AUTOMATED QUESTION GENERATOR AND SUMMARIZER USING MACHINE LEARNING

A DESIGN PROJECT REPORT

Submitted by

KABILESHWARAN K

MELWIN A.B

MOHAMED FAIZUL S

RAVIDHARSHEN M.K

in partial fulfilment for the award of the

degree of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENGE AND DATA SCIENCE

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An autonomous Institution to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM–621112

NOVEMBER 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

SAMAYAPURAM–621112

BONAFIDE CERTIFICATE

Certified that this design project report titled **“AUTOMATED QUESTION GENERATOR AND SUMMARIZER USING MACHINE LEARNING”** is the bonafide work of **KABILESHWARAN K (REG NO: 811721243022) MELWIN A.B (REG NO: 811721243027) MOHAMED FAIZUL S (REG NO: 811721243030) RAVIDHARSHEN M.K (811721243043)** who carried out the project under my supervision.

SIGNATURE	**SIGNATURE**
Dr.T. Avudaiappan M.E., Ph.D.,	Mrs. P.Jasmine Jose M.E.,
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Associate Professor	Assistant Professor
Department of AI	Department of AI
K. Ramakrishnan College of Technology	K. Ramakrishnan College of Technology
(Autonomous)	(Autonomous)
Samayapuram – 621112	Samayapuram – 621112

Submitted for the viva-voce examination held on ………………

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We jointly declare that the project report on **“AUTOMATED QUESTION GENERATOR AND SUMMARIZER USING MACHINE LEARNING”** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **“ANNA UNIVERSITY CHENNAI”** for the requirement of Degree of **BACHELOR OF TECHNOLOGY**. This design project report is submitted on the partial fulfilment of the requirement of the award of Degree of **BACHELOR OF TECHNOLOGY**.

SIGNATURE

____

KABILESHWARAN K

____

MELWIN A B

____

MOHAMED FAIZUL S

____

RAVIDHARSHEN M.K

PLACE : SAMAYAPURAM

DATE :

ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and in - debt to our institution “**K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)**” ,for providing us with the opportunity to do this project.

We are glad to credit honourable chairman **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director **Dr.S. KUPPUSAMY, MBA., Ph.D.,** for forwarding to our project and offering adequate duration in completing our project.

We would like to thank our Principal **Dr.N. VASUDEVAN, M.E., Ph.D.,** who gave opportunity to frame the project the full satisfaction.

We whole heartily thanks to **Dr.T. AVUDAIAPPAN**, **M.E., Ph.D.,** HEAD OF THE DEPARTMENT, **ARTIFICAL INTELLIGENCE** for providing his encourage pursuing this project.

I express my deep and sincere gratitude to my project guide **Mrs. P. JASMINE JOSE M.E.,** ASSISTANT PROFESSOR, **ARTIFICIAL INTELLIGENCE** for his incalculable suggestions, creativity, assistance and patience which motivated me to carry out the project successfully.

I render my sincere thanks to my project coordinator **Mr.P.B. ARAVIND PRASAD ,BE., M.Tech.,** other faculties and non-teaching staff members for providing valuable information during the course. I wish to express my special thanks to the officials & Lab Technicians of our departments who rendered their help during the period of the work progress.

ABSTRACT

In educational technology, automated question generation using machine learning is gaining attention for its potential to enhance personalized learning and streamline assessment processes. This paper proposes a robust framework for an automated question generator that utilizes machine learning, particularly natural language processing (NLP) and deep learning models, to produce a diverse array of questions tailored to different topics, cognitive levels, and learning objectives. The system leverages techniques such as text classification, entity recognition, and semantic analysis to identify core concepts within educational content and generate meaningful questions, ranging from factual queries to higher-order analytical prompts.The generator is trained on a large, curated dataset comprising educational materials and sample question-answer pairs, enabling it to capture nuanced language structures and varying question types. To enhance relevance and adapt to different subject domains, the model is fine-tuned through a combination of supervised and reinforcement learning, incorporating feedback from educators and learners to iteratively improve question quality. Further, an assessment module evaluates generated questions for clarity, complexity, and alignment with predefined standards, ensuring their suitability for adaptive learning systems. Experimental results indicate that the proposed system achieves high accuracy in generating relevant and context-aware questions, positioning it as a valuable tool for scalable, automated assessments in online learning and interactive educational applications.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ML	Machine Learning
NLP	Natural Language Processing
DL	Deep Learning
QG	Question Generation
NLG	Natural Language Generation
QA	Question Answering
BERT	Bidirectional Encoder Representations from Transformers
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
BLEU	Bilingual Evaluation Understudy

CHAPTER 1
INTRODUCTION

In recent years, advancements in machine learning (ML) have revolutionized various domains, including natural language processing (NLP). One of the key applications of NLP is in automating the generation of questions and summarizing texts, which holds immense potential in educational technology, content management, and accessibility. The "Automated Question Generator and Summarizer Using Machine Learning" project aims to leverage state-of-the-art ML algorithms to automate the creation of meaningful questions and concise summaries from a given body of text. By analyzing input data through various NLP techniques such as text extraction, tokenization, and deep learning models, this system generates questions that test comprehension and summarizes complex information into easy-to-understand formats. This project seeks to enhance learning experiences by providing automated tools for content review, helping educators and learners interact with materials more efficiently and effectively.

1.1 BACKGROUND

The need for effective and efficient information retrieval systems has grown significantly in recent years, particularly in education and content management. Traditional methods of summarizing text and generating questions often require significant manual effort and are time-consuming. However, with the rapid advancements in machine learning, especially in natural language processing (NLP), automated systems have shown promise in addressing these challenges. NLP techniques such as text summarization, question generation, and sentiment analysis are now being used to analyze large datasets, transforming the way information is processed and presented. In particular, question generation algorithms help create questions from texts to enhance comprehension and facilitate learning.

1.2 PROBLEM STATEMENT

In the current educational landscape, both students and educators face significant challenges when it comes to efficiently reviewing and processing large volumes of textual content. Manually summarizing materials and formulating relevant questions for comprehension testing is time-consuming and resource-intensive. As the volume of available information increases, the need for automated systems that can assist in these tasks becomes more critical. Existing methods for automatic summarization and question generation often lack accuracy, coherence, and context-awareness, limiting their effectiveness in real-world applications. The absence of an integrated system that can both summarize text and generate relevant questions based on the content further complicates this issue.

1.3 AIMS AND OBJECTIVES

1.3.1 AIM

- Develop an automated system that generates meaningful and contextually accurate questions from a given body of text using machine learning techniques.
- Design an algorithm capable of summarizing large and complex texts into concise, coherent summaries without losing essential information.
- Enhance learning and content review processes by providing a tool that automates question generation and summarization for educational purposes.
- Utilize state-of-the-art natural language processing (NLP) models, such as transformers and deep learning, to improve the accuracy of both question generation and summarization tasks.
- Create an intuitive and scalable system that can handle diverse topics and content types, making it applicable across various domains and user needs.

1.3.2 OBJECTIVES

- Investigate and analyze existing machine learning and NLP techniques for question generation and text summarization.
- Develop a machine learning-based question generation model that extracts relevant questions from a given text.
- Design and implement a text summarization model that condenses large documents into shorter, more manageable summaries while retaining core information.
- Optimize the question generation model for accuracy, coherence, and relevance to the content.
- Assess the performance of the summarization model by comparing it with manually created summaries.
- Evaluate the quality of the generated questions through user feedback and comparison with domain-specific benchmarks.
- Create a user-friendly interface that allows users to input text and receive automated summaries and questions.
- Integrate a feedback mechanism into the system to continuously improve both question generation and summarization quality.
- Test the system's effectiveness across different types of texts, such as academic papers, news articles, and educational content.
- Ensure that the system can handle a wide variety of languages and topics to be used in diverse educational and professional contexts.
- Incorporate advanced NLP techniques such as transformers or BERT to improve the system's understanding of text and context.
- Ensure the scalability of the system to handle large datasets and texts of varying lengths efficiently.
- Maintain the ethical use of data by ensuring the system only processes publicly available or user-provided content.

CHAPTER 2
LITERATURE SURVEY

2.1 TITLE: Opinerium: Subjective Question Generation Using Large Language Models

AUTHOR: Pedram Babakhani, Doreen Sacker, Fikret Sivrikaya, and Sahin Albayrak

YEAR OF PUBLICATION: 2024

ABSTRACT: Opinerium is a pioneering tool focused on generating subjective questions to promote public engagement with media content. Unlike conventional models designed for factual question generation, Opinerium creates opinion-based inquiries, asking users to reflect on and share personal views on topics such as news articles.

ALGORITHM USED: Subjective Question Generation Using Large Language Models Opinerium employs fine-tuned transformer-based models like flan-T5 and GPT-3, using a Sequence-to-Sequence (Seq2Seq) framework. This approach is tailored to generate subjective, opinion-based questions from given textual contexts. The model is fine-tuned on a dataset of 40,000 news articles, which includes both English translations of multilingual content and binary subjective questions.

MERIT: By generating subjective questions, Opinerium encourages users to express personal opinions, fostering deeper engagement with content

DEMERIT: Although the model is powerful for news articles, its generalizability to other domains (e.g., social media, academic texts) may be limited unless further fine-tuned

2.2 TITLE: A Survey of Automatic Text Summarization: Progress, Process and Challenges

AUTHOR: M. F. Mridha, Aklima Akter Lima, Kamruddin Nur, Sujoy Chandra Das

YEAR OF PUBLICATION: 2021

ABSTRACT: The research highlights ATS as a critical tool for managing the vast amounts of text generated daily. It focuses on summarizing text to retain core information, covering both traditional methods and more recent deep learning techniques, including neural network-based models like sequence-to-sequence architectures

ALGORITHM USED: The paper "A Survey of Automatic Text Summarization: Progress, Process, and Challenges" offers a comprehensive analysis of advancements and methodologies in automatic text summarization (ATS). It provides a detailed taxonomy of the field, covering classical and deep learning-based approaches, and outlines the primary algorithms used for extractive and abstractive summarization. The paper also discusses key issues, such as feature extraction methods, dataset requirements, and performance evaluation metrics used in summarization research

MERIT: It presents an extensive overview of various ATS techniques, which is valuable for researchers seeking to understand the field's scope and evolution.

DEMERIT: The paper may lack real-world applications and examples, making it harder for readers to understand the practical utility of ATS models outside theoretical frameworks.

2.3 TITLE: Questionator - Automated Question Generation using Deep Learning

AUTHOR: Animesh Srivastava, Shantanu Shinde

YEAR OF PUBLICATION: 2020

ABSTRACT: The Questionator project addresses the growing need for automated question generation due to increased data and the demand for scalable education tools. It generates questions based on textual and visual inputs, making it suitable for both academic testing and interactive virtual assistant applications. The system uses CNNs for feature extraction and LSTMs to generate coherent natural language questions from these features, allowing educators and other users to efficiently create assessments without manual question writing.

ALGORITHM USED: The Questionator model focuses on automated question generation using deep learning, specifically in educational settings. It leverages deep learning architectures like Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks for processing both text and image inputs, enabling it to create a variety of question types from given content.

MERIT: Questions generated are designed to be relevant and informative, potentially increasing student engagement and comprehension.

DEMERIT: The deep learning models used in Questionator (CNNs and LSTMs) require significant computational power, which may limit accessibility for institutions without advanced hardware.

2.4 TITLE: Automatic Question Generation Using Natural Language Processing and Transformers

AUTHOR: Abheer Hamdy

YEAR OF PUBLICATION: 2024

ABSTRACT: The study addresses the challenge of generating relevant and high-quality questions from textual data, a task essential in educational technology and content analysis. The T5 model, pre-trained on large datasets like the Stanford Question Answering Dataset (SQuAD), is fine-tuned to understand the relationship between context and potential questions, producing questions that align with the information in the text.

ALGORITHM USED: The T5 model, with its encoder-decoder structure, forms the backbone of the question-generation system. During training, T5 is fine-tuned on a dataset where each question is paired with a context, which allows the model to learn to generate questions based on given content. Self-attention is used in each layer to understand the contextual relationship within sentences, enhancing the model’s ability to produce coherent and contextually relevant questions.

MERIT: The T5 model generates questions that are contextually accurate, benefiting educational and assessment tools by aligning questions closely with source material.

DEMERIT: The T5 model and similar transformers require significant computational resources, which can be a constraint for large-scale or resource-limited applications.

2.5 TITLE: Question and Assessment Generator: Deep Learning Approach for Customizable and Intelligent Assessment Creation

AUTHOR: Snehal Rathi

YEAR OF PUBLICATION: 2024

ABSTRACT: The main goal of this approach is to simplify the process of creating educational assessments. The system leverages content analysis to identify key concepts and relationships, which it then uses to generate relevant and diverse questions. By using deep learning models, it can analyze context and create high-quality questions that align with learning goals. This generator is also able to produce distractors for multiple-choice questions, enhancing the overall assessment quality. This tool aims to reduce the time and effort educators spend on question generation, enabling faster and more adaptable assessments.

ALGORITHM USED: The core algorithms involve deep learning models that include neural networks and transformer-based NLP architectures. These models allow the generator to analyze large volumes of text, extract key concepts, and formulate relevant questions. Template-based and rule-based methods are also employed to ensure variety and alignment with educational standards, while quality assurance processes check for clarity, grammar, and factual accuracy.

MERIT: Significantly reduces the manual effort required for question creation, saving time and resources.

DEMERIT: The accuracy and relevance of questions are highly dependent on the quality of the input content, meaning low-quality source material can lead to poorly generated questions.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Existing automated question generation and summarization systems leverage machine learning and NLP to efficiently create relevant questions and concise summaries from large volumes of text. These systems often use transformer-based models like BERT, GPT, and T5, which analyze text context to generate questions and identify key information for summarization. Question generation models are commonly trained on datasets like SQuAD, enabling them to understand context and formulate relevant questions, while summarization models, such as BART and PEGASUS, generate coherent, compressed summaries. By combining these capabilities, modern systems support educational and content management applications, streamlining tasks such as creating assessment materials and summarizing lengthy documents. However, these models still face challenges in ensuring contextual relevance and minimizing computational requirements, which are crucial for effective, scalable deployment

3.1.1 ALGORITHM USED

1. Text-to-Text Transfer Transformer (T5)

The T5 model, developed by Google, is a transformer-based algorithm that treats every NLP task as a text-to-text problem. In question generation, T5 can encode input text into a format that enables the model to generate coherent questions related to the content, providing flexibility across various domains. It’s particularly effective due to its ability to pre-train on large datasets and adapt to specific tasks like question generation through fine-tuning on question-answering datasets such as SQuAD

2. Bidirectional and Auto-Regressive Transformers (BART)

BART is another transformer model, notable for its effectiveness in both summarization and question generation tasks. It combines a bidirectional encoder (similar to BERT) and an autoregressive decoder, enabling it to capture context from the entire input text for accurate summarization. BART is widely used for abstractive summarization, making it suitable for generating human-like summaries that retain essential information while also performing well in generating questions

3. GPT (Generative Pre-trained Transformer)

GPT, especially GPT-3, is widely used for question generation due to its ability to generate coherent and contextually relevant questions without extensive task-specific training. Leveraging its powerful language generation capabilities, GPT-3 can create high-quality questions based on context, offering flexibility across domains. Its strength lies in its large-scale language model, pre-trained on diverse datasets, making it highly versatile for both summarization and question generation

3.1.2 DISADVANTAGE OF EXISTING SYSTEM

- The models can sometimes generate questions or summaries that are irrelevant or lack coherence, especially when dealing with ambiguous or complex text.
- Ensuring the output aligns perfectly with the input context remains a challenge.
- Transformer-based models like BERT, GPT, T5, BART, and PEGASUS are computationally intensive.
- Deploying these systems for large-scale applications requires significant hardware resources (e.g., GPUs), which can be cost-prohibitive.

- Poorly written or noisy input text (e.g., with grammatical errors or lack of structure) may lead to suboptimal outputs, reducing system reliability.
- While models are often trained on general datasets like SQuAD, they may not perform well in domain-specific contexts (e.g., medical, legal, or technical content) without additional fine-tuning.
- The quality of generated questions and summaries heavily depends on the diversity and quality of training datasets.
- Models may inadvertently reflect biases present in the training data.
- These systems do not cater well to individual user needs, such as generating questions with specific difficulty levels or summaries of particular lengths.
- Real-time processing for large datasets or multiple users simultaneously can result in delays due to the heavy computational load.
- Automated evaluation metrics like BLEU and ROUGE for summarization and question generation are not always reliable indicators of quality.
- Human evaluation is still needed, making the development cycle longer and costlier.

3.2 PROPOSED SYSTEM

The proposed system for an Automated Question Generator and Summarizer Using Machine Learning aims to streamline educational content creation by generating relevant questions and concise summaries from vast textual information. This system integrates state-of-the-art NLP techniques, leveraging transformer-based models like T5, BART, and GPT for both tasks. The summarizer component would generate coherent, concise summaries to provide quick insights into lengthy content, while the question generator would create questions that are contextually accurate and aligned with educational objectives. Using a fine-tuned model approach, the system adapts to various subject domains, enhancing its versatility and relevance in different educational contexts. The generated questions may include multiple-choice, short-answer, and true/false formats, while the summaries capture core ideas, facilitating learning and comprehension. By automating these tasks, the system seeks to reduce manual workload for educators and improve learning accessibility, though it may require quality checks to ensure content accuracy and relevance.

3.2.1 TECHNIQUES USED

1. Transformer-Based Language Models

Transformers, such as T5, BERT, and GPT, are the backbone of both question generation and summarization. These models process text using self-attention mechanisms, which allow them to understand context across long sentences. They are pre-trained on vast datasets and can be fine-tuned for specific tasks like generating questions or summarizing text.

2. Sequence-to-Sequence Modeling

Sequence-to-sequence (Seq2Seq) models, especially with encoder-decoder structures, are widely used in NLP tasks. In question generation, these models encode input text and decode it into a question format. For summarization, they encode the main text and decode it into a compressed version, maintaining the essence of the original content.

3. Extractive Summarization

In extractive summarization, key sentences are selected directly from the text to form a summary. Techniques like TextRank or embeddings-based methods rank sentences by importance, creating summaries by extracting the most relevant ones. This technique is often combined with other models for a more comprehensive approach.

4. Abstractive Summarization

Abstractive summarization models, often using transformers, generate new phrases to express the main ideas of a text. Unlike extractive methods, these models paraphrase content, creating summaries that feel more natural and coherent. This approach is particularly useful in producing summaries that are closer to human expression.

5. Template-Based Question Generation

Template-based question generation involves creating question structures or templates that are filled with relevant content based on the input text. By defining question types such as "What is...?" or "Explain how...", this technique can produce accurate questions, although it may lack the creativity of more advanced models.

6. Attention Mechanism

Attention mechanisms allow models to focus on the most relevant parts of the input text when generating questions or summaries. By assigning different weights to words or phrases, the model can prioritize significant information, leading to more contextually accurate questions and summaries.

7. Named Entity Recognition (NER)

NER is used to identify entities like people, places, dates, and events in text. In question generation, NER helps the model identify important subjects around which to frame questions. It also aids in summarization by highlighting key information that should be included in a summary.

3.2.2 ADVANTAGES OF PROPOSED SYSTEM

- Automates the time-consuming tasks of generating questions and summaries, reducing the manual workload for educators and content creators.
- The use of fine-tuned transformer models (e.g., T5, BART, GPT) ensures adaptability to various subject domains, making the system applicable in diverse educational contexts.
- By leveraging state-of-the-art NLP techniques, the system generates questions and summaries that are contextually accurate and aligned with educational goals.
- The system can produce a variety of question types, such as multiple-choice, short-answer, and true/false, catering to different assessment styles and learning needs.
- Summaries provide quick insights into lengthy content, facilitating easier understanding and comprehension for learners with limited time or varying literacy levels.
- The automated nature of the system minimizes variability in the quality of questions and summaries, compared to manual efforts which can vary significantly across individuals.
- Capable of processing vast amounts of text efficiently, making it suitable for applications involving large-scale educational content or document management.
- With fine-tuning, the system can be customized to generate questions or summaries based on specific difficulty levels or educational requirements.
- The system can be integrated with learning management systems (LMS) and other educational platforms, streamlining workflows for educators and learners.
- Reduces reliance on human effort for repetitive tasks, potentially lowering the cost of educational content creation in the long term.

CHAPTER 4

SYSTEM SPECIFICATION

4.1 HARDWARE SYSTEM CONFIGURATION

- Processor (CPU): Multi-core processor such as AMD Ryzen 9 or Intel Core i9 (10th or 11th generation).
- Graphics Processing Unit (GPU): NVIDIA A100, RTX 3090, or Tesla V100
- Memory (RAM): 64 GB DDR4 or higher
- Storage: 1 TB SSD (NVMe preferred) and additional HDD for backups (2 TB)
- Network: High-speed internet connection (at least 1 Gbps).
- Power Supply: 750W or higher PSU (with high efficiency rating like 80+ Gold or Platinum).
- Operating System: Ubuntu 20.04 LTS or Windows 10 Pro.

4.2 SOFTWARE SYSTEM CONFIGURATION

- Operating System: Ubuntu 20.04 LTS (Linux-based)
- Deep Learning Frameworks: TensorFlow and PyTorch
- NLP Libraries: Hugging Face Transformers, spaCy
- Web Frameworks for Deployment: Flask or FastAPI
- Version Control: Git, GitHub
- Data Processing Libraries: Pandas, NumPy
- Visualization Tools: TensorBoard, Matplotlib

4.3 SOFTWARE DESCRIPTION

The software system for the Automated Question Generator and Summarizer Using Machine Learning project integrates several advanced tools and libraries to facilitate natural language processing, deep learning, and real-time deployment. The system utilizes Ubuntu 20.04 LTS for its stability and compatibility with machine learning tools. TensorFlow and PyTorch are employed for model training and fine-tuning, leveraging pre-trained transformer models from Hugging Face Transformers for tasks like question generation and summarization. spaCy is used for text preprocessing and tokenization, ensuring efficient data handling. The system's backend is deployed using Flask or FastAPI, enabling real-time interaction with users through web-based APIs. Data is processed using Pandas and NumPy, allowing for efficient manipulation and analysis. TensorBoard is used to monitor and visualize model performance during training, while Matplotlib helps visualize data and results. Version control is managed via Git and GitHub, ensuring seamless collaboration and code management throughout the project lifecycle. This combination of tools ensures that the system can effectively process large datasets, train complex models, and deliver accurate, real-time results for both question generation and summarization tasks.

4.3.1 LIBRARY

1. Hugging Face Transformers

Overview:

Hugging Face Transformers is a library designed to simplify the use of pre-trained transformer models for various NLP tasks. It provides state-of-the-art models like BERT, GPT-2, T5, and others, which can be fine-tuned on custom datasets for specific tasks such as question generation and text summarization. The library supports both PyTorch and TensorFlow, making it highly versatile.

Usage in the Project:

In this project, Hugging Face Transformers is used to fine-tune models like T5 or BERT for generating context-aware questions and summaries. These pre-trained models, trained on vast datasets, have shown strong performance in NLP tasks and can be quickly adapted to new tasks with minimal computational overhead.

Key Features:

- Easy access to pre-trained models and datasets.
- Pre-processing tools like tokenization to prepare text for models.
- Fine-tuning capabilities for specific applications such as question generation.
- Extensive support for various NLP tasks including text generation, classification, and summarization.

2. spaCy

Overview:

spaCy is a fast and efficient open-source library for NLP, designed to handle large-scale text processing. It offers pre-trained models for tokenization, part-of-speech tagging, named entity recognition (NER), dependency parsing, and more. spaCy is highly optimized for performance and can handle a variety of NLP tasks efficiently.

Usage in the Project:

spaCy is primarily used in this project for text preprocessing tasks such as tokenization, sentence segmentation, and named entity recognition (NER). For question generation, extracting important entities and key phrases helps in generating meaningful questions from input text. For summarization, spaCy helps identify the core content by recognizing named entities and important terms to ensure relevant information is included in the summary.

Key Features:

- Fast and efficient processing of large text datasets.

- Pre-trained models for a wide range of languages.
- In-depth support for syntactic parsing and entity recognition, which aids in generating accurate summaries and questions.
- Integration with deep learning frameworks like TensorFlow and PyTorch.

3. TensorFlow

Overview:

TensorFlow is an open-source machine learning framework developed by Google, widely used for training and deploying machine learning models, especially deep learning models. It provides a comprehensive ecosystem for building and fine-tuning models, with support for both CPU and GPU execution. TensorFlow is particularly known for its ability to scale and handle large datasets efficiently.

Usage in the Project:

In this project, TensorFlow is used for building and training deep learning models, particularly for tasks like summarization and question generation. Models like T5 and BERT are fine-tuned using TensorFlow, taking advantage of its ability to efficiently handle the large-scale computations required for transformer models.

Key Features:

- Scalable and flexible for training deep learning models, including large transformer architectures.
- Built-in support for GPU acceleration, which speeds up training.
- TensorFlow Serving allows for deploying models into production environments.

- Extensive library support for neural network layers, optimizers, and loss functions to fine-tune and customize models for specific tasks.

4.3.2 DEVELOPING ENVIRONMENT

1. Programming Languages

Python (Recommended Version: 3.8 or higher)

- Reason: Python is the primary language for machine learning, NLP, and AI development. It supports numerous libraries like TensorFlow, PyTorch, Hugging Face Transformers, and spaCy, which are essential for this project.
- Package Management: Use pip or conda to install and manage Python dependencies.

2. Integrated Development Environment (IDE)

VS Code or PyCharm

- Reason: VS Code is a lightweight and highly customizable IDE, perfect for Python development. It supports extensions for Git, Jupyter notebooks, and linting tools to maintain code quality. PyCharm is a more feature-rich IDE with built-in support for Python, deep learning frameworks, and version control.
- Features: Code completion, debugging, version control integration, and Jupyter notebook support.

3. Machine Learning Frameworks

TensorFlow 2.x or PyTorch

- Reason: TensorFlow and PyTorch are the most widely used deep learning frameworks in the industry. Both libraries offer advanced capabilities for training and deploying neural networks, including pre-trained models for NLP tasks like question generation and summarization.

- Version Control: Use TensorFlow 2.x or PyTorch 1.x for compatibility with modern transformer models.

4. NLP Libraries

Hugging Face Transformers

- Reason: Hugging Face provides pre-trained transformer models and fine-tuning capabilities for NLP tasks like summarization and question generation. It integrates seamlessly with TensorFlow and PyTorch.

spaCy

- Reason: spaCy is a fast NLP library used for tokenization, syntactic parsing, and named entity recognition (NER). It helps preprocess the text data and extract key entities for better question generation and summarization.

5. Data Science Libraries

Pandas and NumPy

- Reason: Pandas is used for data manipulation, while NumPy handles numerical operations that are essential for preprocessing and model input handling. These libraries help efficiently manage and preprocess large datasets for training.

6. Virtual Environment

Anaconda (Recommended) or venv

- Reason: Anaconda helps manage Python dependencies and environments, providing a consistent setup across different machines. It comes with preconfigured libraries and tools that are essential for data science and machine learning projects.

- Alternatively, use venv (Python's built-in tool) to create a lightweight, isolated environment for managing project dependencies.

7. Version Control

Git and GitHub

- Reason: Git is essential for version control, allowing developers to track changes in the code and collaborate effectively. GitHub or GitLab repositories provide cloud-based hosting, easy collaboration, and continuous integration.

8. Containerization and Deployment

Docker

- Reason: Docker enables the creation of containerized environments, ensuring the project runs consistently across different machines. It simplifies the deployment of models in production.
- Integration: You can build Docker containers to encapsulate your model training and inference environment for easier scalability and deployment.

9. Cloud Computing Platforms (Optional)

Google Colab or AWS EC2

- Reason: For intensive model training, you can use cloud services like Google Colab (free GPUs for small-scale tasks) or AWS EC2 instances with powerful GPU configurations for large-scale model training.
- Usage: Google Colab is suitable for prototyping and testing models, while AWS EC2 instances are ideal for full-scale training and deployment.

10. Database and Storage

MongoDB or PostgreSQL

- Reason: Use databases like MongoDB for NoSQL storage (useful for storing structured or unstructured data like text) or PostgreSQL for more structured data requirements. These databases store training data, model outputs, and user inputs.

Cloud Storage (Google Cloud Storage, AWS S3)

- Reason: Cloud storage solutions provide scalability and reliability for storing large datasets, model checkpoints, and logs.

By setting up the above development environment, the project will be optimized for handling the machine learning, natural language processing, and real-time deployment tasks involved in automated question generation and summarization.

CHAPTER 5

ARCHITECTURAL DESIGN

5.1 SYSTEM DESIGN

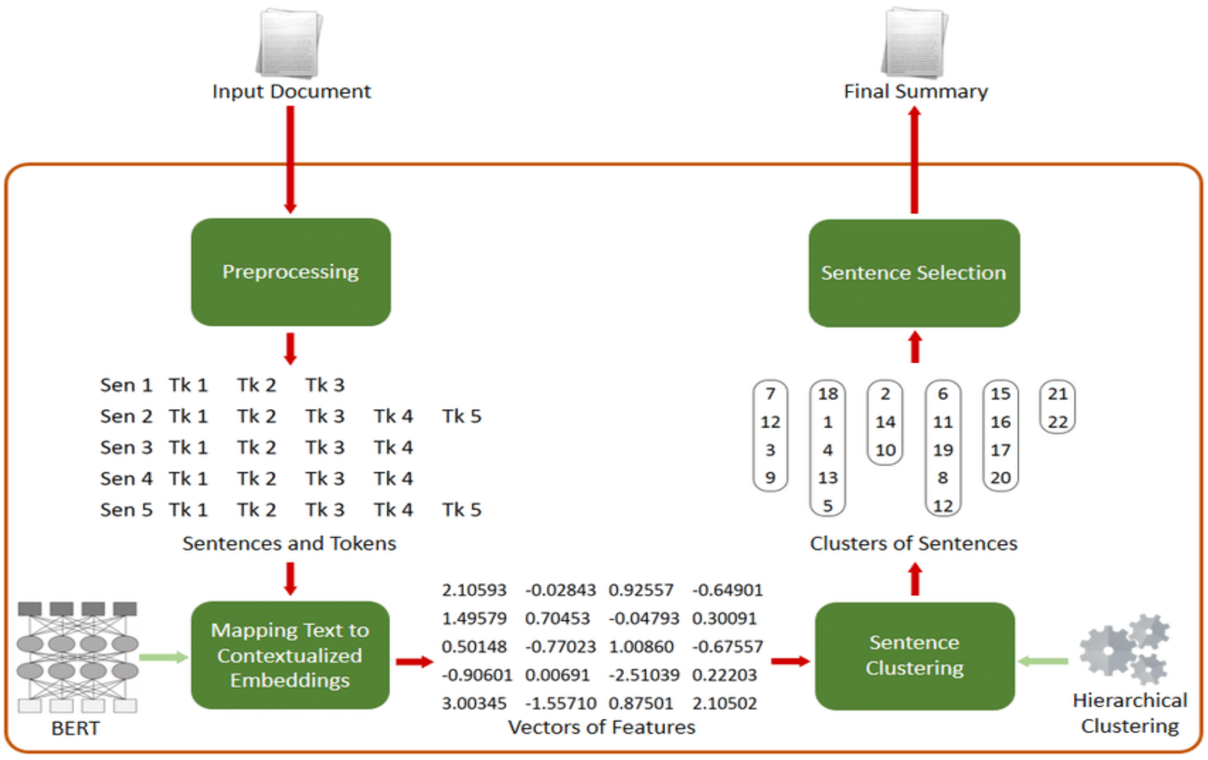


FIG:5.1 SYSTEM DESIGN

Fig 5.1 architecture of the Automated Question Generator and Summarizer system begins with the Input Layer, where raw text data, such as articles or books, is collected. The text then undergoes Preprocessing, including tokenization, cleaning, and normalization to prepare it for analysis. Feature Extraction follows, where the text is transformed into machine-readable vectors using methods like TF-IDF or embeddings (e.g., Word2Vec or BERT). The system employs two distinct Machine Learning Models: one for Summarization, which can be either extractive or abstractive, and another for Question Generation, often using transformer-based models like T5. Finally, the Output Layer generates a summary and relevant questions, with evaluation metrics such as ROUGE or BLEU used to assess performance and accuracy.

5.2 DATA FLOW DIAGRAM

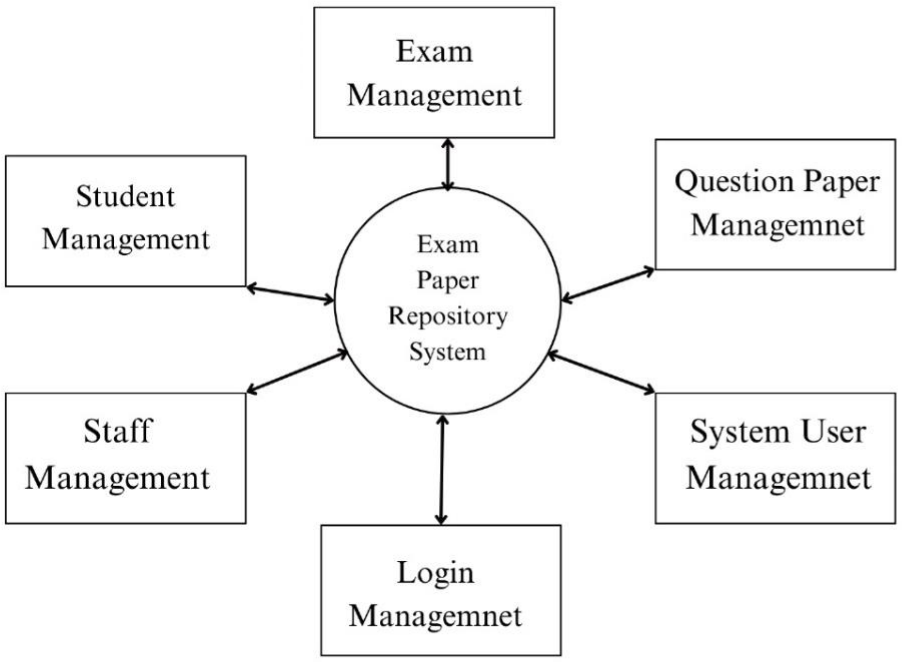


FIG:5.2 DATA FLOW DIAGRAM

Fig 5.2 Diagram Shows The Exam Paper Repository System is a centralized platform that integrates and manages tasks related to exams, students, staff, and system operations. It encompasses various subsystems, such as Exam Management for scheduling and maintaining exam data, and Question Paper Management for creating, storing, and retrieving papers, including automated generation. System User Management oversees user roles and permissions, while Login Management ensures secure access through authentication processes. Staff Management handles roles, materials, and exam reviews, while Student Management maintains records like exam registrations and results. These interconnected subsystems streamline operations and ensure efficient and seamless functionality.

5.3 USE CASE DIAGRAM

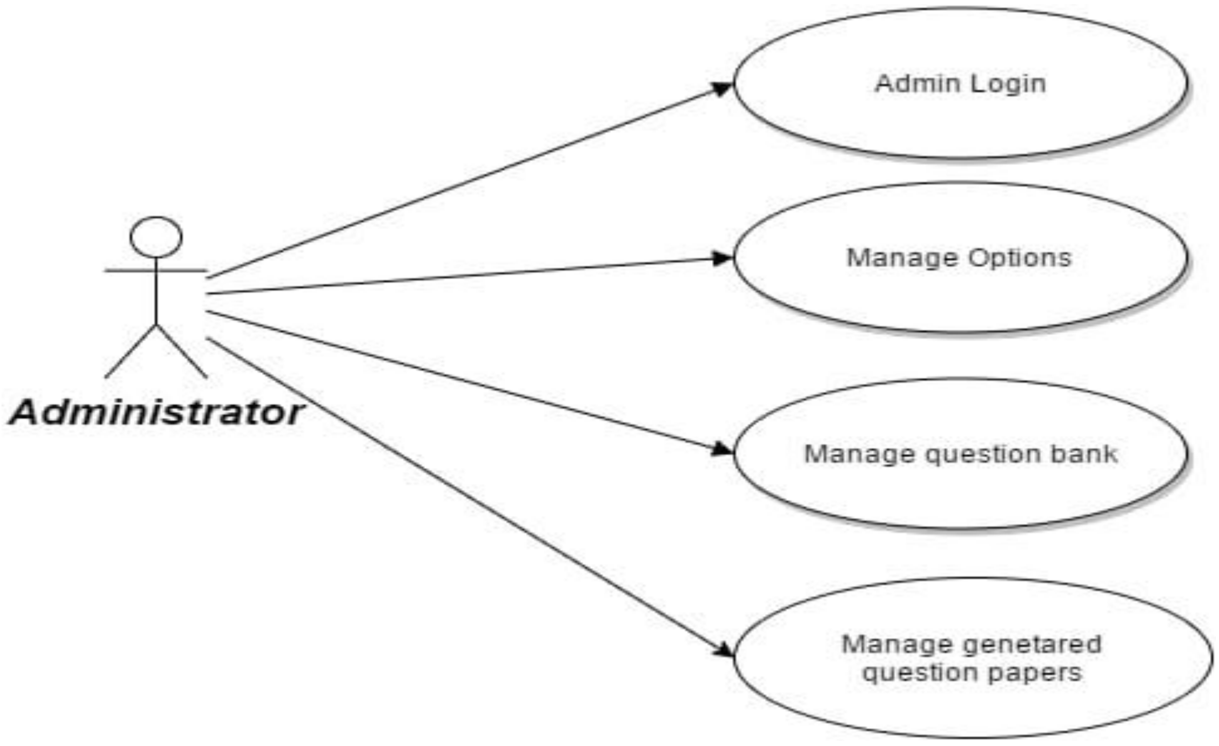


FIG:5.3 USE CASE DIAGRAM

Fig 5.3 diagram illustrates the administrator's interaction with key system functionalities. The administrator securely logs in using authentication mechanisms like passwords or multi-factor authentication. They manage system settings, such as configuring exam parameters and user permissions, and oversee the question bank by adding, editing, or deleting questions to ensure alignment with curriculum requirements. Additionally, the administrator reviews, approves, or modifies automatically generated question papers. The lines connecting the administrator to use cases highlight their access to all core operations. This diagram emphasizes the administrator’s role in ensuring system efficiency, content quality, and operational flexibility.

CHAPTER 6

MODULE DESCRIPTION

6.1 MODULES

- Data Collection and Preprocessing
- Text Representation (Feature Extraction)
- Summarization Models
- Question Generation Models
- Evaluation Metrics
- Deployment and Integration
- Fine-Tuning and Optimization

6.1.1 Data Collection and Preprocessing

- BeautifulSoup, Requests, NLTK, and SpaCy are essential Python libraries that collectively streamline the process of data collection and text preprocessing for natural language processing (NLP) tasks like question generation and summarization.
- BeautifulSoup and Requests are fundamental tools for web scraping, enabling users to extract raw text and data from online sources such as Wikipedia and news websites. Requests simplifies sending HTTP requests to retrieve web pages, while BeautifulSoup parses the HTML or XML content, allowing for easy navigation and extraction of specific elements, such as titles, paragraphs, or links.
- NLTK (Natural Language Toolkit) is a versatile library ideal for text preprocessing. It offers functionalities like tokenization (breaking text into smaller units), stopword removal (filtering out common words), and POS (part-of-speech) tagging, which are critical for cleaning and structuring raw text before analysis. NLTK is widely used to prepare textual data for machine learning models, ensuring better accuracy and performance.

- SpaCy, a modern and robust NLP library, complements NLTK by providing advanced features for complex tasks. It excels in named entity recognition (NER), dependency parsing, and syntactic analysis, delivering fast and accurate results even for large datasets. Its optimized architecture makes it suitable for real-world applications requiring detailed linguistic insights.

6.1.2 Text Representation (Feature Extraction)

- TF-IDF, Word2Vec, and BERT are key techniques in natural language processing (NLP) that enable the conversion of text into numerical representations, each serving distinct purposes and excelling in different scenarios.
- TF-IDF (Term Frequency - Inverse Document Frequency) is a statistical method used to quantify the importance of words within a document relative to a larger collection of documents (corpus). It combines term frequency (how often a word appears in a document) with inverse document frequency (how unique the word is across all documents), emphasizing words that are both frequent in a document and unique to it. TF-IDF is widely applied in traditional machine learning models and is particularly effective for smaller datasets or simpler tasks like text classification and information retrieval.
- Word2Vec, a neural network-based method, goes beyond frequency-based approaches by generating word embeddings—dense, high-dimensional vector representations of words. These embeddings capture semantic relationships and contextual meanings, enabling Word2Vec to identify similarities between words and understand their roles in broader contexts. This makes it ideal for tasks like question generation, where understanding the relationships between words is critical for generating coherent and contextually appropriate questions.

- BERT (Bidirectional Encoder Representations from Transformers) represents a significant advancement in NLP through its use of contextualized word embeddings. Unlike traditional embeddings, BERT generates representations that dynamically adjust based on the surrounding text, capturing the nuanced meanings of words in context. Pre-trained on massive datasets, BERT excels in complex NLP tasks such as summarization and question generation, where deep contextual understanding and accurate linguistic representation are essential. Its transformer-based architecture allows for bidirectional analysis of text, ensuring that every word's meaning is interpreted within its full context.

6.1.3 Summarization Models

- BART, T5, and BERTSUM are advanced transformer-based models tailored for summarization and other natural language processing (NLP) tasks, each leveraging unique approaches to generate or extract meaningful content.
- BART (Bidirectional and Auto-Regressive Transformers) is a transformer model specifically designed for abstractive summarization, where the goal is to generate novel summaries that rephrase and condense content rather than merely extracting sentences. BART combines the strengths of bidirectional transformers, which analyze input text context from both directions, and auto-regressive transformers, which predict text in a sequential manner. This dual capability enables BART to create high-quality summaries that are coherent, context-aware, and semantically rich, making it a leading choice for rewriting and summarizing complex content.
- T5 (Text-to-Text Transfer Transformer) is an extremely versatile model that treats all NLP tasks—from summarization to translation and question generation—as text-to-text problems. This unified framework simplifies task implementation and ensures consistent performance across various

applications. For summarization, T5 effectively generates concise and fluent summaries, while its ability to handle diverse text inputs also makes it highly suitable for generating contextually appropriate questions. The availability of pre-trained T5 models enhances its adaptability, allowing it to deliver strong results even with limited fine-tuning.

- BERTSUM, a specialized variant of BERT fine-tuned for extractive summarization, takes a different approach by identifying and selecting the most relevant sentences from a document. This extractive method ensures that the generated summaries retain the original phrasing and structure of the source content, making BERTSUM ideal for applications where maintaining the authenticity and accuracy of the input text is critical. It excels in tasks requiring concise and factually precise summaries, particularly for structured documents like research papers or reports.

6.1.4 Question Generation Models

- T5, GPT-3, and fine-tuned BERT are highly effective models for question generation, each bringing unique strengths to the task and excelling in different scenarios.
- T5 (Text-to-Text Transfer Transformer) is specifically designed for text-to-text tasks, making it a natural fit for generating questions from input text. By reframing question generation as a text-to-text problem, T5 can be fine-tuned to produce high-quality, contextually aware questions. This adaptability allows T5 to handle a wide range of queries, from straightforward factual questions to more nuanced and complex ones. Its design ensures that the generated questions are not only accurate but also aligned with the underlying context, making it a reliable choice for applications requiring precision and relevance.
- GPT-3 (Generative Pre-trained Transformer 3) represents a breakthrough in language modeling with its unprecedented scale and capability. As one

of the most powerful language models, GPT-3 excels in generating creative, varied, and detailed questions. Its extensive training on diverse datasets enables it to understand and generate questions that are rich in context and semantic depth. GPT-3 is particularly suited for advanced question-generation tasks, including dialogue systems and applications where creativity and diversity in question formulation are critical.

- Fine-tuned BERT (Bidirectional Encoder Representations from Transformers) offers a robust solution for question generation by leveraging its strong contextual understanding. When fine-tuned with domain-specific or task-specific datasets, BERT can generate highly relevant and precise questions tailored to the input text. Its bidirectional nature allows it to analyze and incorporate the full context of a passage, ensuring that the generated questions are both meaningful and well-grounded. This makes fine-tuned BERT an excellent choice for fine-grained tasks where accuracy and contextual alignment are paramount.

6.1.5 Evaluation Metrics

- ROUGE (Recall-Oriented Understudy for Gisting Evaluation) and BLEU (Bilingual Evaluation Understudy) are widely used metrics for assessing the quality of machine-generated text, each tailored to specific tasks. ROUGE is primarily designed for evaluating text summarization. It measures the overlap of n-grams, sequences, or word pairs between the generated summary and reference summaries, providing insights into the accuracy, relevance, and overall quality of the summary. This makes ROUGE a standard for evaluating machine-generated summaries in tasks like abstractive or extractive summarization.
- On the other hand, BLEU is focused on evaluating machine-generated translations or structured outputs like question-answer pairs. It calculates the precision of n-grams in the generated text compared to reference texts,

emphasizing exact matches and penalizing discrepancies in sequence. BLEU is particularly effective in assessing the quality of generated questions or answers, ensuring they align closely with human-created references. Both metrics play crucial roles in benchmarking the performance of natural language processing models across different text-generation tasks.

6.1.6 Deployment and Integration

- Flask, FastAPI, and Streamlit are powerful Python frameworks that enable seamless deployment and interaction with machine learning (ML) models, each serving distinct purposes in creating APIs and user interfaces.
- Flask and FastAPI are widely used web frameworks for developing and deploying RESTful APIs, making it easy to expose ML functionalities as services. With these frameworks, you can create APIs that allow users to input text and receive real-time outputs such as summaries and questions generated by the model. Flask is known for its simplicity and flexibility, while FastAPI offers additional benefits such as automatic data validation, asynchronous support, and high performance, making it ideal for building scalable and efficient APIs. These frameworks are particularly useful when you want to deploy your ML project as a service that other applications or systems can interact with.
- Streamlit, on the other hand, focuses on providing an intuitive and efficient way to build user interfaces for ML models. It is a lightweight framework designed to create interactive data applications with minimal coding effort. Streamlit enables you to quickly build a user-friendly interface where users can input text and view the generated summaries or questions in real-time. This makes it an excellent choice for projects that require a straightforward and fast-to-deploy interface for end-user interaction.

6.1.7 Fine-Tuning and Optimization

- Hugging Face Transformers is a powerful and versatile library that has revolutionized the way developers and researchers use transformer-based models in natural language processing (NLP) tasks. The library provides an extensive collection of pre-trained models, such as T5, BERT, GPT-2, and many others, covering a wide range of use cases, including question generation, summarization, sentiment analysis, text classification, and translation. These pre-trained models serve as a robust foundation for various applications, significantly reducing the time and computational resources required for training models from scratch.
- One of the standout features of Hugging Face is its highly intuitive and user-friendly API, which simplifies the fine-tuning process. Fine-tuning involves taking a general-purpose pre-trained model and customizing it on specific datasets to address domain-specific problems. This process allows developers to adapt state-of-the-art models to unique requirements with minimal effort. Hugging Face supports seamless integration with popular deep learning frameworks like PyTorch and TensorFlow, making it accessible to a wide audience of developers with varying preferences.
- Moreover, the library offers tools for tokenization, training, and evaluation, ensuring that every step of the NLP pipeline is covered. It also includes functionalities for distributed training, mixed-precision training, and export to production-ready formats like ONNX. The community-driven nature of Hugging Face, combined with its extensive documentation and active forums, ensures that users have access to continuous support and updates.
- In essence, Hugging Face Transformers is not just a tool for fine-tuning transformer models but a complete ecosystem for developing cutting-edge NLP solutions. Its ability to bridge the gap between state-of-the-art research and practical application makes it indispensable for anyone looking to leverage the power of transformers for real-world tasks.

CHAPTER 7

7.1 RESULTS AND DISCUSSION

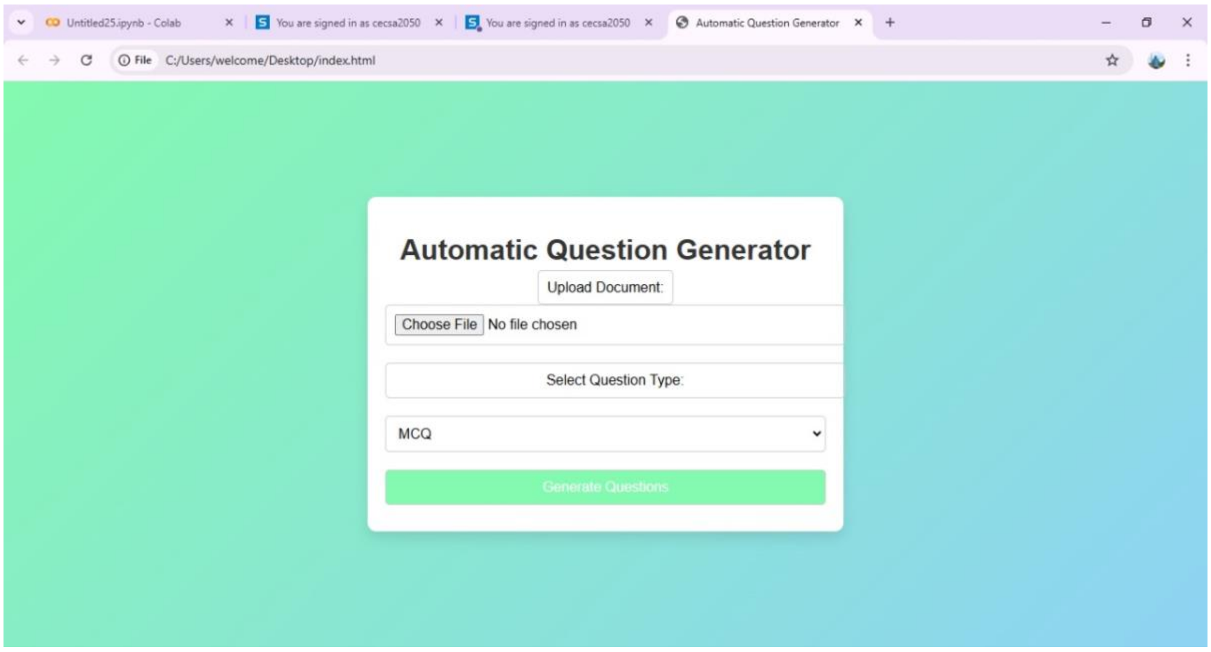

Untitled25.ipynb - Colab
You are signed in as cecsa2050
You are signed in as cecsa2050
Automatic Question Generator
C:/Users/welcome/Desktop/index.html
Automatic Question Generator
Upload Document:
Choose File
No file chosen
Select Question Type:
MCQ
Generate Questions

FIG:7.1 HOME WEB PAGE INTERFACE

Figure 7.1 illustrates the web interface for an "Automatic Question Generator" application. The interface features an option for users to upload a document by selecting a file from their device. Users can choose the type of question they wish to generate, with "MCQ" (Multiple Choice Questions) set as the default option in a dropdown menu. The dropdown menu allows users to explore other question types as needed. At the bottom of the interface, there is a prominently displayed "Generate Questions" button. Clicking this button triggers the question generation process. The layout is designed for simplicity, ensuring ease of use and efficient interaction. The overall goal is to provide a seamless experience for generating questions from user-uploaded documents.

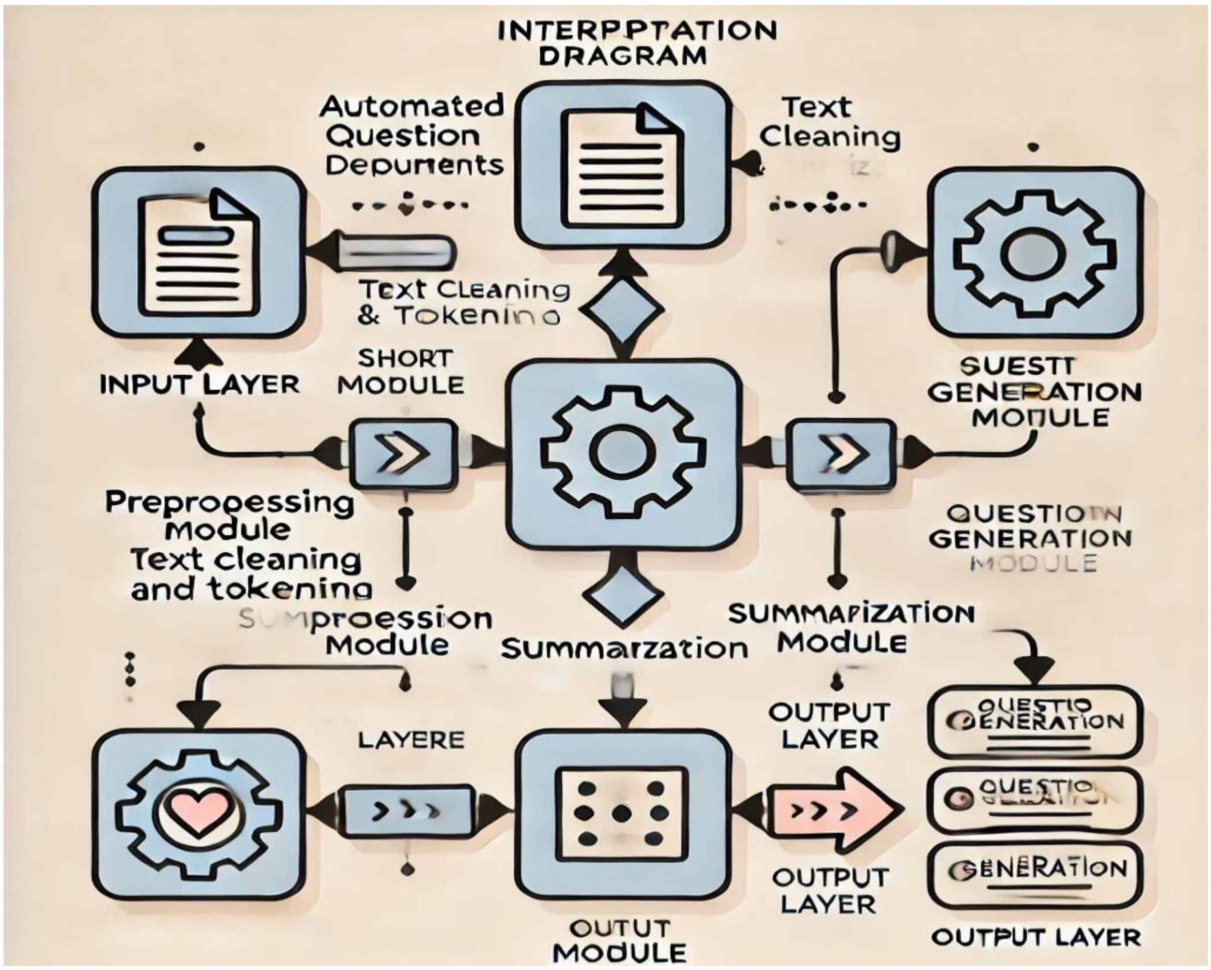


FIG:7.2 INTERPRETATION DIAGRAM

Figure 7.2 illustrates the flow of an Automated Question Generator and Summarizer system. The process starts with the Input Layer, where a user uploads a document. The document is then passed to the Preprocessing Module for cleaning and tokenization. The cleaned text moves to the Summarization Module, which generates a concise summary. This summary is passed to the Question Generation Module, where relevant questions, such as MCQs or short-answer questions, are created. Finally, in the Output Layer, the summary and generated questions are displayed for user review. The diagram shows the system's seamless flow from document upload to output.

CHAPTER 8

CONCLUSION & FUTURE ENHANCEMENTS

8.1 CONCLUSION

In conclusion, the development of an Automated Question Generator and Summarizer using Machine Learning has the potential to revolutionize the way we process and interact with large volumes of text data. By leveraging advanced NLP models such as T5, BERT, and GPT-3, the system can efficiently generate concise summaries and relevant, context-aware questions. This technology not only enhances information retrieval and comprehension but also offers a scalable solution for applications in education, content management, and more. While challenges such as model accuracy and text ambiguity remain, the integration of machine learning and natural language processing provides a promising foundation for further improvements and innovation in automated content understanding**.**

8.2 FUTURE ENHANCEMENTS

Future enhancements for the Automated Question Generator and Summarizer could focus on improving the model’s ability to handle longer and more complex documents, allowing for better contextual understanding. Incorporating multi-language support would expand the system’s usability across different languages and regions. Further fine-tuning of models to generate more domain-specific questions and summaries could improve accuracy for specialized fields. Real-time, interactive summarization could be explored, allowing users to input data and receive outputs instantly. Finally, integrating user feedback mechanisms could help continuously improve the system’s performance and adapt to evolving language trends.

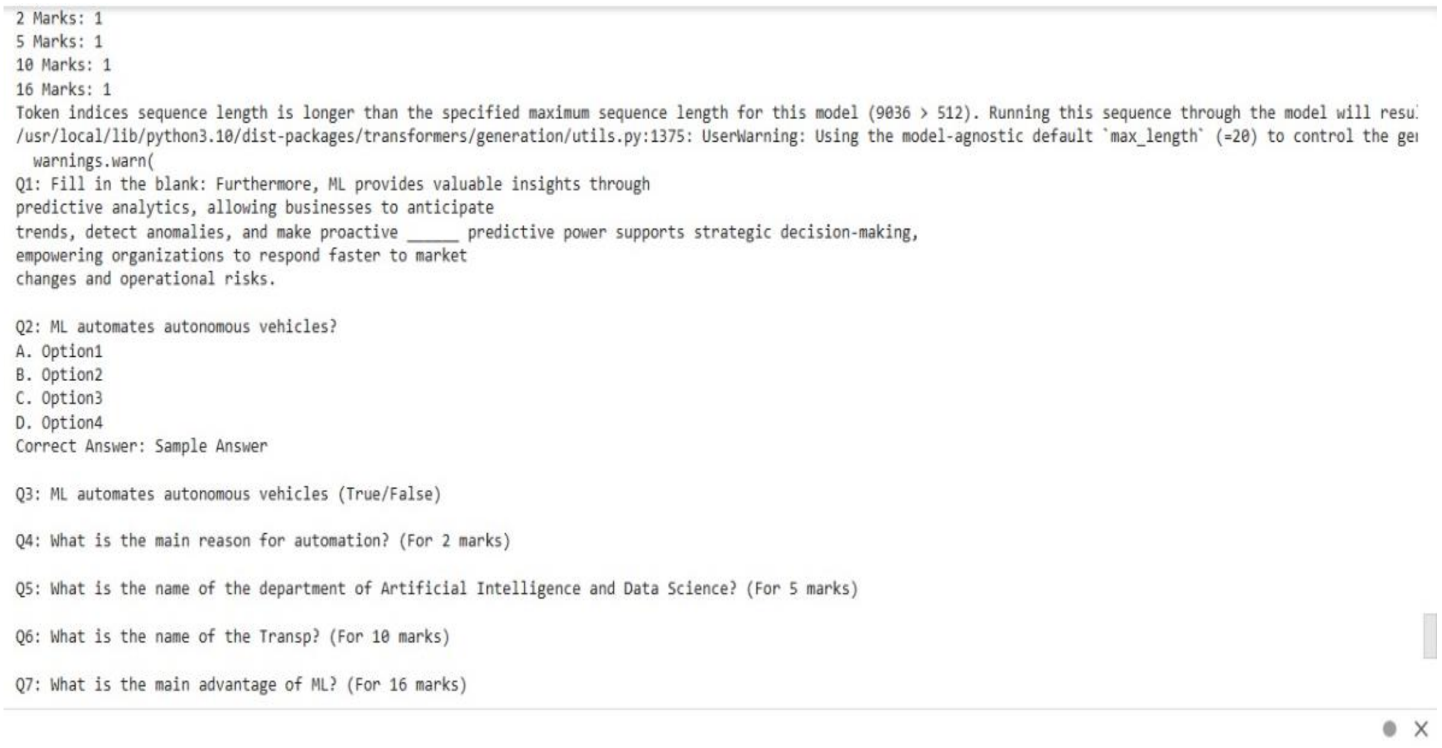
APPENDIX 1.SAMPLE CODE

```
!pip install transformers torch nltk
import nltk
nltk.download('punkt')
from transformers import T5ForConditionalGeneration, T5Tokenizer
import torch
import random
from nltk.tokenize import sent_tokenize
model_name = "t5-small"  # For more accuracy, use "t5-large" or a fine-tuned model on QA
tokenizer = T5Tokenizer.from_pretrained(model_name)
model = T5ForConditionalGeneration.from_pretrained(model_name)
def generate_ai_question(text, question_type="fill_in_the_blanks", num_questions=5):
sentences = sent_tokenize(text)
questions = []
for i in range(num_questions):
sentence = random.choice(sentences)
input_text = f"generate {question_type} question: {sentence}"
input_ids = tokenizer.encode(input_text, return_tensors="pt")
outputs = model.generate(input_ids, max_length=50, num_beams=5, early_stopping=True)
question = tokenizer.decode(outputs[0], skip_special_tokens=True)
questions.append(question)
return questions
text = "Enter some sample text from a PDF or Word document here for question generation."
questions = generate_ai_question(text, question_type="multiple choice", num_questions=3)
for i, question in enumerate(questions, 1):
```

```
print(f"Q{i}: {question}")
with open("ai_generated_questions.txt", "w") as file:
for question in questions:
file.write(f"{question}\n\n")
print("Questions saved to ai_generated_questions.txt")
from sentence_transformers import SentenceTransformer, util
embedder = SentenceTransformer('all-MiniLM-L6-v2')
def create_multiple_choice(sentence, correct_answer):
embeddings = embedder.encode([correct_answer] + sentence.split(),
convert_to_tensor=True)
similar_words = util.semantic_search(embeddings[0:1], embeddings[1:],
top_k=3)
distractors = [sentence.split()[idx['corpus_id']] for idx in similar_words[0]]
options = [correct_answer] + distractors
random.shuffle(options)
return options
def generate_ai_question(text, question_type, num_questions):
generator = pipeline("text2text-generation", model="valhalla/t5-small-qa-qg-
hl")
questions = []
sentences = nltk.sent_tokenize(text)
for i in range(num_questions):
if question_type == 'fill_in_the_blanks':
sentence = random.choice(sentences)
word_to_remove = random.choice(sentence.split())
blanked_sentence = sentence.replace(word_to_remove, "")
questions.append(f"Fill in the blank: {blanked_sentence}")
elif question_type == 'multiple_choice':
question = generator(f"generate question: {text}")[0]['generated_text']
```

```
question_text = f"{question}?\nA. Option1\nB. Option2\nC. Option3\nD. Option4\nCorrect Answer: Sample Answer"

questions.append(question_text)

elif question_type == 'true_false':

question = generator(f"generate question: {text}")[0]['generated_text']

questions.append(f"{question} (True/False)")

elif question_type in ['2_marks', '5_marks', '10_marks', '16_marks']:

random_sentence = random.choice(sentences)

question = generator(f"generate question: {random_sentence}")[0]['generated_text']

questions.append(f"{question} (For {question_type.replace('_marks', '')} marks)")

return questions

def main():

from google.colab import files

uploaded = files.upload()

file_name = list(uploaded.keys())[0]

text = load_text(open(file_name, "rb"))

print("Choose question type:")

print("1. Fill in the Blanks\n2. Multiple Choice\n3. True/False\n4. 2 Marks\n5. 5 Marks\n6. 10 Marks\n7. 16 Marks")

question_type = int(input("Enter choice (1-7): "))

num_questions = int(input("Enter the number of questions: "))

type_map = {1: 'fill_in_the_blanks', 2: 'multiple_choice', 3: 'true_false', 4: '2_marks', 5: '5_marks', 6: '10_marks', 7: '16_marks'}

chosen_type = type_map.get(question_type, 'fill_in_the_blanks')

questions = generate_ai_question(text, chosen_type, num_questions)

for i, question in enumerate(questions, 1):

print(f"Q{i}: {question}\n")
```

APPENDIX 2.SCREEN SHOT



```
2 Marks: 1
5 Marks: 1
10 Marks: 1
16 Marks: 1
Token indices sequence length is longer than the specified maximum sequence length for this model (9036 > 512). Running this sequence through the model will resu
/usr/local/lib/python3.10/dist-packages/transformers/generation/utils.py:1375: UserWarning: Using the model-agnostic default `max_length` (=20) to control the ge
  warnings.warn(
Q1: Fill in the blank: Furthermore, ML provides valuable insights through
predictive analytics, allowing businesses to anticipate
trends, detect anomalies, and make proactive ______ predictive power supports strategic decision-making,
empowering organizations to respond faster to market
changes and operational risks.

Q2: ML automates autonomous vehicles?
A. Option1
B. Option2
C. Option3
D. Option4
Correct Answer: Sample Answer

Q3: ML automates autonomous vehicles (True/False)

Q4: What is the main reason for automation? (For 2 marks)

Q5: What is the name of the department of Artificial Intelligence and Data Science? (For 5 marks)

Q6: What is the name of the Transp? (For 10 marks)

Q7: What is the main advantage of ML? (For 16 marks)
```

FIG:A.2.1 WORKFLOW DIAGRAM FOR AUTOMATED QUESTION GENERATION

REFERENCES

1.Bloom, B. S. (Ed.). (1956). Taxonomy of Educational Objectives: The Classification of Educational Goals. New York: Longmans, Green.

2.Heilman, M., and Smith, N. A. 2010. Good question! statistical ranking for question generation. 609–617. Los Angeles, California: Association for Computational Linguistics.

3.Kishore Papineni, Salim Roukos, Todd Ward, and WeiJing Zhu. (2002). Bleu: a method for automatic evaluation of machine translation. In ACL, pages 311–318. ACL.

4.Rey, G. A.; Celino, I.; Alexopoulos, P.; ´ Damljanovic, D.; Damova, M.; Li, N.; and Devedzic, V. 2012. Semi-automatic generation of quizzes and learning artifacts from linked data.

5.Rus, V. and Graesser, A.C. (2009). Workshop Report: The Question Generation Task and Evaluation Challenge, Institute for Intelligent Systems, Memphis, TN, ISBN: 978-0-615-27428-7Nan Duan, Duyu Tang, Peng Chen, Ming Zhou, Proceedings of the (2017) Conference on Empirical Methods in Natural Language Processing, pages 866–874.

6.Serban, Iulian & García-Durán, Alberto & Gulcehre, Caglar & Ahn, Sungjin & Chandar, Sarath & Courville, Aaron & Bengio, Y. (2016). Generating Factoid Questions with Recurrent Neural Networks: The 30M Factoid Question-Answer Corpus. 588-598. 10.18653/v1/P16-1056.

7.Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. (2018). Paragraph-level neural question generation with max-out pointer and gated self-attention networks.