A Comparative Study of RAG and Fine-Tuned Transformer Models for Domain-Specific Chatbots

Taksh Dhabalia
TY CSE CORE
MIT WPU
Pune, India
dhabalia.taksh@gmail.com

Samanyu Bhate
TY CSE CORE
MIT WPU
Pune, India
bhatesamanyu@gmail.com

Kushagra Singh
TY CSE CORE
MIT WPU
Pune, India
kushagraa.n@gmail.com

Brandon Cerejo
TY CSE CORE
MIT WPU
Pune, India
brandoncerejo39@gmail.com

Abstract—**This paper explores the development of a customized chatbot for interactive website support, implemented using two separate methodologies: a Retrieval-Augmented Generation (RAG) framework and a fine-tuned transformer-based model. The chatbot is designed to provide accurate, domain-specific responses for the IRIS Club at MIT-WPU, handling questions about club activities, projects, events, and member information. We perform a comparative analysis of the two methods based on factors such as implementation difficulty, response accuracy, and resource consumption. The RAG-based solution integrates LangChain, HuggingFace embeddings, and the LLaMA-3 model accessed via Groq's API, whereas the alternative leverages a DistilBERT model trained on relevant data. To evaluate performance, we use metrics like cosine similarity, BERT score, ROUGE scores, and Flesch-Kincaid readability. Our findings indicate that both strategies are viable, each offering distinct advantages and limitations in terms of precision, ease of development, and computational efficiency.**

Index Terms—**Conversational AI, Retrieval-Augmented Generation, Transformer Models, Large Language Models, NLP, Domain-specific Chatbot**

I. Introduction

The rapid advancement of Large Language Models (LLMs) and transformer architectures has significantly reshaped the field of natural language processing (NLP), particularly in the development of conversational AI systems. At the core of this revolution lies the transformer. Transformers are a type of neural network architecture that can process sequential data through a self-attention mechanism. This enables the models to understand the context and relationship between elements in a sequence.

Large Language Models (LLMs) are the most advanced outcome of this innovation. LLM models, such as GPT-4, LLaMA, and Claude are massive transformer-based models typically containing billions of parameters that are first pre-trained on an extensive corpora covering diverse domains of human knowledge. This pre-training phase enables them to develop a comprehensive understanding of language structures, semantics, and world knowledge. Once trained, they can be used for a variety of tasks and can be fine-tuned for specific domains to achieve even greater performance.

In parallel, conversational agents have also evolved from simple rule-based systems to highly developed AI assistants capable of understanding context, maintaining conversation history, and providing accurate domain-specific information. This paper explores two strategies to building specialized chatbots for domain-focused applications, systems designed to operate within constrained knowledge domains rather than as general-purpose conversational agents. The first approach implements a Retrieval-Augmented Generation (RAG) pipeline, while the second utilizes a fine-tuned transformer model, both optimized for targeted use cases.

Domain-specific chatbots offer both unique benefits and challenges. On the one hand, they are expected to demonstrate deep expertise, handle specialized vocabulary accurately, and remain focused on the relevant subject matter. On the other hand, designing such systems requires careful attention to scope and precision. When done effectively, these chatbots can outperform general-purpose models in scenarios such as technical customer support or enterprise knowledge management. This paper presents the development of a domain-specific chatbot tailored for the IRIS Club at MIT-WPU, designed to answer queries related to the club's events, projects, and member activities.

II. Background

To establish the technical foundation for this work, we first look at the core technologies and methodologies used in our domain-specific chatbot implementation. We first start by looking into transformer language model and the Large Language Models (LLMs). We then look into detail about the specific tools used in both our Retrieval-Augmented Generation (RAG) pipeline and fine-tuned transformer approaches.

A. Transformer Language Model

The Transformer model changed how we handle sequences like sentences by removing the need for traditional methods like RNNs and CNNs. Instead, it uses attention mechanisms to understand relationships between words, no matter how far apart they are in the sequence.

At its core, the model uses an encoder-decoder structure [1]. Both the encoder and decoder consist of six stacked layers. Each encoder layer contains two main components: multi-head self-attention and a feed-forward neural network. These components are connected using residual links followed by

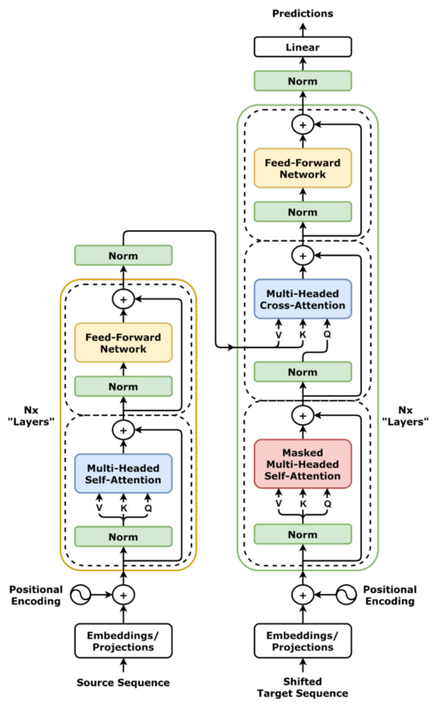

Fig. 1: Transformer Architecture [1]

layer normalization. The output dimensionality throughout the model is fixed at 512.

The decoder also has 6 layers, but with one extra layer in each that lets it look at the encoder's output (called encoder-decoder attention). It also uses something called masked self-attention, which means during training, it can only look at the words that come before the current one, this stops it from "*cheating*" and seeing future words.

The heart of the Transformer is its attention mechanism, which helps the model decide which parts of the input are most important. The formula used is:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

where Q, K, and V represent queries, keys, and values respectively, and d_k is the size of the key vectors. Dividing by d_k makes sure the values don't get too large, which could otherwise make learning unstable.

Instead of using a single attention function, the Transformer splits the attention process into multiple heads. This is called multi-head attention, and it lets the model learn different types of word relationships at the same time.

Since this model doesn't use sequences in the traditional sense, it needs a way to understand word order. To solve this, it adds positional encodings to the input. These are based on sine and cosine functions:

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{\frac{2i}{d_\text{model}}}}\right) \tag{2}$$

$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{\frac{2i}{d_\text{model}}}}\right) \tag{3}$$

These patterns help the model know the position of each word in a sentence. In short, the Transformer is fast and efficient because it doesn't rely on step-by-step processing like RNNs. It can handle full sentences all at once, learns multiple kinds of relationships using multi-head attention, and understands word order through positional encoding. This design is why Transformers power models like GPT, BERT, and many others used in today's AI applications.

B. Large Language Models and Architecture

Modern Large Language Models (LLMs) are the result of continuous progress in the field of natural language processing, built primarily on the transformer architecture. These models exhibit advanced capabilities such as contextual comprehension, multi-step reasoning, and adaptability to specific domains through fine-tuning. The evolution from early statistical models to LLMs can be attributed to three significant advancements:

1) Parameter Scaling: Current LLMs contain billions (and sometimes trillions) of parameters, enabling them to recognize complex patterns and represent knowledge with great accuracy.
2) Architectural Innovations: Techniques such as grouped-query attention, as seen in models like LLaMA-3, and mixture-of-experts architectures have improved model efficiency and scalability.
3) Training Paradigms: Two-stage training (pretraining followed by fine-tuning) coupled with reinforcement learning from human feedback (RLHF) has enhanced the quality and relevance of generated outputs.

C. BERT and Distilled BERT for Question Answering

Our custom fine-tuned model for implementing a chatbot through transformers is based on DistilBERT, which is a lighter and faster variant of BERT that maintains approximately 97% of BERT's performance while reducing the model size by 40% [2]. This makes it well-suited for deployment in resource-constrained environments. Some of its features include:

1) Bidirectional Context:Unlike decoder-only architectures (e.g., GPT), BERT and DistilBERT process text bidirectionally, enabling deeper contextual understanding. On the SQuAD 1.1 benchmark, BERT achieves an F1 score of 93.2%, while DistilBERT retains 91.3% with fewer parameters [2], [3].
2) SQuAD Optimization: We use the distilbert-base-cased-distilled-squad variant, pre-fine-tuned on the Stanford Question Answering Dataset (SQuAD), where it achieves an 85.1 F1 score [2]. For domain adaptation, we further fine-tune the model using a span-based loss function:

$$L = L_\text{start} + L_\text{end} \tag{4}$$

Here, L_{start} and L_{end} represent the cross-entropy losses for predicting the starting and ending positions of the answer in the context passage. By minimizing this combined loss during training, the model becomes more accurate at identifying the correct span of text that answers a given question.

D. The Llama 3.3-70B Versatile Model

For the generation component of our RAG pipeline, we leverage Meta's LLaMA-3 70B model via GroqCloud's API. This model is designed for scalable deployment while retaining strong reasoning capabilities, thanks to its 70 billion parameters. It also includes additional features such as:

- Grouped-Query Attention: Enhances inference efficiency by lowering memory demands without compromising model performance.
- Extended Context Window: Supports inputs up to 8,000 tokens, enabling more coherent responses over longer conversations or documents.
- Open Weights: As an open-source model, LLaMA-3 provides full transparency and flexibility in customization, unlike closed models such as GPT-4.
- Versatile Task Handling: Particularly effective for instruction-following tasks which is essential in chatbot applications

This combination of high parameter count and optimized transformer design allows LLaMA-3 70B to balance reasoning tasks with practical deployment across various real-world NLP tasks.

E. FAISS (Facebook AI Similarity Search)

FAISS (Facebook AI Similarity Search) is a powerful library designed for fast and scalable similarity search in high-dimensional vector spaces. It employs techniques such as product quantization, which compresses vectors to reduce memory usage by up to 8x while maintaining $\tilde{9}0\%$ accuracy in nearest-neighbor search [4] . Additionally, it uses an inverted file indexing structure, allowing for efficient approximate nearest neighbor (ANN) lookups by grouping similar vectors into clusters.

In our system, FAISS is used to index 768-dimensional vector embeddings generated from segmented document chunks. This setup achieves query times of ¡10 milliseconds even for billion-scale datasets [4], making it ideal for real-time retrieval from large document collections.

F. LangChain and HuggingFace

The LangChain framework plays a central role the implementation of our RAG pipeline by offering modular components that streamline various stages of the workflow. These include tools for loading and chunking documents, enabling the transformation of raw text into segments suitable for embedding and retrieval. LangChain also provides interfaces for vector database operations, along with tools for constructing and managing prompts for the generation tasks.

We also use several tools from the HuggingFace ecosystem including the Transformers library which provides pre-trained model architectures and the Sentence-Transformers which are used to generate dense vector representations using the all-mpnet-base-v2 embedding model. For performance evaluation, we rely on HuggingFace's built-in support for metrics like BERTScore, which helps assess the quality of generated responses based on semantic similarity.

III. Methodology

Our chatbot implementation uses two distinct technical approaches, a Retrieval-Augmented Generation (RAG) pipeline and a fine-tuned transformer model - each requiring specialized data processing and system architecture. These approaches were applied specifically to build a chatbot for the IRIS Club at MIT-WPU, enabling it to handle queries related to club events, projects, and member information effectively.

A. Dataset Preparation

The development of our chatbot system required careful curation of domain-specific knowledge, with distinct approaches tailored for the Retrieval-Augmented Generation (RAG) pipeline and the fine-tuned transformer model implementations.

The knowledge base for our chatbot was derived from comprehensive documentation about IRIS Club at MIT-WPU. For the RAG pipeline, we employed unstructured text data comprising 247 distinct information segments including club details, project descriptions, event records, and member profiles. This content was preprocessed through text normalization, special character removal, and URL standardization before being segmented using LangChain's RecursiveCharacterTextSplitter with a chunk size of 300 tokens and 50-token overlap, ensuring contextual continuity between segments.

For the transformer-based model, the dataset was manually structured into question-answer-context triples. A total of 34 examples were compiled in the following format:

```
{
”question”: ”What is Project X?”,
”answer”: ”Project X is...”,
”context”: ”Project X is...”
}
```

Each entry was crafted to mirror the format used in the SQuAD dataset, aligning with the pretraining structure of the distilbert-base-cased-distilled-squad model. Unlike the RAG system, which draws from full-text documents, this approach leverages focused QA pairs, offering a more targeted form of knowledge representation and fine-tuning.

B. RAG Pipeline Implementation

The Retrieval-Augmented Generation (RAG) pipeline begins by breaking down the input documents into smaller, manageable chunks using a recursive character-based splitter. In our implementation, each chunk is approximately 300 characters with an overlap of 50 characters to preserve contextual continuity.

Each chunk is then encoded into a high-dimensional vector using the all-mpnet-base-v2 model from HuggingFace's

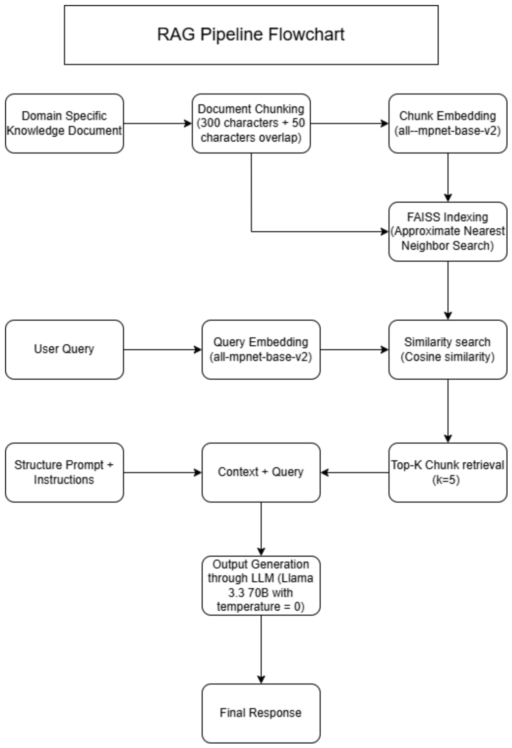
RAG Pipeline Flowchart
Domain Specific Knowledge Document
Document Chunking (300 characters + 50 characters overlap)
Chunk Embedding (all--mpnet-base-v2)
FAISS Indexing (Approximate Nearest Neighbor Search)
User Query
Query Embedding (all-mpnet-base-v2)
Similarity search (Cosine similarity)
Structure Prompt + Instructions
Context + Query
Top-K Chunk retrieval (k=5)
Output Generation through LLM (Llama 3.3 70B with temperature = 0)
Final Response

Fig. 2: RAG Pipeline Flowchart

SentenceTransformers library. This model generates 768-dimensional dense embeddings optimized for semantic similarity tasks. To compare two pieces of text, such as a user query and a document chunk, we use cosine similarity, a standard metric for determining the closeness of two vectors in high-dimensional space.

The cosine similarity between two vectors *u* and *v* is defined as:

$$\text{cos\_sim}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|} \tag{5}$$

To enable efficient search, we use the FAISS library for vector storage and retrieval. The FAISS index supports rapid approximate nearest neighbor search, allowing the system to quickly identify the top-k (in our case, k=5) most relevant chunks for any incoming query. This makes the system both fast and scalable for real-time applications.

For answering questions, the user's query is embedded using the same model and compared to the stored vectors. The five most similar chunks are retrieved and passed into a structured prompt for the language model. The prompt is carefully crafted to limit the model's generation strictly to the retrieved context, thus reducing hallucination and ensuring more accurate responses. The prompt template used is:

"You are an AI assistant for the IRIS club at MIT WPU. Answer the question using the context provided below. If the answer is not explicitly in the context, try to infer it based on the available information. If you still can't find the answer, say "I don't know."

Context: context

Question: question

Answer: "

The final response is generated by a high-performance LLM, specifically the LLaMA-3 70B variant deployed via GroqCloud. To ensure a controlled and deterministic output, we configure the generation with a temperature of 0. The answer is then finally returned.

C. Fine Tuned Transformer Implementation

To implement the domain-specific chatbot, we used the distilbert-base-cased-distilled-squad model—a compact and efficient version of BERT optimized for question answering. It strikes a balance between speed and accuracy by using 6 transformer layers and 82 million parameters.

Each training sample consists of a question, answer, and context. In our case, the answer text itself was reused as the context to simulate document-style QA, allowing the model to focus on extracting precise answers from short, domain-relevant snippets.

Tokenization & Preprocessing**:** We used the HuggingFace tokenizer to combine questions and contexts with attention to overflow handling. Truncation was applied only to the context (to preserve full questions), and a stride of 128 was used to allow for better handling of answers near chunk boundaries.

The preprocessing pipeline ensured that the start and end token positions of the answer were correctly mapped within each input sequence using HuggingFace's offset mapping mechanism.

Training Details:

- Optimizer: AdamW, a popular choice for transformers
- Learning rate: 3×10-5, with linear warm-up over the first 500 steps
- Batch size: 8
- Epochs: 10
- Loss function: Cross-entropy over start and end positions
- Framework: HuggingFace's Trainer API

This configuration led to a stable convergence, with training loss dropping from 1.77 to near-zero and validation loss also approaching zero—indicating effective learning.

Inference & Answer Generation: At inference time, we embed the user query using a SentenceTransformer (all-MiniLM-L6-v2) and compare it with precomputed embeddings of training questions using cosine similarity. If the top match exceeds a threshold of 0.7, the corresponding answer is returned immediately. Otherwise, the QA model runs over the full set of context passages to find the most likely answer span.

Answer Extraction: The span with the highest product of start and end probabilities is selected, and a 15-token window constraint is applied to ensure concise and relevant outputs.

To make sure Domain Specific Responses we use the following methods:

1) Keyword Filtering: Ensures the query includes at least one domain-specific keyword (e.g., "IRIS", "MIT-WPU", "nanotechnology", etc).

2) Semantic Similarity Thresholding: Blocks answers if the query is too dissimilar from known questions (similarity ¡ 0.6).
3) Response Sanitation: Ensures that generated answers contain domain-relevant content before returning them.

This careful layering of checks helps minimize hallucinations and keeps responses tightly aligned with the training data and context.

IV. Evaluation Metrics

To comprehensively assess the quality and performance of our chatbot implementations, for both the RAG-based and the fine-tuned transformer models, we used several metrics to evaluate the generated answers. Each metric highlights a different aspect of the system's output, from semantic similarity and contextual accuracy to readability and language fluency.

A. Cosine Similarity

Cosine similarity [5] measures how closely two vectors, typically text embeddings, align in a high-dimensional space. This metric is particularly suited for evaluating semantic similarity between generated responses and reference answers, capturing how contextually similar they are even when they differ lexically.

Formula: For a predicted answer vector *a* and a reference answer vector r, the cosine similarity is computed as:

$$\text{CosineSim}(\vec{a},\vec{r}) = \frac{\vec{a}\cdot\vec{r}}{\|\vec{a}\|\|\vec{r}\|} = \frac{\sum_{i=1}^{n} a_i r_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} r_i^2}} \quad (6)$$

where *n* is the dimensionality of the embedding vectors (768 in our case using the all-mpnet-base-v2 model).

B. BERTScore

BERTScore [6] improves upon traditional word-matching metrics by using contextual embeddings to evaluate token-level similarity. Unlike surface-level overlap methods, it captures the meaning of words in context, making it highly effective for Question Answering (QA) and generative tasks. It computes three key values: BERT-Precision (measures how much of the generated answer matches the reference), BERT-Recall (measures how much of the reference answer is covered), and their harmonic mean BERT-F1 (balanced score). In QA tasks, it achieves 0.922 Pearson correlation with human judgments on question answering tasks, outperforming ROUGE-L by 0.281 and BLEU by 0.347 [6]. BERTScore serves as a powerful tool to validate if the chatbot's responses maintain the same semantic intent as the expected answers, even when phrased differently.

Core Calculation: Given two sequences, the score is computed by finding the best contextual match for each token in the prediction against the reference, and vice versa:

$$\text{BERTScore} = \frac{1}{|a|}\sum_{w_i\in a}\max_{w_j\in r}(w_i^\top w_j) \quad (7)$$

C. ROUGE Score

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [7] is a set of metrics that measures lexical overlap between a system-generated answer and reference texts. While originally developed for summarization tasks, it remains useful in QA for evaluating factual and syntactic similarity. ROUGE score provides a measure of factual alignment and lexical similarity, which is especially helpful when evaluating extractive QA systems like our transformer-based model. It is widely accepted as a benchmark for QA and summarization.

Variants and Formulas:

- ROUGE-N: Based on matching n-grams
- ROUGE-L: Focuses on the longest common subsequence (LCS) between prediction and reference:

$$R_{\text{lcs}} = \frac{\text{LCS}(a,r)}{|r|}, \quad P_{\text{lcs}} = \frac{\text{LCS}(a,r)}{|a|} \quad (8)$$

- ROUGE-W and ROUGE-S: Account for weighted LCS and skip-bigram overlap respectively

D. Flesch-Kincaid Readability Score

This metric assesses how easy it is to read a given text based on sentence length and syllable count. While not a direct measure of correctness or factual accuracy, it plays a crucial role in evaluating the accessibility of chatbot responses. By monitoring the Flesch-Kincaid score, we ensure that the responses generated are not overly complex for general users.

Formula:

$$\text{FK Grade Level} = 0.39\left(\frac{\text{Total Words}}{\text{Total Sentences}}\right) + 11.8\left(\frac{\text{Total Syllables}}{\text{Total Words}}\right) - 15.59 \quad (9)$$

Interpretation Scale:

- 90–100: Very easy (5th-grade level)
- 60–70: Standard (8th–9th grade level)
- Less than 30: Difficult, suitable for academic/professional readers

V. Results and Interpretation

To assess the effectiveness and reliability of our IRIS Club-specific chatbot implementations, we conducted an evaluation using five user queries which were either related or unrelated to the IRIS community and its projects to see how are chatbot's would respond in both scenarios. The selected questions were:

1) *What is IRIS?*
2) *What was the Inter Campus Innovation Hackathon?*
3) *What is the Tarzan project?*
4) *How can I join IRIS?*
5) *How do I make pizza dough?*

TABLE I: Metrics for answers generated by fine-tune transformer model

Sr. No	Answers Given	Cosine Similarity	Bert F1	Rouge -L	Flesch Reading Ease
1	I am the IRIS Assistant! IRIS is the Innovation, Research, Incubation, and Startup team at MIT WPU. We focus on research, innovation, and entrepreneurship.	1.00	1.00	1.00	29.52
2	Teams formed across universities, and the event culminated in a 12-hour grand finale at MIT-WPU Pune. Highlights included teamwork, mentorship from faculty and industry experts, prototyping, and final presentations to a panel of judges.	1.00	1.00	1.00	35.98
3	The Tarzan Project is an advanced autonomous vehicle control system that integrates computer vision and CAN bus communication.	1.00	1.00	1.00	26.30
4	You can access the Recruitment Form at https://www.iris-club.in/recruitments. Applicants can upload a resume or CV and provide details about their past experiences.	1.00	1.00	1.00	42.38
5	I'm sorry, I can only answer questions about IRIS and its related projects.	0.79	0.86	0.23	59.80

A. Interpretation of results from Fine-Tuned Transformer Model

The fine-tuned transformer showed consistently perfect or near-perfect scores across all in-domain questions. Cosine similarity, BERT F1, and ROUGE-L were all equal to 1.00 for the first four responses, indicating high fidelity to the expected answers both in terms of semantic and lexical similarity.

Additionally, the readability scores (Flesch Reading Ease) for these responses range from 26 to 42, implying a moderate reading difficulty—suitable for technical or semi-technical users.

For the out-of-domain query (*How do I make pizza dough?*), the model appropriately responded with a domain-specific fallback ("I'm sorry, I can only answer questions about IRIS...").

B. ***Interpretation of results from RAG Pipeline***

The RAG-based approach yielded more varied results, reflecting its dependence on retrieved documents. In-domain responses demonstrated moderate to good performance across similarity metrics, but never reached perfect alignment with ground-truth answers. Cosine similarity values ranged from 0.50 to 0.65, while BERT F1 values remained relatively strong (0.88–0.95), showcasing the model's ability to preserve semantic relevance even when the wording differs.

However, the ROUGE-L scores were more scattered, with some answers (e.g., the Tarzan project) achieving only 0.25, indicating lesser lexical overlap. This is expected from retrieval-based systems which may return verbose or contextually different passages.

The Flesch Reading Ease scores (Flesh-Kincaid scores) for the RAG model varied more drastically. For instance, the response to the Hackathon question had a readability score of just 6.17, suggesting a complex sentence structure, while the fallback response ("I don't know") had a very high score of 119.19, reflecting simplicity.

TABLE II: Metrics for answers generated by RAG pipeline

Sr. No	Answer Given	Cosine Similarity	BERT F1	ROUGE -L	Flesch Reading Ease
1	IRIS is the Innovation, Research, and Intelligence Support team at MIT WPU, which focuses on research, innovation, and entrepreneurship. It is a tech community that aims to provide a platform for tech enthusiasts to collaborate, learn, and innovate through hands-on projects and events.	0.60	0.92	0.58	24.27
2	The Inter-Campus Open Innovation Hackathon was held on 16th February 2024 at the MIT-WPU Campus. It was hosted by MIT-TBI in collaboration with ASPIRE, Engineer's Cradle, and I.R.I.S. The event brought together the brightest minds from Pune campuses to tackle real-world challenges through creativity and innovation. The hackathon included expert mentorship, prototyping, and final presentations to a panel of judges.	0.65	0.95	0.72	6.17
3	The Tarzan Project is an advanced autonomous vehicle control system that integrates computer vision and CAN bus communication, using YOLOv8 for real-time object detection to detect road conditions, obstacles, and threats.	0.58	0.88	0.25	29.14
4	To join IRIS, you can fill out the Recruitment Form on the IRIS website, which can be accessed at https://www.iris-club.in/recruitments. The form will ask for your details such as name, PRN, email, phone number, school, branch, current year, and areas of interest. Once you submit the form, your data will be stored securely in the IRIS database for follow-up.	0.50	0.91	0.30	59.94
5	I don't know.	0.00	0.86	0.23	119.19

C. Comparison and Insights from the results

- The fine-tuned transformer consistently delivered accurate, concise, and semantically aligned answers for in-domain questions, with high similarity scores and controlled readability. This highlights its robustness and suitability for domain-specific use cases.
- The RAG pipeline, while flexible and able to retrieve relevant context, showed inconsistencies in fluency and similarity, especially when documents were lengthy or contained extraneous information.
- Both models successfully identified the out-of-domain question, but the transformer provided a clearer, more context-aware fallback, while RAG defaulted to "I don't know."

D. Key Takeaways

- RAG models are adaptable to new and evolving information, making them suitable for retrieval-heavy or open-domain systems.

- Fine-tuned transformer models provide more reliable responses in fixed-domain setting, especially where answer accuracy and control over content are essential.
- For use cases such as institutional or academic chatbots, where factual precision and consistency are prioritized, the fine-tuned transformer outperforms the RAG pipeline.

VI. Conclusion

This research compared two approaches to building a domain-specific chatbot (more specifically for the IRIS Club at MIT-WPU) through a Retrieval-Augmented Generation (RAG) pipeline and a fine-tuned transformer model. Through evaluation across multiple metrics, we observed that the RAG pipeline offers flexibility and is well-suited for retrieval-heavy or evolving knowledge bases. However, it showed variability in precision and readability due to dependence on retrieved content.

In contrast, the fine-tuned transformer model consistently produced accurate, semantically aligned responses, making it more reliable for structured, closed-domain environments. Its strong performance across all in-domain queries, along with better fallback handling, highlights its effectiveness for academic or institutional use.

Overall, while RAG provides adaptability, the fine-tuned model offers superior control and consistency—making it the better choice where accuracy and domain specificity are critical.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention Is All You Need*, (2017).

[2] Devlin, Jacob, Ming-Wei Chang, Kenton Lee and Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, (2019).

[3] Sanh, Victor & Debut, Lysandre & Chaumond, Julien & Wolf, Thomas, *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*, (2019).

[4] Johnson, Jeff, Matthijs Douze, and Hervé Jégou, *Billion-scale similarity search with GPUs*, (2017).

[5] Tim vor der Brück and Marc Pouly, *Text Similarity Estimation Based on Word Embeddings and Matrix Norms for Targeted Marketing*, (2019).

[6] Zhang, Tianyi, Varsha Kishore, Felix Wu, Kilian Q. Weinberger and Yoav Artzi, *BERTScore: Evaluating Text Generation with BERT*, (2019).

[7] Chin-Yew Lin, *ROUGE: A Package for Automatic Evaluation of Summaries*, (2004).

[8] H. Chia, A. I. Oliveira and P. Azevedo, *Implementation of an intelligent virtual assistant based on LLM models for irrigation optimization*, (2024).

[9] K. Pearce, T. Zhan, A. Komanduri, and J. Zhan, A Comparative Study of Transformer-Based Language Models on Extractive Question Answering, (2021).

[10] A. Balaguer et al., RAG vs Fine-Tuning: Pipelines, Tradeoffs, and a Case Study on Agriculture, (2024).

[11] Z. Li, H. Zhu, Z. Lu, and M. Yin, *Synthetic Data Generation with Large Language Models for Text Classification: Potential and Limitations*, (2023).

[12] G. Mialon, C. Fourrier, C. Swift, T. Wolf, Y. LeCun, and T. Scialom, *GAIA: A Benchmark for General AI Assistants*, (2023).

[13] Barbella, Marcello and Tortora, Genoveffa, Rouge Metric Evaluation for Text Summarization Techniques. Available at SSRN: https://ssrn.com/abstract=4120317 or https://dx.doi.org/10.2139/ssrn.4120317.

[14] Solnyshkina, Marina & Zamaletdinov, Radif & Gorodetskaya, L.A. & Gabitov, A.I., *Evaluating Text Complexity and Flesch-Kincaid Grade Level*, (2017).

[15] P. P. Ghadekar, S. Mohite, O. More, P. Patil, Sayantika and S. Mangrule, *Sentence Meaning Similarity Detector Using FAISS*, (2023).