, and parallel adapters. These methods enable the successful adaptation of large pre-trained language models to new tasks with a limited number of learnable parameters. One such approach, LoRA, works by adding a low-rank update to the model's weight matrices. This update is represented as the multiplication of two low-rank matrices and is typically added to both the attention and MLP layers of the model. This structure reduces the number of training parameters and maintains model performance. QLoRA is a further development of this method. By combining LoRA with 4-bit quantization, large models can be fine-tuned with low memory capacities [16]. For example, a model with 65 billion parameters can be run efficiently on a single GPU with only 48 GB of memory.

C. Agentic RAG Approaches

Agentic RAG technique is used in our study to observe the performance of handling document based and non-document-based questions integration with the fine-tuned model in a systematic way.

a) Classic RAG System & Towards Agentic RAG: Multistep Reasoning Needs

Large pre-trained language models have been shown to store factual knowledge in their parameters and achieve state-of-the-art results when fine-tuned in downstream NLP tasks [17]. However, pre-trained models with a differentiable access mechanism to explicit non-parametric memory are still limited in their ability to access and precisely process information, and therefore their performance lags behind task-specific architectures in knowledge-intensive tasks. They cannot easily expand or review their memories, they cannot provide direct insight into their predictions, and they can produce “hallucinations”. Therefore, the work of Lewis et al. [17] introduced RAG models, where the parametric memory is a pre-trained seq2seq model and the non-parametric memory is a dense vector index of Wikipedia accessed by a pre-trained neural receiver. RAG is a method that uses external knowledge sources to ensure that LLMs have access to up-to-date, verifiable, and consistent information.

While RAG systems offer improved accuracy and adaptability, they face critical pitfalls that undermine their reliability in real-world applications [18]. One of the most common problems is incomplete content, where a user query cannot be answered due to the absence of relevant information in indexed documents; however, the system may attempt to respond by assuming an answer rather than accepting the ambiguity. Another significant limitation is ranking errors: Even if a relevant document is found, it may not be among the top k results retrieved, making the underlying context generation process ineffective.

Even when accurate information is present in the context, the language model [18] may be unable to extract it due to noise, ambiguous wording, or contradictory data. RAG systems also struggle with format-specific queries, such as requests for structured output (tables, lists); these queries are often ignored or poorly formatted by the model. Furthermore, when answers are too general or overly detailed, this can lead to user dissatisfaction. Finally, incomplete responses pose a subtle but critical problem: although partial information is returned, important elements present in the context are overlooked, reducing the overall usefulness of the response. All these drawbacks have led to the emergence of the concept of agentic rag to achieve multi-step reasoning and fill the gaps of classical rag systems.

b) Approach of Agentic RAG

Agentic Retrieval-Augmented Generation transcends the limitations of RAG by embedding autonomous AI agents into the classic RAG pipeline [19]. These agents dynamically manage retrieval strategies by leveraging agentic design patterns of planning, tooling, and multi-agent collaboration, iteratively refining contextual understanding, and adapting workflows through clearly defined operational structures ranging from sequential steps to adaptive collaboration and with this integration, agentic RAG systems offer unparalleled flexibility, scalability, and context awareness.

The paper [19] demonstrates their superiority in dynamic, particularly when hallucination avoidance, multi-step reasoning, and contextual adaptation are required. The paper also argues that agentic RAG frameworks extend the capabilities of static RAG systems by embedding agency in the retrieval and generation stages. The agentic rag system has some disadvantages as well, along with its advantages. For example, inference time can be extended due to factors such as the planning process and the use of multiple tools, an incorrect retrieval strategy can be selected in the system, or the hallucination problem may not be completely eliminated.

c) *Contribution of Modular Multi Agent RAG System*

The Agentic RAG architecture proposed in our study offers a modular and task-specific agent execution strategy that extends the classification proposed by Singh et al. [19]. Unlike many agentic RAG systems based on classical agent loops, this system uses clearly separated agents, including web search, translator, hallucination checker, and document grader, each specialized for a specific subtask in the pipeline. Additionally, the integration of language-aware translation agents in both the query input and response output stages enhances multilingual interaction. Furthermore, a hallucination feedback loop provides dynamic validation and retrieval optimization, compatible with feedback-driven agent patterns. Furthermore, the pipeline includes a metadata-aware document rating stage that goes beyond simple vector similarity and includes semantic relevance assessment before rendering. This layered and verifiable architecture provides enhanced adaptability and transparency, making it particularly suitable for enterprise or educational deployments where authenticity and traceability are critical.

III. Methodology

This study declares the development of IntelliMate, an artificial intelligence-driven document understanding system designed to generate context-aware and accurate answers from user-uploaded unstructured documents.

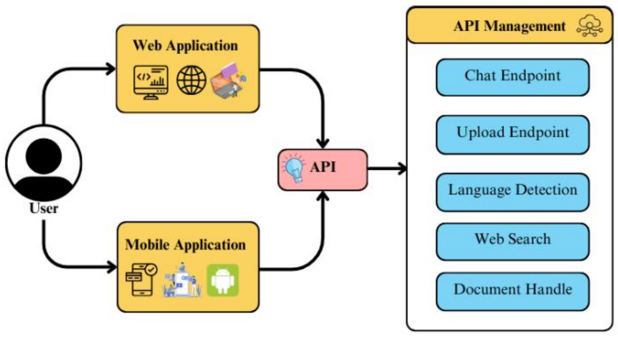


Fig. 1. System architecture

A multi-stage agentic RAG pipeline, 4-bit quantization for efficiency, and the customized fine-tuning of the Llama-3.1 model by Meta compose the advanced components that involve the system architecture. As shown in Fig. 1, the system architecture consists of three main components: Web application, mobile application and a main API layer.

1. The web application is developed using HTML, CSS and JavaScript technologies for the user interface. NoSQL based MongoDB database is preferred for storing and querying user data. Server-side operations and API integration are provided through the .NET framework.
2. The mobile application was developed using Flutter and Dart, offering a user-friendly and cross-platform compatible interface. The device uses an SQLite database for local data processing.
3. The API layer is used in common by both web and mobile applications. This layer provides services such as chat bot, file upload, language detection, document processing and web search. Haystack, OpenAI, Meta AI models and Jupyter environment are integrated for natural language processing and advanced data analysis via API. All user data is securely stored in a main database, and applications access this data via API.

This section outlines our approach, which includes key stages of dataset, fine-tuning, and agentic RAG.

A. *Dataset*

It is the stage of preparing the data set for fine tuning. The most appropriate dataset selection and data preprocessing stages occur.

a) *Dataset Selection*

In this study, we adopted the NQ dataset, which is widely used for the training of question-answer systems. The NQ dataset [2] is a large, open-ended question answering corpus built by extracting real user queries from Google and matching them with corresponding Wikipedia articles. Human annotators contribute both long and short answers, creating a rich and realistic benchmark for QA research. The original release of the dataset [2], [7] contributes 307,373 training examples, 7,830 for development data, and 7,842 for testing data. It is JSON-based, and the answers are mostly contained

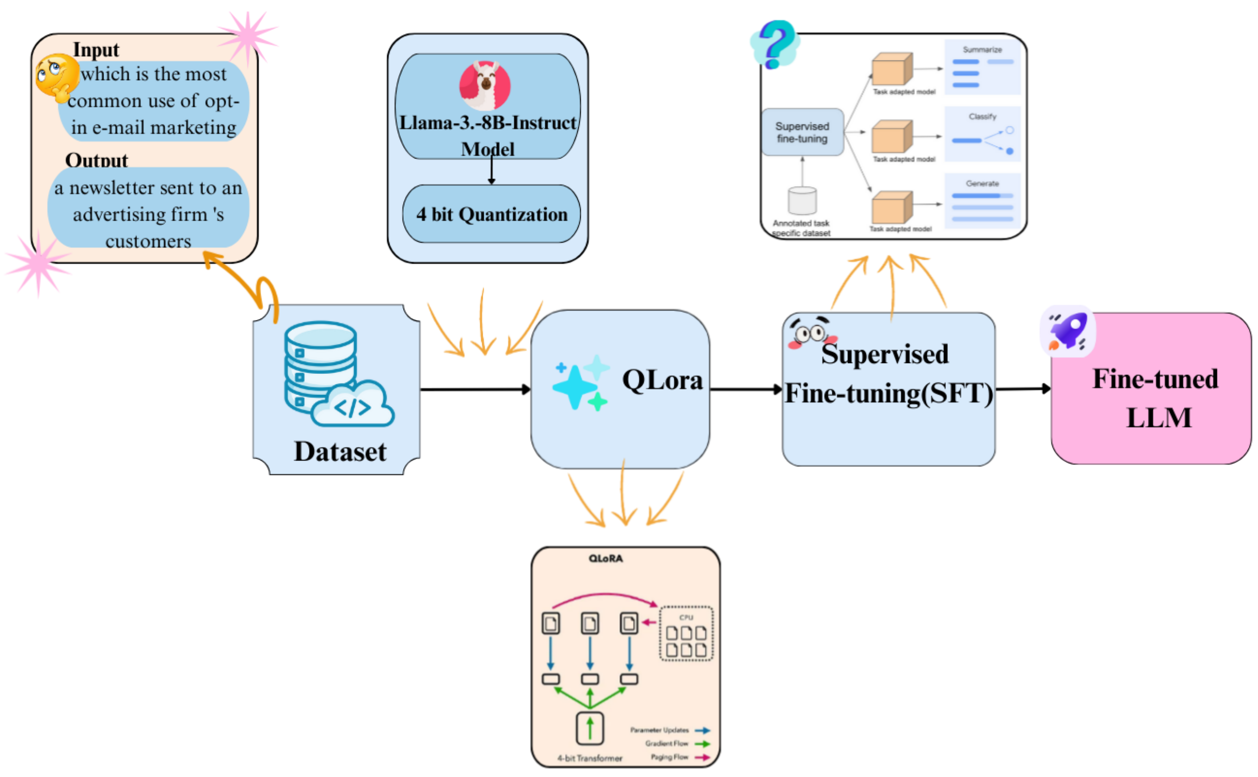


Fig. 2. Fine-Tuning workflow of Llama 3.1 model using QLoRA

in HTML blocks. However, the original NQ dataset is quite large and complex, as it contains more than 300,000 examples. The answers to some queries are contextually inconsistent or inadequate. Moreover, the answers in the original data are mostly embedded in HTML format, parsing and cleaning tags requires an additional preprocessing process. Accordingly, a filtered version called NQ-Filtered, which was published via Kaggle [7], was preferred in our study. According to Kaggle, the file is structured for easy access and analysis, with each record containing the processed question, a detailed answer, and concise answer snippets. The dataset contains a total of 86,212 questions, 65,013 long answers and 57,257 short answers. The answers contained in the columns are not in HTML format as in the original Natural Questions dataset, but in pre-processed and directly converted into text format. According to the data obtained from Kaggle, it is stated that this version allows for more targeted execution of research activities and application development studies. Therefore, the dataset offers a clean and more practical structure for model training.

b) Data Preprocessing

Before training the model, the filtered NQ dataset was transferred to the working environment via Google Drive, and the ZIP file format was extracted. Initially, the CSV file was read using the Pandas library, missing data were checked, and only the complete rows were considered for further processing. During the data processing phase, the total number of examples were limited to 100,000. Each line has been reconstructed in the dictionary structure using the "instruction" (question), "input" (context/long answer) and "output" (short answer) fields and added to the list. This configuration enabled the model to be trained in the instruction-response format. The obtained structured data is divided into three subsets: 80% training, 10% validation and 10% test. These files were subsequently converted into the JavaScript Object Notation Lines (JSONL) Hugging Face Dataset format. In accordance with the input format expected by the model, each sample has been reformatted in the following appropriate format: <s>[INST] instruction input [/INST] output </s>.

As a result, the model learned to generate the correct answer according to the given context by capturing both the user query and the relevant context information. Since the filtered dataset used was truncated of HTML content, no additional parsing process was required. All these preprocessing steps made it possible to fine-tune the model more consistently and efficiently.

B. Fine-Tuning

Fine-tuning is the process of retraining a pre-trained large language model to perform better on a specific task or dataset.

a) Model Selection

The Meta Llama 3.1 collection [1] is a pretrained collection of multilingual large language models as well as instruction tuned generative models in 8B, 70B and 405B sizes (text in/text out). The Llama 3.1 instruction tuned text only models (8B, 70B, 405B) [20] are optimized for multilingual dialogue use cases and outperform many of the available open source and closed chat models on common industry benchmarks.

Llama 3.1 is an auto-regressive language model that uses an optimized transformer architecture, and the tuned versions use SFT and reinforcement learning with human feedback to align with human preferences for helpfulness and safety. Also, English, German, French, Italian, Portuguese, Hindi, Spanish, and Thai are supported languages. The Llama model selected to use for this study was released July 23, 2024, and it includes a custom commercial license, the Llama 3.1 Community License.

The Llama 3.1 8B Instruct model, which has 8 billion parameters, and 128K token 16 times increase comparing Llama 3, is selected because 8B size is suitable for fine-tuning with parameter-efficient methods like QLoRA with limited GPU memory like the A100. Also, the 128K token window makes it possible to contextualize more than one document in a single session in our agentic RAG architecture.

b) Trainer (QLoRa, 8-bit Quantization & SFT Trainer)

At the beginning, the LoRA method, which does not provide memory reduction and only enables training by updating fewer parameters, was adopted and model training was initiated. However, due to higher-than-expected resource consumption, this approach was discontinued. Since GPU and RAM resources were limited for fine-tuning large language models in the project, and the only high-performance GPU accessible was the A100-40GB available via Colab, a more memory-efficient alternative was required. So that, the model needed to be quantized, despite its smaller parameter count compared to other Llama 3.1 models and its 8B size.

The QLoRa method, which provides 8-bit quantization and was deemed suitable for downsizing the model, was chosen. This process decreased the model's weights to a lower bit precision, to reduce both memory consumption and processing time. Quantization further reduced the model's parameter count, and the model was downsized. Concretely, the original model's 8B parameter count was approximately 24GB, and after the model was 8-bit quantized, this memory consumption was reduced to 5-6GB approximately. This makes it easier to fine-tune the model with low resource requirements.

SFT is the customization of a large language model to a specific task or dataset. Because our project's primary goal is a document-based question-answering system, the SFT method was applied to the filtered NaturalQA dataset to better adapt the model to the task. As mentioned earlier, QLoRa, a Parameter-Efficient Fine-Tuning method, was used to ensure less costly and efficient training. Paged Adamw with 8bit was selected as the training optimizer.

c) Model Evaluation

Tests of the fine-tuned model were initially conducted manually, using traditional methods for direct observation. Each sample result was then tested using a custom dataset created for the test, and these results were included in the calculation of the test data. Exact Match (EM), ROUGE-L, BLEU, and F1 scores were used to test the model.

Exact Match assesses whether the model's answer matches the ground truth. This metric is particularly important for questions requiring short and precise answers. EM formula is given EQ. 1 below. N represents the total number of samples, the prediction represents the model's $prediction_i$ prediction, and the reference represents the correct answer for the $prediction_i$ sample. The general function returns 1 if there is a match between the prediction and the reference, 0 otherwise, and to obtain the EM result by dividing the total result by the total number of samples.

$$EM = \frac{1}{N}\sum_{i=1}^{N} 1(prediction_i = reference_i)$$

EQ. 1

F1 score measures the semantic similarity between the model and the reference response by comparing them with a pre-trained language model. F1 formula is shown in EQ. 2 below. It measures semantic similarity between prediction and reference sentences by converting sentences into word-level embeddings. It works with cosine similarity and matches each word to its closest equivalent, producing precision, recall, and F1 scores. For each prediction word, the average of the cosine similarity scores of the closest reference word embedding is precision. For each reference word, the average of the cosine similarity scores of the closest prediction embedding is recall.

$$F1 = 2 \times \frac{\Pr e\,cision \times Recall}{\Pr e\,cision + Recall}$$

EQ. 2

ROUGE-L is more tolerant and measures the longest common subsequence (LCS), or the degree to which the model's answer matches the actual answer. ROUGE-L formula is given EQ. 3 below. β is the hyperparameter that determines the weight of recall relative to precision and is typically set to 1. The LCS represents the longest sequence of words that passes between the predicted and reference texts, preserving their order. The ROUGE-L score provides a balanced measure of similarity by considering both the degree of overlap (recall) and the specificity (precision) of the prediction.

$$ROUGE - L = \frac{(1+\beta^2) * \text{Precision} * \text{Recall}}{Recall + \beta^2 * Precision}$$

$$\text{Precision} = \frac{\text{Longest Common Subsequence}}{Length\ of\ Prediciton}$$

$$\text{Recall} = \frac{\text{Longest Common Subsequence}}{Length\ of\ Reference}$$

EQ. 3

The BLEU (Bilingual Evaluation Understudy) score measures the similarity of the n-grams in the model's response to the n-grams in the reference response. BLEU formula is shown in EQ. 4 below. p_n is the n-gram match rate between the prediction and the reference, or precision. w_n is the weight for each n-gram. If the answer is too short, there is Brevity

Penalty (BP). BP which is given in the EQ. 5. is calculated based on the length of the reference and the prediction.

$$BLEU = BP * \exp\left(\sum_{n=1}^{N} w_n * \log p_n\right)$$

p_n= n-gram overlap between prediction and reference

w_n= weight

BP= brevity penalty

EQ. 4

$$BP = \begin{cases} 1 & if\ c > r \\ e^{1-\frac{r}{c}} & if\ c \leq r \end{cases}$$

EQ. 5

In other words, it measures the model's accuracy at the semantic level, not the word level. As a result, the performance of the final model was analyzed from various aspects.

C. Agentic RAG

The classical RAG architecture essentially consists of query retrieval, information retrieval, and response generation. This architecture delivers effective results in straightforward and simple question-and-answer tasks based on data contained in knowledge bases. However, the classical RAG architecture is limited in complex information requests that require multi-step reasoning or the use of external tools.

In this context, instead of the classical RAG approach, we chose the Agentic RAG architecture, which consists of independent components (agents) with flow control and decision-making capabilities. Agentic RAG analyzes user queries and utilizes not only static knowledge bases but also sources such as vector search engines and web searches to make the information retrieval process more efficient.

In the architecture shown in Fig. 3., the process begins with receiving the user's natural language query. The query first undergoes a language detection phase. If the query language is not directly supported by the model, the Translator Agent is activated in this step, translating the query into a language the model can understand. The system then checks whether the knowledge base contains content addressing this question. If the knowledge base contains an answer, the documents are split into smaller chunks (sentences, phrases, etc.) using recursive chunking. Here, we used the Recursive Character Text Splitter algorithm to split the chunks. With this algorithm, we divided each document into 500 chunks, ensuring context preservation by overlapping the chunks by 100 characters.

These parts are vectorized using a sentence embedding model provided by Hugging Face all-MiniLM-L6-v2, which offers fast and lightweight semantic embeddings [21]. The resulting embeddings are then stored in the Qdrant vector database. This model was chosen for its fast operation and high semantic similarity performance despite producing low-dimensional vectors. Similarly, the user's query is converted into a text embedding, which converts text into numerical representations, and matched with the most relevant content. The most relevant documents are retrieved by the Retriever Agent and ranked by relevance by the Grade Agent. The relevant answer is then retrieved from the most relevant document. If a direct answer is not available in the knowledge base, the system gathers information from external sources through the Web Search Agent, and the answer is returned from the web. The Duckduckgo API is used for the Web search agent. After the answer is generated, the output is checked for compatibility with the user's language; if necessary, the Translator Agent reactivates and translates the output into the user's query language.

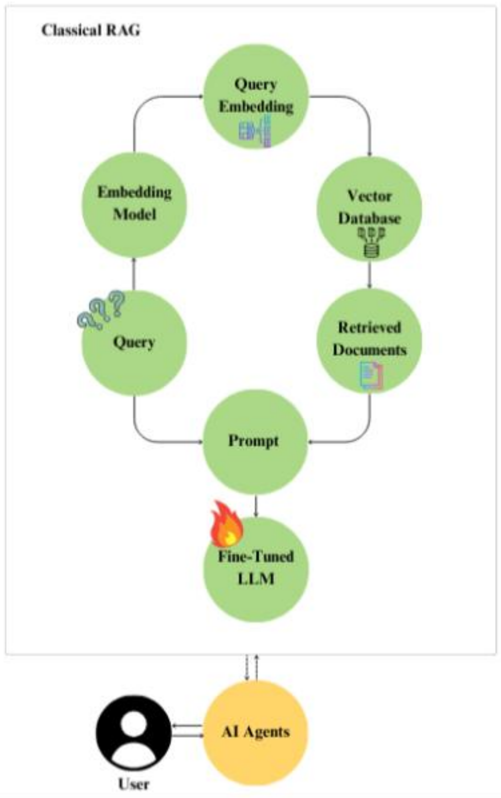


Fig. 3. Agentic RAG workflow

The Agentic RAG architecture is developed with multilingual support, dynamic knowledge acquisition, and context-specific reasoning. With this structure, our model has successfully generated relevant and accurate answers to complex user queries.

IV. EXPERIMENTS

A. Dataset Revision Process

In the first phase, the raw, unfiltered version of the NQ dataset was preferred for the fine-tuning process of the project. This version contributes a rich resource in that it includes real user queries from the Google search engine and related long answers and short answers. Nevertheless, the fact that the long answer fields were contained within HTML blocks in the unfiltered data set significantly complicated the extraction of information. Moreover, the fact that it contains irregular examples and some examples have missing or empty context and answer fields has caused serious difficulties in the data preprocessing process. Therefore, a filtered version of the NQ dataset was used in the fine-tuning process of the model.

B. Fine-Tuning

During the training process of the model, both the effective use of hardware resources and training stability were targeted; accordingly, methods such as quantization, QLoRA and memory-friendly optimization techniques were preferred.

a) Transition from CPU to A100 GPU and Memory Insufficiency

Although the initial tests were carried out on the CPU, the CPU environment was both very slow in terms of training time and the memory limits were exceeded due to the Llama-3.1 model by Meta [1] model having approximately 8 billion parameters. Therefore, a more powerful GPU-supported hardware was preferred.

As of 2025, various accelerator options such as L4, T4, TPU v6e, v5e, and v2-8 are available in the Google Colab environment; however, the NVIDIA A100 (80 GB HBM2e) [22] stands out as the clear leader in terms of computational power and memory capacity. The reasons for this situation are explained as follows. The A100 [23], [5] is approximately 5 times faster than the T4 in FP16 performance, and 5–10 times faster in INT8 mode, which is critically important for training large-scale models. The A100 offers [24] approximately 1.5 TB/s of bandwidth with 40/80 GB of HBM2e memory, which is significantly higher compared to the T4's 16 GB of GDDR6 memory and 320 GB/s bandwidth. Based on the Ampere architecture [25], the A100 delivers high performance in formats such as FP16, BF16, and TF32 through its third-generation Tensor Cores, whereas the L4 is optimized solely for inference and is not suitable for training workloads. These features [26] make the A100 a widely adopted, reliable, and cost-effective cloud solution.

Given the A100's technical superiority, it was preferred as the most powerful, stable and scalable solution for model training. Nevertheless, CUDA Out-Of-Memory (OOM) errors were observed during the full fine-tuning of an 8B-parameter model; therefore, optimizations such as PEFT with QLoRA and 4-bit quantization were employed.

b) Optimizing and Loading the Model with 4-bit Quantization

Due to the approximately 8B parameters of the model [1] causing memory constraints on the GPU, the 4-bit quantization method was applied [27]. Using the Bitsandbytes library [28], this quantization process reduced the model's memory footprint, making both model loading and fine-tuning feasible.

Quantization reduces the model size [27], [28] by approximately 75%, allowing large models to be used more efficiently even on high-end GPUs. Specifically in the QLoRA method, model compression is achieved using the 4-bit NormalFloat (NF4) format [29], which significantly reduces memory usage with minimal impact on performance. Furthermore, research [30] titled Memory-Efficient Fine-Tuning via sub-4-bit Quantization demonstrates that training adaptation is possible even though the model weights remain constant after quantization.

Within the scope of this project, the model was quantized with 4-bit NF4, reducing the model loading memory footprint by approximately 50% and making it applicable for fine-tuning. The code consists of a special method called "nf4" during the 4-bit quantization process and loading the model with 4-bit precision.Therefore, the model [1] with 80 GB A100 GPU memory and 8 B parameters has become fine-tunable [29], and OOM errors have been significantly reduced during the training process thanks to the reduced model size.

Although the model size [27] was reduced as a result of the quantization process [29] the performance loss was minimized with techniques such as QLoRA, thus retaining the model accuracy in NLP tasks to a considerable extent.

c) Token Inconsistency and Evaluation Loss

In the tokenization process of the model, the padding token was set to the same as the end of sequence or EOS token, which indicates the end of the sentence in some prior training sessions. Yet, this assignment led to incorrect loss calculations in the test and validation processes since the padding token and the end-of-sequence (EOS) token were the same. This caused the model to detect the EOS token as masked, resulting in the test metrics being reported as zero.

According to the Hugging Face forum [31], this operation may cause the model to fail to produce an EOS token [32], resulting in continuous text generation. Additionally, it has been pointed out that setting the pad token [33],[34] equal to the EOS token means the EOS token is not backpropagated, which may cause the model to lose its ability to stop generation.

Through the new method, pad token is configured as a cleaned and isolated token, separated from EOS token. After this correction, the model calculated the appropriate loss function during test and validation, and the test results reached the expected values.

This solution [32] requires cautious management of pad and EOS tokens, particularly in decoder-only models, otherwise the correct functioning of the loss function may be disrupted.

d) Transition from LoRA to QLoRA: Memory and Performance Optimization

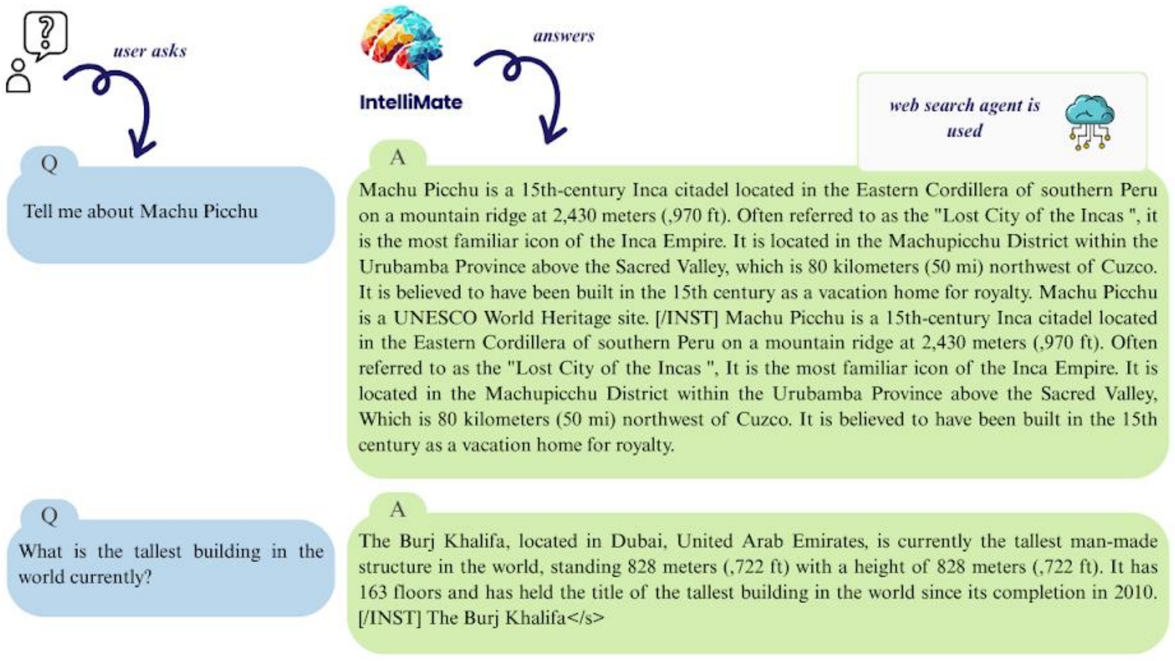
In the initial stages, the model was fine-tuned using the LoRA method. Yet, this method [35] has been noticed to induce high GPU memory consumption, particularly in LLMs (for instance, the 8B parameter Llama-3.1 model by Meta). Therefore, the QLoRA method, which is more memory-friendly and efficient, was preferred.

QLoRA represents the model's weights with 4-bit quantization [5], significantly reducing GPU memory usage. This technique performs low-rank updates on adapter layers while freezing the core parameters of the model, thereby minimizing memory overhead. According to Dettmers et al. (2023), 65-billion parameter models were fine-tuned on a single 48 GB GPU, achieving results comparable in accuracy to those obtained using LoRA.

In conclusion, within the scope of the project, a transition to QLoRA was made, which resulted in more efficient utilization of GPU resources without any significant decrease in model accuracy.

e) Transition from IFT to SFT and Dataset Adjustment Processes

At the beginning of the project, the use of the IFT method [3] was planned; however, IFT required rigorous preparation of each example in a specific instruction format (e.g., system, user roles, etc.).This process [4] has posed challenges , particularly in making it applicable to large-scale Q&A based datasets (NQ). On the contrary, in SFT, data preparation is significantly simplified as it can work directly with the "instruction-input-output" structure in each example.



Tell me about Machu Picchu

Machu Picchu is a 15th-century Inca citadel located in the Eastern Cordillera of southern Peru on a mountain ridge at 2,430 meters (,970 ft). Often referred to as the "Lost City of the Incas ", it is the most familiar icon of the Inca Empire. It is located in the Machupicchu District within the Urubamba Province above the Sacred Valley, which is 80 kilometers (50 mi) northwest of Cuzco. It is believed to have been built in the 15th century as a vacation home for royalty. Machu Picchu is a UNESCO World Heritage site. [/INST] Machu Picchu is a 15th-century Inca citadel located in the Eastern Cordillera of southern Peru on a mountain ridge at 2,430 meters (,970 ft). Often referred to as the "Lost City of the Incas ", It is the most familiar icon of the Inca Empire. It is located in the Machupicchu District within the Urubamba Province above the Sacred Valley, Which is 80 kilometers (50 mi) northwest of Cuzco. It is believed to have been built in the 15th century as a vacation home for royalty.

What is the tallest building in the world currently?

The Burj Khalifa, located in Dubai, United Arab Emirates, is currently the tallest man-made structure in the world, standing 828 meters (,722 ft) with a height of 828 meters (,722 ft). It has 163 floors and has held the title of the tallest building in the world since its completion in 2010. [/INST] The Burj Khalifa</s>

Fig. 4. Web search agent usage

This scenario is also emphasized in academic studies. According to Li et al. [4], it has been demonstrated that through SFT, the model can be easily adapted to classification or Q&A tasks, and the data preparation process is less complex compared to IFT. According to Zhang et al. [3], the SFT and IFT methods are technically very similar; however, SFT is noted to be more user-friendly in practical applications, especially when working with labeled training data.

For this reason, SFT has been preferred in the project. Instead of the complex data format required in the IFT process, the model was trained with supervised labeled examples that were adapted to the "instruction-input-output" structure, thus making the training process easier and maintaining high Q&A task performance.

f) Configuration of Training Arguments Parameters

The Training Arguments parameters used in the fine-tuning process of the model have been thoroughly optimized to balance both GPU memory constraints, training time, and to achieve the best test results.

The batch size was set to 4. According to Marek [36], smaller batch sizes both reduce GPU memory consumption and limit the risk of overfitting. It is also a common method to prevent OOM (Out-of-Memory) errors in large models.

The gradient accumulation value was set to 4. As noted by Huang [37], the accumulation technique enables efficient memory usage, particularly in large models, by grouping gradient updates before applying them. With this setting, an effective batch size of 16 (4 × 4) was maintained, allowing training to proceed without overloading the GPU.

The learning rate value was set to 0.001. This ratio has been specially adjusted to shorten the training time and increase the learning speed. Based on Jin's research [38], Low rates can cause the model to learn too slowly, while high rates can destabilize it. The selected ratio is the optimized version of this balance.

The value of warmup steps was set to 100 and the value of learning scheduler type was set to linear. With the linear scheduler, the learning rate was gradually increased at the beginning of the training process, then decreased. According to the findings of Li et al. [39], this technique provides a more stable training process and prevents sudden deviations.

The number of training epochs were set to *1*. Recent research [40] shows that even a single-epoch fine-tuning can lead to significant performance gains in large language models when learning rate scaling techniques like SGG are applied effectively. Considering the GPU capacity and time constraint, training was done with only one epoch.

The value of eval steps was set to 200, save steps was set to 200 and logging steps was set to 50. These values have been determined in a way that balances both regular monitoring of the model and time management. As noted by [41], since taking too many recordings would be inefficient in terms of time and disk space, these values kept the training time at a reasonable level.

The values of the parameters were set to epoch as the evaluation strategy and the evaluation process as effective. As noted by [41], an epoch-based strategy was applied to evaluate the training at the end.

The value of the parameters was set to AdamW with 8bit as a memory-friendly optimization method. In order to save memory, the paged version of the 8-bit AdamW algorithm was preferred [42]. According to [43], this optimizer prevents memory overflows while also keeping the training process stable.

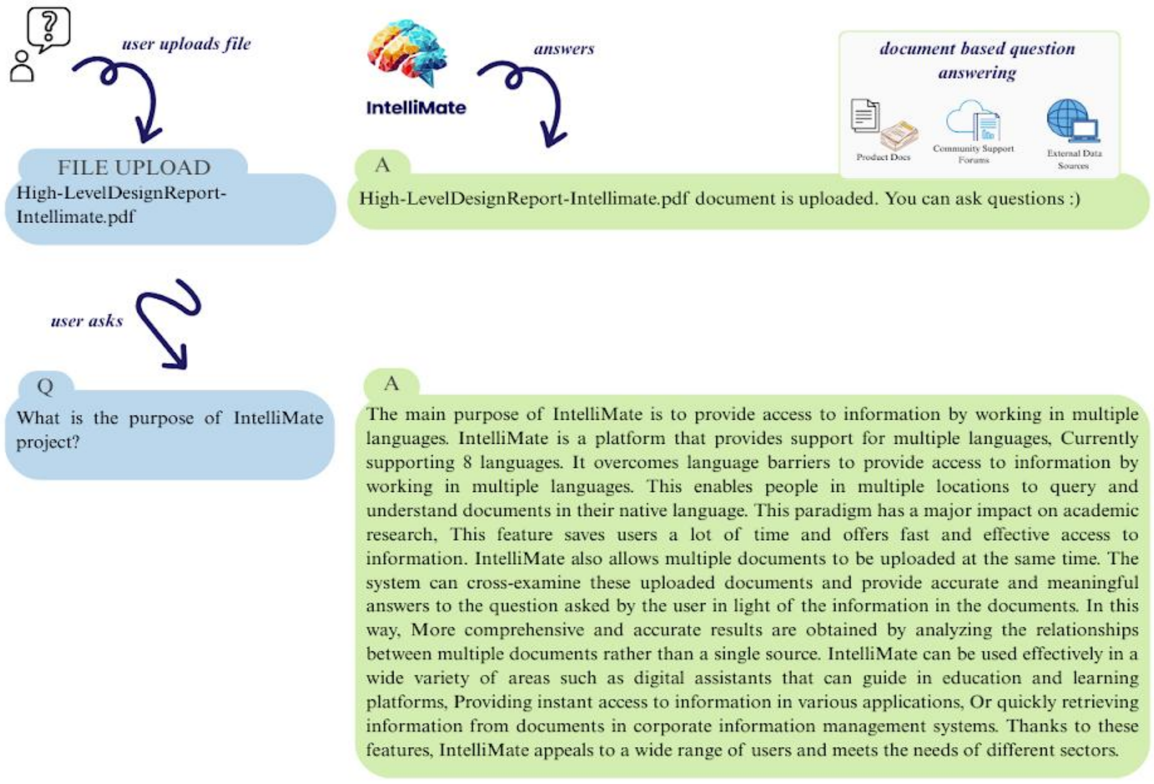
C. Agentic RAG Integration to The Model

A comprehensive agentic RAG pipeline significantly improves the model's performance and user experience. This pipeline also allows the project to integrate diverse applications, such as multilingual support, web search, and hallucination checks, rather than just document-based responses.

systematically. This systematic work has created the opportunity to communicate with the model multilingually, including in Turkish, for both document-based and non-document questions. Furthermore, implemented controls such as hallucination did not enable the model to provide more human-like and consistent responses. Test results for the model with and without agentic RAG can be seen below.

Examples of results of web search agent usage without hallucination check can be seen in the Fig. 4. Web search agent usage As can be figured out from the example tests in Fig. 4, answers should be filtered, and hallucination check should be applied. Example of test after the High-Level Design Report-Intellimate.pdf document uploaded can be seen in the Fig. 5.

D. Model Selection Analyses



FILE UPLOAD
High-LevelDesignReport-Intellimate.pdf

A
High-LevelDesignReport-Intellimate.pdf document is uploaded. You can ask questions :)

Q
What is the purpose of IntelliMate project?

A
The main purpose of IntelliMate is to provide access to information by working in multiple languages. IntelliMate is a platform that provides support for multiple languages, Currently supporting 8 languages. It overcomes language barriers to provide access to information by working in multiple languages. This enables people in multiple locations to query and understand documents in their native language. This paradigm has a major impact on academic research, This feature saves users a lot of time and offers fast and effective access to information. IntelliMate also allows multiple documents to be uploaded at the same time. The system can cross-examine these uploaded documents and provide accurate and meaningful answers to the question asked by the user in light of the information in the documents. In this way, More comprehensive and accurate results are obtained by analyzing the relationships between multiple documents rather than a single source. IntelliMate can be used effectively in a wide variety of areas such as digital assistants that can guide in education and learning platforms, Providing instant access to information in various applications, Or quickly retrieving information from documents in corporate information management systems. Thanks to these features, IntelliMate appeals to a wide range of users and meets the needs of different sectors.

Fig. 5. Document based agentic RAG usage

Although the pre-trained version of the model included support for eight different languages, it lacked Turkish. Therefore, a translation agent was included in the pipeline. Furthermore, despite the pre-trained model being context-based and the SFT-type fine-tuning process applied using the Natural Questions dataset to improve context-based performance, agentic RAG enabled the model to have a more systematic decision-making mechanism and enabled the model to converse with historical information. In addition, it provides generate answers to questions that do not have an answer in the previously constructed vector database and uploaded documents using web search agent in case of requirement.

In conclusion, it can be said that the agentic RAG pipeline implemented enabled the fine-tuned model to function more

To determine the most suitable model for IntelliMate, DistilBERT, BERT, and Llama 3.1 models were tested and compared with the Filtered Natural Question dataset. Four primary metrics were used in the evaluation: Exact Match (EM), ROUGE-L, BLEU, and F1 score. Exact Match measured whether the model's answer matched the target. ROUGE-L assessed the structural integrity of the model's responses and their relationship to the reference response. The BLEU metric demonstrated word-level accuracy and keyword selection, while the F1 score demonstrated the model's semantic closeness. These metrics were used as the primary criteria to determine which model was more suitable for the task.

Analysis revealed that Llama 3.1 outperformed the other models in all metrics (see Table 1). The reason for the Llama model's higher EM and F1 scores compared to other models is its high parameter count and context-processing power, which allow it to accurately understand complex questions and provide strong predictive capabilities. Moreover, the superior ROUGE-L and BLEU scores compared to other models are related to the model's ability to produce consistent answers and select the right keywords. On the other hand, it was found that BERT performed better than DistilBERT across all metrics. This outcome has been linked to BERT's deeper and broader architecture, which enables a more accurate representation of grammatical and contextual relations. DistilBERT, designed with speed and efficiency in mind, was noted to perform behind the other two models in these metrics. The comparison revealed that Llama 3.1 surpasses both BERT and DistilBERT in both parameter size and enhanced contextual representation. Therefore, it was decided to continue the work with Llama 3.1.

V. Results

We implemented the BERT and DistilBERT models and Llama 3.1, during the beginning of our study, while choosing the models and evaluated their performance. The performance of the models is presented in Table 1 for comparison purposes. The comparisons determined that the Llama 3.1 model surpassed other models on various measures.

Finally, the accuracy of our targeted Llama 3.1 model was evaluated using four main metrics: EM, ROUGE-L, BLEU, and F1. From the values in Table 1, the model was extremely good on all the metrics. EM was determined to be 62%, showing an extreme accuracy rate. From Table 1, the ROUGE-L and BLEU were 80% and 50%, respectively. These results vindicated the extent to which the model could provide valid responses and effectively utilize keywords. Finally, the F1 score of 85% validated that the model validated to possess a consistent performance between lexical and semantic accuracy.

After model selection and optimizing Llama 3.1, the model's precision and recall percentages were calculated to be 85%. This revealed that the model responses were relevant. From the data in Table 1, the BERT and DistilBERT models performed relatively less effectively as compared to Llama 3.1. Particularly, DistilBERT outperformed BERT with 41% EM, 60% ROUGE-L, 19% BLEU and 81% F1 score, while BERT had 45% EM, 68% ROUGE-L, 35% BLEU and 83% F1 score.

Table 1. Model performance across EM, ROUGE-L, BLEU, and F1 scores

Model	Exact Match (EM)	ROUGE-L	BLEU	F1 Score
DistilBERT	41%	60%	19%	81%
BERT	45%	68%	35%	83%
Llama 3.1 (IntelliMate)	62%	80%	50%	85%

With the integration of the Agentic RAG architecture into our model, our model began to provide more effective answers. Before Agentic RAG, the model failed to answer some questions or produced blank answers. This was primarily due to the model operating solely on the parameters it was trained on and not having access to external resources. Agentic RAG eliminates these issues. Thanks to the Web Search Agent, the model can retrieve information from the web when it cannot find an answer in the knowledge base. For example, while the model previously could not answer a question like "When was the Republic of Türkiye founded?", it can now accurately answer "October 29, 1923" using information retrieved from the web. Furthermore, the ability to generate answers by analyzing the content of uploaded documents has increased the accuracy of document-based questions. For example, the answer to the question "What is the purpose of the IntelliMate project?" was taken directly from the uploaded document and presented contextually. Furthermore, the model previously supported only eight languages. Thanks to the Translator Agent built into the Agentic RAG architecture, the model can now understand queries in all languages and translate responses into the desired language. Ultimately, the integration with Agentic RAG has significantly enhanced the model's multilingual support, retrieval of information from external sources, and the generation of more contextual answers.

Four metrics were used to evaluate the effect of Agentic RAG integration on model performance. Answerability Rate quantifies how well and meaningfully model responses can be produced. Hallucination Rate indicates the frequency with which the model generates out-of-context, inaccurate information. When these two metrics are evaluated together, they reveal the model's accuracy and reliability. Average Response Time represents the average time it takes for the model to respond to a query and is critical to system efficiency. Finally, Language Support indicates the number of languages supported by the model, reflecting the system's multilingual capacity and ability to serve different user groups.

The quantitative effects of Agentic RAG integration on the fine-tuned model are presented in Table 2. While the model's answerability rate was 67% without RAG, this rate increased to 72% with RAG integration. Similarly, the hallucination rate decreased from 48% to 31%. Furthermore, while the average response time was 8.42 seconds in the No RAG scenario, it decreased to 4.63 seconds with RAG, representing an improvement of approximately 45%. These results indicate Agentic RAG remarkably contributes to the model in terms of accuracy, speed, and multilingualism.

Table *2*. Comparison of Model Performance With and Without Agentic RAG

Metric	No Agentic RAG	With Agentic RAG
Answerability Rate	67%	72%
Hallucination Rate	48%	31%
Average Response Time	8.42 s	4.63 s
Language Support	8 languages	All Languages

Recent studies indicate that RAG systems consistently outperform stand-alone fine-tuned LLMs in both accuracy and reliability. Lakatos et al.[45] shows distinguished ROUGE and

BLEU improvements when RAG is applied, while Rashtchian et al. [46] demonstrate that external context considerably reduces hallucinations in LLM outputs. Consistent with this finding, our approach not only achieves similar improvements but also extends the evaluation with practical metrics such as answerability rate, hallucination reduction, multilingual support, and response latency. By integrating LLaMA, QLoRA-based SFT, and a multi-agent RAG workflow within a unified architecture, IntelliMate delivers a wider and designed more applicable evaluation compared to existing RAG studies.

In addition to the improvements in accuracy and efficiency, the experimental findings provide strong empirical evidence supporting the originality of the proposed IntelliMate pipeline. Unlike previous works that evaluate only isolated components such as SFT or basic RAG, our experiments indicate the impact of integrating LLaMA, QLoRA-based SFT, and a multi-agent RAG architecture within a unified system. The results in Table 2 show that this integrated design yields substantial gains in answerability (+5%), a remarkable reduction in hallucination rate (–17%), and nearly a duplicate improvement in response time compared with the non-RAG architecture. Therefore, this results empirically confirm that the proposed end-to-end architecture introduces a novel and effective interaction between fine-tuning, retrieval, multilingual processing, and agent-based reasoning. This study has not been experimentally demonstrated in prior literature, and the empirical results of this study provide the strong evidence supporting its effectiveness.

VI. CONCLUSION

In conclusion, the IntelliMate study provided an environment that could work in both web and mobile applications. The large language model Llama integrated into this project, was first fine-tuned using the SFT method after quantization. The fine-tuned model is used to achieve usable performance within applications applying the agentic RAG architecture; this process enables the model to answer both document-based and non-document-based questions with multilingual support. This study adds a significant perspective to the literature by combining various methodologies. The first aspect of the contribution to the literature is about using the SFT method with NQ dataset which provides a high increase in performance. This study contributes to the literature by showing that it is possible to achieve high performance with low resources and that QLoRA displays a significant role in this process. The other contribution to the literature is about using agentic RAG methodology in a multi-step way to get a high and systematic performance to answer the questions properly. The special agentic RAG application implemented in the study fills an important gap in the literature. This study presents an end-to-end pipeline that simultaneously combines quantized LLaMA fine-tuned with QLoRA/SFT and a multi-agent RAG architecture. Such a fully integrated system is rarely explored in existing literature. We believe that this study will be a reference point and a different perspective for further studies on LLM, SFT and agentic RAG in the literature.

Authors' Contributions

The authors' contributions to the paper are equal.

Statement of Conflicts of Interest

There is no conflict of interest between the authors.

Statement of Research and Publication Ethics

The authors declare that this study complies with Research and Publication Ethics.

REFERENCES

[1] Meta AI, *Llama 3.1 8B Instruct*, 2024. [Online]. Available: https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct [Accessed: Jul. 17, 2025].

[2] K. Kwiatkowski, et al., "Natural Questions:a Benchmark for Question Answering Research," *Trans. Assoc. Comput. Linguistics*, vol. 7, pp. 453–466, 2019.

[3] S. Zhang et al., "Instruction Tuning for Large Language Models: A Survey,"arXiv,Aug.21,2023.[Online]. Available:https://arxiv.org/abs/2308.10792

[4] Z. Li et al., "Label Supervised Llama Finetuning (LS- Llama)," arXiv preprint 2310.01208, Oct. 2023. [Online]. Available: https://arxiv.org/abs/2310.01208

[5] T. Dettmers, "The Best GPUs for Deep Learning in 2023," Tim DettmersBlog,2023.[Online].Available:https://timdettmers.com/2023/01/30/which-gpu-for-deep-learning/ [Accessed: 18-Jul-2025].

[6] J. Liang, G. Su, H. Lin, Y. Wu, R. Zhao, and Z. Li, "Reasoning RAG via System 1 or System 2: A Survey on Reasoning Agentic Retrieval-Augmented Generation for Industry Challenges," *arXiv preprint*, arXiv:2506.10408, Jun. 2025. [Online]. Available: https://arxiv.org/abs/2506.10408

[7] "Natural Questions Filtered Dataset," Kaggle, [Online]. Available: https://www.kaggle.com/datasets/allen-institute-for-ai/natural-questions.

[8] TensorFlow Datasets, "natural_questions_open" dataset, Splits: train (87,925), validation (3,610),2022 [Online]. Available:https://www.tensorflow.org/datasets/catalog/natural_questions_open.

[9] K. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ Questions for Machine Comprehension of Text," in *Proc. EMNLP*, 2016.

[10] datarpit, *distilbert-base-uncased-finetuned-natural-questions model card*,2022.[Online].Available: https://huggingface.co/datarpit/distilbert-base-uncased-finetuned- natural-questions

[11] K. Lee, M.-W. Chang, and K. Toutanova, "Latent Retrieval for Weakly Supervised Open Domain Question Answering," in *Proc. ACL*, pp. 6086–6096, Jul. 2019.

[12] Reece Shuttleworth, Jacob Andreas, Antonio Torralba, Pratyusha Sharma. "LoRA vs Full Fine-tuning: An Illusion of Equivalence." MIT CSAIL, 2023. [Online]. Available: https://arxiv.org/abs/2410.21228

[13] Tejaskumar Pujari, Anshul Goel, Ashwin Sharma. "Ensuring Responsible AI: The Role of Supervised Fine-Tuning (SFT) in Upholding Integrity and Privacy Regulations." Independent Researchers,India,2023.[Online].Available:https://www.researchgate.net/profile/TejaskumarPujari/publication/390944911_Ensuring_Responsible_AI_The_Role_of_Supervised_FineTuning_SFT_in_Upholding_Integrity_and_Privacy_Regulations/links/68085448bd3f1930dd630e4f/Ensuring-Responsible-AI-The-Role-of-Supervised-Fine-Tuning-SFT-in-Upholding-Integrity-and-Privacy-Regulations.pdf

[14] Joey Hong, Anca Dragan, Sergey Levine. "Q-SFT: Q-Learning for Language Models via Supervised Fine-Tuning." University of California,Berkeley,2023.[Online].Available:https://arxiv.org/abs/2411.05193

[15] Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, Roy Ka-Wei Lee. "LLM-Adapters: An Adapter Family for Parameter-Efficient Fine-Tuning of Large Language Models." Singapore University of Technology and Design, 2023. [Online]. Available: https://arxiv.org/abs/2304.01933

[16] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, Luke Zettlemoyer. "QLoRA: Efficient Finetuning of Quantized LLMs." University of Washington,2023.[Online].Available:https://proceedings.neurips.cc/paper_files/paper/2023/hash/1feb87871436031bdc0f2beaa62a049b-Abstract-Conference.html

[17] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Riedel, S. (2021). Retrieval-augmented generation for knowledge-intensive NLP tasks. *arXiv preprint* arXiv:2005.11401. https://arxiv.org/pdf/2005.11401.pdf

[18] Barnett, S., Kurniawan, S., Thudumu, S., Brannelly, Z., & Abdelrazek, M. (2024). Seven failure points when engineering a retrieval augmented generation system. *arXiv preprint* arXiv:2401.05856. https://arxiv.org/pdf/2401.05856.pdf

[19] Singh, A., Ehtesham, A., Kumar, S., & Khoei, T. T. (2025). Agentic retrieval-augmented generation: A survey on agentic RAG. *arXiv preprint* arXiv:2501.09136. https://arxiv.org/pdf/2501.09136.pdf

[20] D. Patterson *et al*., "The Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink," *arXiv preprint arXiv:2204.05149*, 2022. [Online]. Available: https://arxiv.org/abs/2204.05149

[21] Y. Wang, F. Wei, S. Feng, M. Wang, and M. Zhou, MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers, in Advances in Neural Information Processing Systems *33 (NeurIPS 2020) Proceedings*, 2020, *arXiv preprint arXiv:2002.10957*.[Online].Available:https://arxiv.org/abs/2002.10957

[22] "Comparing Enterprise NVIDIA GPUs: A100, H100, T4...," Server-Parts.eu, 2024. [Online]. Available: https://www.server-parts.eu/post/nvidia-gpu. [Accessed: 18-Jul-2025].

[23] O. Wehrens, "OpenAI Whisper Benchmark Nvidia Tesla T4 / A100," Owehrens.com,Jan.2023.[Online].Available:https://owehrens.com/openai-whisper-benchmark-on-nvidia-tesla-t4-a100/ [Accessed: 18-Jul-2025].

[24] "NVIDIA A100 Tensor Core GPU Architecture In-Depth," NVIDIA Blog,2020.[Online].Available:https://blogs.nvidia.com/blog/2020/05/14/nvidia-a100/. [Accessed: 18-Jul-2025].

[25] "NVIDIA L4 vs. A100 GPUs: Choosing the Right Option...," e2enetworks.com,2024.[Online].Available:https://e2enetworks.com/blog/nvidia-l4-vs-a100-gpus. [Accessed: 18-Jul-2025].

[26] Baseten, "NVIDIA A10 vs A100 GPUs for LLM and Stable Diffusion Inference," Baseten Blog, 2024. [Online]. Available: https://www.baseten.co/blog/nvidia-a10-vs-a100-gpus-for-llm-and-stable-diffusion-inferenc/ [Accessed: 18-Jul-2025].

[27] Hugging Face, "Making LLMs even more accessible with bitsandbytes, 4-bit quantization and QLoRA," *Hugging Face Blog*, May 24, 2023. [Online]. Available: https://huggingface.co/blog/4bit-transformers-bitsandbytes

[28] A. Coelho, "Quantization in LLMs: Why Does It Matter?", *Dataiku Blog*,Jan 10, 2024.[Online].Available:https://blog.dataiku.com/quantization-in-llms-why-does-it-matter/

[29] S. Bajpai, "Shrinking Elephants: A Funny Guide to 4bit and 8bit Quantization for LLMs with LoRA," *Medium*, Sep 27, 2024. [Online].Available: https://medium.com/@shikharstruck/shrinking-elephants-a-funny-guide-to-4-bit-and-8-bit-quantization-for-llms-with-lora-ddf9f1a62070

[30] J. Kim, J. Lee, and S. Moon, "MemoryEfficient FineTuning of Compressed Large Language Models via sub4bit Integer Quantization," *arXiv preprint*, arXiv:2305.14152, May 23, 2023. [Online]. Available: https://arxiv.org/abs/2305.14152

[31] "Why does the falcon QLoRA tutorial code use eos_token as pad_token?", Hugging Face Discuss, Jul. 7, 2023. [Online]. Available: https://discuss.huggingface.co/t/why-does-the-falcon-qlora-tutorial-code-use-eos-token-as-pad-token/45954

[32] "Finetuning GPT: problems with padding", Issue #8452, Hugging FaceTransformers,Oct. 2019.[Online].Available:https://github.com/huggingface/transformers/issues/8452

[33] Brando_Miranda, "How does one set the pad token correctly (not to eos) during finetuning to avoid model not predicting EOS?", PyTorch Forums,Nov. 29, 2024.[Online].Available:https://discuss.pytorch.org/t/how-does-one-set-the-pad-token-correctly-not-to-eos-during-finetuning-to-avoid-model-notpredictingeos/213619

[34] mlabonne, "Issue with pad_token == eos_token: model not 'learning when to stop'", GitHub, Apr. 17, 2023. [Online]. Available: https://github.com/mlabonne/llm-course/issues/33

[35] E. J. Hu et al., "LoRA: LowRank Adaptation of Large Language Models," *ICLR*, Feb. 2021. [Online]. Available: https://arxiv.org/abs/2106.09685

[36] M. Marek et al., "Small Batch Size Training for Language Models," *arXiv preprint*, Jul. 2025. [Online]. Available: https://arxiv.org/abs/2507.07101

[37] Z. Huang et al., "Measuring the Impact of Gradient Accumulation on Cloud-Based Distributed Training," *CCGrid 2023*. [Online]. Available: https://www.researchgate.net/publication/372262958_Measuring_the_Impact_of_Gradient_Accumulation_on_Cloud-based_Distributed_Training

[38] H. Jin, W. Wei, X. Wang, W. Zhang, and Y. Wu, "Rethinking Learning Rate Tuning in the Era of Large Language Models," *arXiv preprint arXiv:2309.08859*, Sep. 2023. [Online]. Available: https://arxiv.org/abs/2309.08859

[39] C. Li et al., "The Stability-Efficiency Dilemma: Investigating Sequence Length Warmup for Training GPT Models," *arXiv*, Aug. 2021. [Online]. Available: https://arxiv.org/abs/2108.06084

[40] S. Li et al., "Taming LLMs by Scaling Learning Rates with Gradient Grouping," *arXiv preprint*, Jun. 2025. [Online]. Available: https://arxiv.org/abs/2506.01049

[41] Hugging Face, "Trainer Arguments Explained," *Hugging Face Docs*, 2025.[Online].Available:https://huggingface.co/docs/transformers/main_classes/trainer

[42] T. Dettmers et al., "8-bit Optimizers via Block-wise Quantization," *arXivpreprint*,Dec. 2021.[Online].Available:https://arxiv.org/abs/2110.02861

[43] Hugging Face, "8-bit Optimizers and Memory Efficiency," *Hugging FaceDocs*,2024.[Online].Available:https://huggingface.co/docs/bitsandbytes/optimizers

[44] FoundationAI,"Llama-3.1-FoundationAI-SecurityLLM-8B-Instruct Technical Report," Technical Report, 2024. [Online]. Available: https://www.arxiv.org/pdf/2508.01059

[45] R. Lakatos, P. Pollner, A. Hajdu, and T. Joo, "Investigating the Performance of Retrieval-Augmented Generation and Fine-Tuning for the Development of AI-Driven Knowledge-Based Systems," arXiv preprint, 2024. [Online]. Available: https://arxiv.org/abs/2403.09727

[46] C. Rashtchian, H. Joren, J. Zhang, C.-S. Ferng, D.-C. Juan, and A. Taly, "Sufficient Context: A New Lens on Retrieval-Augmented Generation Systems," arXiv preprint, 2025. [Online]. Available: https://arxiv.org/abs/2411.06037