A project report on

QUESTION GENERATION AND EVALUATION ACCORDING TO BLOOM'S TAXONOMY USING LLMs AND NLP TECHNIQUES

Submitted in partial fulfillment for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

by

B SHREYA (21BCE6167)

ASHISH RAM J A (21BCE6193)

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April, 2025

QUESTION GENERATION AND EVALUATION ACCORDING TO BLOOM'S TAXONOMY USING LLMs AND NLP TECHNIQUES

Submitted in partial fulfillment for the award of the degree of

Bachelor of Technology in in Computer Science and Engineering

by

B SHREYA (21BCE6167)

ASHISH RAM J A (21BCE6193)

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April, 2025

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

DECLARATION

I hereby declare that the thesis entitled "Question Generation and Evaluation according to Bloom's Taxonomy using LLMs and other NLP Techniques" submitted by Ashish Ram J A(21BCE6193), for the award of the degree of Bachelor of Technology in Computer Science and Engineering, Vellore Institute of Technology, Chennai is a record of Bonafide work carried out by me under the supervision of Dr. Janaki Meena M.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Chennai

Date: 8th April, 2025 Signature of the Candidate

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

School of Computer Science and Engineering

CERTIFICATE

This is to certify that the report entitled **"Question Generation and Evaluation According to Bloom's Taxonomy using LLMs and Other NLP Techniques"** is prepared and submitted by **Ashish Ram J A(21BCE6193)** to Vellore Institute of Technology, Chennai, in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** is a bonafide record carried out under my guidance. The project fulfills the requirements as per the regulations of this University and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma and the same is certified.

Signature of the Guide

Name: Dr. Janaki Meena M
Date: 8th April, 2025

Signature of the Examiner

Name:

Date:

Signature of the Examiner

Name:

Date:

Approved by the Head of Department,
B.Tech. Computer Science and Engineering

Name: Dr. Nithyanandam P
Date: 8th April, 2025

ABSTRACT

Question generation is a crucial component of educational assessment, yet existing automated approaches often fail to ensure both relevance to study material and alignment with Bloom's Taxonomy. This project introduces a novel question generation technique involving Large Language Models (LLMs) and Natural Language Processing (NLP) within a Hypothetical Document Embedding (HyDE)-based architecture. Our system automatically generates questions from uploaded study materials while ensuring adherence to Bloom's Taxonomy's cognitive levels. To this end, we have implemented a three-tier evaluation framework: (1) LLM-guided assessment, (2) comparison against a golden standard dataset, and (3) quantitative scoring via the QSTS metric—a cosine similarity measure between candidate questions and the reference context. Across varying document sizes, we observed robust performance in generating high-quality questions that match with the required Bloom's taxonomy levels as well as ensuring the scope of the question stays within the input material uploaded . Specifically, for a short document (6 pages on Breadth First Search), the system achieved a QSTS of 0.828; for a medium-size document (203 pages on Sorting Algorithms), a QSTS of 0.774 was obtained; and for a long document (1312 pages from Introduction to Algorithms), the QSTS was 0.628. Additionally, LLM-guided evaluations confirmed that the generated questions were understandable, relevant, and aligned with the appropriate level of Bloom's Taxonomy. These quantitative findings underscore the system's capacity to produce questions that are both contextually grounded and pedagogically sound. By addressing the taxonomy-aware challenge in question generation, our approach significantly enhances automated assessment tools.

ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to Dr. Janaki Meena M, Professor, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai, for her constant guidance, continual encouragement, understanding; more than all, she taught me patience in my endeavor. My association with her is not confined to academics only, but it is a great opportunity on my part to work with an intellectual and expert in the field of Text Mining, Image Processing, BigData Analytics, Algorithms and Data Science.

It is with gratitude that I would like to extend my thanks to the visionary leader Dr. G. Viswanathan our Honorable Chancellor, Mr. Sankar Viswanathan, Dr. Sekar Viswanathan, Dr. G V Selvam Vice Presidents, Dr. Sandhya Pentareddy, Executive Director, Ms. Kadhambari S. Viswanathan, Assistant Vice-President, Dr. V. S. Kanchana Bhaaskaran Vice-Chancellor, Dr. T. Thyagarajan Pro-Vice Chancellor, VIT Chennai and Dr. P. K. Manoharan, Additional Registrar for providing an exceptional working environment and inspiring all of us during the tenure of the course.

Special mention to Dr. Ganesan R, Dean, Dr. Parvathi R, Associate Dean Academics, Dr. Geetha S, Associate Dean Research, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai for spending their valuable time and efforts in sharing their knowledge and for helping us in every aspect.

In jubilant state, I express ingeniously my whole-hearted thanks to Dr. Nithyanandam P, Head of the Department, B.Tech. Computer Science and Engineering and the Project Coordinators for their valuable support and encouragement to take up and complete the thesis.

My sincere thanks to all the faculties and staff at Vellore Institute of Technology, Chennai who helped me acquire the requisite knowledge. I would like to thank my parents for their support. It is indeed a pleasure to thank my friends who encouraged me to take up and complete this task.

Place: Chennai

Date: 8th April, 2025 **Ashish Ram J A**

CONTENTS **PAGE NO**

LIST OF FIGURES

PAGE NO

LIST OF TABLES **PAGE NO**

LIST OF ACRONYMS

LLM	Large Language Model
NLP	Natural Language Processing
HyDE	Hypothetical Document Embeddings
QG	Question Generation
DB	Database
GPT	Generative Pre-Trained Transformer
AI	Artificial Intelligence
SME	Subject Matter Expert
RAG	Retrieval-Augmented Generation
VLM	Vision Language Model
NA	Not Applicable
OCR	Optical Character Recognition

Chapter 1

Introduction

1.1 INTRODUCTION TO THE PROJECT

In today's fast-changing world of education, teachers need better ways to check how well students understand what they learn. As learning materials become more digital and classrooms more diverse, it's harder and more time-consuming for teachers to create good questions for tests and assignments, especially ones that encourage deeper thinking, not just memorization.

Bloom's Taxonomy is a well-known system that helps organize questions by the level of thinking they require, from basic recall to creative problem-solving. But writing questions that match these levels properly is not easy and can take a lot of effort.

This project is inspired by the need to make that process easier and more effective. With the help of powerful language models and natural language processing (NLP), we can now build tools that help generate meaningful questions automatically. These tools can save time for teachers and make sure that the questions are closely related to the study material and cover a wide range of thinking skills. The main aim is to support better learning and fairer assessment by using smart technology in education.

1.2 OVERVIEW INTO BLOOM'S TAXONOMY

Bloom's Taxonomy is a hierarchical framework used in education to classify learning objectives based on cognitive complexity. Developed in 1956 by Benjamin Bloom and his collaborators [1], it provides a structured approach to designing assessments, ensuring that learners progress from basic recall to higher-order thinking skills.

1.2.1 INITIAL LEVELS OF BLOOM'S TAXONOMY (1956)

The original taxonomy consisted of six levels of cognitive learning, arranged in increasing order of complexity:

Knowledge – Recalling facts, definitions, and basic concepts.
Comprehension – Understanding and explaining ideas.
Application – Applying learned concepts to new situations.
Analysis – Breaking down information to understand relationships.
Synthesis – Combining elements to form a new whole.
Evaluation – Judging based on criteria and evidence.

1.2.2 REVISED BLOOM'S TAXONOMY (2001)

In 2001, Anderson and Krathwohl revised the taxonomy [2], making two key changes: the categories were changed from nouns to verbs to emphasize the cognitive process. "Synthesis" was replaced with "Creating," and "Evaluation" was placed before it, emphasizing the highest cognitive level. The revised taxonomy consists of:

Remembering – Recalling basic facts and concepts.
Understanding – Explaining and interpreting information.
Applying – Using information in new contexts.
Analyzing – Examining patterns and relationships.
Evaluating – Making judgments based on criteria.
Creating – Generating new ideas or products.

Bloom's Taxonomy is widely used in education for curriculum design, structuring lessons to promote deeper learning; assessment development, creating questions that evaluate different cognitive levels; and personalized learning, adapting instruction to students' cognitive abilities. By ensuring a structured progression from basic to advanced thinking skills, it plays a critical role in enhancing educational effectiveness, critical thinking, and problem-solving abilities.

1.3 MOTIVATION BEHIND THE PROJECT

Creating high-quality educational questions is a time-consuming and inconsistent process. Educators often rely on manual question formulation, which can lead to variability in difficulty, cognitive depth, and topic relevance. This makes it challenging to maintain a structured approach that effectively evaluates student learning.

This project aims to automate question generation using LLMs and NLP techniques, ensuring that questions are not only accurate and relevant but also aligned with Bloom's Taxonomy. Unlike manual methods, which depend on individual expertise and intuition, this system follows a structured approach to generate questions that consistently match the intended cognitive levels. By maintaining taxonomy-based alignment, it ensures that assessments cover all necessary levels of understanding, from basic recall to higher-order thinking.

Additionally, the scalability of this system addresses the growing need for efficient and unbiased question creation across various subjects and educational domains. The system ensures that all questions are solvable using the given study material, eliminating the risk of ambiguity or external dependencies.

By reducing the workload on educators and standardizing question formulation, this project enhances the efficiency, reliability, and effectiveness of assessment processes, ultimately improving the overall quality of education.

1.4 STATE-OF-THE-ART METHODS

Today, several advanced methods are being used to create questions for learning and assessment. Some systems use transformer-based models trained on special datasets to generate high-quality questions. Others rely on a database of pre-written questions, which are tagged with Bloom's Taxonomy levels, and fetch the most suitable ones. There are also template-based approaches that use fixed formats to create questions for specific topics. More recently, large language models (LLMs) have made it possible to generate questions using simple prompts, making the process faster and more flexible. While each of these methods has its strengths, there's still a growing need for smarter systems that can combine accuracy, context, and cognitive depth—motivating the development of more advanced, hybrid techniques.

1.5 CHALLENGES IN EXISTING METHODOLOGIES

Existing QG methodologies include rule-based, retrieval-based, and transformer-based approaches. Rule-based and template-based methods lack flexibility, while database (DB) retrieval-based models struggle with unstructured text. Transformer-based models (BERT, T5, GPT, etc.) generate fluent but often taxonomy-misaligned questions, making them unsuitable for structured assessments. Hybrid approaches like Retrieval-Augmented Generation (RAG) improve contextual accuracy but introduce retrieval-generation balance challenges. Key issues across these methods include lack of taxonomy alignment, uncontrolled question complexity, contextual mismatch, and evaluation difficulties. Our approach addresses these limitations by integrating NLP, HyDE-based architectures, and multi-agent LLM pipelines to ensure structured, taxonomy-aligned, and contextually relevant question generation.

1.6 SCOPE OF THE PROJECT

This project aims to automate the generation of descriptive questions from academic study materials using Large Language Models (LLMs), Natural Language Processing (NLP) techniques, and a HyDE-based approach, ensuring alignment with Bloom's Taxonomy. The system is designed to process structured educational content such as textbooks, lecture notes, and scholarly articles to generate taxonomy-aligned questions suitable for academic assessments.

The project is particularly relevant for use in educational institutions, where it can support educators in designing curriculum-aligned assessments and facilitate scalable question paper creation. It will produce a comprehensive dataset of generated questions and a detailed report outlining the methodology, implementation, and evaluation strategies.

The quality of generated questions will be assessed through multiple techniques, including alignment with a golden standard dataset and classification across Bloom's Taxonomy levels. While the system offers broad applicability, its effectiveness depends on the quality and structure of the input study materials. Additionally, subject-specific customization may be required to optimize question relevance and accuracy.

Chapter 2

Background Study

2.1 LITERATURE SURVEY OF EXISTING WORKS

Automated question generation (QG) has been extensively researched, focusing on different methodologies, including rule-based, transformer-based, and retrieval-augmented approaches. However, ensuring question relevance to study materials and aligning generated questions with Bloom's Taxonomy remains a critical challenge.

2.1.1 TRADITIONAL AND TRANSFORMER-BASED APPROACHES

Early QG systems tagged questions with cognitive levels and generated question sets based on requirements. However, this system required human intervention for question preparation and tagging, limiting full automation, as discussed by Paul, A. et. al. in [6]. Later, Paul, R. J. et. al. in [7] leveraged the T5 transformer model to generate questions, achieving 70% contextual relevance. However, transformer models remain computationally expensive and prone to biases.

2.1.2 SECURED AND STRUCTURED QUESTION GENERATION

Ragasudha, R., and Saravanan, M. in [8] incorporated a database of pre-tagged questions aligned with Bloom's Taxonomy and utilized cryptographic techniques for secure question transfer. However, reliance on a predefined database limited the generation of complex, novel questions. Similarly, Scaria, N. et. al. in [9] evaluated GPT-3.5 and GPT-4 for generating high-quality questions in the Indian high school social science curriculum. While the study highlighted GPT models' effectiveness, it relied on human evaluators, indicating the need for an automated, scalable evaluation mechanism.

2.1.3 NEURAL AND MULTILINGUAL QUESTION GENERATION

A broader perspective is offered by Guo, S. et. al. in [10], which reviews deep neural network-based QG methods, acknowledging their promising results but emphasizing their inability to proactively generate questions based on diverse user requirements. Ackerman, R., and Balyan, R [11] expanded QG research into multilingual contexts, specifically for English and Spanish health-related content. The study found that LLMs struggled to maintain quality when generating multiple questions from short texts, underscoring limitations in handling compact information.

2.1.4 CONTEXTUAL AND DOMAIN-SPECIFIC QUESTION GENERATION

Lohr, D. et. al. in [12] examined retrieval-augmented generation (RAG) to enhance contextual relevance in computer science education. Despite improvements in annotation generation, relational annotation and alignment with educational standards remained inconsistent, requiring manual intervention. Li, K., and Zhang, Y. introduced a novel LLM-guided QG approach in [13] that generates an answer plan before producing questions, ensuring content and difficulty control. However, the absence of expert-annotated labels hindered its reliability.

2.1.5 EVALUATION AND OPTIMIZATION OF LLM-BASED QUESTION GENERATORS

Several studies have explored evaluating LLM-generated questions. Meißner, N. et. al. presented an automated MCQ generation system for software engineering education in [14], reducing educators' workload but struggling with limited PDF support and question diversity. Scaria, N. et. al. in [15] systematically assessed LLM-generated questions using zero-shot, few-shot, and chain-of-thought (CoT) prompting. The study highlighted challenges in generating high-quality 'Create'-level questions. Elkins, S. et. al. in [16] analyzed teacher adoption of LLM-generated questions and found that while educators preferred AI-generated quizzes, optimizing prompt engineering remained challenging.

2.1.6 FINE-TUNING AND ALIGNMENT WITH BLOOM’S TAXONOMY

To address taxonomy alignment challenges, Duong-Trung, N. et. al. in [17] fine-tuned an LLM to generate taxonomy-aware questions, outperforming GPT-4 in semantic interdependence and hallucination reduction. However, a lack of domain-specific evaluation frameworks persisted. Similarly, Babakhani, P et. al. in [18] explored fine-tuning Flan-T5 and GPT-3 for subjective question generation in news media but struggled with robust evaluation metrics and contextual coherence.

2.1.7 APPLICATIONS IN STANDARDIZED TESTING

Chen, Y., and He, L. investigated LLMs in [19] for structured question generation in national teacher certification exams. While expert evaluations were conducted, further optimization was necessary for complex question formulation.

2.2 IDENTIFIED GAPS FROM EXISTING WORKS

The identified gaps in automated question generation (QG) research primarily revolve around limitations in automation, alignment with Bloom's Taxonomy, contextual relevance, evaluation methodologies, and domain-specific adaptability. Many early QG systems require human intervention for question selection, tagging, or validation, reducing full automation potential. Transformer-based approaches, while effective, are computationally expensive and prone to biases, limiting accessibility and consistency. Reliance on predefined databases for structured QG restricts the ability to generate novel and complex questions, while multilingual QG struggles with quality degradation in compact texts. Retrieval-augmented generation (RAG) and LLM-guided QG techniques improve contextual relevance but often require manual intervention for semantic annotation and standard alignment. Existing evaluation frameworks depend heavily on human assessment, making large-scale validation inefficient. Furthermore, fine-tuned LLMs have shown promise in taxonomy-aware question generation, but the absence of robust domain-specific evaluation frameworks hinders their reliability. Addressing these gaps requires a scalable, automated evaluation mechanism and taxonomy-aware question generation techniques that ensure both contextual relevance and alignment with educational standards.

2.3 PROJECT STATEMENT

In traditional education, the process of designing high-quality assessment questions that effectively evaluate different levels of cognitive understanding is both labor-intensive and time-consuming for educators. Manually crafting questions that align with Bloom's Taxonomy requires deep subject matter expertise, careful consideration of learning objectives, and significant effort to ensure diversity in question types and difficulty levels.

This project seeks to revolutionize question generation by automating the process using Large Language Models (LLMs), Natural Language Processing (NLP) techniques, and a HyDE-based (Hypothetical Document Embeddings) architecture. By leveraging these advanced AI-driven methods, the system will generate contextually relevant and taxonomy-aligned questions directly from study materials.

A core objective of this project is to ensure that the generated questions accurately correspond to the six levels of Bloom's Taxonomy—ranging from basic knowledge recall to higher-order critical thinking and problem-solving skills. This structured approach to automated question generation enhances the quality of assessments, making them more systematic, scalable, and adaptive to learners' cognitive abilities.

By integrating LLM-guided evaluation techniques, benchmarking against golden standard datasets, and employing the QSTS (Question-Set Taxonomy Suitability) score for assessment quality, the system will iteratively improve its question-generation capabilities. This not only aids educators by reducing manual workload but also enhances learning outcomes by providing well-structured and pedagogically sound questions tailored to different levels of understanding.
Ultimately, this project aims to bridge the gap between AI-driven automation and effective pedagogy, ensuring that learning assessments remain robust, fair, and aligned with the principles of cognitive skill development.

2.4 RESEARCH OBJECTIVES

The primary objective of this research is to automate the generation of structured, taxonomy-aligned questions using LLMs and NLP techniques. This system aims to ensure that the generated questions are contextually relevant, solvable from the given study material, and mapped to appropriate Bloom's Taxonomy levels.

To achieve this, the research focuses on the following key objectives:

1. Develop a systematic approach for extracting meaningful information from study materials and transforming it into well-formed questions.
2. Ensure taxonomy alignment by categorizing generated questions based on Bloom's Taxonomy, ranging from lower-order cognitive skills (remembering, understanding) to higher-order thinking (analyzing, evaluating, creating).
3. Optimize question relevance by verifying that all generated questions can be answered using the provided study material.
4. Support multiple question formats, including multiple-choice, short-answer, and descriptive questions, ensuring adaptability across different educational needs.

5. Implement evaluation techniques to assess the quality of generated questions through LLM-based analysis, comparison with a golden standard dataset, and structured scoring metrics.
6. Enhance scalability and efficiency by designing a system that can generate high-quality questions across various subjects with minimal manual intervention.

By addressing these objectives, the research aims to streamline and standardize question generation, reducing the manual effort required for assessment design while ensuring high-quality, structured, and taxonomy-aligned educational content.

2.5 RESEARCH QUESTIONS ADDRESSED

RQ1: To what extent can trained language models generate questions that align with those created by subject matter experts?

This question explores the capability of language models to replicate human-like question generation while maintaining quality, accuracy, and depth. Subject matter experts (SMEs) craft questions based on their understanding of the topic, considering factors such as difficulty, relevance, and cognitive demand. The study aims to determine how well LLM-generated questions compare to expert-crafted ones in terms of clarity, taxonomy alignment, and appropriateness for assessment. It also examines whether the models introduce biases or inconsistencies that may affect question quality.

RQ2: How effectively can questions generated by language models be applied in practical scenarios, such as educational assessments?

The practical utility of AI-generated questions depends on their applicability in real-world educational settings. This research evaluates whether these questions can be seamlessly integrated into exams, quizzes, and formative assessments across different subjects. It investigates their usability for educators and students, ensuring that they are neither too simplistic nor overly complex. Additionally, the study examines how well these questions enhance learning outcomes, foster critical thinking, and maintain fairness across various educational levels.

RQ3: How can the questions generated by language models be evaluated to judge whether they meet the expected academic standards?

For AI-generated questions to be credible, they must be evaluated against predefined academic standards. This research explores different assessment techniques, including comparison with expert-generated questions, alignment with Bloom's Taxonomy, and structured scoring metrics. It also considers automated and human evaluation methods to measure aspects such as relevance, difficulty, linguistic clarity, and correctness. By defining a robust evaluation framework, the study ensures that the generated questions meet the expectations of educational institutions and educators.

Chapter 3

Proposed Methodology

3.1 OVERVIEW OF THE HYDE-BASED ARCHITECTURE

The HyDE-based architecture for question generation follows a structured approach that integrates document processing, retrieval-based augmentation, and fine-tuned language models to ensure the generation of high-quality, contextually relevant questions. The overall architecture is depicted in Fig 3.1 and the HyDE-based architecture is depicted in Fig. 3.2.

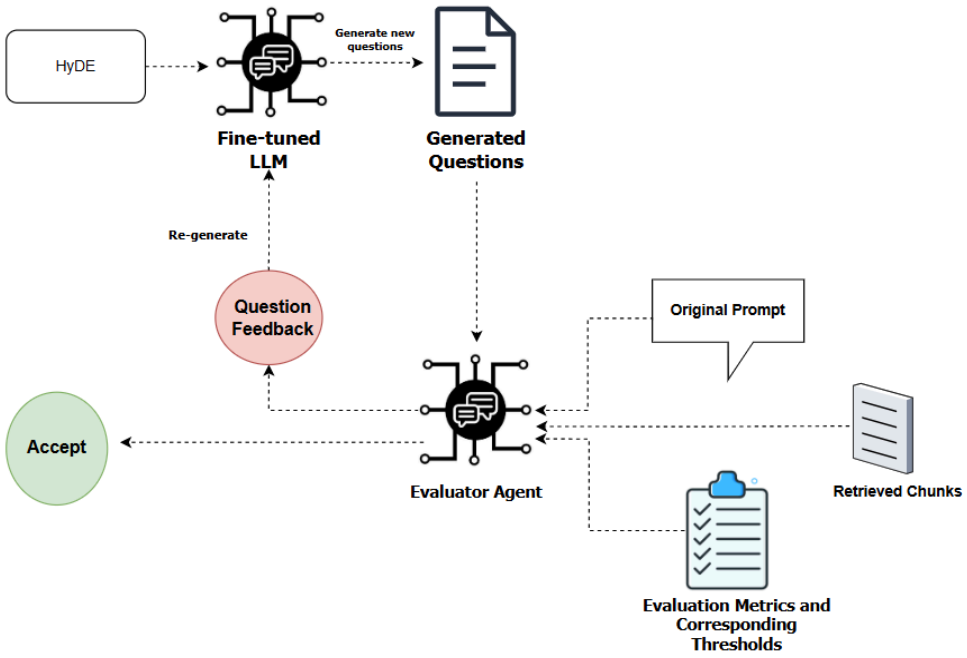

Fig. 3.1: Overall Architecture of the System

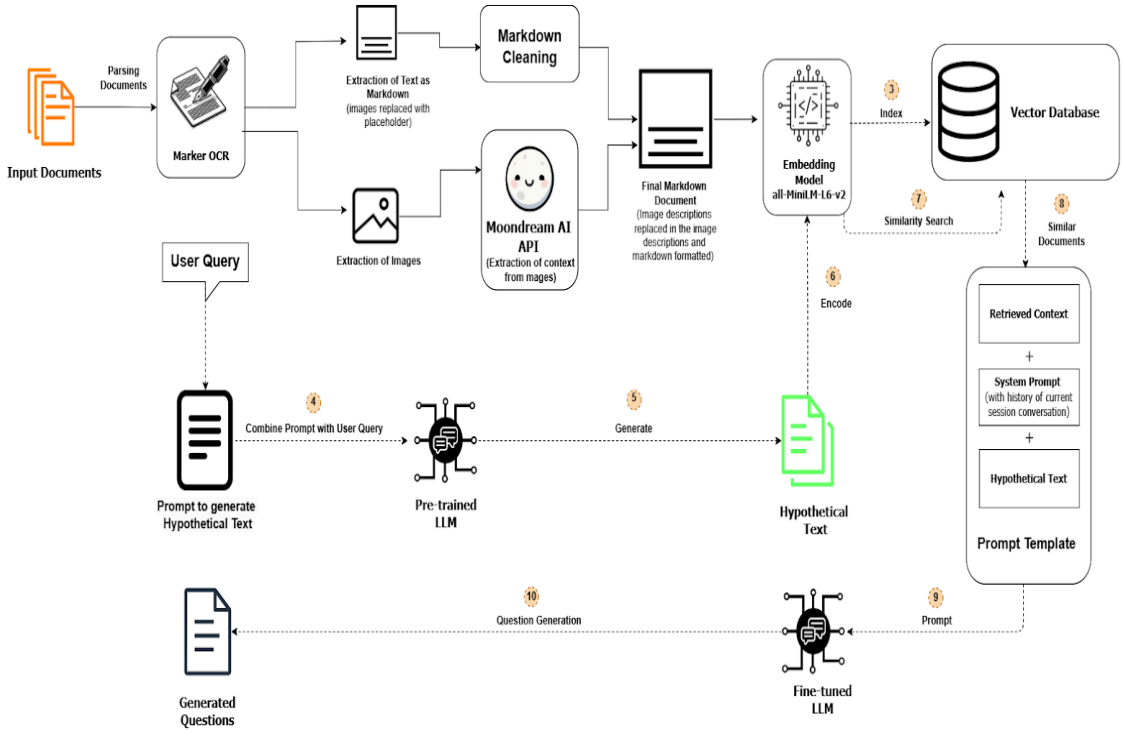

Fig. 3.2: HyDE Component of the Architecture

The process begins with document parsing, where input materials are analyzed to extract both text and images. Extracted text undergoes pre-processing, while images are processed using the Moondream AI API to extract contextual details. The pre-processed text is then encoded using an embedding model (all-MiniLM-L6-v2) and stored in a vector database for efficient similarity-based retrieval.

When a user submits a query, a pre-trained LLM first generates hypothetical text based on the query. This hypothetical text is then encoded and used for similarity search within the vector database to retrieve relevant study materials. The retrieved content, along with system prompts and session history, is structured into a prompt template, forming the input for question generation.

The fine-tuned LLM is then used to generate questions based on the retrieved study content. To ensure quality control, an evaluator agent assesses the generated questions using predefined academic metrics. If the questions do not meet the required educational standards, the evaluator provides feedback, prompting the fine-tuned LLM to re-generate improved questions. This iterative process enhances the coherence, relevance, and accuracy of the generated questions.

Once the evaluator agent validates the output, the final set of questions is accepted. This approach ensures that the generated questions are not only contextually appropriate but also aligned with academic standards, making them useful for educational assessments.

3.1.1 TRADITIONAL RAG

Retrieval-Augmented Generation (RAG) is a framework that enhances the capabilities of language models by combining information retrieval with text generation. Instead of relying solely on pre-trained knowledge, a RAG-based system dynamically retrieves relevant information from an external knowledge source, such as a vector database, and incorporates it into the generated response.

The process begins with query encoding, where the user input is transformed into an embedding representation. This is then used to perform similarity search in a vector database containing pre-processed and indexed documents. The most relevant documents are retrieved and provided as additional context to the language model. This retrieved information is appended to the prompt, allowing the model to generate more informed and contextually accurate responses.

RAG ensures that the generated output is not limited to static model knowledge but is enriched with up-to-date and relevant data, making it particularly useful for applications like question answering, summarization, and automatic question generation. It also improves accuracy, reduces hallucinations, and ensures responses remain grounded in verifiable sources. Studies also suggest that RAG provides a more specific, diverse and factual content particularly in language generation tasks [3].

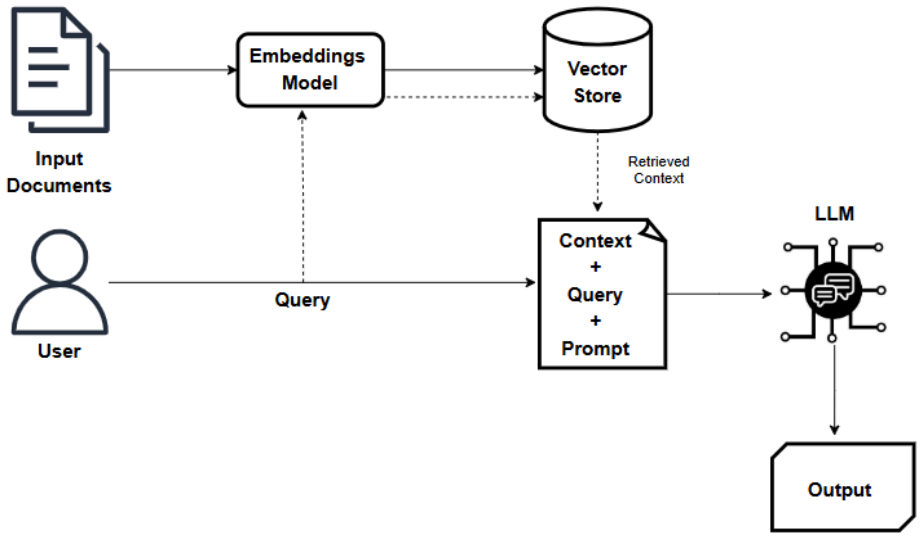

Fig 3.3: Traditional RAG Architecture

3.1.2 HyDE ARCHITECTURE:

Hypothetical Document Embeddings (HyDE) is a retrieval technique that enhances dense retrieval by generating synthetic document representations. Instead of directly retrieving documents based on a query, HyDE first generates hypothetical passages using a language model and then retrieves real documents that are semantically similar to these generated passages. This method helps in improving retrieval effectiveness, especially in zero-shot and few-shot scenarios.

The core idea behind HyDE is that traditional dense retrieval methods rely on pre-trained embeddings that might not generalize well to unseen queries. By leveraging a generative model, HyDE creates a richer representation space that aligns better with query intent. This approach has been particularly useful in open-domain question answering and knowledge-intensive tasks, where precise retrieval of relevant information is critical.

HyDE has been shown to improve retrieval quality without requiring extensive labeled data. It aligns closely with retrieval-augmented generation (RAG) by enhancing the retrieval component, ensuring that retrieved documents are more contextually relevant. This makes it an effective approach for tasks requiring high recall and precision in document retrieval. Studies show the superiority of performance while using the HyDE-based architecture [4].

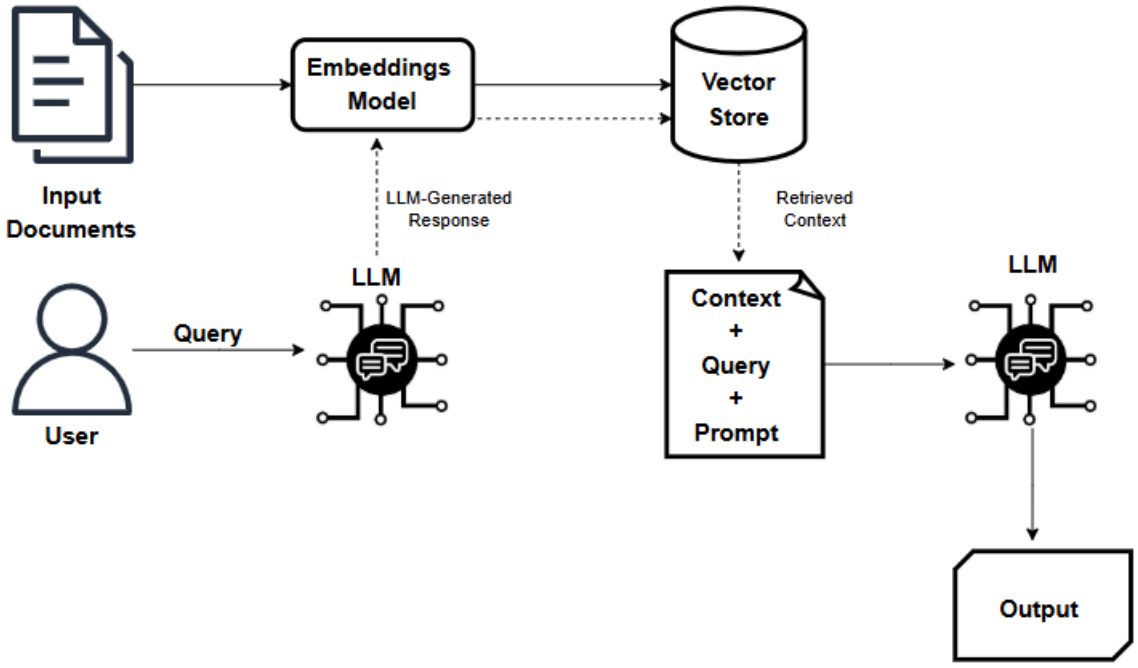

Fig 3.4: HyDEArchitecture

3.1.3 TRADITIONAL RAG VS HyDE RAG

Hypothetical Document Expansion (HyDE) enhances retrieval by generating an additional set of documents, H, using an LLM based on the input query Q. Instead of relying solely on direct retrieval, HyDE first expands the query by creating a hypothetical response, enriching the search process with more context. This generated document H is then embedded using a Contriever model, which filters out irrelevant details while preserving key information, producing an embedding E. This embedding E is then used to query the vector database, retrieving relevant real-world documents C that closely match the hypothetical passage. Finally, the retrieved context C, along with the original query Q, is fed into the LLM to generate a well-informed response. This method significantly improves retrieval performance because the search is performed using enriched details beyond just the query, leading to the retrieval of documents that contain more comprehensive and relevant information. Compared to traditional retrieval techniques, HyDE provides better recall, reduces lexical mismatches, and ensures that retrieved documents are semantically richer, making it particularly effective for open-domain question answering and complex information retrieval tasks.

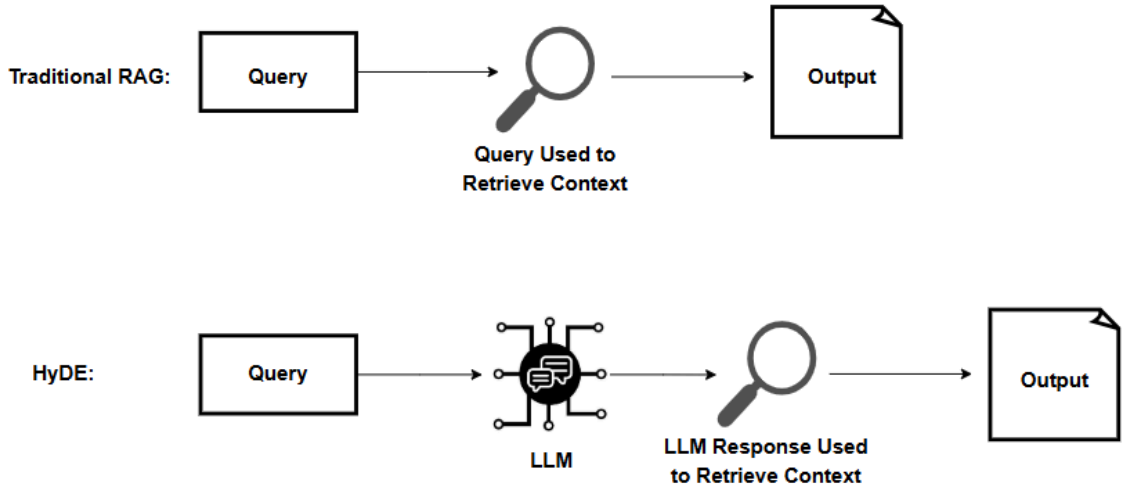

Fig. 3.5: Traditional RAG vs HyDE

3.2 TEXT AND IMAGE EXTRACTION TECHNIQUES

Marker is a high-performance tool designed to extract structured content from various document formats, including PDF, images, PPTX, DOCX, XLSX, HTML, and EPUB. It accurately converts these documents into Markdown, JSON, and HTML while preserving essential structures such as tables, forms, equations, inline math, links, references, and code blocks. This makes it an ideal solution for tasks requiring precise data extraction while maintaining document integrity.

In terms of text extraction, Marker follows a systematic process to ensure accuracy and consistency. It begins by parsing and preprocessing the document to identify its structure, distinguishing between plain text, formatted content, and special elements. It then removes unnecessary artifacts such as headers, footers, watermarks, and other irrelevant elements that might interfere with the extracted content. Additionally, it ensures that formatting features like bold, italics, bullet points, numbered lists, and inline references are preserved. Finally, the extracted text is converted into Markdown for better readability and further processing, with options to export in JSON or HTML for structured applications.

Beyond text, Marker is also highly efficient in image extraction from documents. It identifies and segments images, figures, and charts embedded within files while ensuring that associated metadata, such as captions and references, are retained. The extracted images can either be stored separately or embedded within the converted Markdown, JSON, or HTML file, ensuring proper alignment with the text.

One of Marker's key advantages is its exceptional performance and optimization. It works efficiently across GPU, CPU, and MPS environments, significantly outperforming cloud-based services like Llamaparse and Mathpix in terms of speed and accuracy. While its serial processing of individual PDF pages already delivers high accuracy, its batch mode capabilities allow it to process up to 122 pages per second on an H100 GPU, making it a robust tool for large-scale document processing.

Additionally, Marker can be integrated with Large Language Models (LLMs) to further enhance text recognition, refine extracted content, and handle complex document layouts more effectively. This combination ensures high accuracy while improving the adaptability of the extracted data for downstream applications.

By incorporating Marker for text and image extraction, we ensure a structured and reliable format for further use in document analysis, knowledge retrieval, and AI-driven applications.

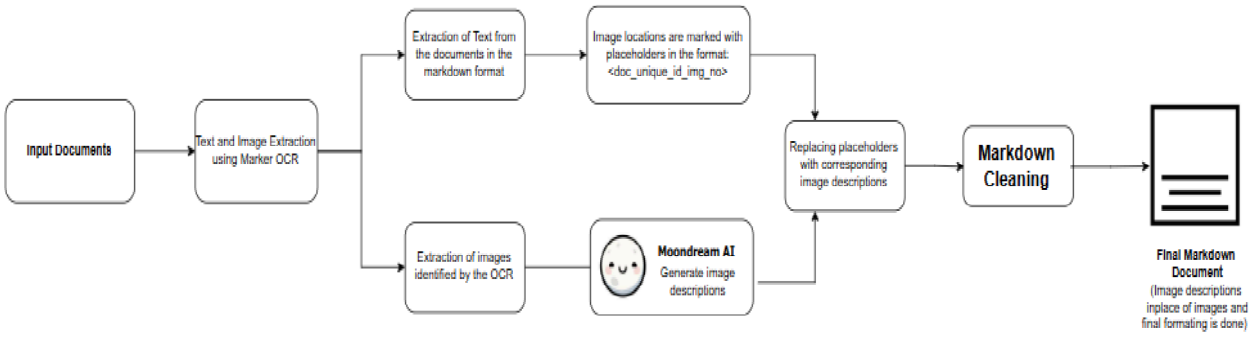

Fig. 3.6 Extraction and Conversion to Markdown Format of Text and Images

3.3 IMAGE INSIGHT GENERATION

Once images are extracted from uploaded documents, they are processed using the Moondream API, a powerful vision-based AI system designed to analyze, interpret, and extract meaningful insights from images. Unlike standard OCR tools, Moondream goes beyond basic text extraction by applying deep learning techniques for scene understanding, object detection, and visual reasoning. This makes it particularly useful for processing diagrams, charts, handwritten notes, schematics, and mathematical notations, ensuring that extracted images contribute to the overall knowledge representation of the document.

The Moondream API performs multi-level analysis on images. First, it applies OCR and text detection, extracting embedded text from scanned documents, labels in charts, or handwritten notes. Next, it conducts semantic scene understanding, identifying key visual elements such as objects, graphs, or equations, and determining their relevance in context. The API also interprets structured data, such as tables, charts, and visual statistics, converting them into machine-readable formats that can be queried for insights. Additionally, Moondream generates descriptive captions for images, summarizing their contents in a meaningful way that can be used for contextual search and retrieval.

All extracted insights—text, objects, and metadata—are formatted into Markdown to ensure structured representation. These insights are embedded along with text-based content in a vector database, allowing for multi-modal retrieval where both text and images contribute to accurate search and AI-driven analysis. By integrating Moondream, the system enhances document comprehension, improves retrieval accuracy, and ensures that images play an active role in knowledge extraction and question generation.

3.4 HIERARCHICAL CHUNKING TECHNIQUE

Once both text and image insights are extracted, the document is processed using hierarchical chunking to ensure structured storage and efficient retrieval. Traditional document processing treats content as linear text, but hierarchical chunking allows for better organization and context preservation, improving searchability and downstream AI applications.

The first level of chunking splits documents into major sections, typically based on headings, subheadings, and logical separations such as chapters or key topics. The second level breaks these sections into paragraph-sized chunks, ensuring that each part remains cohesive and contextually meaningful. Finally, the third level extracts individual sentences or key phrases, enabling fine-grained retrieval for highly targeted search queries. This multi-level approach ensures that retrieved content maintains its logical structure, preventing fragmented or out-of-context results.

Once the hierarchical chunks are created, they are embedded and stored in Qdrant DB, a powerful vector database optimized for semantic search. Unlike traditional keyword-based search methods, Qdrant uses dense vector embeddings to retrieve conceptually similar content, even when different words or phrasing are used. This makes question generation and knowledge retrieval more accurate and relevant, as the system can find semantically related content rather than relying on exact keyword matches.

By linking Moondream-generated image insights with corresponding text chunks, the system enables multi-modal retrieval, where both textual explanations and visual references contribute to search results. This approach ensures that question generation aligns with the logical flow of study material, making AI-generated assessments more structured, contextually relevant, and easy to understand.

3.5 FINE-TUNING OF SYSTEM WITH CUSTOM DATASET

Fine-tuning a language model for automatic question generation involves adapting a pre-trained model to the custom dataset, ensuring alignment with taxonomy levels and domain-specific knowledge of Data Structures and Algorithms (DSA). The dataset includes university question papers, exercise questions from reference books, and web resources, all annotated with Bloom's Taxonomy levels (BT1–BT6). Before fine-tuning, the data undergoes preprocessing, including tokenization, noise removal, standardization, and dataset splitting into training (80%), validation (10%), and test sets (10%). A pre-trained transformer-based seq2seq model, such as LLaMA 3, is chosen for training. The input representation involves prefixing taxonomy labels to the questions, allowing the model to learn structured question generation patterns. Training involves optimizing a cross-entropy loss function, tuning hyperparameters like batch size and learning rate, and applying regularization techniques such as dropout and gradient clipping to prevent overfitting. After fine-tuning, the model is evaluated with expert human review to assess relevance, clarity, and taxonomy alignment. An iterative refinement process further enhances the model based on evaluation feedback. This fine-tuning approach ensures that the generated questions align with academic standards, covering various cognitive levels, and making them suitable for educational assessments and exam preparation.

3.6 QUESTION GENERATION

Our approach to question generation follows a prompt-based methodology, ensuring alignment with user-specific requirements and reference materials. The process begins with user input, which includes details such as the topic, the number of questions required, taxonomy level, academic level, and any additional customization preferences. Using this topic information, an LLM first generates a hypothetical text that outlines key subtopics and related concepts, helping to provide contextual grounding for the question generation process.

This generated text is then embedded and used to query a vector database, which contains pre-processed, chunked, and embedded reference materials. The retrieval step ensures that the system fetches the most relevant context from trusted academic resources, improving the factual accuracy and depth of generated questions. The retrieved context, along with the original user query, is then fed into a fine-tuned LLM trained specifically for structured question generation across different Bloom's Taxonomy levels.

Our question generation pipeline follows a structured three-prompt approach to ensure accuracy, contextual relevance, and alignment with academic standards.

1. User Prompt: The process begins with user input, which includes essential details such as the topic, number of questions, taxonomy level, academic level, and any specific customization requirements. This information is structured into a well-defined prompt that guides the system in generating relevant hypothetical content.

2. Hypothetical Text Generation Prompt: Using the user prompt details, an LLM generates a hypothetical text that expands on the given topic, covering subtopics and related concepts. This serves as an enriched context for the retrieval step, improving the accuracy of the final questions. The generated text is then embedded and used to query a vector database containing structured reference materials, ensuring that relevant information is retrieved.

3. Evaluation Prompt: Once the retrieved context and user specifications are processed, the final step involves sending this information to a fine-tuned LLM via an evaluation prompt. This ensures that the generated questions adhere to the expected taxonomy levels, maintain coherence, and align with academic standards. The evaluation step acts as a quality control mechanism, refining the output before presenting it to the user.

Below are the structured prompt templates used in this workflow:

3.6.1 INITIAL USER PROMPT OVERVIEW

An instructor persona is defined (based on academic level, major, and course). The task is to generate a specific number of questions targeting a selected Bloom's Taxonomy level, grounded in particular topics. The generation is guided by a predefined explanation of the taxonomy levels to ensure alignment with cognitive objectives.

3.6.2 HYPOTHETICAL TEXT GENERATION PROMPT OVERVIEW

A set of main topics is provided along with the associated Bloom's Taxonomy level, academic level, major, and course context. The task is to generate a fixed number of sub-topics per main topic, ensuring alignment with the specified taxonomy level. The generation is informed by predefined descriptions of Bloom's levels to reflect appropriate cognitive depth and academic rigor. The output consists solely of relevant sub-topics—excluding any questions, explanations, or unrelated content.

3.6.3 FINAL QUESTION GENERATION PROMPT OVERVIEW

The instruction to generate a specific number of questions for a given course, aligned with a chosen Bloom's Taxonomy level is provided. The generation must strictly adhere to content retrieved from course materials and reflect the expectations of the academic level and major. The taxonomy level definition is explicitly provided to guide question depth and style. No external knowledge should be used—only what's in the retrieved content.

3.6.4 EVALUATION AND FINAL FEEDBACK GENERATION PROMPT OVERVIEW

The persona is an experienced instructor responsible for evaluating a set of questions based on their alignment with a specified Bloom's Taxonomy level, course topic, and academic context. A set of well-defined evaluation criteria (e.g., clarity, grammatical correctness, relevance, cognitive level alignment, etc.) guides the assessment. The evaluator must provide justified feedback and a final verdict on the quality and acceptability of the questions without extra embellishment.

This multi-step prompt approach ensures that the generated questions are well-structured, educationally relevant, and aligned with the specified learning objectives.

3.7 GENERATION OF GOLDEN REFERENCE DATASET

The synthetic data generation process follows a structured loop-based approach to systematically create 5,000 questions per Bloom's Taxonomy level using NVIDIA's Llama 3.1-405B model. This model, trained on a vast corpus of text using supervised fine-tuning and reinforcement learning with human feedback (RLHF), is designed for instruction-following tasks and high-quality text generation. The loop iterates through 60 predefined topics related to Data Structures and Algorithms, such as sorting algorithms, graph traversal, and recursion. For each topic, the system executes five iterations, corresponding to the six higher-order levels of Bloom's Taxonomy (Analyze, Apply, Evaluate, Create, etc.), where each iteration generates 100 taxonomy-aligned questions using a carefully designed prompt. This ensures a total of 5,000 questions per taxonomy level, ultimately producing 30,000 structured questions covering the entire syllabus. The generated questions are streamed, parsed, and stored in a CSV file for further use.

3.7.1 PURPOSE OF GENERATING SYNTHETIC DATA

This automated large-scale question generation approach significantly benefits AI-powered educational assessments. It eliminates manual effort, ensuring a consistent supply of high-quality, diverse, and structured questions aligned with cognitive learning objectives. The use of LLM-based synthetic data allows the system to adaptively generate, refine, and expand educational datasets, making it ideal for training AI-driven tutors, adaptive learning systems, and intelligent assessment models. The generated dataset can be fine-tuned and integrated into a vector database, enabling better RAG pipelines and hierarchical chunking for enhanced question-context alignment. By leveraging NVIDIA's TensorRT-LLM and H100 GPUs, this process achieves high-speed generation, scalability, and precision, ensuring taxonomy-aligned question sets that improve automated assessment tools, intelligent tutoring systems, and personalized learning platforms.

3.8 PSEUDOCODE

3.8.1 INPUT

01. DOCUMENT_PATH → Path to the input document (e.g., PDF)

02. COURSE_NAME → Name of the course

03. NUM_QUESTIONS → Number of questions to generate

04. ACADEMIC_LEVEL → Target academic level (e.g., undergraduate)

05. TAXONOMY_LEVEL → Bloom's Taxonomy level (e.g., Analyze)

06. TOPICS_LIST → List of course-relevant topics (e.g., BFS, DFS)

07. MAJOR → Major subject area (e.g., Computer Science)

08. LLM_CHOICE_GENERATION → LLM for question generation (e.g., GPT-3.5)

09. LLM_CHOICE_EVALUATION → LLM for evaluation (e.g., Gemini)

10. QSTS_THRESHOLD → QSTS score threshold for regeneration

3.8.2 OUTPUT

01. GENERATED_QUESTIONS → Final set of generated questions

02. EVALUATION_FEEDBACK → Evaluation feedback from LLM

03. EVALUATION_SCORES → QSTS scores

3.8.3 MODULES AND FUNCTIONS

1. DOCUMENT CONTENT EXTRACTION MODULE
   1.1. **Function:** EXTRACT_DOCUMENT_CONTENT(DOCUMENT_PATH)
   1.2. Load the document from DOCUMENT_PATH and utilize Marker to extract the document's content and convert it into Markdown format.
   1.3. Insert unique placeholders for every detected figure during extraction.
   1.4. Save the extracted images in a directory with filenames corresponding to their placeholders.
   1.5. Retrieve detailed image descriptions using the Moondream API and replace the placeholders accordingly.
   1.6. Clean the Markdown to remove extraneous artifacts.
   1.7. Return the structured Markdown content along with the associated image metadata.

2. VECTOR DATABASE MANAGEMENT MODULE
   2.1. Function: INITIALIZE_VECTOR_DATABASE()
   2.2. Initialize and connect to a vector database client.
   2.3. Set up the vector database collection if it does not already exist.
   2.4. Return the vector database client.
   2.5. Function: EMBED_AND_INSERT_CHUNKS(VECTOR_DATABASE_CLIENT, COLLECTION_NAME, TEXT_CHUNKS, DOCUMENT_METADATA, EMBEDDING_MODEL)
   2.6. For each TEXT_CHUNK in TEXT_CHUNKS, generate a text embedding using the EMBEDDING_MODEL.
   2.7. Prepare a data point with a unique ID, the embedding vector, and a payload containing the TEXT_CHUNK, SESSION_ID, and DOCUMENT_METADATA.
   2.8. Insert the data point into the specified COLLECTION_NAME in the vector database client.

2.9. Function: SEARCH_VECTOR_DATABASE(VECTOR_DATABASE_CLIENT, COLLECTION_NAME, QUERY_EMBEDDING, SESSION_ID_FILTER, SEARCH_LIMIT)
2.10. Use the QUERY_EMBEDDING to perform a semantic search in the COLLECTION_NAME.
2.11. Filter results by SESSION_ID_FILTER if provided.
2.12. Limit the results to SEARCH_LIMIT.
2.13. Return the search results.

3. TEXT PROCESSING AND EMBEDDING MODULE
3.1. Function: SPLIT_TEXT_INTO_CHUNKS(TEXT, CHUNK_SIZE, CHUNK_TOLERANCE)
3.2. Split the TEXT into chunks approximately equal to CHUNK_SIZE.
3.3. Ensure that each chunk ends at a sentence boundary within CHUNK_TOLERANCE.
3.4. Return the resulting TEXT_CHUNKS.
3.5. Function: INITIALIZE_EMBEDDING_MODEL()
3.6. Initialize the sentence transformer model for generating text embeddings.
3.7. Return the EMBEDDING_MODEL.
3.8. Function: EMBED_TEXT(TEXT, EMBEDDING_MODEL)
3.9. Generate a text embedding for TEXT using the EMBEDDING_MODEL.
3.10. Return the resulting TEXT_EMBEDDING.

4. QUESTION GENERATION AND EVALUATION MODULE
4.1. Function: GENERATE_HYPOTHETICAL_TEXT(PROMPT_TEMPLATE_PATH, PLACEHOLDERS, LLM_CHOICE_GENERATION)
4.2. Load the prompt template from PROMPT_TEMPLATE_PATH.
4.3. Fill in the prompt template using the provided PLACEHOLDERS.
4.4. Call the specified language model (LLM_CHOICE_GENERATION) to generate the hypothetical text.
4.5. Return the HYPOTHETICAL_TEXT.
4.6. Function: GENERATE_QUESTIONS_FROM_CONTEXT(PROMPT_TEMPLATE_PATH, PLACEHOLDERS, RETRIEVED_CONTEXT, LLM_CHOICE_GENERATION)
4.7. Load the prompt template from PROMPT_TEMPLATE_PATH.
4.8. Populate the template with PLACEHOLDERS and RETRIEVED_CONTEXT.
4.9. Call LLM_CHOICE_GENERATION to generate the questions.
4.10. Return the generated questions text.

4.11. Function: EVALUATE_GENERATED_QUESTIONS_LLM_GUIDED(EVALUATION_PROMPT_PATH, PLACEHOLDERS, GENERATED_QUESTIONS_TEXT, LLM_CHOICE_EVALUATION)
4.12. Load the evaluation prompt template from EVALUATION_PROMPT_PATH.
4.13. Fill in the template with PLACEHOLDERS and the GENERATED_QUESTIONS_TEXT.
4.14. Call the evaluation language model (LLM_CHOICE_EVALUATION) to obtain evaluation results.
4.15. Return the LLM-guided evaluation results.
4.16. Function: EVALUATE_GENERATED_QUESTIONS_METRICS(CANDIDATE_QUESTIONS_TEXT, REFERENCE_TEXT, EMBEDDING_MODEL)
4.17. Calculate the BLEU-4 score between CANDIDATE_QUESTIONS_TEXT and REFERENCE_TEXT.
4.18. Compute the QSTS (Question Sentence Text Similarity) using the EMBEDDING_MODEL.
4.19. Return the evaluation scores (BLEU-4, QSTS).
4.20. Function: REGENERATE_QUESTIONS_WITH_FEEDBACK(INITIAL_QUESTIONS, REFERENCE_CONTEXT, BASE_PROMPT_PATH, FEEDBACK_HANDLER, THRESHOLDS, LLM_CHOICE_GENERATION, LLM_EVALUATION_RESULTS)
4.21. Initialize BEST_QUESTIONS using INITIAL_QUESTIONS.
4.22. For a limited number of iterations:
4.23. Evaluate BEST_QUESTIONS against REFERENCE_CONTEXT using defined metrics.
4.24. Generate automatic feedback based on the evaluation scores.
4.25. Combine automatic feedback with optional user feedback using the FEEDBACK_HANDLER.
4.26. Augment the base prompt from BASE_PROMPT_PATH with the combined feedback to create an iteration prompt.
4.27. Generate new questions using LLM_CHOICE_GENERATION with the augmented prompt.
4.28. Evaluate the newly generated questions against REFERENCE_CONTEXT.
4.29. If the new questions show improvement over BEST_QUESTIONS based on THRESHOLDS, update BEST_QUESTIONS; otherwise, cease iterations.
4.30. Return BEST_QUESTIONS, the combined feedback, and the best evaluation scores.

5. SESSION MANAGEMENT AND FEEDBACK HANDLING MODULE
   5.1. Function: INITIALIZE_SESSION_MANAGER(VECTOR_DATABASE_CLIENT)
   5.2. Configure the session manager using the VECTOR_DATABASE_CLIENT.
   5.3. Generate a unique SESSION_ID.
   5.4. Return the session manager and the SESSION_ID.
   5.5. Function: LOG_SESSION_INTERACTION(SESSION_MANAGER, INTERACTION_TYPE, INTERACTION_DATA)
   5.6. Log the interaction type and data along with a timestamp and SESSION_ID using the session manager.
   5.7. Function: INITIALIZE_FEEDBACK_HANDLER()
   5.8. Set up and configure the feedback handler.
   5.9. Return the initialized feedback handler.
   5.10. Function: COLLECT_FEEDBACK(FEEDBACK_HANDLER, AUTOMATIC_FEEDBACK, USER_FEEDBACK)
   5.11. Gather automatic feedback and any optional user feedback using the FEEDBACK_HANDLER.
   5.12. Return the combined feedback.
   5.13. Function: GENERATE_ITERATION_PROMPT(FEEDBACK_HANDLER, BASE_PROMPT_PATH, FEEDBACK)
   5.14. Load the base prompt from BASE_PROMPT_PATH.
   5.15. Augment the base prompt with the provided FEEDBACK using the FEEDBACK_HANDLER.
   5.16. Return the resulting iteration prompt.
6. FINAL OUTPUT AND SESSION END

   6.1. Consolidate the final set of generated questions, evaluation feedback, and evaluation scores after all iterations and tests are completed.

   6.2. Log the final output details and evaluation results with the session manager.

   6.3. Return the final set of questions/outputs as the overall result of the process.

   6.4. Log the session end with the unique session ID.

3.8.4 MAIN ALGORITHM STEPS

1. Initialization
   1.1. Set up the vector database client and the text embedding model.
   1.2. Initialize the session manager and generate a unique session ID.
   1.3. Initialize the feedback handler and define document metadata.
   1.4. Log the session start.

2. Document Content Extraction
   2.1. Log the start of text extraction.
   2.2. Extract document content, including text, figures, and tables.
   2.3. Clean the extracted Markdown and update image placeholders with detailed descriptions.
   2.4. Log the completion of extraction.

3. Text Chunking, Embedding, and Vector Database Insertion
   3.1. Split the extracted content into text chunks.
   3.2. Log the text chunking process.
   3.3. Generate text embeddings for each chunk.
   3.4. Insert the embedded chunks into the vector database with associated metadata.
   3.5. Log the completion of the database insertion.

4. Hypothetical Text Generation and Semantic Search
   4.1. Prepare the required placeholders for generating context.
   4.2. Generate hypothetical text using the designated LLM.
   4.3. Generate a query vector from the hypothetical text.
   4.4. Search the vector database for relevant context using the query vector.
   4.5. Extract the retrieved context.

5. Question Generation, Evaluation, and Iteration
   5.1. Prepare the placeholders for the question generation prompt.
   5.2. Generate the initial set of questions using the designated LLM.
   5.3. Evaluate the initial questions using the evaluation LLM.
   5.4. Iterate the regeneration process by incorporating feedback until metric thresholds are met or improvements cease.
   5.5. Log the completion of question generation.

6. Final Output and Session End
   6.1. Output the generated questions, evaluation feedback, and evaluation scores.
   6.2. Log the session end with the unique session ID

Chapter 4

Experimental Setup and Results

4.1 EXPERIMENTAL SETUP

The experimental environment for development and evaluation was configured on a Windows 11 operating system to represent a common user desktop environment. Hardware specifications included an Intel Core i5 11th generation processor, providing a balance of processing capability for both CPU-intensive tasks and general system responsiveness. The system was equipped with 16GB of RAM, which served as sufficient memory for running the Python scripts, handling document processing libraries, and managing moderate-sized language models and vector databases in memory. The software environment was based on Python 3.11, selected for its robust ecosystem of scientific computing and natural language processing libraries. Development and experimentation were primarily conducted within Python notebooks, leveraging their interactive nature for iterative code development, immediate execution, and visualization of intermediate results. For interfacing with Large Language Models (LLMs), the OpenRouter API platform was employed. This choice provided access to a diverse range of LLMs, including Gemini and Deepseek models, allowing for exploration of different model capabilities and potentially facilitating model comparison within the question generation task. Local installations of Poppler and Tesseract-OCR were configured to support comprehensive PDF document processing and Optical Character Recognition (OCR) functionalities, respectively. Furthermore, a SentenceTransformer model ("all-MiniLM-L6-v2") was initialized for text embedding, forming a crucial component of the semantic search and evaluation pipeline. Session management and feedback mechanisms were integrated into the experimental design to systematically track interactions and enable iterative refinement of the question generation process.

4.2 TOOLS AND LIBRARIES USED

This architecture utilizes a diverse set of libraries and tools spanning several categories. For document content extraction and manipulation, the system now leverages Marker—a high-performance tool that converts various document formats (PDF, DOCX, PPTX, etc.) into structured Markdown while preserving formatting and automatically inserting unique placeholders for figures. These placeholders are later enriched with detailed descriptions generated by the Moondream API for comprehensive visual content analysis. Additionally, text processing and cleaning are performed using regular expressions and standard Python libraries. Natural language processing tasks, including sentence embedding and semantic similarity, are facilitated by SentenceTransformers. Interaction with large language models is achieved through the requests library and the OpenRouter API, accessing models such as Gemini and Deepseek, while the openai library supports GPT-3.5. Vector storage and retrieval are managed using QdrantClient with a free instance of Qdrant Cloud, and evaluation of generated questions is conducted using metrics from NLTK (BLEU), rouge-score, bert-score, BARTScore, and bleurt.

4.3 EVALUATION OF GENERATED QUESTIONS

The evaluation of generated questions incorporated a multifaceted approach - combining automatic evaluation metrics assessment , semantic metric evaluated by the llm and a pipeline for iterative refinement monitored and managed by a Large Language Model evaluator agent/judge. Automatic evaluation of generated questions for evaluation [21] primarily relied on QSTS (Question-Sentence Text Similarity) from the automatic metrics , which analyze the discriminative ability of the models across different data inputs [20]. QSTS uses Sentence-BERT embeddings to quantify the quality and relevance of generated questions against the source context. Predefined thresholds for these metrics guided a regeneration process, aiming to improve question quality. Furthermore, the system incorporated a feedback mechanism which provides scores based on various human evaluation metrics, producing results closer to the human baseline compared to the direct prompting approach [22]. Thus we integrate this feedback mechanism to log both automatic evaluation scores and potential user feedback[22], allowing for prompt augmentation and iterative question refinement within the experimental session. The results of the final regeneration is used for the benchmarking.

4.4 EXPERIMENTAL RESULTS

4.4.1 RESULTS FOR SHORT DOCUMENT

Name of material - Lecture 13: Graphs I: Breadth First Search - Introduction to Algorithms (6 pages)
course_name = "Data Structures and Algorithms"
num_questions = "15"
academic_level = "undergraduate"
taxonomy_level = "Create"
topics_list = "Breadth First Search, Shortest Path"
major = "Computer Science"

Generated Questions:

1. Design an algorithm to perform a breadth-first search on a directed graph starting from a specific vertex 's'. Include pseudocode and explain the key steps involved in the process.

2. Develop a strategy to find the shortest path in a directed graph using the parent pointers and level information. Illustrate this approach with a detailed example for better understanding.

3. Propose a method to represent a graph using adjacency lists for efficient traversal and exploration. Discuss the advantages of this representation over other methods in terms of space and time complexity.

4. Create a Python class that implements a graph structure with vertices and edges. Include methods to add vertices, establish connections between vertices, and perform a breadth-first search starting from a given vertex.

5. Formulate an approach to determine the number of possible states and configurations in a Rubik's Cube based on the number of vertices and edges in the cube. Provide a step-by-step explanation of the calculations involved.

6. Devise a plan to explore a graph level by level from a designated starting vertex 's' using the breadth-first search algorithm. Explain how to build each level based on the previous level's information.

7. Construct a graph to represent a network of nodes connected by edges, with each node indicating a potential state. Incorporate the concept of vertices and edges to illustrate the connectivity between different states within the network.

8. Create a methodology to find the diameter of a cube configuration graph, such as a Rubik's Cube, and determine the optimal number of moves required to solve the cube. Discuss the implications of this metric in puzzle-solving strategies.

9. Invent an algorithm to compute the God's Number for different types of Rubik's Cubes based on their dimensions. Explain the significance of God's Number in the context of solving complex puzzles efficiently.

10. Develop a technique to represent a graph using object-oriented variations, such as creating an object for each vertex and storing information about neighbors. Discuss how this representation enhances graph manipulation and analysis.

11. Design a data structure to store and manage incidence lists for edges in a graph, allowing efficient access to outgoing edges from each vertex. Compare this method with other representations in terms of scalability and ease of traversal.

12. Create a step-by-step guide for implementing the breadth-first search algorithm on a graph to explore vertices level by level. Include considerations for handling different edge cases and optimizing the search process.

13. Sketch a graph illustrating the hierarchical structure of vertices labeled with levels and parent connections. Describe the relationship between levels and parent vertices in the context of graph traversal and exploration.

14. Formulate a plan to efficiently list vertices unreachable from a specified vertex by incorporating the concept of levels assigned to each vertex. Discuss the implications of identifying unreachable vertices in graph analysis.

15. Develop a program in Python that utilizes adjacency lists to represent a graph and performs a breadth-first search from a given starting vertex. Demonstrate the output of the search process and analyze the efficiency of the algorithm.

Evaluation Results:

QSTS: 0.82788987326

4.4.2 RESULTS FOR MEDIUM SIZE DOCUMENT

Input parameters:

Name of material = “Sorting Algorithms PDF(203 pages)”
course_name = "Data Structures and Algorithms"
num_questions = "5"
academic_level = "undergraduate"
taxonomy_level = "Create"
topics_list = "Sorting Algorithms"
major = "Computer Science"

Generated Questions:

1. Create a step-by-step guide illustrating the implementation of the Quicksort algorithm with a focus on the partitioning process and how elements are categorized based on a pivot element. Include explanations and code snippets to demonstrate the algorithm effectively.

2. Design a new sorting algorithm that combines the efficiency of Merge Sort with the partition approach of Quicksort. Explain the key components of your algorithm, how it handles dividing elements, and the process of recombining them into a sorted list. Provide detailed insights into the time complexity analysis and potential advantages of your proposed algorithm.

3. Develop a comprehensive comparison between the time complexities of Mergesort, Quicksort, Insertion Sort, and Selection Sort algorithms. Discuss in detail how each algorithm performs in best-case, worst-case, and average-case scenarios, highlighting the importance of understanding these variations in choosing the most suitable sorting method for a given dataset.

4. Formulate a detailed plan for implementing a divide-and-conquer sorting algorithm for a linked list structure. Outline the necessary helper functions for the divide and join steps, emphasizing the iterative nature of their implementation. Provide a roadmap for testing these helper functions and the overall sorting functionality of the algorithm.

5. Propose a novel approach to enhance the efficiency of a sorting algorithm by integrating a recursive sorting mechanism with an iterative component for certain operations. Explain how this hybrid approach can contribute to improving overall performance and scalability, particularly in scenarios where traditional sorting methods may exhibit limitations.

Evaluation Results:

QSTS: 0.7735986709594727

4.4.3 RESULTS FOR LARGE SIZE DOCUMENT

Name of the material = “Introduction to Algorithms (1312 pages)”
course_name = "Data Structures and Algorithms"
num_questions = "15"
academic_level = "undergraduate"
taxonomy_level = "Create"
topics_list = "graph algorithms, shortest path"
major = "Computer Science"

Generated Questions:

1. Create a new pathfinding algorithm that combines characteristics of both Breadth First Search and Depth First Search. Describe its approach and discuss its potential advantages over existing algorithms.

2. Design a novel graph search algorithm that can optimize route planning for a transportation system considering real-time traffic updates. Explain how this algorithm functions and its potential impact on logistics.

3. Propose an algorithm that can find the second-shortest path between two nodes in a weighted graph, considering scenarios where the shortest path is blocked. Provide a step-by-step explanation of how this algorithm works.

4. Develop a unique algorithm that utilizes random walks in graph traversal to analyze connectivity patterns in a network dataset. Discuss the significance of such an algorithm in understanding network behavior.

5. Devise a graph algorithm that calculates the cumulative weights of all paths connecting a set of nodes in a network. Explain how this algorithm can be beneficial for evaluating complex relationships within a graph.

6. Create an optimization algorithm that minimizes the total cost of traversing a graph with weighted edges. Illustrate how this algorithm can be applied in scenarios such as resource allocation or network routing.

7. Invent a graph algorithm that identifies nodes with the highest influence based on centrality metrics. Discuss the methodology employed and the significance of such an algorithm in analyzing network dynamics.

8. Develop an algorithm that efficiently computes the shortest path between all pairs of nodes in a graph, emphasizing its usefulness in scenarios like network reliability analysis or urban planning.

9. Propose a novel approach for calculating the minimum spanning tree in a graph with varying edge weights. Elaborate on the algorithm's implementation and its implications on network connectivity.

10. Create a customized pathfinding algorithm that adapts the concept of Eulerian and Hamiltonian cycles to find unique traversal paths in a graph. Explain how this algorithm can aid in exploring diverse routes efficiently.

11. Design an algorithm that leverages machine learning techniques to predict the most optimal paths in a graph based on historical data. Discuss the methodology used and potential applications of such an algorithm.

12. Develop a decision-making algorithm that evaluates multiple alternative routes between two locations in a graph considering various metrics such as time, distance, and cost. Describe the strategy employed and its advantages over standard routing algorithms.

13. Invent a graph search algorithm that incorporates dynamic weighting of edges based on changing conditions in the network. Explain how this algorithm can adapt to real-time data updates for efficient pathfinding.

14. Create an algorithm that identifies common patterns in graph structures through community detection techniques. Discuss the algorithm's role in identifying clusters of interconnected nodes and its significance in network analysis.

15. Design a graph optimization algorithm that minimizes the overall distance traveled in a network while visiting specific nodes with constraints. Discuss the algorithm's approach and potential applications in logistics and planning scenarios.

Evaluation Results:

QSTS: 0.6275986709594727

4.4.4 LLM-GUIDED EVALUATION RESULTS

Input:

Name of the material = “Graph Algorithms”
course_name = "Data Structures and Algorithms"
num_questions = "3"
academic_level = "undergraduate"
taxonomy_level = "Create"
topics_list = "graph algorithms, trees"
major = "Computer Science"

Generated Questions:

1. Design a new algorithm based on the principles of minimum spanning trees that can efficiently compute the second-best minimum spanning tree of a connected graph. Provide a detailed explanation of the algorithm and analyze its time complexity in terms of the number of vertices and edges in the graph.

2. Create a novel approach to preprocess a very sparse connected graph G before applying Prim's algorithm to improve the running time. Propose a method that selects minimum-weight edges incident on each vertex and contracts these edges to reduce the number of vertices. Justify the efficiency of your approach and compare it to the standard Prim's algorithm runtime complexity.

3. Develop an innovative technique to identify the maximum-weight edge on the unique simple path between any two vertices u and v in a given minimum spanning tree T. Formulate an O(V^2) time algorithm that can compute the maximum weight for all pairs of vertices in the tree T. Explain the steps of your algorithm and discuss its practical applications in graph analysis and optimization.

Evaluation:

1. Design a new algorithm based on the principles of minimum spanning trees that can efficiently compute the second-best minimum spanning tree of a connected graph. Provide a detailed explanation of the algorithm and analyze its time complexity in terms of the number of vertices and edges in the graph.

- Understandable: yes
- TopicRelated: yes
- Grammatical: yes
- Clear: yes
- Rephrase: NA
- Answerable: yes
- Central: yes
- WouldYouUseIt: yes
- SkillLevel: yes

2. Create a novel approach to preprocess a very sparse connected graph G before applying Prim's algorithm to improve the running time. Propose a method that selects minimum-weight edges incident on each vertex and contracts these edges to reduce the number of vertices. Justify the efficiency of your approach and compare it to the standard Prim's algorithm runtime complexity.

- Understandable: yes
- TopicRelated: yes
- Grammatical: yes
- Clear: yes
- Rephrase: NA
- Answerable: yes
- Central: yes
- WouldYouUseIt: yes
- SkillLevel: yes

3. Develop an innovative technique to identify the maximum-weight edge on the unique simple path between any two vertices u and v in a given minimum spanning tree T. Formulate an O(V^2) time algorithm that can compute the maximum weight for all pairs of vertices in the tree T. Explain the steps of your algorithm and discuss its practical applications in graph analysis and optimization.

- Understandable: yes
- TopicRelated: yes
- Grammatical: yes
- Clear: yes
- Rephrase: NA
- Answerable: yes
- Central: yes
- WouldYouUseIt: yes
- SkillLevel: yes

Feedback:
All the questions meet the evaluation criteria effectively. They are relevant to the course topic of Data Structures and Algorithms, are grammatically correct, clear, answerable, and aligned with the Bloom's taxonomy level specified. Thus, all questions are acceptable for assessment in an undergraduate Data Structures and Algorithms course.

4.5 DISCUSSION OF RESULTS

The experimental results demonstrate a promising application of LLMs in generating higher-order questions based on Bloom's taxonomy, particularly at the "Create" level, across varying document sizes and subjects within the domain of computer science. Overall, the system exhibits a robust performance in generating questions that stimulate critical thinking and problem-solving skills, as evidenced by the quality scores obtained (QSTS).

QSTS score is obtained by evaluating the semantic similarity of generated questions with the relevant contextual information retrieved from the database of uploaded study material. A high QSTS score ensures that the generated questions are majorly from the study material. It is obtained by the formula:

QSTS = cosine_similarity(candidate_questions, reference context)

Category	Title of the Document	Number of Pages	QSTS Score
Short Size Document	Breadth First Search	6	0.828
Middle Size Document	Sorting Algorithms	203	0.774
Long Size Document	Introduction to Algorithms	1312	0.628

Table 4.1: Results Achieved using the Proposed Methodology

4.5.1 PERFORMANCE ACROSS DOCUMENTS OF DIFFERENT SIZE

1. **Short Size Document:**
   For the short document (6 pages on Breadth First Search), the generated questions were detailed and multifaceted, covering algorithm design, implementation, and analytical tasks. With a QSTS of approximately 0.828, the system proved effective in formulating questions that require synthesis and creative problem solving. The diversity in question types—from designing pseudocode to conceptualizing new algorithms for computing metrics such as God's Number—indicates that the LLM effectively interpreted the content and produced inquiries that are both challenging and relevant.

2. **Medium Size Document:**
   The middle-sized document, a 203-page PDF on Sorting Algorithms, yielded a slightly lower QSTS of around 0.774. Although the generated questions maintained an emphasis on algorithm creation and comparative analysis, the reduced score might suggest that the system encountered challenges in managing the breadth of content. The questions still maintained a high academic standard, focusing on practical implementations, detailed explanations, and novel algorithmic proposals. This result points to a generally successful adaptation of the system to longer, more complex materials, albeit with some degradation in question specificity and quality.

3. **Large Size Document:**
   For the extensive document (1312 pages from “Introduction to Algorithms”), the QSTS dropped further to approximately 0.628. The generated questions in this case, while still adhering to the “Create” taxonomy level, tended to be more generic and less contextually rich compared to the shorter documents.
   The complexity and volume of the source material appear to have posed significant challenges for the LLM, due to the difficulty in distilling key concepts from such a large dataset. Consequently, the questions remain valid and thought-provoking, their relative quality and depth of specificity are impacted with a higher context length, indicating potential areas for improvement in handling extensive texts.

Based on the inferences from tests across different input document lengths , we infer that there exists a tradeoff between document’s token length and the quality of the question according to the QSTS score [20]. In all cases, the questions target an undergraduate level of understanding and consistently push toward higher-order thinking as defined by Bloom's taxonomy. This confirms the effectiveness of our LLM-based architecture in educational settings by generating questions that not only assess knowledge but also encourage the application, synthesis, and evaluation of learned concepts with suitable compliance to the input parameters and the bloom’s taxonomy.

Chapter 5

Conclusion and Future Work

5.1 CONCLUSION

The experimental results demonstrate the effectiveness of the proposed architecture in generating high-quality questions aligned with Bloom's Taxonomy, specifically at the "Create" level. The generated questions exhibit strong relevance to the input materials, spanning various algorithmic concepts, including graph traversal, sorting, and optimization. The quality of generated questions, measured by QSTS, showed a decreasing trend as document length increased, with scores of 0.827 for short documents, 0.773 for medium-sized documents, and 0.627 for long documents. This suggests that while the model maintains coherence in shorter texts, its performance slightly degrades when handling more extensive materials, potentially due to increased complexity and a wider scope of topics. Additionally, the system successfully formulated complex, open-ended questions that encourage problem-solving and creativity, indicating its suitability for educational applications in computer science.

5.2 FUTURE WORK

- Fine-tune the model on domain-specific datasets to improve contextual understanding and maintain question relevance and improve quality when deadline with a narrow range of input context.

- Implement human-in-the-loop feedback mechanisms for educators to evaluate and iteratively improve the generated questions.

- Addition of Grounding - access to web/verified external material for the LLM judge to improve the quality and robustness of the evaluation pipeline.

- Establish a Reinforcement Learning pipeline to improve the generation and evaluation iteratively using human user feedback thus solving the low relevancy or semantic issues in the generation.

- Distill smaller yet accurate models from powerful open source LLM's using reward based techniques like GRPO using humanly and synthetically generated data to reduce computation time and resources.

- Usage of an LLM based text , image and tables extraction pipeline to retain more minute details and structures from the input documents using OCR and by refining them using an LLM agent into a more LLM friendly formats such as Markdown or DITA-XML.

Appendices

Appendix 1

Understanding Vision Language Models (VLMs)

Vision Language Models (VLMs) represent a significant advancement in artificial intelligence, seamlessly integrating large language models with vision encoders. This fusion enables machines to process and reason about both textual and visual data, facilitating natural language interactions about images. The emergence of VLMs is revolutionizing the development of visual AI applications, paving the way for more intuitive and adaptable machine learning systems.

Moondream, a 1.6 billion parameter model meticulously developed by Vikhyatk, is the result of an innovative integration of SigLIP, Phi-1.5, and the extensive LLaVa training dataset [5]. This model represents a groundbreaking achievement in AI research, specifically designed for academic exploration and restricted to non-commercial applications. By leveraging cutting-edge methodologies and robust datasets, Moondream pushes the boundaries of artificial intelligence, setting a new standard for computational efficiency and innovation.

Unlike conventional computer vision models that rely on predefined classes and specific tasks, Moondream's VLMs offer remarkable versatility, allowing them to:

- Seamlessly process both text and image inputs
- Execute sophisticated visual reasoning
- Adapt to a broad spectrum of vision-related tasks
- Generate in-depth textual responses regarding visual content

This evolution in AI represents a transformative shift, democratizing access to powerful vision-based AI solutions.

How Vision Language Models Work

Recent advancements in transformer architectures and training methodologies have enabled the creation of compact yet efficient VLMs without compromising on performance. Moondream capitalizes on these innovations to deliver enterprise-grade visual AI accessible to a broad range of users.

The fundamental components of a VLM include:

1. **Vision Encoder:** Typically CLIP-based, this module extracts key features from images.
2. **Projector:** Converts visual features into a format compatible with the language model.
3. **Large Language Model (LLM):** Processes the transformed visual data to generate meaningful textual interpretations.

Through an optimized integration of these components, Moondream achieves performance levels comparable to models with 7 billion parameters, while requiring only 0.5B-2B parameters. This efficiency enables deployment on low-resource devices, such as Raspberry Pis and older smartphones, making high-quality vision AI more accessible.

Historical Evolution of Computer Vision

The journey of computer vision has seen significant milestones, evolving through distinct phases:

1. **1960s-1970s: Rule-Based Systems**
    - Early computer vision systems relied on manually coded rules.
    - These systems were highly specialized, rigid, and required extensive domain expertise.
2. **1980s-1990s: Classical Vision**
    - Geometric and statistical methods emerged.
    - Feature detection algorithms facilitated early pattern recognition.
    - Computational constraints limited their broader applicability.
3. **2000s-2011: Machine Learning Era**
    - Machine learning algorithms, such as Support Vector Machines (SVMs), became popular.
    - These approaches depended on manually crafted features.
    - They demonstrated limited generalization across diverse tasks.
4. **2012-2020: Deep Learning Revolution**
    - The success of AlexNet in the 2012 ImageNet competition marked a turning point.
    - Convolutional Neural Networks (CNNs) achieved human-level performance.
    - Transformer architectures and advanced feature learning techniques emerged.

5. **2021-Present: Multimodal AI Era**
    - The introduction of CLIP (2021) facilitated effective visual-language integration.
    - Vision encoders began working alongside LLMs.
    - Advanced architectures like SigLIP enhanced model efficiency.
    - Open-source models like Moondream increased accessibility and affordability.

Moondream exemplifies the latest evolution, enabling high-performance vision AI to function effectively in resource-constrained settings.

Challenges and Moondream's Solutions

Despite their advantages, VLMs still face several challenges:

- **Limited Input Resolution:** Most models operate within fixed image sizes, such as 224x224 or 336x336 pixels.
- **Spatial Understanding Constraints:** Precise localization and spatial reasoning remain challenging.
- **Video Understanding Limitations:** Many models struggle with extended temporal contexts.
- **Domain-Specific Fine-Tuning:** Some specialized applications require additional training.

Moondream addresses these challenges through:

- Efficient tiling techniques to handle high-resolution images.
- Optimized spatial reasoning models.
- Advanced context-handling mechanisms for improved video understanding.
- Streamlined fine-tuning approaches that maintain efficiency while improving domain adaptability.

The Breakthrough: A Paradigm Shift in Computer Vision

Traditional computer vision models are inherently limited. They rely on fixed-class classifications, require extensive retraining for new tasks, and lack the ability to understand images using natural language. In contrast, VLMs like Moondream represent a fundamental shift by enabling:

- **Flexible, Task-Agnostic Approaches:** Unlike traditional models, VLMs do not need task-specific training.
- **Natural Language Interaction:** Users can query images conversationally.
- **Zero-Shot Capabilities:** VLMs can generalize to new tasks without additional training.
- **Multimodal Understanding:** The integration of textual and visual data enhances contextual comprehension.

Moondream's Innovations: Efficient and Scalable AI

One of the biggest challenges in VLM adoption is the high computational cost. Many VLMs exceed 7 billion parameters, making them impractical for edge devices. Moondream overcomes this limitation through several innovations:

- **Advanced Model Pruning:** Eliminating redundant parameters while preserving performance.
- **Architecture Optimizations:** Enhancing cross-modal learning efficiency.
- **Specialized Training Approaches:** Achieving high performance with fewer parameters.
- **Breakthrough Vision-Language Alignment Methods:** Improving textual understanding of visual content.
- **Optimized Inference Paths:** Reducing resource consumption during real-time processing.

These breakthroughs allow Moondream to function effectively in environments with limited computational power, enabling its deployment across cloud servers and edge devices alike.

Benefits of Moondream's VLMs

Technical Advantages:

- Faster inference speeds.
- Lower resource consumption.
- Reduced infrastructure costs.
- Smaller carbon footprint, promoting sustainable AI.

Developer Benefits:

- Ability to run locally on consumer-grade hardware.
- Simplified deployment and scaling.
- Lower operational costs.
- Increased accessibility to advanced vision-language AI.

Applications Across Industries

VLMs are transforming numerous sectors by enabling new use cases:

- **E-Commerce:** Automating product tagging, visual search, and catalog management.
- **Healthcare:** Assisting in medical image analysis and report generation.
- **Accessibility:** Enhancing digital accessibility through automated alt text generation.
- **Content Moderation:** Enabling visual content filtering and compliance monitoring.
- **Education:** Facilitating interactive visual learning experiences.
- **Manufacturing:** Enhancing quality control and automated inspection processes.

The advent of VLMs marks a transformative moment in AI, bridging the gap between vision and language processing. Moondream exemplifies this revolution by delivering high-performance, resource-efficient models that democratize access to advanced visual AI. By overcoming traditional computer vision limitations, Moondream sets the stage for a future where multimodal AI is both powerful and accessible, driving innovation across diverse industries.

REFERENCES

[1] Bloom, B. S., Engelhart, M. D., Furst, E. J., Hill, W. H., & Krathwohl, D. R. (1964). Taxonomy of educational objectives (Vol. 2). New York: Longmans, Green.

[2] Conklin, J. (2005). [Review of *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives Complete Edition*, by L. W. Anderson, D. Krathwohl, P. Airasian, K. A. Cruikshank, R. E. Mayer, P. Pintrich, J. Raths, & M. C. Wittrock]. *Educational Horizons*, *83*(3), 154–159. http://www.jstor.org/stable/42926529

[3] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. NeurIPS 2020. arXiv. https://doi.org/10.48550/arXiv.2005.11401

[4] Gao, L., Ma, X., Lin, J., & Callan, J. (2023). Precise Zero-Shot Dense Retrieval without Relevance Labels. In A. Rogers, J. Boyd-Graber, & N. Okazaki (Eds.), Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 1762–1777). doi:10.18653/v1/2023.acl-long.99

[5] Ghosh, A., Acharya, A., Saha, S., Jain, V., & Chadha, A. (2024). Exploring the Frontier of Vision-Language Models: A Survey of Current Methodologies and Future Directions. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2404.07214

[6] Paul, A., Sabu, A., Abdulkader, B., George, P., & Sreedevi, S. (2021). QGen: An automatic question paper generator. In Advances in intelligent systems and computing (pp. 555–564). https://doi.org/10.1007/978-981-16-5157-1_43

[7] Paul, R. J., Jamal, S., Bejoy, S., Daniel, R. J., & Aju, N. (2024). QGen: Automated Question Paper Generator. 2024 5th International Conference on Innovative Trends in Information Technology (ICITIIT), 1–4. doi:10.1109/ICITIIT61487.2024.10580391

[8] Ragasudha, R., & Saravanan, M. (2022). Secure Automatic Question Paper Generation with the Subjective Answer Evaluation System. 2022 International Conference on Smart Technologies and Systems for Next Generation Computing (ICSTSN), 1–5. doi:10.1109/ICSTSN53084.2022.9761323

[9] Scaria, N., Chenna, S. D., & Subramani, D. N. (2024). How Good are Modern LLMs in Generating Relevant and High-Quality Questions at Different Bloom's Skill Levels for Indian High School Social Science Curriculum? BEA, 1–10. Retrieved from https://aclanthology.org/2024.bea-1.1

[10] Guo, S., Liao, L., Li, C., & Chua, T. (2024). A survey on Neural Question Generation: Methods, Applications, and Prospects. 8038-8047. 10.24963/ijcai.2024/889., 8038–8047. https://doi.org/10.24963/ijcai.2024/889

[11] Ackerman, R., & Balyan, R. (2023). Automatic Multilingual question generation for health data using LLMs. In Communications in computer and information science (pp. 1–11). https://doi.org/10.1007/978-981-99-7587-7_1

[12] Lohr, D., Berges, M., Chugh, A., Kohlhase, M., & Müller, D. (2024a). Leveraging large language models to generate Course-Specific semantically annotated learning objects. Journal of Computer Assisted Learning, 41(1). https://doi.org/10.1111/jcal.13101

[13] Li, K., & Zhang, Y. (2024). Planning first, Question second: An LLM-Guided method for controllable question generation. Findings of the Association for Computational Linguistics: ACL 2022, 4715–4729.

https://doi.org/10.18653/v1/2024.findings-acl.280

[14] Meißner, N., Speth, S., Kieslinger, J., & Becker, S. (2024). EvalQuiz – LLM-based Automated Generation of Self-Assessment Quizzes in Software Engineering Education. Software Engineering im Unterricht der Hochschulen 2024 (pp. 53–64). doi:10.18420/seuh2024_04

[15] Scaria, N., Chenna, S. D., & Subramani, D. (2024). Automated educational question generation at different Bloom's skill levels using large language models: strategies and evaluation. In Lecture notes in computer science (pp. 165–179). https://doi.org/10.1007/978-3-031-64299-9_12

[16] Elkins, S., Kochmar, E., Cheung, J. C. K., & Serban, I. (2024). How teachers can use large language models and bloom's taxonomy to create educational quizzes. Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence. doi:10.1609/aaai.v38i21.30353

[17] Duong-Trung, N., Wang, X., & Kravčík, M. (2024). BloomLLM: Large Language Models based question Generation combining Supervised Fine-Tuning and Bloom's Taxonomy. In Lecture notes in computer science (pp. 93–98). https://doi.org/10.1007/978-3-031-72312-4_11

[18] Babakhani, P., Lommatzsch, A., Brodt, T., Sacker, D., Sivrikaya, F., & Albayrak, S. (2024). Opinerium: subjective question generation using large language models. IEEE Access, 12, 66085–66099. https://doi.org/10.1109/access.2024.3398553

[19] Chen, Y., & He, L. (2025). The Practice of Large Language Models in Automated Question Generation: A Case Study of ChatGLM in High School Information Technology Curriculum. Education Reform and Development, 6(12), 311–318. https://doi.org/10.26689/erd.v6i12.9384

[20] Fu, Weiping; Wei, Bifan; Hu, Jianxiang; Cai, Zhongmin; and Liu, Jun. (2024). "QGEval: Benchmarking Multi-dimensional Evaluation for Question Generation." arXiv preprint arXiv:2406.05707. Available at: https://arxiv.org/abs/2406.05707.

[21] Ushio, Asahi; Alva-Manchego, Fernando; and Camacho-Collados, Jose. (2023). "Generative Language Models for Paragraph-Level Question Generation." arXiv preprint arXiv:2210.03992. Available at: https://arxiv.org/abs/2210.03992.

[22] Deroy, Aniket; Maity, Subhankar; and Sarkar, Sudeshna. (2025). "MIRROR: A Novel Approach for the Automated Evaluation of Open-Ended Question Generation." arXiv preprint arXiv:2410.12893. Available at: https://arxiv.org/abs/2410.12893.