REC QUESTION PAPER GENERATOR

MINI PROJECT REPORT

Submitted by

Priya Dharshini	**210701197**
Sajay Prakash K	**210701222**

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI ANNA UNIVERSITY::
CHENNAI 600 025

MAY 2024

BONAFIDE CERTIFICATE

Certified that this Report titled "**REC QUESTION PAPER GENERATOR**" is the bonafide work of **"Priya Dharshini (210701197) and Sajay Prakash K (210701222)"** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. Rakesh Kumar M, M.E, PhD.,

Assistant Professor,

Department of Computer Science and Engineering,

Rajalakshmi Engineering College,

Chennai – 602015

Submitted to Mini Project Viva-Voce Examination held on _____

Internal Examiner **External Examiner**

ABSTRACT

Today, institutions spend many days preparing well-articulated question papers for students across various subjects. The questions themselves need to be structured so that each subject's units are given equal weight. Additionally, the questions must guarantee that students receive proper educational value. The professors have to put in a lot of time and effort to formulate, check, and turn in the question papers. This application aims to streamline the process of creating these question papers by leveraging large language models, the Retrieval Augmented Generation (RAG) technique, and an intuitive interface. The content from the PDF notes is extracted using Natural Language Processing (NLP) techniques, and it is then saved in a vector database. The retrieval augmented generation (RAG) technique and the LangChain framework are used by the large language model to generate question papers based on predefined question paper patterns and the context provided by the Portable Document Format (PDF) notes.

ACKNOWLEDGEMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavour to put forth this report. Our sincere thanks to our Chairman **Mr. S.MEGANATHAN, B.E, F.I.E.**, our Vice Chairman **Mr. ABHAY SHANKAR MEGANATHAN, B.E., M.S.,** and our

respected Chairperson **Dr. (Mrs.) THANGAM MEGANATHAN**, **Ph.D.,** for providing us with the requisite infrastructure and sincere endeavoring in educating us in their premier institution.

Our sincere thanks to **Dr. S.N. MURUGESAN, M.E., Ph.D.,** our beloved Principal for his kind support and facilities provided to complete our work in time. We express our sincere thanks to **Dr. P. KUMAR, Ph.D.**, Professor and Head of the Department of Computer Science and Engineering for his guidance and encouragement throughout the project work. We convey our sincere and deepest gratitude to our internal guide, **Dr. Rakesh Kumar M, M.E, PhD.,** Assistant Professor, Department of Computer Science and Engineering. Rajalakshmi Engineering College for his encouragement and guiding us throughout the project to build our project.

Priya Dharshini - 210701197

Sajay Prakash K - 210701222

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

1. **LLM** - Large Language Model
2. **RAG** - Retrieval Augmented Generation
3. **NLP** - Natural Language Processing
4. **PDF** - Portable Document Format
5. **AI** - Artificial Intelligence
6. **GPU** - Graphics Processing Unit
7. **API** - Application Programming Interface
8. **OCR** - Optical Character Recognition
9. **ML** - Machine Learning
10. **DL** - Deep Learning
11. **QA** - Question Answering
12. **IR** - Information Retrieval
13. **SVM** - Support Vector Machine
14. **ANN** - Artificial Neural Network
15. **BERT** - Bidirectional Encoder Representations from Transformers
16. **GPT** - Generative Pre-trained Transformer
17. **T5** - Text-to-Text Transfer Transformer

CHAPTER 1
INTRODUCTION

1.1 GENERAL

The process of generating examination papers is an essential but daunting task in schools. Achieving the objective of balancing the units of each subject and having the questions with the required educational content is a very difficult task and requires a lot of time and effort from professors. Historically, this process takes place in many stages; for example, drafting, review, and finalization stages, which together require extensive time and effort to develop question papers.

With reference to the fact of the digital era of the modern world it has been turned out to be necessary and essential to employ technology in order to enhance processes in education. The next future of question papers and NLP: towards using Large Language Models. These technologies can not only minimize much of the effort in this tedious process but also help in the development of better questions and those relevant to learning.

The following services are included in the proposed application: This paper describes the application for automating and improving the creation of question papers. This system is expected to be useful in relieving professors in terms of workload on question papers and enhancing the efficiency of the question paper formulation process using large language models and the RAG technique along with an easy user interface. This implies that the first step involves use of NLP techniques to extract contents from PDF notes which are stored in a vector database for easy retrieval. The RAG technique and the LangChain framework are established in generating questions based on the patterns defined in generating the question and adding the context from the notes extracted in the question paper.

1.2 OBJECTIVE

The main target of this project is building an automated framework that combines large language models with RAG and easy-to-use interface to make the process of creating question papers easier.

The proposed system aims at simplifying the work of getting questions papers ready for assessment processes and also saving time for the professors in formulating, reviewing and finalizing questions instead of expending a lot of time on these processes. A major objective is to minimize the bias of question papers generated so that they will cover the correct number of units in a balanced manner for the subject which ensures adequate evaluation and educational worth. The implementation of the project involves the application of the latest Natural Language Processing (NLP) to transform PDF notes into relevant content that can be stored and indexed in the vector database for fast retrieval.

On the basis of the RAG technique and LangChain framework, the system suggests and produces contextually adequate questions, the answers to which follow the defined template and content obtained. Second, the project aims to improve the quality of teaching and learning with appropriately worded, logically organized, and pedagogically valuable question content aligned with the course outcomes. Ease of use will be achieved with the development of an easy to understand user interface that will enable the educators to enter specifications and choose the pattern of the question paper for the generation of questions from the virtual question bank and advice on how the generated questions can be modified before the final versions are downloaded. By achieving these goals, the project will better the process of question paper formation in institutions of learning with regards to the effectiveness, quality, and reliability.

1.3 EXISTING SYSTEM

Manual methodology for designing question papers is the standard practice that is applied in most educational institutions today. I understand that the faculty members need to prepare questions; ensure the subjects are covered to a balanced extent; and preserve the educational purpose of the tests. This involves several key steps: developing questions for each unit on their own without using recall templates; proofreading the questions several times to eliminate errors, ensure proper wording, and represent every unit in a similar fashion, which is a time-consuming activity; maintaining the balance and arrangement of the questions between units; finalizing and formatting the question paper for the printing or the e-learning distribution, including clear explanations and spacing; and their critical review for educational and pedagogical quality and adherenceSuch traditional approaches may be resource and time-consuming and are open to, albeit unintentional, uneven quality and coverage of questions.

It takes longer than the given time frame and the question papers remain unavailable for a longer time; and the quality and balance of the questions are dependent on the expert and he can have different perceptions about the quality and balance of the questions than the others. Still using these methods is the second reason why many institutions continue to rely on conventional methods despite the challenges because most of the automated tools are currently designed for different fields rather than for the field of question.

1.4 PROPOSED SYSTEM

The proposed system is capable of automating the process of creation and improvement of question papers and assisting the examiner by using large language models and RAG i. e retrieval augmented generation and a simple UI. This system aims to solve the problems and disadvantages associated with the current manual process of handling educators' reservations which are not as efficient, cost effective, and consistent as they should be.

Key features of the proposed system include:Key features of the proposed system include

Content Extraction using NLP: It uses the latest Natural Language Processing (NLP) to scan through PDF notes and other class materials to derive useful content. The former is then stored in a vector database which enables easier retrieval of the content and use in question generation.

Retrieval Augmented Generation (RAG): i) RAG technique: The system first extracts the relevant content using the RAG technique and then integrates it with the capabilities of large language models to create contextually relevant questions. This not only makes the questions factual but also makes them relevant to the context of the information drawn from the extracted notes.

Predefined Question Patterns: It has logic built into it that dictates the frequency of questions and their structure. These patterns mean that the question papers are constructed meaningfully, suitable units of the subject studied are covered, and a reasonable fare of assessment is given.

Intuitive User Interface: The interface for the educators has been simplified to make it possible to enter their needs, select appropriate question paper patterns based on the needs, and review the questions generated by the system. This user interface will guarantee that the users can be easily able to manage the process of creating the question paper without much technical knowhow.

Automated Balancing and Structuring: The system balances the coverage of all subject units being under control of the system which also solves one of the problems that were mentioned in relation to the manual system. This feature ensures that all units are evenly represented or proportioned in the question paper.

Quality Assurance: The RAG technique is used along with advanced capabilities of LLM to translate the learning objectives in easy language and also provides the pedagogically sound and clear students' questions. This leads to increased general level of educational content of the papers.

CHAPTER 2

LITERATURE SURVEY

This application will automate the generation of question papers, using subject notes and a given pattern of question paper. The manual generation of question papers is one of the laborious jobs that takes a lot of effort from professors. In addition to framing questions that would correctly assess the understanding of topics by the students, this needs to conform to the educational objectives in addition. There are also additional issues, such as question papers are revised several times and have to be generated in different sets.Our application automates all the above tasks, and we lessen the burden on the educators and ensure high standard quality for the generated question paper. The application allows users to upload their notes through a simple drag-and-drop user interface, and with one button click, a whole question paper is generated. This removes the need for manual work and excessive editing.Large language models, like the ChatGPT-4, can be used to achieve results similar to that but are not optimized for this task. Such general models require elaborated prompts and a lot of human intervention to produce well-structured questions. In contrast, our application is designed only for the generation of question papers, making our application a much easier and user-friendly alternative.Flexibility is one of the main application advantages. The user can easily change the question paper pattern by changing the pattern variable in the code, thus enabling its customization for specific needs. In addition, the application supports the use of different open-source models, giving the user a choice between using a Llama2 model by Meta AI, Mixtral models by Mistral AI, and many more. This makes sure that a user is not limited to just ChatGPT-4 but is free to choose the best model that would work for his needs.Besides saving teachers some much-needed effort, our application ensures that the question papers so generated maintain quality and relevance. To ensure that professors are more focused on lecturing and reaching out to students other than administrative roles, we automate tasks like framing of questions and revision cycles.The application is also characterized by a user-friendly interface whereby users can upload their subject notes using drag-and-drop functionality. With a single click, the question paper is generated without any need for human labor and heavy editing. Our approach is efficient and contrasts with the use of giant language models such as ChatGPT-4, which can be used but needs detailed prompts and human inputs to finally come up with questions well-structured for an examination.

Our application also offers flexibility; users can easily modify the pattern of the question paper just by changing the variable pattern within the code. This ability ensures that question papers meet specific educational needs and objectives. Further, the application accepts several open-source models such as Llama2 by Meta AI, Mixtral models by Mistral AI, and so forth. This freedom in our application allows the user to choose the most relevant model for the work, thus enhancing the effectiveness and relevance of the question papers generated.Thus, our application is an important advancement in automating educational processes, where a simple and efficient solution is provided in generating question papers of good quality and customized to suit the educational context of the individual.

CHAPTER 3

SYSTEM DESIGN

3.1 DEVELOPMENT ENVIRONMENT

3.1.1 HARDWARE SPECIFICATIONS

This project uses minimal hardware but in order to run the project efficiently without any lack of user experience, the following specifications are recommended

Table 3.1.1 Hardware Specifications

PROCESSOR	Intel Core i5
RAM	4GB or above (DDR4 RAM)
GPU	Intel Integrated Graphics
HARD DISK	6GB
PROCESSOR FREQUENCY	1.5 GHz or above

3.1.2 SOFTWARE SPECIFICATIONS

The software specifications in order to execute the project has been listed down in the below table. The requirements in terms of the software that needs to be pre- installed and the languages needed to develop the project has been listed out below.

Table 3.1.2 Software Specifications

FRONT END	Streamlit
BACK END	Python
FRAMEWORKS	LangChain, Streamlit, Groq, Ollama
SOFTWARES USED	Python IDLE, Google Chrome

3.2 SYSTEM DESIGN

3.2.1 ARCHITECTURE DIAGRAM

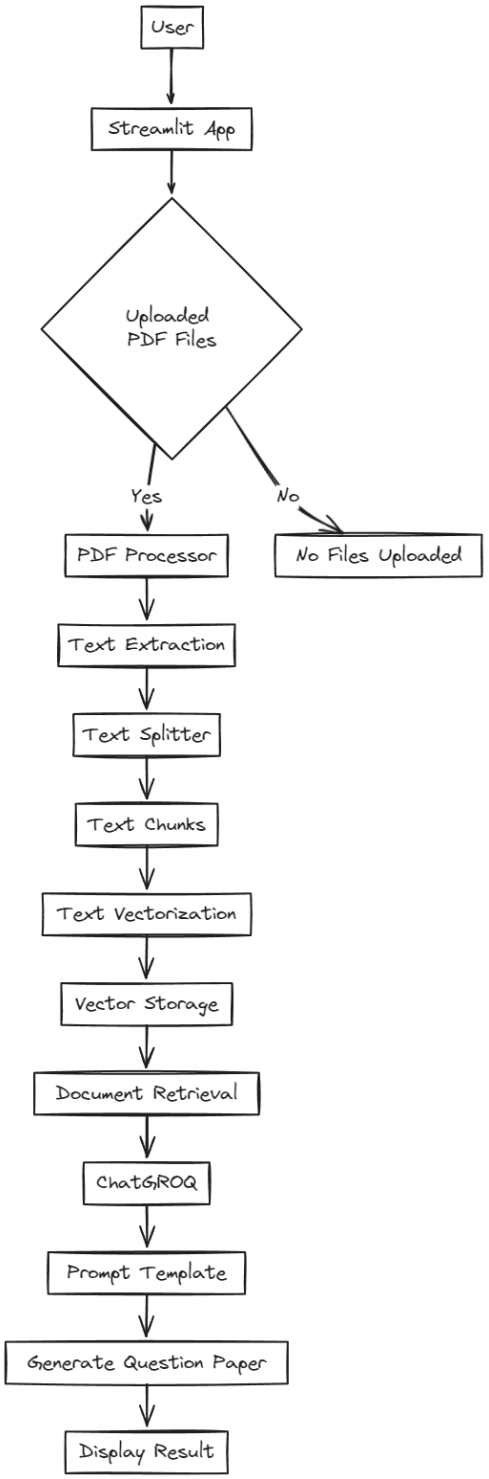


Fig 3.2.1 Architecture Diagram

The system architecture, as seen in Fig 1, represents the process of generating question papers based on the PDF notes uploaded via a StreamLit application. Users interact with the application for file upload of PDF files containing study materials. After uploading, PDF Processor initiates the process of text extraction from each document, which will be further segmented by the Text Splitter component into manageable chunks.Further, the text chunks are taken to Text Vectorization, where they are transformed into vector representations in things like Ollama Embeddings. These vectors then go into Vector Storage, usually managed by FAISS for efficient retrieval.From here, when a user specifies the exam type, for example, CAT-1 or End Semester, through prompts on the Streamlit interface, the document retrieval module retrieves relevant text chunks based on the stored vectors. This retrieval process will be done using a retrieval chain, ensuring the system identifies and gathers all the relevant content for question paper generation.ChatGroq, through the Groq API and a prompt template specified for that exam type, uses the generated question chunks to return structured question papers.It uses the template to stipulate the format and contents that should be expected for every type of exam. The displayed question paper that has been generated shows to the user with Display Result via the Streamlit interface. The application, hence, harnesses text processing, vectorization, and retrieval techniques, integrated with an API, to make it super efficient and advanced with regard to automated generation of question papers, which supports efficiency and personalization in educational settings.

3.2.2 Natural Language Processing

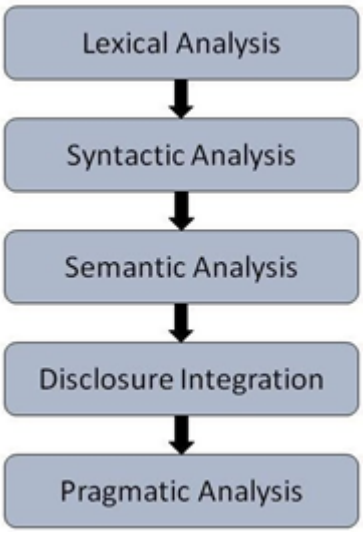


Fig 3.2.2 Steps in Natural Language Processing

Natural language processing is one of the many applications of Artificial Intelligence (AI). Using deep learning algorithms we can improve the ability of the AI to interpret and generate natural language which would be useful in this research[1]. Through Natural Language Processing(NLP) techniques, we are able to extract the contents of the notes and store it using a Vector Database.

3.2.3 Vector Database

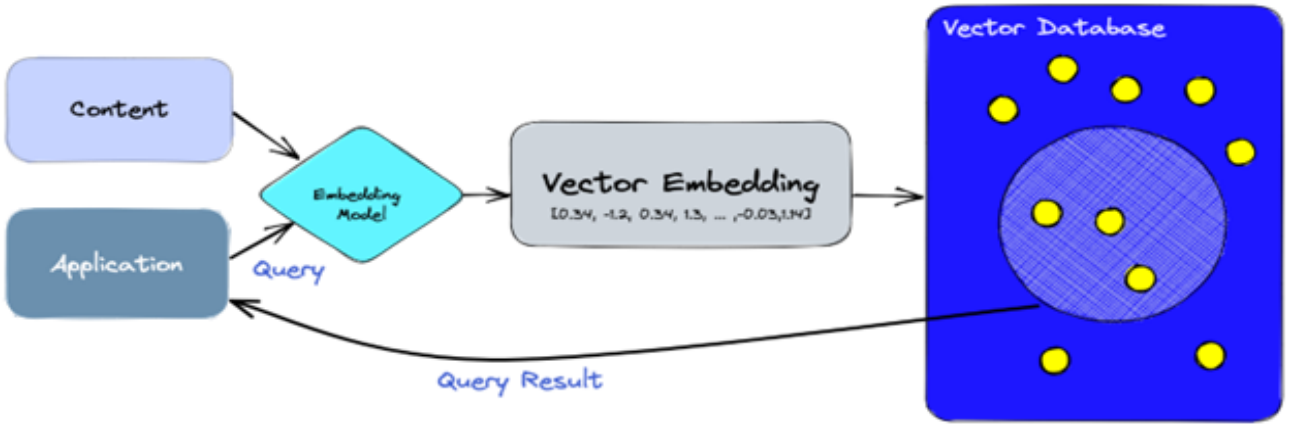


Fig 3.2.3 Role of Vector Database in Applications

A vector database indexes and stores vector embeddings for fast retrieval and similarity search.[2] One of the main advantages of vector databases is their ability to handle various data types such as text, images, audio, etc. which is essential for storing information from the notes and perform similarity search, clustering and classification. Each paragraph and sentence is transformed into a vector representation and are indexed and stored into the database.

3.2.4 Working of a Vector Database

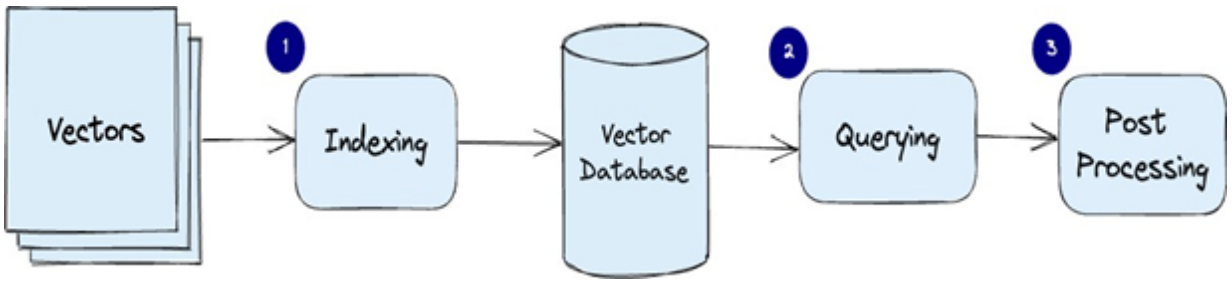


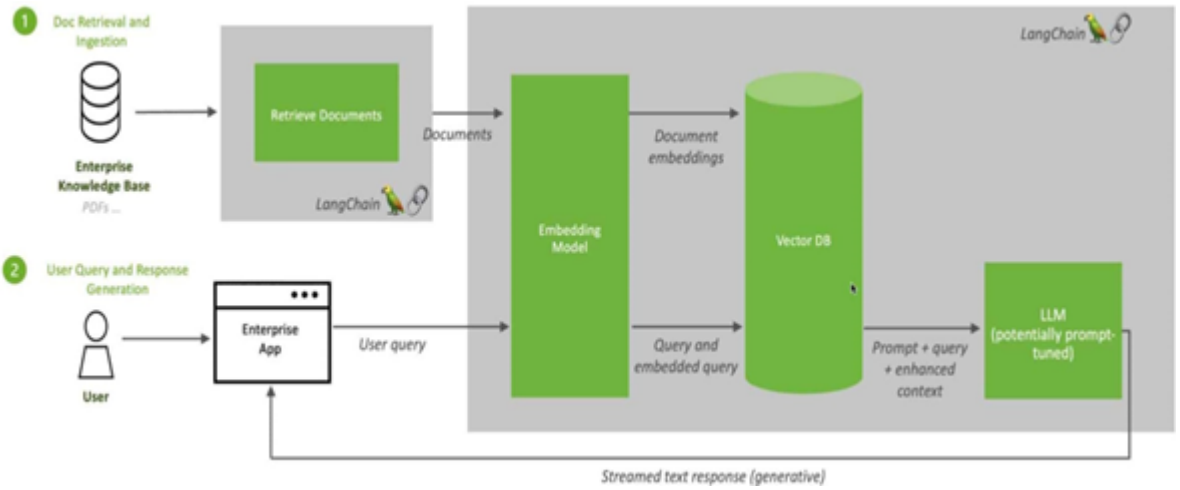
Fig 3.2.4 Working of a Vector Database

1. Indexing: The vector database indexes vectors using an algorithm such as PQ, LSH, or HNSW (more on these below). This step maps the vectors to a data structure that will enable faster searching.
2. Querying: The vector database compares the indexed query vector to the indexed vectors in the dataset to find the nearest neighbors (applying a similarity metric used by that index)

3. Post Processing: In some cases, the vector database retrieves the final nearest neighbors

from the dataset and post-processes them to return the final results. This step can include re-ranking the nearest neighbors using a different similarity measure.[3]

3.2.5 Retrieval-Augmented Generation (RAG)



3.2.5 Retrieval-Augmented Generation (RAG)

RAG is an AI framework for retrieving facts from an external knowledge base to ground large language models (LLMs) on the most accurate, up-to-date information and to give users insight into LLMs' generative process.[4] Retrieval-Augmented Generation (RAG) plays an important role in ensuring the accuracy and focus of the generated questions.

While LLMs like Llama-2 are powerful tools for text generation, their knowledge is limited to the data they were trained on. The training data used might not include the exact content of the notes that are required for the questions in the question paper. Thus, RAG is used to bridge this gap by acting as a mechanism to retrieve relevant information from a knowledge base (Vector Database).

CHAPTER 4

PROJECT DESCRIPTION

4.1 MODULE DESCRIPTION

4.1.1 Content Extraction Module

Function: This module is concerned with retrieving the relevant content of PDF notes and other educational resources using Natural Language Processing (NLP) algorithms.

Components:

PDF Parser: Reads the content from pdf and retrieves the text from it

Text Preprocessing: Cleaning and pre-processing is carried out here.

Entity Recognition and Key Phrase Extraction: Tagging – identifies the most important concepts, topics and entities in the text.

4.1.2 Vector Database Module

Function: Organizes the processed content in a way that makes it easy to access when generating questions.

Components:

Vectorization Engine: Transforms textual data into numerical vectors by using approaches such as TF-IDF, word embedding models (Word2Vec), or BERT.

Database Management System: Responsible for vectorized data storing and indexing and retrieving.

4.1.3 Question Generation Module

Function: Retrieves questions from content and predefined patterns using the Retrieval Augmented Generation (RAG) and large language models.

Components:

Retrieval Engine: Retrieves content from the vector database that meets the context necessary for question generation.

Language Model Interface: It performs great with LLMs. The content that is retrieved is then fed to GPT to generate questions from the content that was retrieved.

Pattern Application: Prescribes specific question styles to facilitate the preparation of balanced question papers.

4.1.4 Question Pattern Module

Function: Ensures systematic and consistent coverage of the subject units through the use of pre-designed patterns for question papers.

Components:

Pattern Repository: Stores ready templates and patterns for different types of questions and subjects.

Customization Interface: Enables instructors to create and configure their own question formats.

4.1.5 User Interface Module

Function: Offers an easy way for educators to use the system, set criteria, specify the patterns, and check the created questions.

Components:

Input Forms: Collect the requirements and preference from the user for the question paper.

CHAPTER 5

IMPLEMENTATION AND RESULTS

5.1 IMPLEMENTATION

1. Setup Environment

Install Necessary Libraries: Set up natural language processing libraries such as transformers, pdf miner. Six is for the PDF extraction of the application, langchain for building language model applications and Faiss for functioning of vector databases.

2. First, extract text from from the PDFs

Extract Text: Use pdfminer. six to scan handwritten notes or documents If you have a lecture note or some study material in a PDF file it can use six to extract the raw text from it.

3. Preprocess Text

Text Preprocessing: Being a machine learning algorithm, it involves preprocessing of the extracted text where parameters such as lower casing, tokenization, removal of stop words among others are involved to prepare the text for the generation of its embedding.

4. Vector Database for Text Store

Convert Text to Embeddings: For instance, you should feed the preprocessed text to a pre-trained language model such as BERT or similar and then convert the prediction of the model into embeddings (numbers).

Store Embeddings: Store these embeddings in a vector database such as FAISS so that similarity search and information retrieval are possible for inquiring about similar information.

5. Setup RAG Model: Set up an RAG model, where the user input can be retrieved from a database, and the model can further generate an answer to a question.

Generate Questions: In order to create questions for the study materials, apply the RAG model to the amount of concrete contexts and topics extracted from the materials. This means that by querying the vector database, one gets to identify relevant passages and, by use of the language model, formulate questions.

6. Interface for Professors

User Interface (UI): Create an efficient way for professors to input the type of information they require for the course, i. e. topic or unit, web application or command line tool. It should also provide them with convenient tools to review the questions, modify them from the preset options, and confirm the generated questions.

5.2 OUTPUT SCREENSHOTS

REC QPG - This is a web application designed to generate question papers based on uploaded PDF notes. The design of the application is given below.

5.2.1 Upload Notes

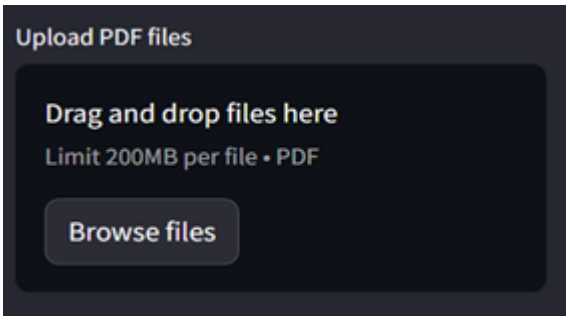


Fig 5.2.1 PDF Upload Screen

First, the user has to drag and drop the PDF files of the notes that have to be used as context for the question paper. Optionally, the user can also choose to open file explorer and select the files that are to be uploaded.

5.2.2 Process PDF Files

Fig 5.2.2 Process PDF Files

The PDF files are then processed in phases. First, the PDF is converted into plain text and the large amounts of text is split into smaller chunks. Each of these chunks represent some meaningful data. This chunk is then stored into a vector database in a mathematical representation and can be used for generation of the question papers.

5.2.3 Choose Question Paper Pattern

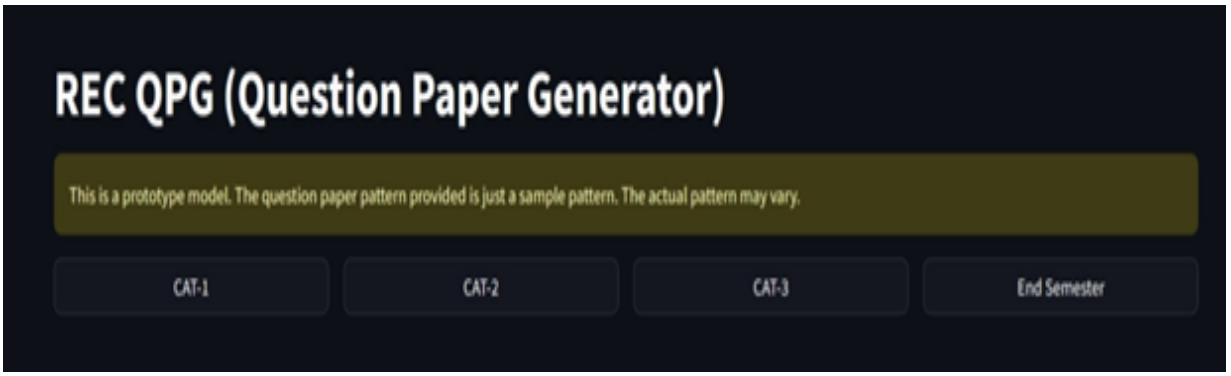


Fig **5.2.3 Choose Question Paper Pattern**

The user has the choice of selecting from the various examinations. Depending on the user's selection a pattern for the question paper will be set. The questions that are drafted will be based strictly upon the basis of the specified question paper pattern.

5.2.4 Generated Question Paper

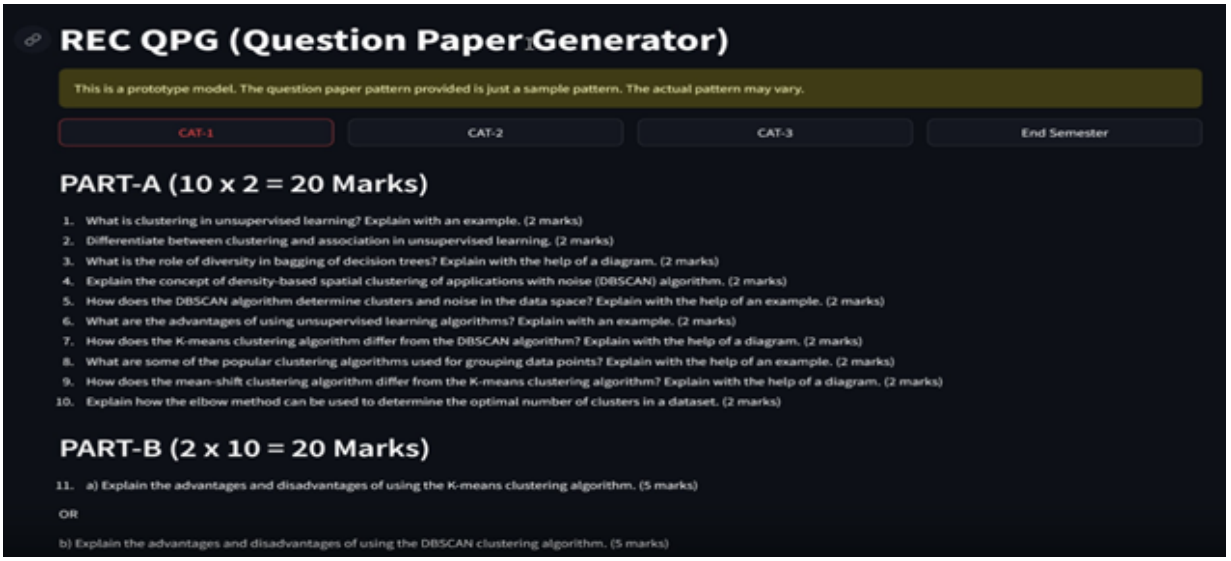


Fig 5.2.5. Final Output

Finally, the question paper is generated based on the specified pattern and the questions are based upon the uploaded notes. Each generation will give a new set of questions, so this can be used to generate multiple sets of a single paper.

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENTS

6.1 CONCLUSION

In conclusion, this application not only accelerates the question paper generation process but also improves the quality and relevance of the papers generated. Automating the monotonous aspects of question paper creation enables professors to spend more time focusing on other aspects of teaching and engaging their students. Furthermore, this application creates a level playing field for all assessment aspects by ensuring that all subjects and topics are well-represented in the question papers. Moreover, institutions can customize the question papers to align with their individual curriculum demands and assessment objectives. Be it based on the difficulty of questions, some topics that should be given weight, or the types of questions to include, the educators will have the freedom to adapt question papers to themes that suit their various categories. By leveraging technologies such as natural language processing, large language models, and retrieval augmented generation, the application empowers educators to create high-quality question papers efficiently, benefiting both students and institutions.

Through NLP, LLM, and RAG, the application facilitates educators in generating the best question papers with high efficiency, benefiting students and the institution in various ways. With the integration of NLP, the application delves into the contents of subject notes and thus derives questions that are contextually relevant and meet the educational criteria. The application aims to create appropriate and meaningful questions that will test the comprehension and critical thinking skills of students with paramount effectiveness.Furthermore, the RAG increases precision and relevance by integrating the information retrieval capability. This approach not only makes the assessment process much more enriching but also includes the capacity for the educator to draw from a wide range of sources while formulating questions. The application saves time by automating tasks that an educator has to do, like formatting and arranging questions in a specified pattern while generating question papers. This allows professors to spend more time on interactive teaching methods and student support.Most importantly, the application tailors question papers according to specific demands from the curriculum and objectives of the assessment, thus ensuring that educators develop assessments for their specific needs and educational goals.

Whether it is question difficulty, topic weighting, or picking the right type of question, the educator has the freedom to design an assessment that will be best at measuring understanding and progress.This is a leap in educational technology because it makes assessment practices easier, faster, and individualistic. This application automatically removes tedious administrative jobs from educators' tasks and enhances the quality and relevance of assessments. Ultimately improving student learning outcomes and contributing to the excellence of education at the institution.

6.2 FUTURE ENHANCEMENTS

Adaptive Learning Integration: Use adaptive learning strategies to change and revise questions to reflect on individual students' performance. It can also manipulate the type of questions and the level or area of concentration which relate to previous accomplishment by the learner.

Advanced Analytics and Reporting: Leakage – Use advanced analytical tools to issue the reports on quality and difficulty level of the questions covered in the question paper. The use of past assessments can shed light on overall trends on the performance of students.

Integration with Learning Management Systems (LMS):Integration with Learning Management Systems (LMS): Integrate with general LMS platforms – such as Moodle, Blackboard e. g. Instructors should advocate for an active adoption of LMSs such as Moodle, Blackboard, or Canvas for the ease of transferring quiz and assessment questions.

REFERENCES

1. Alawwad, Hessa Abdulrahman, Areej Alhothali, Usman Naseem, Ali Alkhathlan, and Amani Jamal. "Enhancing Textbook Question Answering Task with Large Language Models and Retrieval Augmented Generation." *arXiv preprint arXiv:2402.05128* (2024).
2. Madhav, Dinesh, Sanskruti Nijai, Urvashi Patel, and Komal Champanerkar. "Question Generation from PDF using LangChain." In *2024 11th International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 218-222. IEEE, 2024.
3. Pinecone. "What is a Vector Database?"
4. IBM Research. "Retrieval-Augmented Generation (RAG)."
5. Brown, T. B., et al. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33, 1877-1901.
6. Radford, A., et al. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*.
7. Raffel, C., et al. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140), 1-67.
8. 8. Black, S., et al. (2022). GPT-NeoX-20B: An open-source autoregressive language model. *arXiv preprint*.
9. Lewis, P., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33, 9459-9474.
10. Izacard, G., & Grave, E. (2020). Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint*.
11. Yih, W. T., et al. (2020). Pre-trained text encoders for scalable zero-shot retrieval. *arXiv preprint*.
12. LangChain Documentation. (2022). LangChain.
13. LangChain GitHub Repository. (2022).
14. Sultanum, N., et al. (2019). PDF-tExtract: a flexible framework for PDF data extraction and visualization. *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*.
15. Lozano, A. C., et al. (2019). Tabula: Extracting tables from PDFs. *The Journal of Open Source Software*, 4(36), 1246.

16. Zhong, X., et al. (2019). PubLayNet: largest dataset ever for document layout analysis. *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*.
17. Johnson, J., et al. (2019). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3), 535-547.
18. Guo, R., et al. (2020). Accelerating large-scale inference with anisotropic vector quantization. *Proceedings of the 37th International Conference on Machine Learning*.
19. Harris, C. R., et al. (2020). Array programming with NumPy. *Nature*, 585(7825), 357-362
20. Zhu, Y., et al. (2019). Texar: A modularized, versatile, and extensible toolkit for text generation. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.
21. Agarwal, R., & Mannem, P. (2021). Automatic gap-fill question generation from text books. *Proceedings of the Sixth Workshop on Innovative Use of NLP for Building Educational Applications*.
22. Ghosh, S., et al. (2020). Automated generation of multiple-choice questions for vocabulary assessment. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01), 888-895.
23. Gikandi, J. W., et al. (2020). Online formative assessment in higher education: A review of the literature. *Computers & Education*, 57(4), 2333-2351.
24. Nicol, D. J., & Macfarlane‐Dick, D. (2021). Formative assessment and self‐regulated learning: A model and seven principles of good feedback practice. *Studies in Higher Education*, 31(2), 199-218.
25. Haladyna, T. M., & Downing, S. M. (2020). A taxonomy of multiple-choice item-writing rules. *Applied Measurement in Education*, 2(1), 37-50.
26. Miller, M. D., Linn, R. L., & Gronlund, N. E. (2021). *Measurement and assessment in teaching*. Pearson.
27. Manning, C. D., et al. (2020). The Stanford CoreNLP natural language processing toolkit. *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*.
28. Bird, S., et al. (2020). *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc
29. Bello, M., & Usman, M. (2021). An overview of document layout analysis. *Journal of Computer Science and Technology*, 31(5), 1068-1085
30. Wilkerson, M., & Roos, B. (2022). Extracting data from PDFs using open source tools. *Journal of Open Research Software*, 2(1).

31. Schlegel, V., et al. (2020). Fast and accurate nearest neighbor search with the euclidean k-means tree. *IEEE Transactions on Knowledge and Data Engineering*, 32(1), 82-96.
32. Aumüller, M., et al. (2020). Ann-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Information Systems*, 87, 101374.