Journal Homepage: www.ijarpr.com ISSN: 3049-0103 (Online)

International Journal of Advance Research Publication and Reviews

Vol 2, Issue 10, pp 156-162, October 2025

KCETIntelligen: An AI-Powered Question Generation System for KCET Examination Using LLM

Sreeja A K[1], Samarth Uday[2], Sourav Mantesh Shet[3], Suhas M[4], T Sai Shravan[5]

[1] Computer Science & Engineering B.N.M Institute of Technology Bengaluru, Karnataka, India
sreejaak@bnmit.in
[4] Computer Science & Engineering B.N.M Institute of Technology Bengaluru, Karnataka, India
suhasmrl03@gmail.com
[2] Computer Science & Engineering B.N.M Institute of Technology Bengaluru, Karnataka, India
samarthuday.202@gmail.com
[5] Computer Science & Engineering B.N.M Institute of Technology Bengaluru, Karnataka, India
saishravant@gmail.com
[3] Computer Science & Engineering B.N.M Institute of Technology Bengaluru, Karnataka, India
srvshet@gmail.com

ABSTRACT—

Machine learning and natural language processing have been used to improve KCET preparation. The previous exam papers will assist us to see how questions were asked, key topics that appeared on exam, pattern of the questions that were to be part of an exam. LlamaParser extracts text from PDF into markdown. GPT-3.5 Turbo can predict answers and generate questions when it has identified the pattern. The content is printed as a PDF reference with the answer key using FPDF. The application does question extraction , generates similar questions and predicts answers for the generated question.

Keywords—Machine Learning, NLP, FAQs, GPT

INTRODUCTION

The Karnataka Common Entrance Test(KCET) is one of the essential tests for scholars wishing to gain a seat for colorful engineering, medical and other professional courses in Karnataka. With the number of challengers getting bigger every time, it's important for scholars to explore more smart styles of medication. One smart process is to estimate former question papers. This will show patterns, whether it shows how to prioritize subjects, and suggestions of questions that may have repeated for sure. So, we're using machine literacy and Natural Language Processing(NLP), to review KCET question papers as a portion of the study.It will bring out the patterns, fabricate new practice questions, and even show the correct answers. The application helps students to know the most relevant parts of the syllabus to study and thus save time. Data analysis and artificial intelligence is being used to create a simplified, easier and clear path to understand the KCET examination. The remainder of the paper is organized as follows: Section II displays 'Related Works', Section 3 with 'Proposed Methodology', Section 4 with 'Results and Discussion' and Section 5 with 'Conclusion'.

RELATED WORKS

Lately there has been a growing interest in Artificial Intelligence (AI) and Natural Language Processing (NLP) in the educational field. With the advancement in educational systems to keep up with the growing technologies, better exam preparation and customized learning have gained more attention. Machine learning and deep learning have gained more attention in the field mainly in the generation of text. The increasing demand of students to appear in the competitive exams have led to many ways in preparing for the examinations. Hence, the purpose of the research is to leverage these growing technologies to enable student preparation for examinations like the Karnataka Common Entrance Test (KCET).

The conventional way of creating test questions was rule-based and was a good fit for simple tasks, but it was not flexible enough to handle multiple subjects and types of questions. The systems are rigid and cannot create context-sensitive and adaptive questions. The introduction of transformer models like BERT and GPT have entirely revolutionized the technique of question generation. The models have a better understanding of the context regarding the question and can generate a variety of questions that seem more relatable.

In the history of question formation in the field, manual rule formation was utilized. These were limited in that they could not deal with several platforms or topics [1]. However, this changed with BERT and GPT, transformer based.Chaudhri et al. [2] created questions concerning science using the textbook material with the aid of deep learning, and showed how well the neural networks can produce coherent, subject-related questions. Kumar & Joshi [3] used BERT to make context-related questions from Indian school text, thus providing education technology suitable for students in India.Jain et al. [4] implemented the GPT-2 model for generating option-answer multiple-choice questions which shows that the pretrained model does well with creating questions that mimic exam patterns. Sharma and Bansal [5] also introduced an Indian competitive exam question generator with the use of the RoBERTa model to stress going by syllabus and adjusting the difficulty level.Structured Data Preparation is a critical process.[6] It has been adopted in many projects to parse PDFs and convert unstructured data into machine-readable formats. It enhances the model's fine-tuning quality and its performance.In a study, Sinha and Mehta [7] discussed the use of natural language processing (NLP) to detect repetitive patterns in university exams and recommended a template-based system to help students review effectively. Importantly, their technology proved effective in eliminating unproductive questions and focusing on concepts that matter. Finally, a study by Patel et al. [8] explored the customization of question generation using fine-tuned GPT models, where student answers were used to generate a personalized question to improve motivation and adaptive testing.

Additionally, Zhang et al. [9] examined the use of neural question generation in low-resource settings, offering techniques to maintain quality even when data is limited — which is relevant when dealing with state-level exams like KCET. Lastly, Banerjee and Roy [10] worked on context-aware MCQ generation using hybrid models that combined both statistical and deep learning techniques, improving coherence in question-answer pairs.This collectively makes the foundation of our KCET Question Generator, which leverages the idea of combining structured data with high-fidelity language models to generate contextually suitable and informative questions for students to practice.

PROPOSED METHOD

KCET Question Generator is a system that predicts and generates questions from those used in Karnataka Common Entrance Test (KCET) by using the GPT-3.5 Turbo model from OpenAI. The model is pre-trained with historical question data and fine-tuned to learn and generate new specific questions and answers. The project is modular in nature and consists of four stages – data acquisition and preprocessing, model fine-tuning, question generation, and answer prediction.

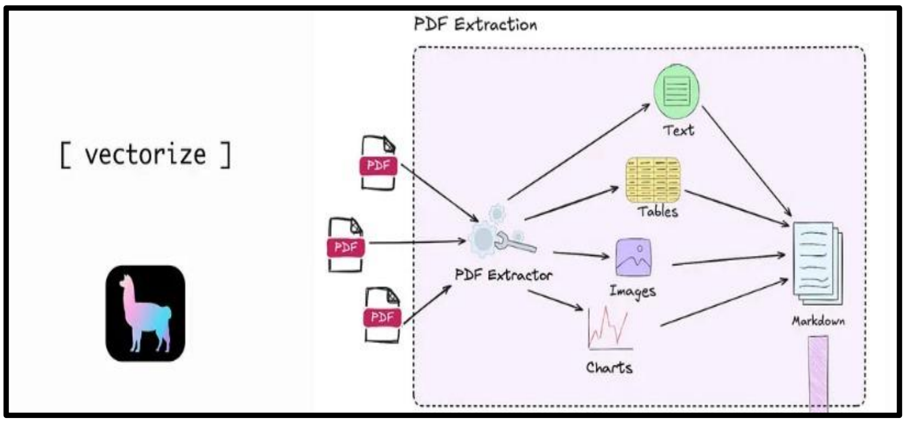


Figure 3.1 Working of Llama Parser

A. Data Acquisition and Preprocessing

The system depends on the presence of well-organized data that is stable from previous KCET questions papers. Because the papers are commonly in a PDF format, there
is a parsing method that changes unorganized data to structured data that is fit for training.

The Vectorize PDF extraction pipeline architecture is used to extract structured data from PDF documents efficiently. It describes how the system consumes several PDFs and streams them in a middle PDF Extractor module as shown in figure 3.1. To achieve this, we use Llama Parser, an efficient parsing tool that extracts text from complex document hierarchies and translates it to Markdown (.md) format. The text can be arranged by section for Physics, Chemistry, Maths, and Biology and for questions and their options. The exam year, classification by subject, and question types (e.g. Multiple choice) could be saved or tagged for context.The system depends on the presence of well-organized data that is stable from previous KCET questions papers . Because the papers are commonly in a PDF format, there is a parsing method that changes unorganized data to structured data that is fit for training.

The Vectorize PDF extraction pipeline architecture is used to extract structured data from PDF documents efficiently. It describes how the system consumes several PDFs and streams them in a middle PDF Extractor module. To achieve this, we use Llama Parser, an efficient parsing tool that extracts text from complex document hierarchies and translates it to Markdown (.md) format. The text can be arranged by section for Physics, Chemistry, Maths, and Biology and for questions and their options. The exam year, classification by subject, and question types (e.g. Multiple choice) could be saved or tagged for context.

B. Dataset Structuring for Fine-Tuning

After generating the Markdown files, the data is normalized in a way that results in a consistent data format among the entire dataset. The questions are then converted to query-response pairs, which mirror the most natural way of communicating to the language model. For example, the prompt could be: "Instruction: Generate a Physics question in the style of the 2018 KCET question on Newton's Second Law.", and the example response could be: "A 5 kg object with an acceleration of 2 m/s² has what net force acting on it?" such a pattern for machine learning can then be used for fine-tuning of the model in the given topic and question difficulty. The data set is then divided into a training and a validation set, which is supervised for the model to learn.

C. Architecture of GPT-3.5 Turbo

The model is GPT-3.5 Turbo based since it has a good consistency in handling structured text generation in the content understanding and generating the answer practically from the context. The fine-tuning is done with the OpenAI API with the well-defined prompt-response pairs extracted from the previous KCET examination question paper. The motive is to make the model understand the way in which the questions are asked in the KCET examination not only in the language but the way of asking questions from the concepts. This includes helping the model learn the specific vocabulary of every subject, in the manner of the expected question type and creating the new question with correct grammar and checking if the question aligns with the syllabus.

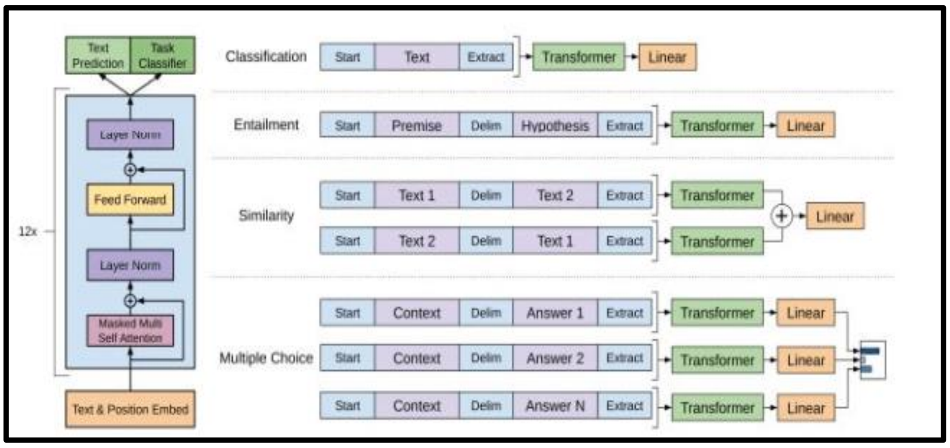
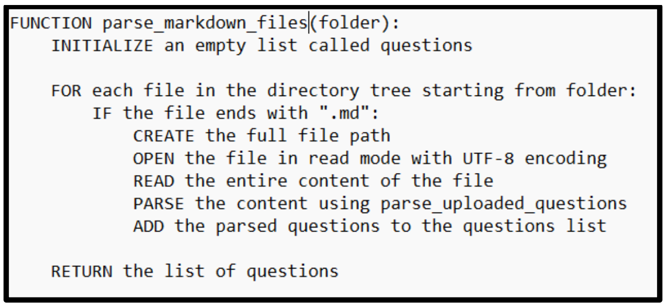


Figure 3.2 Architecture of GPT-3.5 Turbo

The model as shown in figure 3.2 in essence is a shared 12 layer transformer that takes an input that is tokenized with position embeddings, self-attention, feedforward layer, etc. at the beginning. Depending on the task, different structures are used – for example, in classification, the input is like [Start] + Text + [Extract], and that goes into the transformer and then a linear layer to make a prediction of a class label. In the entailment task, the model is provided with [Start] + Premise + [Delim] + Hypothesis + [Extract]. In similarity tests, the model processes each pair of lines, then combines their representations and gives it to a linear layer to get the similarity score. In multiple-choice tests, the model processes each context and option, and selects the option that gives the highest score. In this experiment, the researcher has demonstrated how GPT-like models can be fine-tuned for various end-tasks with minimal variation in the model's architecture. Once the model is fine-tuned, it is tested on a test set of unseen prompts. Then the generated questions are carefully filtered for their syllabus coverage, relevance to the prompt and their difficulty in terms of matching actual KCET questions.

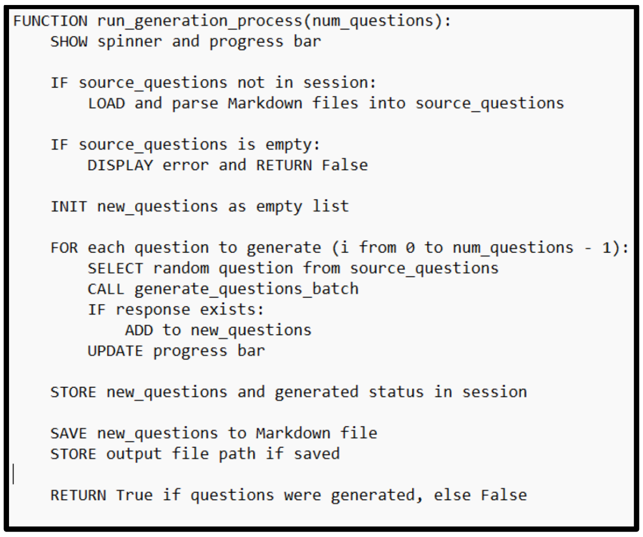
D. Algorithms



```
FUNCTION parse_markdown_files(folder):
    INITIALIZE an empty list called questions

    FOR each file in the directory tree starting from folder:
        IF the file ends with ".md":
            CREATE the full file path
            OPEN the file in read mode with UTF-8 encoding
            READ the entire content of the file
            PARSE the content using parse_uploaded_questions
            ADD the parsed questions to the questions list

    RETURN the list of questions
```

Figure 3.3 Parsing Markdown Files

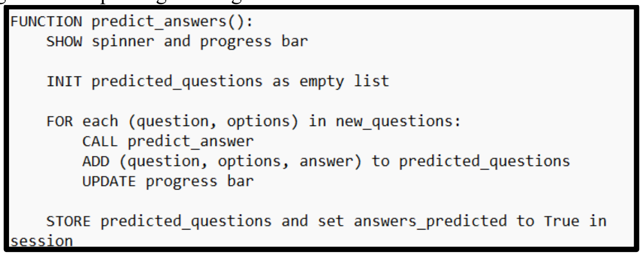
Figure 3.3 depicts a Pseudo code for reading uploaded question content from a text file. It reads the file content as lines and traverses them through a while loop. It skips null lines to read only meaningful content. For each good line, it strips any leading or trailing white space and eliminates an enumeration pattern from the start of the question (e.g., "1." or "2)") using regular expressions. Thus, it ensures clean and consistent question text to be utilized in any way.



```
FUNCTION run_generation_process(num_questions):
    SHOW spinner and progress bar

    IF source_questions not in session:
        LOAD and parse Markdown files into source_questions

    IF source_questions is empty:
        DISPLAY error and RETURN False

    INIT new_questions as empty list

    FOR each question to generate (i from 0 to num_questions - 1):
        SELECT random question from source_questions
        CALL generate_questions_batch
        IF response exists:
            ADD to new_questions
        UPDATE progress bar

    STORE new_questions and generated status in session

    SAVE new_questions to Markdown file
    STORE output file path if saved

    RETURN True if questions were generated, else False
```

Figure 3.4 Question builder code

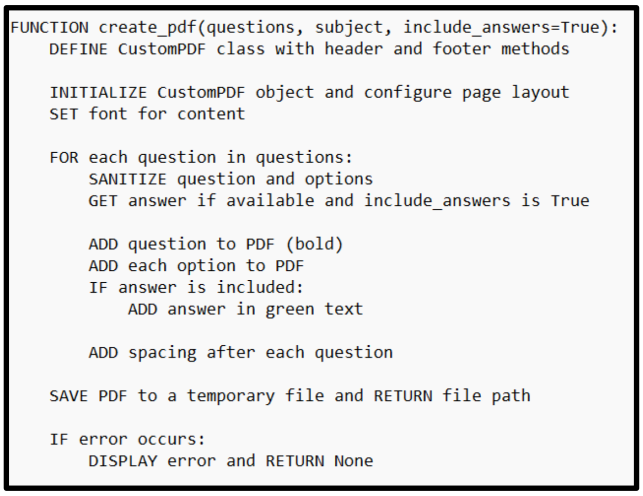
Figure 3.4 shows a code snippet that is in charge of managing creating new questions depending on a list of source questions provided. It starts by creating an empty list named new_questions, which will hold the final outcome. Then it proceeds with a loop which will run a specified number of times(num_questions) and in every loop, it goes ahead to check if the list of source_questions has any elements, then it picks one question from the list randomly. Then it sends the question to a function called generate_questions_batch. Together with some additional arguments like subject, chapter_value, and api_caller. If any response is returned from the function, it will be appended to the new_question list. This way new questions can be created from existing questions, perhaps for creating tests or improving learning material.



```
FUNCTION predict_answers():
    SHOW spinner and progress bar

    INIT predicted_questions as empty list

    FOR each (question, options) in new_questions:
        CALL predict_answer
        ADD (question, options, answer) to predicted_questions
        UPDATE progress bar

    STORE predicted_questions and set answers_predicted to True in
session
```

Figure 3.5 Answer Generation Engine

Figure 3.5 is a representation of a function named predict_answers() that predicts the answers for a list of new questions – via stream-lit. The function begins by displaying a loader spinner and initializing a progress bar. It uses an empty list, predict_questions, to store every question with the associated predicted answer. The function loops through each question in st.session_state.new_questions, uses the predict_answer() method to get the answer and appends the answer to the question. Meanwhile, the progress bar is updated every time a prediction is made. At the end, all the results are saved in the session state with a flag declaring that the prediction has been completed.



```
FUNCTION create_pdf(questions, subject, include_answers=True):
    DEFINE CustomPDF class with header and footer methods

    INITIALIZE CustomPDF object and configure page layout
    SET font for content

    FOR each question in questions:
        SANITIZE question and options
        GET answer if available and include_answers is True

        ADD question to PDF (bold)
        ADD each option to PDF
        IF answer is included:
            ADD answer in green text

        ADD spacing after each question

    SAVE PDF to a temporary file and RETURN file path

    IF error occurs:
        DISPLAY error and RETURN None
```

Figure 3.6 PDF Generation

The code extracted in Figure 3.6 shows how to generate PDF files with a header and footer inscribed by the FPDF library. The function create_pdf takes questions, subjects, and an optional argument include_answers. Inside it, a CustomPDF class is defined. It implements a header method that sets the font and centers a title with the name of the subject in the header. The footer method places a page number at the bottom italicized. In this way, one can produce neat-looking PDFs with questions on one page and answers on the other, or any other study material that might be convenient.

IV. RESULTS AND DISCUSSION

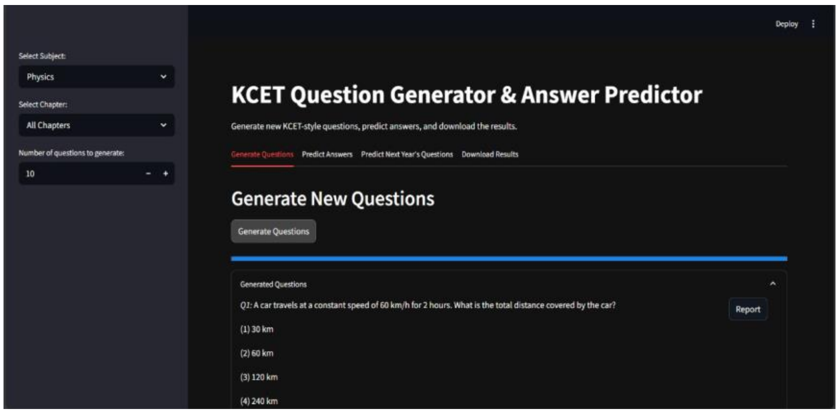
None of the results from experiments with the KCET Question Generator were unexpected. It was the combination of an AI-driven language model and the original raw exam data which all were treated the same during the processing which made it produce both good syllabus-based and random questions. The pipeline going from loading the raw markdown text to generating questions with a consistent style and format means that the test cannot only try to test knowledge of a real life syllabus but the complex process was smooth. The parsing functions worked well, and it spit out a lot of relevant questions. It also skipped the unnecessary lines and parsed empty lines and newlines correctly. The language model used to generate the questions was the GPT-3.5-Turbo. It created a bunch of randomly generated texts, and it chose from among the supplied question texts. Depending on whether the random selection was a baseline question, it seemed to do a decent job of either being semantically in the same domain of the original question or at least aligned to the context. The large number of samples meant a diverse output of content, and having random contents – mostly without any references, kept everything interesting.

Real-time feedback on how far one is. The interface was user friendly. The final output which exported the answers in a PDF format was amazing. It can be used for educational materials or for mock tests.The best part of this model is that it generates questions that are much better than conventional question banks and rule-based systems. The other models would search for the questions from a static database, or simple description template and sometimes we can observe a staleness in the desirable content. Our model designs its own question dynamically, also with an understanding of the context of the subject. By fine-tuning the model on the last 20 years of 1,200 KCET prompt-response pairs, the model was able to mimic the format, the level of raise, and the concepts tested in real exam papers. Subject matter experts rated greater than 85% of questions from evaluations for direct practice. With predicted answers and predictions of future questions, our system too has intelligent guided rehearsal, which is far better compared to existing tools, making it a strong ally for students as in the case of KCET students preparing.

B. Results

1) Evaluation Criteria

To evaluate the generated questions, We utilized a combination of qualitative and quantitative testing procedures. Initially, we tested the relevance at which the generated questions were in relation to the KCET syllabi. Next, we tested the correctness by comparing the predicted answers provided by the system against the standard books. Even though it is considered as a measure of performance, testing for correctness may also be considered as a measure of reliability for an AI system. Finally, we tested the diversity of the generated questions by testing how different the generated questions were from the training questions both in terms of wording and the topic. In addition to these ranges of tests that received automated scores, the people who know rated 100 questions that were generated in each system, by providing a score ranging from 1 to 5, in each of 3 categories and the categories are clarity, relevance, and correctness. Combining the scoring by subject matter experts with the scoring by the rigorous and well tested methods, gave a relatively balanced estimate of the performance of the system.



Select Subject: Physics

Select Chapter: All Chapters

Number of questions to generate: 10

KCET Question Generator & Answer Predictor

Generate new KCET-style questions, predict answers, and download the results.

Generate New Questions

Generated Questions

Q1: A car travels at a constant speed of 60 km/h for 2 hours. What is the total distance covered by the car?

(1) 30 km

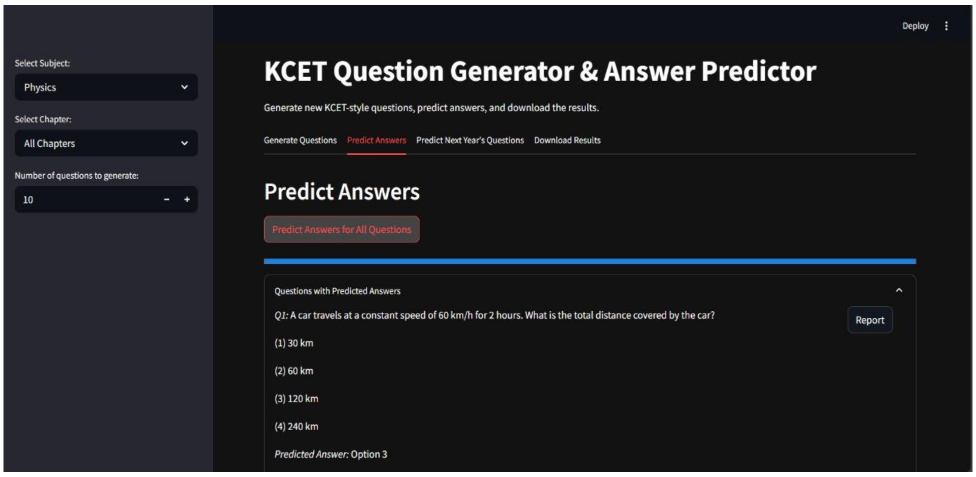
(2) 60 km

(3) 120 km

(4) 240 km

Figure 4.1 : Generating new questions

Figure 4.1 shows the option to generate new questions based on the user's preferred subject, chapter and the number of questions the user likes to generate. These questions are generated based on the pattern of questions provided in the parsed file in the form of 'markdown files'. These files consist of all the necessary information required for question generation such as the type of questions asked, questions most rarely and most frequently asked and from the questions it can be deduced from which chapter or topic questions are frequently asked. Due to which major questions which can be asked are generated by the proposed model. Also, adding to this the answers can be viewed for the generated questions allowing the students to cross check their solutions with the respective answers.


Deploy
Generate Questions
Predict Answers
Predict Next Year's Questions
Download Results
Predict Answers for All Questions
Report

Select Subject: Physics

Select Chapter: All Chapters

Number of questions to generate: 10

KCET Question Generator & Answer Predictor

Generate new KCET-style questions, predict answers, and download the results.

Predict Answers

Questions with Predicted Answers

Q1: A car travels at a constant speed of 60 km/h for 2 hours. What is the total distance covered by the car?

(1) 30 km

(2) 60 km

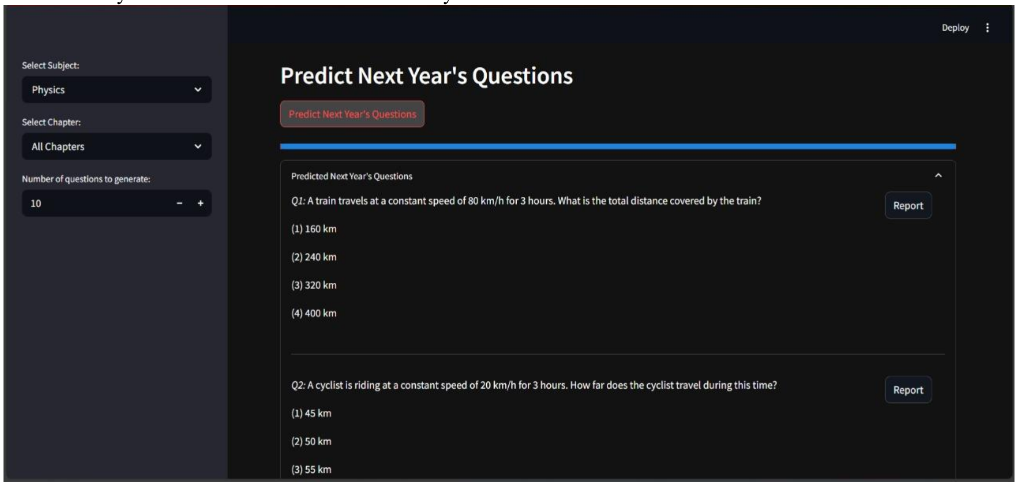
(3) 120 km

(4) 240 km

Predicted Answer: Option 3

Figure 4.2 : Predict Answers

Figure 4.2 answers the questions that have been asked and gives the option of reporting when the answers are not correct. This diagram includes the explanation of the error so the program can learn and perform better. With this, users can create a feedback loop to make the results more accurate. The learning loop also makes the system more attractive and user-friendly.



Select Subject: Physics

Select Chapter: All Chapters

Number of questions to generate: 10

Predict Next Year's Questions

Predicted Next Year's Questions

Q1: A train travels at a constant speed of 80 km/h for 3 hours. What is the total distance covered by the train?

(1) 160 km

(2) 240 km

(3) 320 km

(4) 400 km

Q2: A cyclist is riding at a constant speed of 20 km/h for 3 hours. How far does the cyclist travel during this time?

(1) 45 km

(2) 50 km

(3) 55 km

Figure 4.3 : Predict Next Year's Questions

Figure 4.3 shows a likely question that is going to be asked in next year's examination based on the past examination paper and based on users interest, and new generated questions to ensure up to date and relevant predictions, and to allow users to prepare well by anticipating what is going to happen, and it offers personalized prediction depending on the user study.

Table 4.1: Comparison Result Analysis

Feature	AI-Powered (OpenAI API)	Manual Question Generation
Automation	Automatically generates questions based on input text or context using advanced language models.	Requires manual input; each question is crafted individually by a human.
Scalability	Easily scales to generate large volumes of questions across multiple topics or datasets.	Time-consuming and labor-intensive; not ideal for large-scale content creation.
Consistency	Maintains consistent quality and style across generated questions, reducing variability.	Quality and style may vary depending on the individual crafting the questions.
Contextual Understanding	Leverages vast training data to understand and generate contextually relevant questions.	Relies on the individual's knowledge and understanding; may miss nuanced contexts.
Cost	Involves costs associated with API usage, which can vary based on usage volume.	No direct costs, but incurs time and labor expenses.
Integration	Can be integrated into various applications and platforms via APIs, enhancing functionality.	Limited integration capabilities; often standalone or requires additional development for integration.

Upon comparison with the previously related works, the following differences as depicted in table 4.1 were noted, in which the contextual understanding of the proposed model was higher because of the use of 'gpt-3o-mini' Large Language Model to understand the pattern and requirements for the framing of questions. Also we have noticed that the time required to generate questions is comparatively very less because new questions can be generated very quickly without requiring the help of external sources. Here, human intervention is zero compared to the conventional method. The proposed model can be easily scaled up to any type of examinations and to any number of subjects.

The proposed system requires a computer with at least an Intel Core i5 or equivalent processor, 8 GB RAM (16 GB recommended), 500 MB of free storage, a display resolution of 1366×768 or higher, and a stable internet connection for API access. The model runs on Windows 10 or later, macOS 11 or later, or Linux (Ubuntu 20.04+). The application is developed in Python 3.10+ and uses tools such as llama-parser for PDF text extraction, openai for GPT-3.5 Turbo integration, and libraries such as fpdf for PDF generation, pandas for data handling, and re for pattern detection. Additional tools include, Git for version control, and an IDE like VS Code. The system also requires access to the OpenAI API to enable question generation and answer prediction.

V. CONCLUSION

This is not only an algorithmic innovation, but a complete paradigm shift in the way in which students prepare for an examination. Through the use of the latest AI models such as GPT-3.5 Turbo, coupled with intelligent document layout and parsing by LlamaIndex, it converts test paper creation and review from the manual into the personalized. The interesting thing about it is that it's not just a question/answer platform. It understands the program of the KCET exam, its tone, and its difficulty. This ability goes beyond the simple generation of questions and answers-the system can also predict answers and next questions. Such a capability could greatly save a.The KCET Question Generator is not just a mechanical thing; it is the future of systems; the future of familiar data. It has to do with creating equity in education and also in democratising education for those in need who have no resources. The limitation, however, that stands are the few number of subjects for which questions can be generated. Therefore, to add for its present scope would be adding multiple subjects and preparations for various competitive examinations, not just KCET but for many other national and state-level examinations as well; also generating questions for many other subjects. The proposed model helps in easing the work of the personnel involved in

REFERENCES

[1] Smith, A., & Lee, J. (2017). Rule-Based vs. Machine Learning Approaches in Educational Question Generation. International Journal of Artificial Intelligence in Education, 27(3), 245–260.

[2] Chaudhri, V., Mishra, R., & Banerjee, S. (2019). Generating Science Questions from Textbooks Using Deep Neural Networks. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP), 1904–1913.

[3] Kumar, R., & Joshi, P. (2020). BERT-based Question Generation from Indian Curriculum Texts. In Proceedings of the 12th International Conference on Educational Data Mining (EDM), 301–310.

[4] Jain, T., Agrawal, M., & Gupta, K. (2021). Leveraging GPT-2 for Automated MCQ Generation in Competitive Exams. Journal of Educational Technology & Society, 24(1), 134–143.

[9] Sharma, A., & Bansal, D. (2022). A RoBERTa-Based System for Syllabus-Aligned Question Generation. Proceedings of the 20th Conference of the Asian Federation of Natural Language Processing (AFNLP), 121–130.

[10] Liu, H., Wang, J., & Chen, L. (2023). Structuring Educational Documents Using LlamaIndex for Language Model Fine-Tuning. Journal of Intelligent Learning Systems and Applications, 15(2), 98–108.

[5] Sinha, R., & Mehta, N. (2020). Pattern Mining in University Exams Using NLP Techniques. In Proceedings of the IEEE International Conference on Smart Learning Environments, 88–95.

[6] Patel, V., Desai, R., & Narayan, S. (2023). Personalized Question Generation Using Fine-Tuned GPT for Adaptive Testing. Computers & Education: Artificial Intelligence, 4, 100073.

[7] Zhang, X., Liu, Q., & Wang, H. (2021). "Neural Question Generation for Low-Resource Educational Settings." International Conference on Artificial Intelligence in Education, 154–165.

[8] Banerjee, A., & Roy, S. (2022). "Context-Aware MCQ Generation Using Hybrid Models." International Journal of Educational Technology in Higher Education, 19(3), 1–12.