AUTOMATED QUESTION BUILDER APPLICATION USING GEN-AI

Arthi R[1]
Department of Artificial Intelligence and Data Science
Panimalar Engineering College Chennai, India.
arthirameshkumar22@gmail.com

Nila D[2]
Department of Artificial Intelligence and Data Science
Panimalar Engineering College Chennai, India.
niladhanagopal2003@gmail.com

Nithya Shree K[3]
Department of Artificial Intelligence and Data Science
Panimalar Engineering College Chennai, India.
nithya.shree12304@gmail.com

Dr. M.S.Maharajan [4]
Associate Professor
Department of Artificial Intelligence and Data Science
Panimalar Engineering College Chennai, India.
maha84rajan@gmail.com

Abstract* — The generation of educational content can now be automated because to developments in generative AI, which has important implications for evaluation and personalized learning. In order to generate a variety of questions with different levels of complexity, including multiple-choice, coding, and paragraphbased questions, this work provides an automated question generator application. In order to ensure context and relevance, the application dynamically generates questions based on input themes using Groq's Lemma API, large language models (LLMs), and natural language processing (NLP) approaches. The application successfully generates a range of well-organized questions that are in line with input specifications, according to the results. This study shows how generative AI may improve educational technologies by automating content creation, which saves time and money on creating high-quality questions.

Keywords—Groq's Lemma API, Large language model, Natural language processing.

I. INTRODUCTION:

Automatic Question Generation (AQG) has emerged as a crucial area in Natural Language Processing (NLP) as a result of the need to enhance conversational agents and instructional materials. New developments in AQG systems demonstrate a number of techniques, including as rule-based strategies and neural network structures like Transformers, which have transformed the creation of logical and contextually appropriate queries. There is a noticeable emphasis on a variety of applications in the literature, from visual and conversational contexts to independent question production. In order to overcome the difficulties of preserving quality and relevance, these developments show how models such as GPT, BERT, and XLNet may create questions that nearly resemble content created by humans.

Using Groq's Lemma API, we create an Automated Question Generator Application in this project that effectively creates questions from text inputs. Our application uses the insights from related efforts to improve the quality of question generation by incorporating cutting-edge generative AI techniques. This emphasis on using huge language models is consistent with recent research showing that synthetic data is a powerful tool for training question-answering systems.

Additionally, by filling in gaps in the creation of excellent, pertinent questions that can greatly improve user engagement and learning outcomes, this initiative aims to add to the current conversation in AQG.

A key component of enhancing user engagement and interactive learning in a range of applications, including chatbots and educational systems, is Automatic Question Generation (AQG. The caliber and pertinence of generated questions have been greatly enhanced by the latest developments in AQG techniques, especially those that employ neural network designs like as Transformers. Although template-based and supervised question generation are examples of classic approaches that have established the foundation, their dependence on particular datasets and templates typically limits their flexibility and diversity.

Recent research on the development of open-ended questions that call for multiple-sentence answers has suggested a new question type ontology that more accurately captures the complex nature of questions than conventional question words. This study introduces a labeled dataset with 4,959 questions to demonstrate the significance of developing datasets that more accurately capture the intricacies of natural language

II. RELATED WORKS:

Advanced technologies like as BERT and BART, which are based on transformer topologies, are used in question generating when query suggestion is being used. In order to anticipate the subsequent inquiry based on the order of prior queries, these models are trained on enormous datasets of user sessions. These models can provide inquiries that are pertinent to the context by efficiently capturing the connections and dependencies between queries through the use of attention techniques. By optimizing the probability of noticing the subsequent question based on the preceding ones, the training process allows the models to understand user intent and behavior during intricate search tasks.

Transformer-based models can therefore generate a variety of well-reasoned query recommendations that meet user requirements, surpassing conventional RNN-based methods in managing long-range dependencies and data noise [3].

Transformer-based models like GPT, BERT, and XLNet can be utilised to effectively create questions in Natural Language Generation (NLG) Based on the input text, these models use attention mechanisms to comprehend context and produce logical queries. With its enormous parameter size, GPT-3, for example, may produce questions regardless of the task, exhibiting good performance even in the absence of task-specific fine-tuning. BERT, on the other hand, uses bidirectional context to improve comprehension, which can be used to make pertinent queries from a text. Overall, by successfully evaluating and comprehending the underlying content, these models' improvements make it easier to create high-quality queries [1].

Generative AI technologies are used to generate questions by using textual descriptions of problems as a basis. According to recent empirical studies, generative AI can reliably create conceptual models from these descriptions, including ER, BPMN, and UML. This feature implies that by analyzing and converting textual inputs into structured queries or prompts, generative AI can be used to generate pertinent questions, improving the efficacy of teaching and evaluation resources [10].

Rule-based approaches apply transformation and pattern matching techniques to declarative statements, simplifying the text first, and then using particular patterns to generate questions. Conversely, neural network-based methods use encoder-decoder architectures specifically, RNNs and transformer models to produce queries from input text. To discover the connections between queries and their accompanying answers, these models are trained on huge datasets like WikiAnswers. In order to develop questions that are both syntactically and semantically legitimate, it is imperative to use Natural Language Processing (NLP) techniques to parse the input text and extract relevant semantic information [8].

This approach generates the questions and anticipates their main points simultaneously. With two model variants— one that utilises pre-identified exemplars and the other that uses generated templates to assist in question writing—the research also used templates to enhance controllability and diversity in the generated questionsg.. A new dataset labeled according to a defined question type ontology supports the method, which distinguishes the subtle nature of questions more well than commonly used question phrases [4].

Employing a methodology that presents QG as a process of summarization and inquiry. This entails using publicly accessible summary data and annotating the summaries with semantic roles, dependency parsing, and named entity recognition. The parsed summaries are utilized to produce
to produce questions using a set of heuristics. The generated questions are used in conjunction with the source news articles to train an end-to-end neural QG model [5].

One prominent technique is the sequence-to-sequence system created by Soru et al., which generates SPARQL templates using bi-directional LSTM. Nevertheless, this approach has drawbacks, including the incapacity to handle terms that are not part of the lexicon and a lack of comprehension of the inquiry, which results in improper graph patterns in the queries that are generated. Using large-scale language models to encode linguistic information from natural language queries, SGPT is an additional method that blends end-to-end and modular systems. To produce precise SPARQL searches, our system learns intricate inquiry patterns and adds graph- specific information to the language model's parameters. Furthermore, SGPT adapts common assessment criteria and presents training methods to gauge performance in SPARQL query creation.

Networks of Long Short-Term Memory (LSTM) are used to produce natural language. The model is trained on a dataset utilising input words and context vectors, which encapsulate the semantic meaning of the sentences, in order to predict the next word in a sequence. To improve the model's comprehension of the context and produce more intelligible text, context vectors are used in conjunction with input-output instances during the training phase[7].

A three-step modeling pipeline comprising question generation, question filtration, and unconditional answer extraction from text is the main method used to generate questions in this context. A next-token-prediction language modeling aim is used in conjunction with pretrained models, namely a GPT-2 decoder model. Concatenating context, answer, and question tokens during training enhances the caliber of questions produced by the model. Large generative transformer models, which can have up to 8.3 billion parameters, are also used to improve the caliber of the questions that are produced [2].

Fine-tuning techniques based on transformers that employ a single pre-trained language model without the need for additional features, answer information, or other procedures [1 One of the most widely used Deep Learning-based techniques for QG is Sequence-to-Sequence (Seq2Seq) models, which use LSTM-based neural networks to encode the source context paragraph and decode it to generate a

generated question [12].

Transformers-based fine-tuning approaches leverage a single pre-trained language model without incorporating extra features, answer annotations, or supplementary steps. Among the most prevalent deep learning methods for question generation (QG) are Sequence-to-Sequence (Seq2Seq) models. These models utilize LSTM-based neural networks to process the given context, encoding the source paragraph and decoding it into a generated question.

By utilizing ChatGPT's features, educators can generate open-ended question prompts that correspond with the learning objectives and success standards of the lesson. Furthermore, ChatGPT may be used to create high-quality rubrics that succinctly and clearly outline the precise tasks that students must complete in order to pass the various competence levels. Accordingly, by employing unique and unusual artifacts, generative AI-powered evaluation systems may facilitate the incorporation of ongoing input into learning procedures [15].

III .EXISTING SYSTEM:

Technology	**Function in AQG**	**Strength**	**Limitation**
LSTM	1. Retains long-term dependencies in sequential data. 2. Generates coherent, contextually relevant questions.	Handles long sequences and maintains context well.	Limited vocabulary handling; struggles with out-of-vocabulary terms.
Seq2Seq Model	1. Encodes input context and decodes relevant questions. 2. Suitable for translation, summarization, and AQG tasks.	Captures context effectively; useful in conversational AI.	Struggles with complex dependencies and out-of-vocabulary words.
BERT	1. Generates contextually coherent queries by understanding word relationships. 2. Captures user intent effectively.	Captures nuanced word meanings, handles long-range relationships well.	High computational requirements; less effective for strict sequential dependencies.
Transformer Model	1. Uses attention to produce contextually relevant questions. 2. Efficient at generating coherent text responses.	Efficient processing with attention, captures complex dependencies.	High resource demands; may struggle with strictly sequential data.
BART	1. Generates accurate, contextually varied questions from input. 2. Combines bidirectional and autoregressive models.	Adaptable for diverse tasks, produces coherent and accurate questions.	Requires tuning for complex formats; high computational needs.
RNN	1. Maintains hidden state for sequential question generation. 2. Adapts to user context over a session.	Effective for short dependencies and session-based tasks.	Struggles with long sequences (vanishing gradient); needs LSTM or GRU variants for better long-range dependency handling.

LSTM:
An LSTM (Long Short-Term Memory) network is a form of recurrent neural network (RNN) designed especially to process sequential data by retaining long-term dependencies.. They are made up of input, forget, and output gates that regulate information flow, enabling the network to remove extraneous information while retaining relevant features across long sequencesAs a result, LSTMs are suitable for a variety of natural language processing applications, including text production and questionanswering systems, since they can preserve the history of lengthy sentences. In educational or conversational AI applications, in particular, LSTMs may formulate queries that are both coherent and contextually accurate by processing text word by word while preserving context across phrases.

LSTM functions by preserving long-term information through its distinct design, which consists of several gates: cell states, input, forget, and output gates. These gates establish what historical data should be kept and what should be thrown away. Because of this, LSTMs can retain the history of lengthy sentences, which makes them appropriate for a range of natural language processing uses the history of lengthy sentences, which makes them appropriate for a range of natural language processing uses, such as text production and question-answering systems. In order to generate more coherent text, the model uses context vectors to better comprehend the relationships between the input words and is trained to predict the next word in a sequence depending on the input words [7].

In order to generate SPARQL templates, Soru et al. created a sequence-to-sequence system that incorporates LSTM. Although it has drawbacks, like the inability to handle terms that are not in the lexicon and a lack of comprehension of the query, this system uses bi-directional LSTM to manage the generation process, which may result in improper graph patterns in the queries that are generated [6].

***SQUENCE TO SEQUENCE*:**

A neural network design known as a sequence-to-sequence (Seq2Seq) model converts an input sequence into an output sequence; it is frequently employed for tasks including question creation, summarization, and translation. It consists of two primary parts: an encoder that compresses the input text (such a phrase or paragraph) into a context vector and then uses that context vector to create a logical output sequence, word by word. Seq2Seq models are trained to generate pertinent questions from an input passage in automatic question generation (AQG). The decoder formulates queries that are appropriate for the context based on the encoder's knowledge of the input text. This strategy is especially helpful with conversational AI and educational systems, where creating accurate and meaningful questions can improve learning and engagement.

Sequence-to-Sequence (Seq2Seq) models employ LSTM in question creation. Where an LSTM-based neural network encodes the source context paragraph and another LSTM-based network decodes the embedded data to provide a generated question [12].

Sequence-to-sequence (seq-to-seq) models use an encoderdecoder architecture to transform an input sequence into an output sequence. Once the input sequence has been processed, the encoder creates a fixed-size context vector that contains the input's information. After receiving this context vector, the decoder creates the output sequence, usually one element at a time. Attention mechanisms are frequently added to this method to enable the model to concentrate on various input sequence segments while producing each output element, enhancing the output's quality and relevance [11].

BERT :

In query recommendation, BERT (Bidirectional Encoder Representations from Transformers) provides contextual embeddings for user queries. It helps the model better comprehend user intent by capturing the connections and dependencies between words in a query. BERT is optimized for particular tasks, including anticipating the subsequent inquiry in a series based on earlier inquiries in a user session, and has been pre-trained on huge datasets. Because of this feature, BERT can produce contextually appropriate and coherent query suggestions, surpassing more conventional models like RNNs, particularly when it comes to managing long-range relationships and data noise [3].

BERT (Bidirectional Encoder Representations from Transformers) is used to generate enquiries by leveraging its bidirectional context awareness . This enables BERT to perform a thorough analysis of the input text and produce pertinent queries depending on the information. Without requiring a great deal of task-specific fine-tuning, its architecture allows it to capture the subtleties of language, making it useful for tasks like question generation [1].

***TRANSFORMER MODEL*:**

GPT is based on the transformer architecture, a type of neural network that performs well on tasks involving natural language processing.After being exposed to a large text dataset, which includes books and articles, it learns to generate material that is comparable to the text it was trained on. In response to a prompt or context, the model evaluates the data and generates a response one word at a time, predicting the next word based on the input and the words it has previously created.The model uses attention processes to focus on the most relevant parts of the input in order to generate responses that make sense and are appropriate for the situation [15].

To fine-tune the model for a specific task, T5 is applied by using a task prefix.The task prefix "generate question:" is used at the beginning of the input text to create questions.Parameter optimisation is used during the finetuning phase to determine the model's optimal configuration [14].

***BART*:**

BART functions as a denoising sequence-to-sequence model that has been trained for tasks involving comprehension, translation, and natural language creation. It efficiently generates text based on the input it gets by combining bidirectional and autoregressive transformers. The model is optimized for certain tasks, such question creation, where it may generate questions based on pre-provided answers or circumstances [2].

There are several ways to use BART to generate questions, including using alternative model sizes (BASE and LARGE) and specific parameter settings for optimisation. [14].

An encoder and a decoder make up the entire architecture of the bidirectional and auto-regressive transformer utilised in query suggestion, which makes it particularly helpful for tasks involving text synthesis. It is trained on a variety of tasks, including phrase permutation, text infilling, and token masking, which improves its capacity to produce logical and contextually appropriate inquiries based on prior user interactions. BART uses a gradual unfreezing strategy during training, which enables the model to efficiently adjust its parameters. This method enables BART to produce a variety of query suggestions that are pertinent to the user's search context, outperforming other models like BERT [3].

***RECURRENT NEURAL NETWORK*:**

The ability of recurrent neural networks (RNNs) to evaluate sequential input in Natural Language Generation (NLG) makes them valuable for language modelling and text generated By keeping a concealed state to preserve data from prior time steps, they are able to produce text one word at a time while taking earlier words' context into account. However, as input length increases, RNNs face difficulties such as the vanishing gradient problem that make it more difficult for them to understand long-range relationships in sequences. Because of this constraint, sophisticated systems like Transformers have been created to solve these problems [1].

By analysing user query sequences and keeping a hidden state that records information from prior inputs, recurrent neural networks (RNNs) are used in query suggestion. By changing their representation with every query, RNNs are able to comprehend context and intent and model user behaviour over time. RNNs are appropriate for user sessions with varying query counts since they can handle variable-length sequences. Their ability to completely capture user session context may be hampered by their inability to handle long-range dependencies because to vanishing gradient problems. Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) are two variations that have been developed to better regulate information flow across longer sequences [3].

Recurrent neural networks (RNNs) are used in an encoderdecoder architecture for Automatic Question Generation (AQG) The encoder uses several recurrent neural network layers to handle an input sequence that contains a passage and its matching response. The encoded form of the response is then used by the decoder to create a question. A bi-directional RNN is used to enhance the quality of the produced questions by improving context collection from both directions of the input sequence. When compared to baseline techniques, the RNN model has shown higher BLEU ratings, demonstrating its efficacy in generating pertinent queries [8].

IV. ARCHITECTURE DIAGRAM:

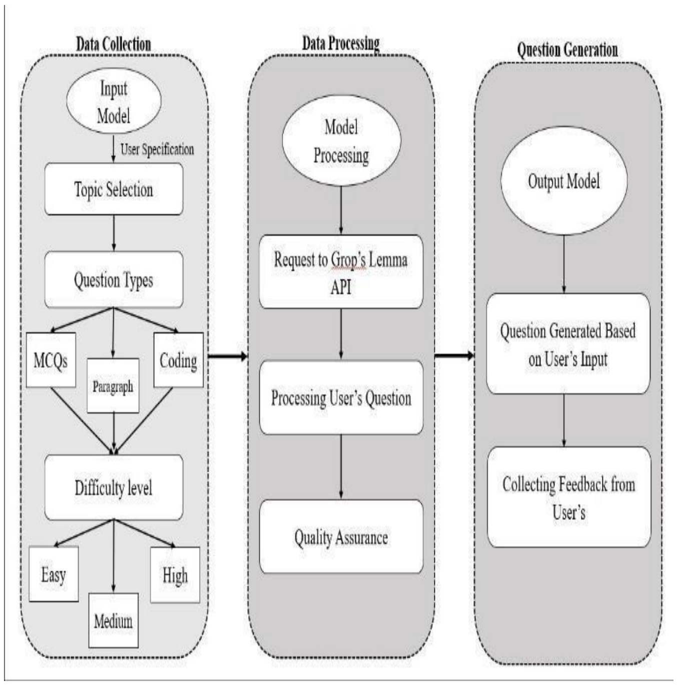

FIG 1: ARCHITECTURE DIAGRAM

With three primary stages Data Collection, Data Processing, and Question Generation—the graphic shows the workflow of a Automatic Question Generator (AQG). Users can set a number of characteristics, such as the topic, question type, and degree of difficulty, during the Data Collection phase, when the system first gets input from the user. Multiple- choice questions (MCQs), code questions, and paragraph- based questions are among the question categories the user can select from. They can also choose an easy, medium, or high difficulty level to match the amount of complexity they want.

In order to generate questions depending on the chosen topic and format, the AQG system first prepares the input data for question creation by sending a request to Groq's Lemma API, which manages natural language processing. Questions are processed using the API.

V. PROPOSED SYSTEM:

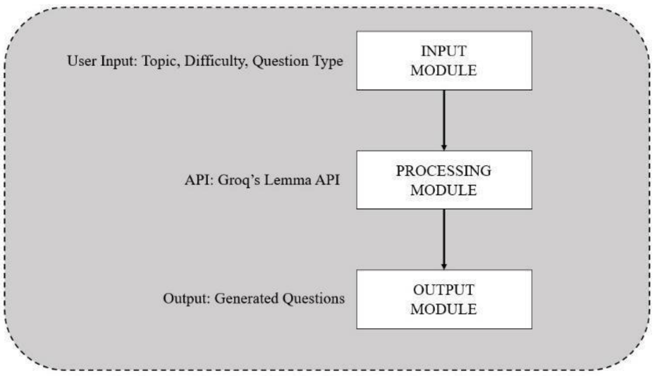
User Input: Topic, Difficulty, Question Type
INPUT MODULE
API: Groq's Lemma API
PROCESSING MODULE
Output: Generated Questions
OUTPUT MODULE

FIG 2: WORKFLOW DIAGRAM

The automated question generator application's suggested system architecture uses a modular approach to expedite the question creation process. Three main parts make up the architecture: the input module, processing module, and output module. Because of the clear data and interaction flow guaranteed by this design, users may quickly create queries based on their needs.
Diagram concept:

- Using a flowchart or diagram, illustrate how the three primary parts are related in order.
- The data flow from the input module to the processing module and then to the output module can be shown using arrows.
- Provide icons or representations for every module, such as a webpage for outputs, a cloud for the API, and text boxes for inputs.

COMPONENT:

1. The input module

The Input Module functions as the user interface via which users, instructors, or trainers can define the specifications needed to generate the questions. This module's user-friendly and intuitive design allows it to record important characteristics.

User field input:

- Selecting a Topic: Users can enter a certain subject or topic to have questions generated for it. Auto-suggestion tools can be added to this to assist users identify pertinent topics more quickly.
- Levels of Difficulty: Users can choose from a list of predetermined possibilities for the questions' difficulty (e.g., Easy, Medium, Hard). You can use a dropdown menu or radio buttons to accomplish this.
- Question Types: Multiple Choice Questions (MCQs), coding questions, and descriptive questions are among the various question styles from which users can select. This can be done using a multi- select dropdown or checkboxes.

Additional features:

- Preview Section: Before creating questions, users can verify their selections when a preview section displays an overview of the inputs they have chosen.
- After all inputs have been filled out, the Submit button will start the question creation process.

2. Processing module

- The system's central component, the Processing Module, is in charge of creating queries using the data that the Input Module provides. This module generates high-quality questions by utilizing a variety of NLP approaches and Groq's Lemma API.

- Data Flow: The module sends a request to Groq's Lemma API with the chosen topic, degree of difficulty, and question type as parameters after receiving the inputs.

Procedure for Creating Questions:

- API Interaction: Using the specified criteria, the Processing Module creates an API call to Groq's Lemma, which yields a JSON response with questions created. The use of AI and NLP models by the API guarantees that the queries are suitably difficult and contextually relevant.
- Post-processing: After the questions are received, they could go through extra steps to ensure proper formatting or to eliminate any irrelevant or duplicate results. This action could entail:
- Natural Language Processing (NLP): To improve the caliber of questions produced, methods like relevance score, keyword extraction, and text normalization can be used.
- Using validation methods to verify the coherence and clarity of the generated questions is known as quality assurance.

3. The Output Module:

The created questions must be shown to users or sent by the output module. This module makes sure that the created material is easy for users to read, engage with, and use.

Display mechanisms:

- Question List: The generated questions are presented in an organized manner, perhaps as a table or list view, with options for each question (if applicable for multiple-choice questions).
- Interactive Components: Add buttons that allow users to:
- Copy: To make sharing or using questions easier, copy them to the clipboard.
- Print: For offline use, print the questions directly.
- Download: For convenience, offer the ability to download the questions in a number of different formats (such as Word and PDF).

User feedback:

- A feedback form embedded into the Output Module allows users to comment on the questions' quality and relevancy once they have viewed the generated questions. The question creation algorithms can be gradually improved with the help of this input.

VI. FUTURE WORK:

When developing your automatic question generator in the future, think about adding adaptive learning paths and user progress tracking to improve user customisation. Include project-based assignments, brief responses, and multimedia materials such as code excerpts and video explanations in your question types. Including gamification elements like challenges, leaderboards, and badges can increase user involvement. Include an automated response explanation system, a feedback system to improve the quality of the questions, and an editor for changing the questions. For wider use, integrate the product with learning management systems and develop APIs. Additional enhancements to the site will include multidisciplinary question support, collaborative learning opportunities, and real-time coding exams.

VII. CONCLUSION:

A strong foundation for producing varied and dynamic coding questions suited to different skill levels is provided by the creation of an automatic question generator with Flask and the OpenAI API. In addition to increasing the effectiveness of content production for training and education, this initiative opens the door for major future growth. Real-time evaluations, multimedia-rich questions, adaptive learning features, and integration with well-known learning systems are examples of possible developments. While security features and AI-driven analytics will ensure dependability and ongoing improvement, gamification and collaborative tools will further increase user engagement. All things considered, this project lays the foundation for a more scalable, customized, and interactive approach to contemporary learning and evaluation tools.

VII. REFERENCES

[1] M. Onat Topal, Anil Bas, Imke van Heerden "Exploring Transformers in Natural Language Generation: GPT, BERT, and XLNet" 2021.

[2] Raul Puri, Ryan Spring, Mostofa Patwary ,Mohammad Shoeybi, Bryan Catanzaro "Training Question Answering Models From Synthetic Data" 22 Feb 2020

[3] Agnès Mustar, Sylvain Lamprier, Benjamin Piwowarski "Using BERT and BART for Query Suggestion" 5 Oct 2021.

[4] Shuyang Cao and Lu Wang "Controllable Open-ended Question Generation with A New Question Type Ontology "1 Jul 2021

[5] Chenyang Lyu, Lifeng Shang, Yvette Graham, Jennifer Foster, Xin Jiang, Qun Liu "Improving Unsupervised Question Answering via Summarization-Informed Question Generation " 16 Sep 2021 .

[6] md rashad al hasan rony,uttam kumar,roman teucher, liubov kovriguina1 , and jens lehmann "A Generative Approach for SPARQL Query Generation From Natural Language Questions "5 July 2022 .

[7] Sivasurya Santhanam "context based text-generation using lstm networks "May 4, 2020 Nikahat Mulla1,Prachi Gharpure "Automatic question generation: a review of methodologies, datasets, evaluation metrics, and applications "30 January 2023

[8] Ghader Kurdi1,Jared Leo1, Bijan Parsia1,Uli Sattler,Salam Al-Emari "A Systematic Review of Automatic Question Generation for Educational Purposes "21 November 2019

[9] Stefan Feuerriegel,Jochen Hartmann,Christian Janiesch, Patrick Zschech "Generative AI ": 12 September 2023

[10] Xiyao Ma,Qile Zhu,Yanlin Zhou, Xiaolin Li2 "Improving Question Generation with Sentence-Level Semantic Matching and Answer Position Inferring "

[11] Luis Enrico Lopez, Diane Kathryn Cruz, Jan Christian Blaise Cruz, Charibeth Cheng "Transformer-based End-to-End Question Generation" 3 May 2020.

[12] Shweta Yadav, Deepak Gupta , Asma Ben Abacha , Dina Demner-Fushman" Question-aware transformer models for consumer health question summarization"

[14] Asahi Ushio and Fernando Alva-Manchego and Jose Camacho-Collados" Generative Language Models for Paragraph-Level Question Generation" 2 Jan 2023.

[15] David Baidoo-Anu1 , Leticia Owusu Ansah" Education in the Era of Generative Artificial Intelligence (AI): Understanding the Potential Benefits of ChatGPT in Promoting Teaching and Learning" 31.12.2023