Team05-Deep Learning-Based Tutor for NCERT Curriculum Using RAG and LLMs

Abijna Rao[a], Jyothsna Shaji[b], Malu Jayachandran[c], Nirmith V[d] and Thangaraj N[e]

[a, b, c, d, e]Indian Institute of Science, Bangalore, India

Abstract. This work introduces a multi-task, deep learning-based tutoring system tailored to the NCERT curriculum, leveraging Retrieval-Augmented Generation (RAG) and specialized Large Language Models (LLMs) to perform curriculum-aligned educational tasks. The system supports three primary functionalities: (1) question answering grounded in textbook content, (2) automated question generation using a fine-tuned T5 model, and (3) multilingual answering via integration with Sarvam, an LLM capable of generating responses in Indian languages.

The core architecture adopts a unified RAG pipeline, where NCERT textbooks are tokenized, semantically embedded, and indexed using dense vectors. Retrieved passages are appended as context to generative models, ensuring curriculum-grounded outputs. LangChain orchestrates the retrieval and generation flow, enhancing domain fidelity and factual accuracy across tasks.

To improve the accuracy and curriculum alignment of automated question generation, we systematically evaluate several fine-tuning strategies for the T5 model. Specifically, we compare (1) the base T5 model, (2) T5 fine-tuned on NCERT questions and solutions, (3) T5 tuned with GPT-4-based self-instruct data, and (4) a combined method employing both parameter-efficient fine-tuning (PEFT) and self-instruct tuning. Among these, the combination of PEFT with self-instruct emerges as the most effective approach, consistently achieving higher accuracy, better lexical alignment, and greater question diversity.

1 Introduction

The integration of large language models (LLMs) into education has opened new avenues for scalable, curriculum-aligned tutoring systems. We present a multi-task AI framework designed around the NCERT curriculum, supporting question answering, automated question generation, and multilingual responses. Leveraging a unified Retrieval-Augmented Generation (RAG) architecture and LangChain for orchestration, the system ensures outputs remain grounded in textbook content, enhancing factual consistency and curricular relevance. A key focus is improving question generation via multiple fine-tuning strategies for the T5 model. We evaluate the base T5, NCERT-tuned variants, a GPT-4-based self-instruct approach, and a hybrid method combining self-instruct with parameter-efficient fine-tuning (PEFT). Results show that the PEFT + self-instruct strategy significantly outperforms others in generating accurate, diverse, and curriculum-aligned questions.

2 System Architecture

The proposed tutoring system is a modular, multi-task framework supporting three core functions: **Question Answering (QA)**, **Question Generation (QG)**, and **Multilingual Answering**. All components are unified through a shared Retrieval-Augmented Generation (RAG) backbone, ensuring outputs remain grounded in NCERT curriculum content.

- **Question Answering (QA):** GPT-4 is used to generate accurate, curriculum-aligned answers using the retrieved context.
- **Question Generation (QG):** A T5 model fine-tuned with multiple strategies, including PEFT and GPT-4–based self-instruct tuning, generates pedagogically relevant questions.
- **Multilingual Answering:** The Sarvam LLM generates accurate responses in multiple Indian languages using the same retrieved context.

The system is orchestrated via LangChain, centered around a **Router Agent** (powered by *LLaMA3-8B-8192*) that classifies user intents and dispatches structured task-specific calls in JSON format. A **Memory** module maintains conversational history, enabling coherent multi-turn interactions. This architecture supports scalable, curriculum-aligned tutoring across diverse educational tasks.

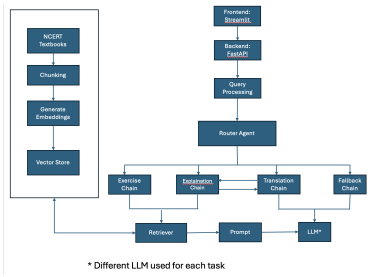

Figure 1: System Architecture

2.1 Question Answering

The primary objective of this project is to assist students in comprehending fundamental technical concepts that are aligned with their academic syllabus, presented in a manner that is easy to consume.

2.1.1 Methodology

Queries are routed through a RAG pipeline powered by gpt-4o-mini, which uses FAISS vector search over pre-indexed academic

content extracted from structured PDFs. The documents are processed with RecursiveCharacterTextSplitter and embedded using text-embedding-3-small to build the vector store.

2.1.2 Results

Experiments were conducted using various combinations of components, including LLaMA 3–8B–8192 [11] and GPT-4o-mini [8] as the underlying language models, NomicAI's nomic-embed-text and OpenAI's text-embedding-3-small as embedding models, and Chroma and FAISS [6] as the vector stores. The chunk size and text splitters were tuned to assess improvement in the quality of content retrieved. Human evaluation was sufficient to confirm the superior performance of GPT-4o-mini over the LLaMA-based model. Notably, when using Chroma for retrieval, the retrieved context often included fragments of textbook exercise questions or figure references. While GPT-4o-mini effectively filtered these for relevance before generating a response, the LLaMA-based model lacked such discernment and frequently incorporated irrelevant content into its explanations.

2.1.3 Evaluation Metrics

The QA system was evaluated on two distinct categories of questions—descriptive and numerical—derived from a curated NCERT-based question-answer dataset.

Table 1: Model Evaluation Scores (ROUGE-L, BLEU, BERTScore, F1)

Metric	Descriptive QA	Numerical QA
BERTScore	0.8495	0.8055
F1	0.3914	0.2125
ROUGE-L	0.2201	0.1191
Bleu	0.7056	0.090

The model showed stronger performance on descriptive questions. Higher F1 and BLEU scores in this indicate better lexical and token-level alignment with reference answers, while consistently high BERTScore suggests strong semantic similarity overall. In contrast, performance on numerical questions degraded significantly, likely due to challenges in generating precise numerical values, mathematical expressions, and maintaining format fidelity—elements critical to structured responses.

To complement traditional metrics, we employed a GPT-4-based evaluator to assess generated answers across human-aligned dimensions such as correctness, relevance, clarity, completeness, factual accuracy, conciseness, and fluency. The model achieved consistently high ratings (8.5/10), indicating that responses were coherent. To address the limitations of traditional metrics, more robust evaluation methods like BEM (BERT Embedding Match), and RAGAS metrics for reference-based correctness and reference-free answer relevance can be integrated as future work. These metrics provides more context-aware framework for evaluating QA systems in retrieval-augmented educational applications. [2]

2.2 Question Generation Using RAG + LLM

2.2.1 Dataset

The training dataset for the QG model is constructed from official NCERT textbooks and solutions. Each question is paired with its corresponding answer and programmatically mapped to a *subtopic* based on the textbook structure. This results in a structured dataframe with fields `[Subtopic, Question, Answer]`, enabling fine-grained learning of curriculum-aligned patterns.

Additionally, the full NCERT textbook corpus is preprocessed and indexed as the retrieval base for the RAG module, ensuring that generation remains grounded in canonical content during both training and inference.

2.2.2 RAG Pipeline for Question Generation

The Retrieval-Augmented Generation (RAG) pipeline ensures that generated questions are grounded in NCERT textbook content through four stages:

1. **Query Embedding:** Each subtopic is embedded using `BAAI/bge-base-en-v1.5` to form a semantic query vector.
2. **Dense Retrieval:** NCERT text is chunked, embedded using the same model, and indexed in a FAISS `IndexFlatIP` structure. The top-k passages (typically $k = 5$) are retrieved based on cosine similarity.
3. **Reranking:** Retrieved chunks are reranked using the `BAAI/bge-reranker-large` CrossEncoder, and the top passages are concatenated into a final context document. BAAI BGE models are used for both embedding and reranking due to their strong performance on BEIR benchmarks and instruction-tuned optimization for semantic retrieval [13].
4. **Question Generation:** This context is fed into a fine-tuned generative model (e.g., T5 or BART) to produce curriculum-aligned questions at the subtopic level.

This two-stage retrieval (dense + reranking) significantly improves context quality, yielding more relevant and pedagogically appropriate questions.

2.2.3 Methodology

Baseline Model Initial experiments was done using BART and T5 architectures to identify a suitable baseline model for curriculum-aligned question generation. Initial trials with BART revealed significant limitations, including incoherent and repetitive output sequences even when prompted with clean, context-rich inputs. For example, outputs such as *"which which is wants is is is belongs is is..."* were frequently observed, indicating a lack of controlled generation capacity. In contrast, the T5 architecture, particularly variants fine-tuned for question generation, demonstrated more fluent and contextually relevant outputs. This is consistent with prior studies [15, 10], which found that T5-based models outperform BART in QG tasks across benchmarks like SQuAD, MS MARCO, and NewsQA, with stronger performance on BLEU, ROUGE-L, and METEOR metrics. Moreover, models such as `valhalla/t5-base-qg-hl`, which adopt a highlight-based encoding strategy, showed improved alignment with target answers and greater fluency. Hence T5 is chosen as the base model for QG.

Fine-Tuning Strategies We employ the following strategies to fine-tune the T5 model for curriculum-aligned question generation:

- **Supervised Fine-Tuning:** The model is trained on a supervised dataset of question–answer pairs extracted from NCERT textbooks and solutions. This forms the foundational approach to align model outputs with curricular content.

- **Self-Instruct Tuning:** To improve question diversity and pedagogical alignment, we augment training data using *self-instruct* [12]. GPT-4 is prompted with a few-shot template to generate synthetic questions from NCERT passages, and is also used to filter out low-quality samples.
- **Parameter-Efficient Fine-Tuning (PEFT) with Self-Instruct:** To mitigate the computational cost of full-model fine-tuning, we adopt the *LoRA* (Low-Rank Adaptation) method [5] in conjunction with the *Self-Instruct* framework [12]. Synthetic question–answer pairs generated and filtered using GPT-4 are used to fine-tune the model efficiently via LoRA, allowing instruction alignment with minimal trainable parameters.

2.2.4 Training Efficiency and Fine-Tuning Observations

Comparison: Full Fine-Tuning vs. PEFT

Full Fine-Tuning. The fully fine-tuned T5 model reaches a validation loss of 0.80 by epoch 3 (from 1.32), indicating rapid convergence. However, this setup incurs high GPU memory usage and slower training speed (0.03 iterations/sec), with just three epochs taking 6.8 hours—making longer training impractical on limited hardware.

PEFT + Self-Instruct. Using LoRA-based PEFT and GPT-4-generated instruction data, the model converges more gradually (validation loss 1.55 at epoch 7.2, from 2.58) but offers significantly reduced resource requirements. Training runs at 0.05 iterations/sec and completes 7.2 epochs in 9.3 hours.

Efficiency Insights.

- PEFT reduces training time, memory consumption, and number of trainable parameters (typically <1%).
- Despite higher validation loss, PEFT + Self-Instruct achieves comparable downstream performance, especially when paired with high-quality synthetic data.
- PEFT acts as an implicit regularizer, preventing overfitting on instruction-tuned data.

Impact of Self-Instruct Framework

Fine-tuning with GPT-4-generated question–answer pairs (via few-shot prompting) significantly boosts question quality:

- Better contextual alignment with textbook content.
- Greater variation in cognitive depth (e.g., factual, inferential, analytical).
- Enhanced grammatical fluency and semantic clarity.

A GPT-4-based filtering loop further improves dataset quality, reducing the need for manual annotations and accelerating curriculum coverage.

2.2.5 Evaluation of Question Quality

Table 2: Model Evaluation Scores (BERT-F1, ROUGE-L, Self-BLEU-2)

Model	BERT-F1	ROUGE-L	Self-BLEU
Base T5	0.8572	0.1835	0.3422
T5 + Supervised FT	0.8531	0.2345	0.3526
T5 + Self-Instruct	0.8485	0.2130	0.3312
T5 + Self-Instruct using PEFT	0.8720	0.2815	0.2984

Table 2 compares four T5-based configurations using three complementary metrics: BERT-F1 (semantic similarity), ROUGE-L (lexical overlap), and Self-BLEU-2 (diversity).

Semantic Accuracy (BERT-F1). All models demonstrate high semantic similarity with ground truth answers. Notably, the PEFT + Self-Instruct variant achieves the highest score (0.8720), indicating improved contextual understanding. Even the base T5 performs reasonably well (0.8572), though slightly behind fine-tuned variants.

Lexical Relevance (ROUGE-L). The PEFT + Self-Instruct model leads with a ROUGE-L of 0.2815, showing better alignment in surface-level phrasing. Supervised fine-tuning also improves ROUGE over the base model (0.2345 vs. 0.1835), while self-instruct tuning offers moderate gains.

Diversity (Self-BLEU-2). Lower Self-BLEU scores indicate greater output diversity. Here, the PEFT + Self-Instruct variant again performs best (0.2984), suggesting it avoids overfitting and repetitive phrasing. In contrast, supervised fine-tuning slightly increases redundancy (0.3526), possibly due to narrow data patterns.

Interpretation. The PEFT + Self-Instruct configuration consistently outperforms others across all metrics, balancing semantic accuracy, lexical precision, and linguistic diversity. These results validate that combining parameter-efficient tuning with high-quality synthetic data enhances question generation effectiveness, while also maintaining efficiency and generalizability.

2.2.6 Critical Evaluation: Limitations of Non-PEFT Fine-Tuning

Although the non-PEFT model trained on GPT-4 self-instruct data shows good token-level convergence (BERT-F1: 0.82–0.89), qualitative analysis reveals significant shortcomings in question quality.

Overfitting to BCE Loss. Low training loss often leads to degenerate outputs such as repetitions (e.g., *"What is the difference between a cis-reaction and a cis-reaction?"*) or incomplete prompts, suggesting that BCE loss alone is not a reliable proxy for educational quality.

Mismatch Between Metrics and Utility. Despite high BERT-F1, low-to-moderate ROUGE-L (0.07–0.34) and fluctuating Self-BLEU (0.18–0.62) indicate limited lexical fidelity and inconsistent diversity—traits not always captured by standard metrics.

Observed Quality Issues. Manual review reveals:

- Grammatically broken or incoherent questions.
- Weak pedagogical framing and vague instructional focus.
- Hallucinations and topic mixing, especially in science-heavy domains.

These issues affirm the need for more structured tuning approaches, like PEFT, to ensure stable, high-quality educational generation.

Conclusion. While BCE loss and semantic similarity metrics may suggest learning progress, the absence of PEFT leads to instability and overfitting, especially when trained on GPT-4-generated data. The results affirm that:

Model parameter control via PEFT (e.g., LoRA) not only improves training efficiency but also acts as an implicit regularizer.

Self-instruct data alone is insufficient unless the model is tuned with care—either via full fine-tuning with adequate regularization, or PEFT to prevent overfitting and structural drift.

3 Translation Pipeline

The translation pipeline utilizes Sarvam Translate V1, a domain-specific model optimized for Indic languages, addressing the limitations of general-purpose LLMs in low-resource translation tasks. Triggered by the Router Agent's query classification, the system follows a RAG-based generation flow, where user queries are first answered in English using a large language model. The generated response is then translated into the target Indic language. This architectural design synergistically combines the strengths of advanced LLMs for content generation with a specialized translation model, thereby ensuring both high-quality answers and accurate multilingual output.

3.0.1 Evaluation Metrics for Translation

Given the use of an off-the-shelf translation model, a comparative benchmarking study was performed to determine the optimal choice for this application. Llama, GPT-4o, and Sarvam models were evaluated across two distinct datasets, employing BLEU[9] and METEOR[1] metrics, alongside anecdotal human assessment

Datasets

1. English to Hindi/Malayalam/Tamil/Kannada datasets from AIKOSH[7].
2. Custom-created dataset of 10 entries of English to Malayalam from SCERT Kerala English[3] and Malayalam[4] medium textbooks.

Translation Evaluation Results Results are updated below.

Table 3: Dataset1: Translation Model Evaluation Scores (BLEU, METEOR)

Language	BLEU			METEOR		
	Sarvam	**Llama**	**GPT-4o**	**Sarvam**	**Llama**	**GPT-4o**
Hindi	0.12	0.07	**0.15**	0.31	0.26	**0.40**
Malayalam	0.05	0.02	**0.05**	0.16	0.07	**0.18**
Tamil	**0.05**	0.02	0.05	0.17	0.09	**0.19**
Kannada	**0.06**	0.03	0.05	**0.18**	0.10	0.17

Table 4: Dataset2: Translation Model Evaluation Scores (BLEU, METEOR)

Language	BLEU			METEOR		
	Sarvam	**Llama**	**GPT-4o**	**Sarvam**	**Llama**	**GPT-4o**
Malayalam	**0.04**	0.02	0.04	**0.18**	0.06	0.16

Translation Evaluation Conclusion Initially, our tests on Dataset 1 showed GPT-4o performing well. However, when humans reviewed the translations, Sarvam AI consistently seemed better. This difference made us look closer at Dataset 1, and we realized its translations (authored by Microsoft) appeared machine-generated and were not high quality, which might have skewed our initial results. To get a fairer evaluation, we created a new dataset (Dataset 2) using real school textbooks. On this new, more domain-relevant dataset, Sarvam AI clearly performed best in both our automated tests and human review. This confirmed that Sarvam AI is the best choice for our application's translation needs.

4 Conclusion and Future Work

The TutorAI system demonstrates the effectiveness of a modular, multi-task architecture for delivering curriculum-aligned educational support. Key capabilities include question answering, automated question generation, and multilingual response generation, all grounded in NCERT content through a unified RAG pipeline.

Future work will focus on enhancing personalization by profiling user learning behavior and adapting tutoring strategies accordingly. For the Question Generation module, we propose integrating reinforcement learning with a reward model trained to assess question quality—optimizing for pedagogical value, diversity, and relevance beyond token-level losses. Additionally, incorporating visual content such as textbook figures and tables may improve the quality of explanations. Multilingual support can be expanded and evaluated using regional board data for broader inclusivity.

The source code for this project can be accessed at our GitHub repository.

5 Contributions

All authors contributed significantly to the conceptualization, design, and implementation of the NCERT AI-Tutor system. Specific contributions are detailed as follows:

- **Abijna Rao**: Conducted an extensive literature review on existing question generation (QG) techniques—including answer-aware QG, encoder-decoder models, and instruction-tuned approaches—to inform model design decisions. Developed the QG module tailored to NCERT content and established a robust evaluation framework using BERT-F1, ROUGE-L, and Self-BLEU metrics. Implemented and benchmarked multiple fine-tuning strategies on the T5 model, with a focus on Parameter-Efficient Fine-Tuning (PEFT) via LoRA and GPT-4–based self-instruct tuning. Powered RAG with the state-of-the-art BAAI embedding model. Systematically analyzed model performance across configurations, highlighting trade-offs in efficiency, quality, and diversity.
- **Jyothsna Shaji**: Designed and implemented the orchestration components like router agent and destination chains, enhanced with apt prompt engineering, memory and session management. Built Streamlit UI and integrated Fast API backend with conversational chat interface, quizzing and evaluation. Evaluated llama, GPT-4 models with Chroma and FAISS for different embedding models for QA. Explored AgenticAI integration, image embeddings and bulk translation with OCR for SCERT texts.
- **Malu Jayachandran**: Developed and implemented the multilingual translation pipeline, integrating Sarvam Translate V1 for Indic languages. Orchestrated its seamless operation within the LangChain framework, including the design and implementation of session memory management. Conducted comparative analysis of translation models using BLEU and METEOR metrics, confirming Sarvam AI's optimal performance for domain-relevant output.
- **Nirmith V**: Developed framework for evaluation of Question Answering (QA) module. Conducted the analysis of QA quality using BERT-F1, ROUGE-L, BLEU and GPT-4 based evaluation. Proposed the enhancements that could be adopted for more contextual evaluation of QA module. Dataset collection from various sources and providing clean data for various modules in this project. Advanced RAG+ method was explored which gives more application aware reasoning. [14]

- **Thangaraj N**: Laid the groundwork for the system architecture, particularly in the implementation of the Retrieval Augmented Generation (RAG) pipeline using LangChain and OpenAI Models (gpt-4o-mini). Responsbile for chunking of NCERT textbooks, indexing, embedding(text-embedding-3-small), and storing in vector store(FAISS). Prompt engineering for QA and Quiz Generation module of the system.

6 Citations and references

References

[1] A. Banerjee and A. Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and Summarization*, pages 65–72, 2005.

[2] F. M. G. A. L. A. B. L. D. G. Ermelinda Oro1, 2 and . Massimo Ruffolo1. Evaluating retrieval-augmented generation for question answering with large language models. https://ceur-ws.org/Vol-3762/495.pdf, 2024.

[3] Government of Kerala, Department of General Education, State Council of Educational Research and Training (SCERT) Kerala. *CHEMISTRY Standard X, Part 1*. State Council of Educational Research and Training (SCERT) Kerala, Thiruvananthapuram, Kerala, 2019. URL https://scert.kerala.gov.in/standard-10/.

[4] Government of Kerala, Department of General Education, State Council of Educational Research and Training (SCERT) Kerala. *CHEMISTRY Standard X, Part 1, Malayalam Medium*. State Council of Educational Research and Training (SCERT) Kerala, Thiruvananthapuram, Kerala, 2019. URL https://scert.kerala.gov.in/standard-10/.

[5] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[6] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.

[7] Microsoft. Aikosh news test references for machine translation evaluation: English to hindi, malayalam, tamil, and kannada datasets, 2025. URL https://aikosh.indiaai.gov.in/web/datasets/. License: CC BY-SA 4.0; Uploaded: February 24, 2025.

[8] OpenAI. Gpt-4o. https://openai.com/index/gpt-4o, 2024. Accessed: 2025-06-22.

[9] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, 2002.

[10] A. Singh, A. Joshi, et al. Comparing neural question generation architectures for reading comprehension. *arXiv preprint arXiv:2301.11345*, 2023.

[11] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, A. Ram, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[12] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, H. Hajishirzi, and Y. Choi. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

[13] H. Xu, Y. Lin, H. Mei, H. Ji, X. Wang, Q. Liu, Y. Wang, Z. Liu, M. Sun, and L. Lin. Bge: An embedding model for general purpose. https://huggingface.co/BAAI/bge-base-en-v1.5, 2022. Accessed: 2025-06.

[14] S. Z. M. F. Z. Y. Z. X. Z. Z. W. H. H. T. L. H. T. L. X. J. U. N. U. Yu Wang1, 2. Rag+: Enhancing retrieval-augmented generation with application-aware reasoning. *arXiv:2506.11555v1*, 2025.

[15] S. Zhou, X. Du, and C. Cardie. Sparta: Improving zero-shot question generation by structured prompting. *ACL*, 2023.