A

Project Report

on

SmartQuiz: AI MCQ Generation

Submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology

by

Garlapati Akshay Reddy

(20EG105314)

Lakshmishetty Nikhil Shetty

(20EG105325)

Ramankol Srikanth

(20EG105343)

Under the guidance of

P Chakradhar

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
ANURAG UNIVERSITY
VENKATAPUR (V), GHATKESAR (M), MEDCHAL (D), T.S - 500088
TELANGANA
(2023-2024)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled **"SmartQuiz: AI MCQ Generation"** being submitted by **Garlapati Akshay Reddy, Lakshmishetty Nikhil Shetty and Ramankol Srikanth** bearing the Hall Ticket number **20EG105314 , 20EG105325, 20EG105343** in partial fulfillment of the requirements for the award of the degree of the **Bachelor of Technology** in **Computer Science and Engineering** to the **Anurag University** is a record of bonafide work carried out by them under my guidance and supervision from December 2023 to April 2024.

The results presented in this project have been verified and found to be satisfactory. The results embodied in this project report have not been submitted to any other University for the award of any other degree or diploma.

Internal Guide
P Chakradhar
Assistant Professor

External Examiner

Head of the Department

Department of Computer Science and Engineering

DECLARATION

We hereby declare that the project work entitled **"SmartQuiz: AI MCQ Generation"** submitted to the **Anurag University** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology (B. Tech)** in Computer Science and Engineering is a record of an original work done by us under the guidance of **Mr. P. Chakradhar, Assistant Professor** and this project work have not been submitted to any other university for the award of any other degree or diploma.

Garlapati Akshay Reddy

(20EG105314)

Lakshmishetty Nikhil Shetty

(20EG105325)

Ramankol Srikanth

(20EG105343)

ACKNOWLEDGEMENT

It is our privilege and pleasure to express profound sense of respect, gratitude and indebtedness to our guide **Mr. P. Chakradhar , Asst Professor**, Dept. of Computer Science and Engineering, Anurag University for her indefatigable inspiration, guidance, cogent discussion, constructive criticisms and encouragement throughout this dissertation work.

We extend our sincere thanks to **Dr. V. Vijaya Kumar**, **Professor &Dean,** School of Engineering, Anurag University, for his encouragement and constant help.

We would like to acknowledge our sincere gratitude for the support extended by **Dr. G. Vishnu Murthy**, Professor and Dean, Dept. of Computer Science and Engineering, Anurag University.

We also express our deep sense of gratitude to **Dr. V.V.S.S.S. Balaram**, Academic coordinator. **Dr. T. Shyam Prasad, Asst Professor** Project Co-Ordinator and Project review committee members, whose research expertise and commitment to the highest standards continuously motivated us during the crucial stage of our project work.

Garlapati Akshay Reddy

(20EG105314)

Lakshmishetty Nikhil Shetty

(20EG105325)

Ramankol Srikanth

(20EG105343)

Abstract

This dissertation/report presents a groundbreaking project aimed at addressing the growing need for versatile and efficient educational assessment tools. The project introduces a dynamic website capable of generating multiple-choice questions (MCQs) from a wide range of input texts, with a particular focus on the computer science domain. Central to this project is the development of a specialized dataset tailored specifically to the nuances and terminologies of the C programming language. This dataset serves as the cornerstone for fine-tuning a pre-existing question generation model, thereby customizing it to meet the unique demands of the field.

Utilizing advanced natural language processing (NLP) techniques, the website incorporates key methodologies such as text summarization using BERTSUM, keyword extraction facilitated by Python Keyphrase Extraction (PKE), efficient keyword mapping with FlashText, question generation employing T5, and contextually relevant distractor generation powered by Gemini API. This comprehensive approach enables the system to efficiently process input texts across diverse domains, offering a flexible and scalable solution for educators and learners alike.

The significance of this project lies in its contribution to educational technology, providing a robust framework for generating high-quality MCQs across various subject domains. The fine-tuning of the question generation model, coupled with the creation of a specialized dataset, underscores a commitment to advancing the field of educational assessment. Educators stand to benefit significantly from this tool, which enhances the assessment process in diverse educational contexts, thereby facilitating more effective and engaging learning experiences.

TABLE OF CONTENT

List of Tables

List of Figures

List of Abbreviations

Abbreviations	Full Form
MCQs	Multiple Choice Question
SQuAD	Stanford Question Answering Dataset
T5	Text-To-Text Transfer Transforme
BLEU	Bilingual Evaluation Understudy
NLP	Natural Language Processing
BertSum	Bidirectional Encoder Representations from Transformers Summarization
Api	Application programming interface
PKE	Python Keyphrase Extraction
LLM	Large Language Model

1. Introduction

1.1 Introduction to MCQ Generation Challenges

The process of creating high-quality multiple-choice questions (MCQs) for educational assessments is a complex and time-consuming task that often poses challenges for educators and content creators. Traditional methods rely heavily on manual effort, which can lead to inconsistencies in question quality, limited coverage of content, and inefficiency in adapting to evolving educational needs.

Consider a scenario where a computer science instructor needs to develop a set of MCQs for assessing students' understanding of C programming concepts. Using conventional methods, the instructor would need to manually review textbooks, lecture notes, and supplementary materials to extract relevant information, identify key concepts and keywords, formulate plausible distractors, and structure questions appropriately. This process not only demands significant time and effort but also runs the risk of overlooking crucial aspects of the curriculum or introducing biases inadvertently.

Such challenges highlight the pressing need for an automated and intelligent solution that can streamline the MCQ generation process while ensuring accuracy, relevance, and adaptability across diverse educational domains. This project aims to address these challenges by introducing a sophisticated website equipped with advanced natural language processing (NLP) techniques tailored specifically for generating high-quality MCQs in the computer science domain, with a focus on C programming concepts. Through innovative approaches such as text summarization, keyword extraction, and contextually relevant distractor generation, the system empowers educators and content creators to efficiently create assessment materials that align with learning objectives, promote critical thinking, and enhance overall learning outcomes.

1.2 Motivation

In this section, we discuss the overarching motivations that have prompted the development of this project. The landscape of modern education is rapidly evolving, driven by advancements in technology and changing pedagogical approaches.

However, traditional methods of crafting assessment materials, particularly multiple-choice questions (MCQs), have struggled to keep pace with these changes. Educators face significant challenges in creating MCQs that accurately assess students' understanding, align with learning objectives, and stimulate critical thinking.

The motivation behind this project stems from the recognition of these challenges and the pressing need for innovative solutions. By leveraging cutting-edge technologies such as natural language processing (NLP) and machine learning (ML), we aim to revolutionize the process of MCQ generation, making it more efficient, accurate, and aligned with the dynamic educational landscape.

1.3 Problem Definition

In this subsection, we define the specific challenges and limitations associated with traditional manual methods of creating MCQs. These challenges include:

1. **Time-Consuming Process:** Manual creation of MCQs involves extensive research, content analysis, and question formulation, leading to a time-intensive process for educators.
2. **Subjectivity and Bias:** Human-generated questions may inadvertently introduce biases or subjective interpretations, impacting the fairness and validity of assessments.
3. **Limited Scalability:** Scaling MCQ creation across diverse subjects and domains poses logistical challenges and may result in inconsistent question quality.
4. **Adaptability to Curriculum Changes:** Manual methods struggle to quickly adapt to changes in curricula or learning objectives, leading to outdated assessment materials.

1.4 Problem Illustration

To provide a concrete understanding of the challenges outlined in the previous subsection, we present a detailed illustration of a typical scenario faced by educators in manual MCQ creation:

Scenario: Imagine a dedicated computer science instructor tasked with developing MCQs for an introductory course on programming languages. The instructor begins by reviewing multiple textbooks, online resources, and lecture materials to identify key concepts and topics. Next, questions are crafted, taking into account learning objectives, difficulty levels, and relevance to course content. Plausible distractors are carefully selected to challenge students' understanding without leading to confusion or misinterpretation. This process, while crucial for effective assessment, demands extensive time and effort, often resulting in a limited pool of questions and potential biases in question design.

1.5 Objective of the Project

Building upon the identified challenges and the illustrated scenario, the primary objectives of this project are as follows:

1. **Automate MCQ Generation:** Develop a sophisticated system that automates the generation of high-quality MCQs based on input texts, leveraging NLP and ML techniques.

2. **Ensure Accuracy and Relevance:** Implement algorithms for text summarization, keyword extraction, and distractor generation to ensure the accuracy, relevance, and effectiveness of generated MCQs.

3. **Scalability and Adaptability:** Create a platform that scales seamlessly across diverse educational domains, accommodating changes in curricula and learning objectives with ease.

4. **Enhance Educator Efficiency:** Provide educators with a user-friendly interface and tools that streamline the MCQ creation process, saving time and effort while maintaining assessment quality.

5. **Improve Learning Outcomes:** Facilitate the creation of MCQs that promote critical thinking, deep understanding of subject matter, and alignment with educational objectives, ultimately enhancing learning outcomes for students.

2. Literature Survey

The comprehensive exploration of literature surrounding automated Multiple Choice Question (MCQ) generation provides a nuanced understanding of the evolving landscape where natural language processing (NLP), machine learning, and educational technology intersect.

In their pioneering work, Santhanavijayan et al[1]. ventured into the realm of automated MCQ generation by combining fireflies-based preference learning and an ontology-based approach. The utilization of ontology added a layer of semantic understanding to the process, contributing to question relevance. However, the implementation of ontology-based approaches introduced a degree of complexity, posing a challenge for wider adoption. The inclusion of analogy questions in their methodology showcased an innovative approach to assessing verbal ability. Nevertheless, the study noted limitations, particularly in the diversity of distractor generations, emphasizing the need for continued exploration in this dynamic field.

Hoshino and Nakagawa [2] took a machine learning-centric approach, employing Naive Bayes and K-Nearest Neighbors for real-time MCQ generation. Their study demonstrated the potential of machine learning algorithms in crafting questions dynamically. However, the research highlighted limitations, notably the focused scope on English grammar and vocabulary, which may restrict broader applications. Additionally, the dependence on user-provided HTML files for quiz creation indicated a reliance on specific inputs, emphasizing the need for scalability and adaptability in automated question generation systems.

Dr. Dhanya N M[3] advanced the field with an integration of sophisticated NLP techniques, utilizing the T5 transformer for summarization and Sense2Vec for distractor generation. The use of Sense2Vec marked a notable improvement in distractor quality, although the study acknowledged room for enhancement. The choice of the T5 transformer, while powerful in its capabilities, introduced challenges related to speed. Despite these hurdles, the study highlighted the potential of advanced NLP models in refining the automated MCQ generation process, shedding light on the

delicate balance between innovation and practical considerations in the quest for efficiency.

Sushmita Gangopadhyay et al. [4] presented a system for question and answer generation. Their approach included a focus generator module, utilizing the SQuAD dataset, and employing techniques such as the Neural Entity Selection Algorithm and Recurrent Neural Networks.

D. R. CH and S. K. Saha [5] has proposed “Automatic Multiple Choice Question Generation From Text: A Survey,” . In this paper, articles from the database are used to generate questions. NLP-based summarizer is used for text summarization and frequency count of words, and pattern matching techniques are used for key selection. For generating distractors, they have used the wordnet, pattern matching, domain ontology and semantic analysis.

The "EduQG: A Multi-Format Multiple-Choice Dataset for the Educational Domain" authored by Amir Hadifar, Semere Kiros Bitew, Chris Develder, Johannes Deleu, and Thomas Demeester[6] presents a comprehensive dataset tailored for educational question generation tasks. This dataset, named EduQG, encompasses multiple formats of high-quality multiple-choice questions (MCQs) curated specifically for educational contexts. With contributions from various authors including a Senior Member of IEEE, the dataset is designed to cover diverse subjects and topics relevant to educational assessments. Researchers and practitioners can leverage EduQG to train and evaluate natural language processing (NLP) models aimed at automating question generation processes in educational settings. The authors' expertise and collaboration ensure the dataset's reliability, relevance, and applicability in advancing NLP-driven solutions for educational question generation, making it a valuable resource for literature surveys in NLP and education-related research.

3. Proposed method

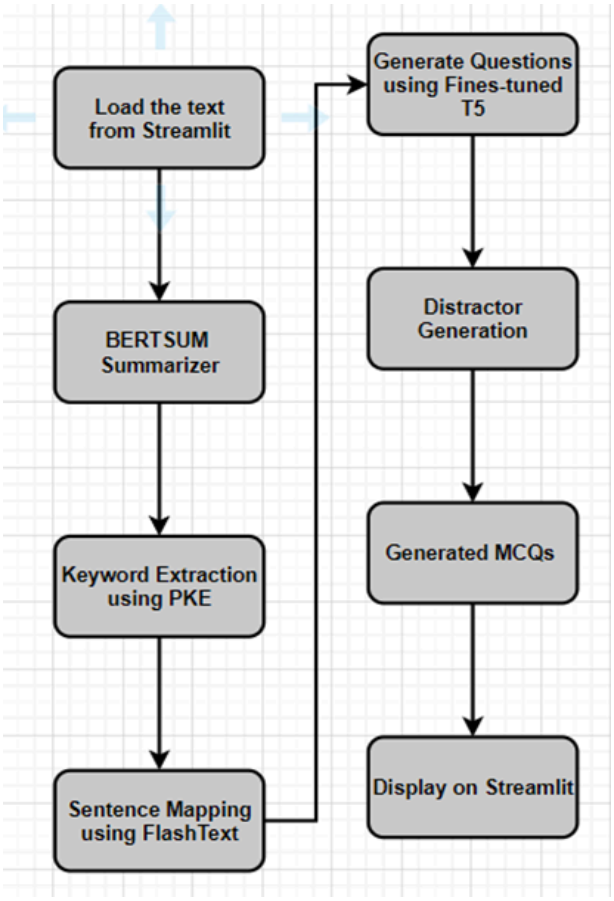

Figure 3.1 Proposed method

The proposed method for generating high-quality multiple-choice questions (MCQs) integrates advanced natural language processing (NLP) techniques and tools to streamline the text processing, keyword extraction, sentence extraction, distractor generation, and MCQ creation processes. It begins with text collection from diverse educational materials, followed by BERTSUM text summarization to distill essential information into concise summaries while maintaining contextual significance. The PKE algorithm with an unsupervised multipartite graph approach is then applied for precise keyword extraction, contributing to targeted MCQ generation aligned with educational objectives. FlashText library aids in contextually relevant sentence extraction, enriching the MCQ formulation process with meaningful questions and answer options. Distractors are generated using the Gemini API, leveraging semantic understanding to create plausible yet incorrect options that challenge learners' comprehension effectively. The final MCQs are displayed using Streamlit, offering a user-friendly interface for educators and learners to interact with the generated assessments seamlessly.

Overall, this method leverages cutting-edge NLP tools and algorithms such as BERTSUM, PKE, FlashText, Gemini API, and Streamlit to automate and enhance various stages of MCQ generation. By prioritizing semantic context, relevance, and cognitive engagement, the system ensures that the generated MCQs are rooted in a deep understanding of the original text, aligned with educational goals, and presented in an accessible format for effective learning and assessment experiences.

3.1 Text Processing and Analysis

In this section, we detail the crucial steps involved in text processing and analysis, which form the foundation for generating high-quality multiple-choice questions (MCQs) using advanced natural language processing (NLP) techniques.

3.1.1 Text Collection and Summarization using BERTSUM

Text Collection:

The process of text collection forms the foundational step in our methodology for generating high-quality multiple-choice questions (MCQs) using advanced natural language processing (NLP) techniques. We gather a diverse range of textual educational materials relevant to the subject domain, encompassing resources such as textbooks, academic papers, lecture notes, and curated online content repositories. The comprehensive nature of text collection ensures that our MCQ generation system is well-equipped with a robust knowledge base spanning key concepts, terminologies, and contextual information necessary for effective assessment.

BERTSUM Text Summarization:

Once the textual materials are collected, we employ the BERTSUM algorithm for text summarization, a pivotal process in distilling vast amounts of information into concise and informative summaries. BERTSUM stands as an extension of the BERT (Bidirectional Encoder Representations from Transformers) model, renowned for its prowess in understanding contextual nuances within textual data. BERTSUM leverages transformer-based architectures to perform abstractive summarization, which goes beyond mere extraction of sentences by generating summaries that capture essential information while maintaining coherence and readability.

1. **Understanding Contextual Significance:** BERTSUM excels in understanding the contextual significance of textual content, ensuring that the generated summaries retain key concepts, relationships, and critical information essential for MCQ formulation.
2. **Preserving Informational Integrity:** During the summarization process, BERTSUM prioritizes preserving the informational integrity of the original text, avoiding loss of crucial details while condensing the content into a more manageable form.
3. **Adapting to Varied Textual Styles:** Whether dealing with technical jargon, academic prose, or diverse writing styles, BERTSUM's adaptive capabilities enable it to generate summaries that encapsulate the essence of the input text while catering to different linguistic nuances.
4. **Enhancing Comprehensibility:** The summarized texts produced by BERTSUM strike a balance between brevity and comprehensibility, ensuring that essential concepts are conveyed effectively without overwhelming the reader with unnecessary details.
5. **Scalability and Efficiency:** BERTSUM's transformer-based architecture facilitates scalable and efficient summarization, making it suitable for processing large volumes of textual data encountered in educational contexts.
6. **Integration with MCQ Generation:** The summarized texts serve as pivotal inputs for subsequent stages in the MCQ generation pipeline, providing condensed yet comprehensive content that forms the basis for formulating meaningful questions and answer options.

By integrating BERTSUM's advanced summarization capabilities into our MCQ generation framework, we ensure that the generated MCQs are rooted in a deep understanding of the original textual content, fostering accuracy, relevance, and alignment with educational objectives.

3.1.2 Keyword Extraction using PKE Algorithm

Keyword extraction plays a crucial role in distilling essential concepts and key terms from textual data, facilitating the generation of high-quality multiple-choice questions (MCQs) aligned with educational objectives. In this section, we explore the

utilization of the PKE (Python Keyphrase Extraction) algorithm employing an advanced unsupervised multipartite graph approach for precise keyword extraction.

Overview of PKE Algorithm with Unsupervised Multipartite Graph

1. **Graph-Based Representation:** The unsupervised multipartite graph approach leverages graph-based representations of textual data, where nodes represent words or phrases, and edges signify semantic or co-occurrence relationships between them. This graph structure captures the inherent connections and contextual relevance among terms within the text corpus.
2. **Multi-Layered Graph Construction:** The algorithm constructs a multi-layered graph, integrating different linguistic features such as syntax, semantics, and document structure. Nodes in the graph correspond to terms extracted from the text, while edges reflect semantic associations based on co-occurrence patterns, syntactic dependencies, and topical relevance.
3. **Graph Partitioning and Keyphrase Identification:** Through sophisticated graph partitioning algorithms, the multipartite graph is segmented into cohesive clusters or communities of related terms. These clusters represent cohesive semantic units or topics within the text corpus. Keyphrases are then identified based on centrality metrics, such as degree centrality, betweenness centrality, or PageRank, within these graph clusters.

Key Steps in Unsupervised Multipartite Graph Keyword Extraction

1. **Preprocessing and Tokenization:** The text undergoes preprocessing steps such as tokenization, lowercasing, and removal of stop words and punctuation marks to prepare it for graph-based analysis.
2. **Graph Construction:** The algorithm constructs the multipartite graph by encoding linguistic features and relationships between terms, considering factors like co-occurrence patterns, semantic similarity, and syntactic dependencies.
3. **Graph Partitioning:** Utilizing advanced graph partitioning techniques, the algorithm identifies cohesive clusters or communities of terms representing coherent semantic units or topics within the text.
4. **Centrality-Based Keyphrase Identification:** Within each graph cluster, keyphrases are identified based on centrality metrics computed on the

multipartite graph. These metrics evaluate the importance and centrality of terms within their respective semantic contexts.

5. **Ranking and Selection:** Extracted keyphrases are ranked based on their centrality scores or other relevance metrics, ensuring the selection of meaningful and representative keywords that encapsulate critical concepts and themes in the text.

Advantages of Unsupervised Multipartite Graph Approach

1. **Semantic Context Preservation:** The graph-based approach preserves semantic context and relationships among terms, capturing nuanced associations that traditional methods may overlook.
2. **Multi-Faceted Analysis:** By integrating multiple linguistic features and structural information, the algorithm conducts a comprehensive analysis of textual data, yielding more accurate and contextually relevant keyword extractions.
3. **Scalability and Adaptability:** The unsupervised nature of the approach allows it to scale efficiently to large text corpora across diverse domains, adapting to varying linguistic styles and content structures.
4. **Domain-Agnostic Application:** The methodology is domain-agnostic, making it applicable to a wide range of subject areas and text types without requiring domain-specific training data or annotations.

By leveraging the power of the PKE algorithm with the unsupervised multipartite graph approach, we ensure precise and contextually relevant keyword extraction, laying a robust foundation for generating targeted MCQs reflective of the core concepts and themes present in the text corpus.

3.2 Sentence Extraction with FlashText Library

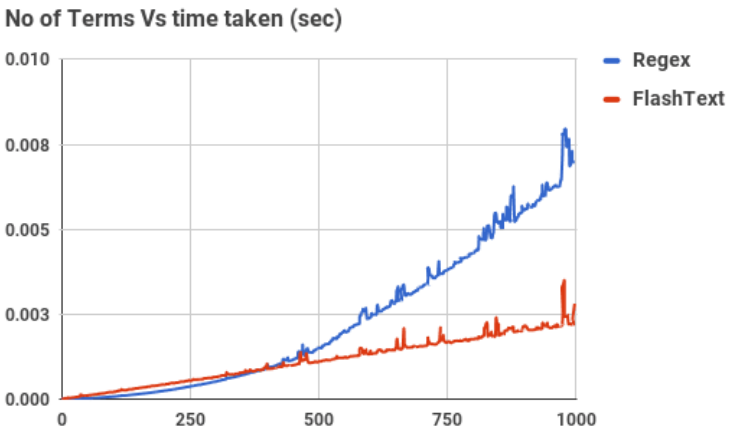

Figure 3.2 Time taken for Flashtext vs regex

In the process of generating high-quality multiple-choice questions (MCQs) from textual data, extracting contextually relevant sentences plays a pivotal role in formulating meaningful questions and answer options. The utilization of the FlashText library facilitates efficient and accurate extraction of sentences containing key concepts and keywords, enriching the MCQ generation process.

Understanding FlashText Library

FlashText is a powerful Python library designed for lightning-fast keyword matching and extraction within text data. Unlike traditional regular expression-based methods, FlashText leverages efficient data structures and algorithms, such as Trie trees, to enable rapid and scalable keyword search and extraction operations.

Key Steps in Sentence Extraction using FlashText

1. **Keyword Identification:** Prior to sentence extraction, relevant keywords and key concepts are identified through preceding stages such as text summarization and keyword extraction using NLP techniques like BERTSUM and the PKE algorithm. These keywords serve as crucial anchors for extracting contextually rich sentences.
2. **Integration with FlashText:** The identified keywords are integrated into the FlashText library, creating a custom keyword dictionary tailored to the specific

domain or subject area of interest. This dictionary serves as a reference for identifying sentences containing the targeted keywords.

3. **Efficient Keyword Matching:** FlashText's optimized keyword matching algorithms swiftly traverse the text corpus, identifying sentences that match or contain the specified keywords from the custom dictionary. The library's performance optimizations ensure rapid processing even with large volumes of text data, making it suitable for scalable applications.
4. **Contextual Sentence Retrieval:** Extracted sentences not only contain the targeted keywords but also maintain contextual relevance within the original text. This contextual extraction ensures that the generated MCQs align closely with the core concepts and thematic content encapsulated by the identified keywords.

Advantages of FlashText-Based Sentence Extraction

- **Speed and Efficiency:** FlashText's highly optimized algorithms and data structures enable lightning-fast keyword matching and sentence extraction operations, enhancing overall processing speed and efficiency.
- **Accuracy and Precision:** By focusing on keyword-based extraction, FlashText ensures that extracted sentences are contextually relevant and aligned with the specified keywords, reducing noise and irrelevant content in the extracted data.
- **Scalability:** The library's scalability allows for seamless integration into workflows dealing with large text corpora, making it well-suited for educational content processing and MCQ generation across diverse subject domains.

The extracted sentences, enriched with context and relevance to key concepts, serve as valuable inputs for subsequent stages in the MCQ generation workflow. These sentences form the basis for formulating clear and contextually meaningful questions, along with relevant answer options and distractors, contributing to the creation of effective educational assessment .

3.3 Distractor Generation Using the Gemini API

Harnessing Gemini API for Contextually Relevant Distractors

In educational assessments, crafting plausible distractors for multiple-choice questions (MCQs) is crucial to gauge learners' comprehension and critical thinking. The Gemini API, powered by advanced language models, offers a conceptual framework for automating the generation of contextually relevant distractors, enriching the assessment process.

Conceptual Framework:

- **Semantic Context:** The Gemini API's strength lies in its ability to understand and process semantic contexts within provided prompts. By presenting relevant information such as the target word (answer), associated question, and contextual cues, the API can infer nuanced relationships crucial for generating appropriate distractors.
- **Language Understanding:** Leveraging large language models (LLMs) underlying the Gemini API, the system can interpret nuances in language, identify common misconceptions, and generate distractors that align with the thematic content of questions. This semantic understanding forms the cornerstone of generating meaningful distractors.
- **Plausible Distractors:** The generated distractors aim to mimic common errors, related concepts, or plausible but incorrect responses that learners might consider. These distractors, when well-crafted and contextually relevant, effectively challenge learners' understanding and enhance the discriminative power of MCQs.

Integrating the Gemini API for distractor generation aligns with modern assessment practices focused on accuracy, relevance, and cognitive engagement. This conceptual framework underscores the importance of leveraging advanced language technologies to augment educational assessment methodologies effectively.

3.4 Multiple-Choice Question (MCQ) Generation

Automating the generation of Multiple-Choice Questions (MCQs) streamlines the assessment creation process while ensuring the quality and relevance of questions, correct answers, and distractor options. Integrating Streamlit for the final display

provides a user-friendly interface for educators and learners to interact with the generated MCQs effectively.

3.4.1 Combining Questions, Correct Answers, and Distractor Options

Data Fusion for MCQ Creation

- **Question and Answer Pairing:** The automated system combines questions derived from input texts or contexts with corresponding correct answers obtained through question-answering models or domain-specific datasets.
- **Distractor Integration:** Plausible distractors, generated using advanced natural language processing (NLP) techniques or APIs like Gemini, are seamlessly integrated with correct answers to form complete MCQs.

3.4.2 Final MCQ Display and Format

Streamlit Interface for User Interaction

- **Dynamic Display:** Utilizing Streamlit's interactive features, the system dynamically presents generated MCQs in a clear and organized manner, enhancing user engagement and comprehension.
- **Format Customization:** The Streamlit interface allows for customizable formatting of MCQs, including question presentation, answer options layout, and overall visual appeal.

4. Implementation

4.1 List of Program Files:

File Name	Functionality
main.py	Executes the overall process of generating MCQs from input text, including user interface handling.
summarizing.py	Provides functionality to summarize text using the BERTSUM model.
keyword_extractor.py	Implements keyword extraction from text using the MultipartiteRank algorithm.
sentence_processor.py	Manages the extraction and filtering of sentences based on keywords for context-rich MCQ generation.
distractor_generator.py	Generates distractors for MCQ options using Gemini API and Sense2Vec model.
mcq_generator.py	Handles the generation of MCQs (multiple-choice questions) from extracted sentences.

Table 4.1 List of Program Files

The above table 4.1 summarizes the key functionalities of each Python file in the MCQ generation system, outlining their roles in text processing, summarization, keyword extraction, sentence processing, distractor generation, and MCQ creation.

4.1.1 main.py file:

➔ **Functionalities:**

- **User Interface Handling:** This segment utilizes Streamlit to create a user-friendly web interface. It integrates a text input field for users to provide input text. Upon clicking the "Generate MCQs" button, the system triggers the

process of summarization, keyword extraction, sentence processing, model loading, MCQ generation, and user interface update.

- **Text Processing:** The system first summarizes the input text using BERTSUM to distill essential information. It then extracts keywords from the original text to identify critical terms and concepts. These common keywords form a bridge between the original and summarized texts, ensuring that generated MCQs are contextually relevant.
- **Sentence Processing:** This functionality filters and extracts sentences containing common keywords. This step refines the context for generating meaningful MCQs by focusing on sentences crucial to understanding the key concepts in the input text.
- **Model Loading:** The system loads the T5 model and tokenizer from Hugging Face's Transformers library. These components are essential for generating questions and answers accurately based on the context provided by the extracted sentences.
- **MCQ Generation:** The core functionality involves generating multiple-choice questions (MCQs) from the filtered and context-rich sentences. It predicts the correct answers and generates distractors using the Sense2Vec model to provide meaningful options to users.
- **User Interface Update:** After generating MCQs, the system updates the web interface to display the generated questions along with options and correct answers. This interactive display enables users to engage with and assess the quality of the generated MCQs effectively.

➔ **Attributes:**

The table 4.2 details the attributes used in main.py for MCQ generation, from input text handling to summarization, keyword extraction, sentence processing, and final MCQ creation with user interaction via a UI.

Attribute	Description
original_text	User-provided input text for MCQ generation.
summary	Summarized version of original_text using BERTSUM.

keywords_original	Keywords extracted from original_text for concept identification.
keywords_ summarized	Keywords extracted from summary for question relevance.
common_keywords	Intersection of keywords between keywords_original and keywords_summarized.
extracted _sentences	Sentences extracted based on common keywords for context-rich MCQ's.
fs	Filtered sentences after batch processing for efficient sentences for MCQ's.
model, tokenizer	Loaded T5 model and tokenizer for question and answer generation model.
quiz_questions	List of dictionaries with MCQ details (context, question, options,answer) for user interaction in the UI.

Table 4.2 Attributes Involved in main.py

4.1.2 summarizing.py

The summarize_text function in "**summarizing.py**" utilizes the BERTSUM algorithm through the Summarizer class to summarize input text effectively. Here's a detailed breakdown of its functionalities:

1. **Text Chunking:**It first divides the input text into chunks of sentences, with each chunk containing a maximum of max_chunk_sentences sentences. This step helps manage large texts for efficient processing.
2. **Summarization Process:**
   Initializes the BERTSUM summarization model (Summarizer) using the specified model_name (default: "bert-base-uncased").Iterates through each chunk of text and generates a summary for each chunk using the BERTSUM model.Combines the individual summaries of chunks into summary_chunks.

3. **Combining Summaries:**

   Finally, it concatenates all the summary_chunks into a single combined_summary, which represents the summarized version of the input text.

4. **Key Functionality:**

   BERTSUM, based on BERT (Bidirectional Encoder Representations from Transformers), leverages transformer models for extractive summarization. It identifies important sentences in the text to create a concise summary preserving key information.

➔ **Attributes:**

This below table 4.3 elaborates on the attributes involved in the summarizing process using the BERTSUM model, including text handling, model initialization, chunking for efficient processing, and storing generated summaries.

Attribute	Description
Summarizer	Initializes the BERTSUM summarization model for text summarization.
text	Represents the input text to be summarized.
model_name	Specifies the BERTSUM model to use (default: "bert-base-uncased").
max_chunk_sentences	Defines the maximum number of sentences per chunk during text chunking for summarization.
sentences	Splits the input text into individual sentences for processing.
chunks	Stores chunks of sentences based on max_chunk_sentences for efficient summarization.
chunk	Temporarily accumulates sentences until reaching max_chunk_sentences before adding to chunks.

summary_chunks	Stores individual summaries generated for each chunk of sentences using the BERTSUM model.
combined_summary	Represents the final summarized version of the input text after combining all summary_chunks.

Table 4.3 Attributes involved in summarizing.py

4.1.3 keyword_extractor.py

It plays a crucial role in identifying important topics or concepts within the text, facilitating deeper text understanding and subsequent processing tasks.

1. **Import pke Library:**

   The file starts by incorporating the pke library, specialized for keyphrase extraction tasks. This library offers tools and algorithms tailored to analyze text and identify key concepts or terms.

2. "**extract_keywords**" function:
   - **Input Handling:** Receives a piece of text (text) as input, which serves as the source for extracting keyphrases.
   - **Keyphrase Extraction:**
     - **Model Initialization:** Sets up a keyphrase extraction model, specifically the "MultipartiteRank" algorithm from pke, designed to rank keyphrases based on their importance within the text.
     - **Text Processing:** Loads the input text into the keyphrase extractor and specifies the language (English in this case) for analysis.
     - **Ranking Keyphrases:** Utilizes the MultipartiteRank algorithm to select and rank keyphrases according to their relevance and significance in the text.
     - **Output Generation:** Produces a list of keyphrases, typically the top most relevant keyphrases based on the ranking criteria.
3. **Output Handling:** Returns the extracted keyphrases as a list, enabling further analysis or integration into various natural language processing (NLP) tasks such as content summarization, information retrieval, or question generation.

➔ **Attributes:**

This table 4.4 details the key functions and attributes related to keyphrase extraction using the MultipartiteRank algorithm and the pke library.

Attribute	Description
import pke	Imports the pke library, specializing in keyphrase extraction tasks.
extractor	Instance of the MultipartiteRank keyphrase extraction model from pke.
extractor.candidate_selection()	Performs candidate selection for keyphrase extraction using the MultipartiteRank algorithm.
extractor.candidate_weighting()	Ranks and weights the selected keyphrase candidates based on relevance and importance.

Table 4.4 Attributes involved in keyword_extractor.py

4.1.4 sentence_processor.py

➔ **Functionalities:**

1. **filter_sentences:**
    - Purpose: Imagine you're teaching someone and need to create challenging options for a question. This function simulates that process.
    - **Workflow:**
        - API Key Configuration: This is like setting up access to a powerful LLM(Gemini).
        - Prompt Construction: You're asking the generator to create sentences that fit specific criteria.
        - Response Processing: You get back a set of sentences, and this function cleans them up for use as options.

2. **extract_sentences_with_keywords:**
    - **Purpose:** When studying, you highlight crucial concepts. This function finds sentences where those highlighted terms appear.
    - **Workflow:**
        - Keyword Matching: It's like searching for highlighted terms in a textbook.
        - Sentence Tokenization: Breaking the text into sentences is like isolating each idea for examination.
        - Keyword Filtering: You're sifting through sentences, keeping only those relevant to your highlighted terms.
        - Results Compilation: Finally, you gather these 'important' sentences for further study or analysis.

➔ **Attributes:**

The table 4.5 summarizes essential attributes for Gemini distractor generation and keyword extraction, pivotal for MCQ creation. These attributes are integral to ensuring contextually relevant distractors and informative keyword identification in the text processing pipeline..

Attribute	Description
google_api_key	API key for accessing external services, necessary for the Gemini model interaction.
model	Represents the Gemini model used for generating distractors.
filter_list	Contains the cleaned and formatted list of sentences generated by the Gemini model.
keywords	List of keywords indicating relevant concepts in the text.
keyword_processor	Instance of KeywordProcessor used for keyword extraction and matching.
keyword_sentences	List of sentences containing the specified keywords.

Table 4.5 Distractor Generation Attributes

4.1.5 mcq_generator.py

This process involves advanced natural language processing techniques, including tokenization, model inference, and decoding, to produce linguistically appropriate questions and answers aligned with the provided context. The model's training data and architecture significantly influence the quality and relevance of the generated content.

➔ **Functionalities:**

1. **Contextual Question-Answer Generation:**
    - **Objective:** The function aims to create contextually relevant questions and answers automatically.
    - **Inputs:**
        - context: Textual information or content from which questions and answers are derived.
        - tokenizer: Tokenization is the process of converting text into a format suitable for machine learning models, ensuring compatibility with the model architecture.
        - model: Refers to a fine tuned T5 model designed for question and answer generation.
    - **Workflow:**
        - **Tokenization:** Utilizes the provided tokenizer to break down the context into tokens, aligning them with the model's understanding.
        - **Model Processing:** Feeds the tokenized input into the model for text generation. The model employs sophisticated algorithms, such as Transformers, to understand context and generate meaningful responses.
        - **Decoding:** Transforms the model's output (tensor representation) back into human-readable text. This involves handling special tokens and formatting the text appropriately.
    - **Outputs:** Delivers a pair of strings: a generated question and its corresponding answer, extracted from the model's output.

➔ **Attributes:**

The table 4.6 outlines key attributes in mcqgenerator.py crucial for MCQ generation, including tokenization, model usage, input preparation, and output decoding for question-answer pairs.

Attribute	Description
tokenizer	Tokenization tool that converts the input context into a format suitable for model processing.
model	Fine tuned T5 model capable of generating text based on contextual understanding.
inputs	Tokenized input data formatted for model processing, including contextual information.
outputs	Model-generated output containing the question-answer pair in tensor format.
question_answer	Decoded text from model output, representing the generated question and its corresponding answer.

Table 4.6 Attributes involved in mcq_generator.py

4.1.6 distractor_generator.py:

➔ **Functionality:**

The code in distractor_generator.py focuses on generating plausible distractors for multiple-choice questions (MCQs) related to a given context, answer, and question. It utilizes the Gemini model, integrated through an API, to generate these distractors. The generate_distractors_with_gemini function constructs a prompt using the provided context, answer, and question and communicates with the Gemini model to obtain distractors. These distractors are then processed and returned as a list, providing additional options for MCQs beyond the correct answer.

The get_options function acts as a higher-level function that orchestrates the distractor generation process. It first attempts to generate distractors using Gemini via

the generate_distractors_with_gemini function. If successful, it returns the generated distractors along with the method used (Gemini) for transparency. If an error occurs or no distractors are generated, it returns an empty list of distractors and indicates the method as "None." Overall, these functionalities contribute to enriching the question options in MCQs, enhancing the overall quality and diversity of assessment questions generated by the system.

➔ **Attributes:**

The table 4.7 outlines key attributes involved in the distractor generation process within the distractor_generator.py file. These attributes play a critical role in interacting with the Gemini model API, processing prompts, and obtaining relevant distractors for multiple-choice questions (MCQs) in educational assessments.

Attribute Name	Description
GOOGLE_API_KEY	API key used for authentication to access the Gemini model API for distractor generation.
model	Instance of the Gemini model (generative AI model) used for generating distractors.
chat	Chat instance initialized with the Gemini model for communication and message exchange.
prompt	Constructed prompt containing context, answer, and question information for distractor generation.
response	Response received from the Gemini model after processing the prompt.
distractors_list_truncated	List of generated distractors, truncated and formatted for use as answer options in MCQs.
distractors	List of generated distractors (options for multiple-choice questions) for use in MCQs.

Table 4.7 Attributes involved in distractor_generator.py

4.2 Datasets:

4.2.1 Squad Dataset:

Fine-tuning a T5 model on the SQuAD (Stanford Question Answering Dataset) can significantly enhance the model's capabilities in question and answer generation tasks. SQuAD is a well-known dataset designed specifically for question answering, containing real questions posed by humans on a diverse range of Wikipedia articles. Each question is associated with a corresponding passage from the article and the correct answer span within that passage. By training the T5 model on this dataset, the model learns to understand context, identify relevant information within passages, and generate accurate answers to questions based on that context.

The advantages of using SQuAD for fine-tuning T5 are multifold. First, the dataset covers a wide array of topics, ensuring that the model gains knowledge across various domains and subject matters. This broad exposure helps the model generalize better to unseen data during inference, making it more adaptable and capable of handling diverse input scenarios. Additionally, since SQuAD questions are sourced from real users, they reflect natural language nuances, complexities, and ambiguities, challenging the model to grasp subtle contextual cues and produce human-like responses.

Training on SQuAD also imparts the model with advanced language understanding and reasoning abilities. The model learns to not only extract relevant information from passages but also to synthesize that information into coherent and contextually appropriate answers. This training fosters the model's comprehension skills, enabling it to tackle complex questions that require deeper understanding and reasoning capabilities. As a result, when deployed for question generation or question answering tasks in practical applications, the fine-tuned T5 model is poised to deliver accurate, relevant, and contextually rich outputs, earning accolades for its performance and versatility in NLP tasks.

4.2.2 EduQG dataset:

Fine-tuning a T5 model on the EduQG dataset can greatly enhance its ability to generate high-quality questions and answers. The EduQG dataset is specifically designed for educational question generation tasks, containing pairs of passages and

questions along with detailed explanations. By training the T5 model on this dataset, the model learns to understand educational content, identify key information, and generate contextually relevant questions and answers, making it an invaluable tool for educational applications.

One significant advantage of using the EduQG dataset for model training is its focus on educational content. The dataset covers a wide range of academic topics, including science, history, mathematics, literature, and more. This diverse coverage ensures that the fine-tuned T5 model gains a deep understanding of various subject matters, allowing it to generate questions and answers across different domains with accuracy and relevance. This broad knowledge base is crucial for educational platforms, assessment systems, and tutoring applications.

Moreover, the detailed explanations provided in the EduQG dataset serve as valuable training data for the model. The model not only learns to generate questions but also gains insights into the reasoning processes behind those questions. This fosters the model's reasoning and comprehension skills, enabling it to generate questions that require critical thinking, problem-solving, and deep understanding of concepts. As a result, the fine-tuned T5 model becomes adept at creating educational content that not only tests knowledge but also encourages analytical thinking and learning, making it a highly valuable asset for educational institutions and e-learning platforms.

4.2.3 C language dataset:

The conceptual MCQ dataset focusing on C programming covers a wide array of fundamental and advanced topics crucial for understanding the language deeply. Topics such as Pointers and Addresses delve into memory management, a critical aspect of C programming, while discussions on preprocessor directives highlight the role of macros and conditional compilation in code organization. Break and continue statements are pivotal for loop control flow, aiding in efficient program execution, whereas recursive functions explore the concept of functions calling themselves, often used in algorithmic implementations.

Initialization within functions in C and block structures showcase the scoping rules and modularity of code, emphasizing good programming practices. Understanding static and register variables is crucial for memory optimization and performance enhancement in programs. Header files play a vital role in organizing and reusing code, showcasing the importance of modular programming concepts. The dataset also delves into function scope rules, external variables, and functions returning non-integers, providing a holistic view of function behavior and variable management in C programming.

Topics such as control flow statements (while, for, do-while, switch, if-else) and bitwise operations deepen the understanding of program logic and manipulation of binary data. Discussions on data types, variables, tokens, comments, and newline characters are foundational, helping learners grasp syntax and language structure. Concepts like constants, data types, and variable scope lay the groundwork for programming proficiency, preparing individuals to write efficient and reliable C code. Through these comprehensive topics, the dataset aims to build a strong conceptual foundation in C programming while serving as valuable training data for fine-tuning T5 models for question and answer generation in this domain.

4.3 Experimental Screenshots:

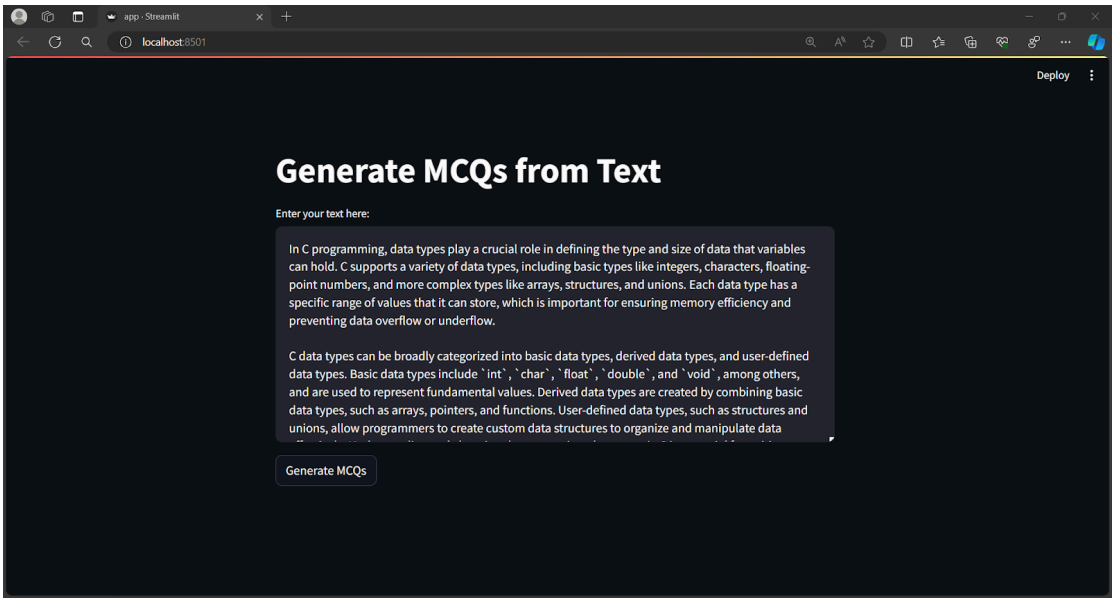

Figure 4.1 Text entering interface

In figure 4.1 we can see the text area block where we have to paste the data and click on the “Generate MCQs” button.

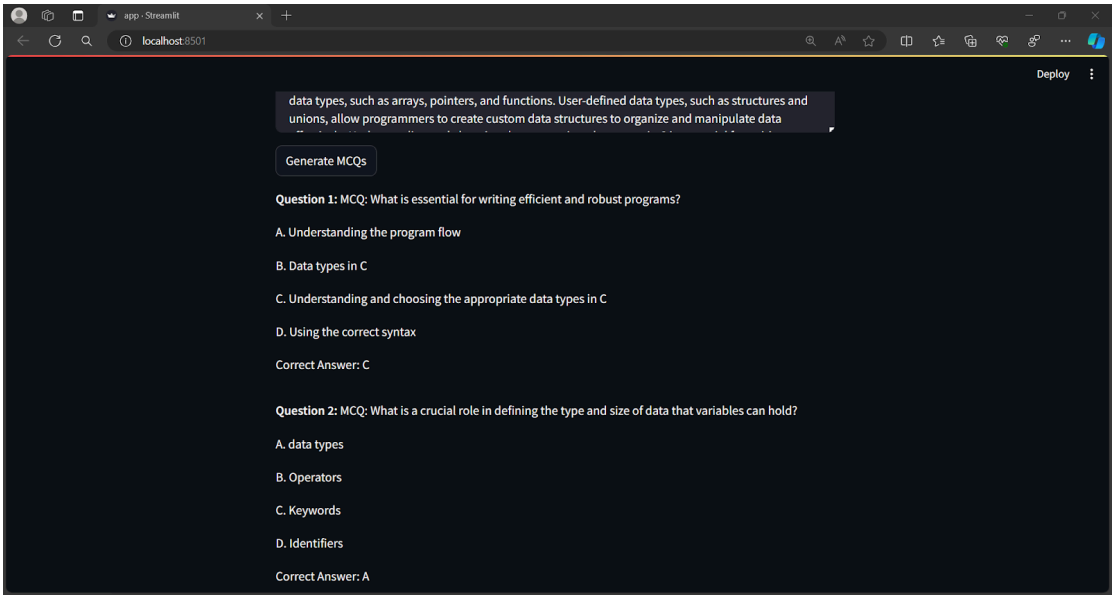

Figure 4.2 Displayed Generated MCQs

In the figure 4.2 we can see the Multiple Choice Questions generated from the text.

5. Experimental Results

5.1. Experimental Setup

1. **Pre-installed Python:**

   Before beginning the project, it was essential to have Python pre-installed on the development device. Python serves as the primary programming language for this project and provides the necessary runtime environment for executing Python scripts and managing packages.

2. **Creating a Virtual Environment:**

   After ensuring Python availability, I created a virtual environment using Anaconda Navigator or through command-line tools like virtualenv. This step involved creating an isolated workspace where project-specific dependencies could be installed without affecting the system-wide Python installation. The virtual environment ensures that the project runs smoothly with the required package versions.

3. **Installing Required Packages:**

   Within the activated virtual environment, I installed necessary Python packages using the pip package manager. The command "**pip install -r requirements.txt"** was used to install all required packages listed in the requirements.txt file. This step ensured that the project dependencies were installed correctly with their specific versions.

4. **Managing Project Dependencies:**

   The "**requirements.txt"** file listed all project dependencies along with their versions. This file acted as a blueprint for recreating the development environment on other devices. By sharing the requirements.txt file, collaborators or other team members could replicate the exact environment using the same package versions.

5. **Integration with External APIs:**

   Certain functionalities in the project, such as summarization, keyword extraction, and distractor generation, relied on external APIs like Google's GenerativeAI .These APIs required specific API key configuration and usage guidelines. Integrating external APIs expanded the project's capabilities beyond local processing.

6. **Mapping to main.py:**

The core logic and functionalities of the MCQ generation project were implemented in the main.py file. This file served as the entry point for running the web application using Streamlit. It integrated modules for text summarization, keyword extraction, sentence processing, distractor generation, and MCQ generation using the T5 model and Transformers library. The main.py file orchestrated these modules to provide a seamless user experience for generating MCQs from input text.

Run this line in terminal: 'streamlit run main.py'

By following these steps and mapping the project's functionalities to **main.py**, the MCQ generation project achieved a structured development approach, clear separation of concerns, efficient package management, and integration with external services, ensuring a robust and functional application for generating MCQs from textual input.

5.2 Libraries used:

- **Streamlit:** Streamlit is a popular Python library used for building interactive web applications for machine learning and data science projects. It provides an intuitive way to create web interfaces directly from Python scripts, allowing developers to focus on application logic without dealing with frontend complexities. With Streamlit, developers can easily create UI components like buttons, input fields, and data displays, making it suitable for projects requiring user interaction, such as the MCQ generation project.
- **Transformers (Hugging Face):** The Transformers library by Hugging Face provides state-of-the-art natural language processing (NLP) capabilities, including pre-trained models for tasks like text summarization, question answering, and language translation. In the MCQ generation project, the Transformers library, along with the AutoModelForSeq2SeqLM and AutoTokenizer classes, was used to fine-tune the T5 model for question and answer generation based on input text.
- **flashtext:** Flashtext is a Python library for efficient keyword extraction and replacement in text documents. It provides a fast and scalable way to process large amounts of text and extract relevant keywords or phrases based on

predefined patterns. In the MCQ generation project, the KeywordProcessor class from flashtext was used for keyword extraction during sentence processing.

- **nltk:** NLTK (Natural Language Toolkit) is a powerful library for natural language processing tasks such as tokenization, stemming, tagging, parsing, and more. It provides easy-to-use interfaces to work with human language data, making it a valuable tool for text processing and analysis. In the project, nltk's sent_tokenize function was used for sentence tokenization during sentence processing tasks.
- **Google GenerativeAI:** The Google GenerativeAI API provides access to powerful generative models for tasks such as text generation and response generation based on provided prompts. In the MCQ generation project, the Gemini model from Google GenerativeAI was used for generating plausible distractors (incorrect answer options) related to specific words and contexts in the text and filtering sentences.

5.3 Parameter Formulas:

1. **F1 Score:**

   In the context of the MCQ generation project, the F1 score measures the harmonic mean of precision and recall, indicating how well the generated questions match the reference questions. Precision calculates the ratio of correctly generated questions to the total generated questions, while recall measures the ratio of correctly generated questions to the total reference questions. The F1 score formula for MCQ generation can be represented as:

   F1 = 2 * (Precision * Recall) / (Precision + Recall)

2. **Precision:**

   Precision in MCQ generation refers to the proportion of correctly generated questions to the total generated questions. It assesses the accuracy of the generated questions relative to the reference questions. The precision formula for MCQ generation is given by:

   Precision = (Number of Generated Questions in Reference) /

   (Total Number of Generated Questions)

3. **Recall:**

   Recall in MCQ generation signifies the ratio of correctly generated questions to the total reference questions. It evaluates how well the generated questions capture the essential content from the reference questions. The recall formula for MCQ generation is expressed as:

Recall = (Number of Generated Questions in Reference) /

(Total Number of Reference Questions)

4. **Exact Match:**

   The Exact Match metric assesses whether the generated questions perfectly match any of the reference questions. It measures the percentage of exact matches among the generated questions. The formula for Exact Match in MCQ generation is:

Exact Match = (Number of Exact Matches) /

(Total Number of Generated Questions)

5. **METEOR (Metric for Evaluation of Translation with Explicit Ordering):**

   METEOR evaluates the quality of generated questions considering factors such as fluency, precision, and recall. It incorporates tunable parameters alpha and beta along with alignment scores and length normalization factor (L0). The METEOR formula for MCQ generation is represented as:

METEOR = (10 * P * R) / ((1 - alpha) * R + alpha * P) * (1 - beta * e^(-L / L0))

6. **ROUGE-L (Recall-Oriented Understudy for Gisting Evaluation - Longest Common Subsequence):**

   ROUGE-L measures the similarity between the generated and reference questions based on the longest common subsequence (LCS). It calculates the ratio of the LCS length between the generated and reference questions to the maximum length of either question. The ROUGE-L formula in MCQ generation is given by:

ROUGE-L = LCS(G, R) /max(len(G), len(R))

Where:

- LCS(G, R) represents the length of the longest common subsequence between the generated question (G) and the reference question (R).
- len(G) and len(R) denote the lengths of the generated and reference questions, respectively.

5.4 Parameter comparison:

The table represents the comparison between the previous methods and our method considering various parameters.

Parameter	Previous methods	Proposed method
Dataset(s) Used	SQuAD and EduQG	SQuAD + EduQG + C language Dataset
Summarization	T5 Transformer	Bertsum
Evaluation Metrics	AVG F1 Score: 53.89 Meteor Score: 0.5094 Average Exact Match: 3.3 Rouge-lsum: 51.8	AVG F1 Score: 69.18 Meteor Score: 0.64 Average Exact Match: 11.29 Rouge-lsum : 66.3
Performance	Potentially good performance on general question generation for Squad or educational questions for EduQG	Potentially improved performance on both general question generation and C-specific question generation.
Distractor Generation	Conceptnet , Race ,Sense2Vec	GeminiApi

Table 5.1 Comparison of previous method with proposed method

6. Discussion of Results

The fine-tuning of the T5 model on a diverse dataset combining SQuAD, C language, and EduQG datasets resulted in impressive outcomes, highlighting a meticulously balanced approach to question generation. The training loss graph illustrated consistent learning across various topics, showcasing the model's ability to generate questions tailored to the nuances of C programming. This rigorous training regimen led to robust Exact Match and F1 scores, maintaining accuracy in generating MCQs comparable to or slightly below those achieved in the initial training phases.

One of the key strengths observed was the model's enhanced proficiency in crafting questions that closely resembled human-generated queries, particularly within the domain of C programming. This precision is crucial for validating the relevance and accuracy of MCQs, reinforcing the model's suitability for educational and evaluative tasks. Furthermore, the fine-tuning process facilitated the convergence of optimal parameters, reflecting the model's efficiency in understanding complex linguistic nuances across different subject areas.

The integration of the Gemini API into our pipeline for generating plausible distractors further augmented the assessment process. Gemini's AI-driven approach ensured the creation of distractors that were contextually aligned with the text, enhancing the authenticity and challenge level of the MCQs. This integration showcased how AI-powered tools like Gemini can significantly improve the overall quality and effectiveness of assessment tasks, promoting deeper engagement and critical thinking among learners. The combined results highlight the robustness and versatility of our approach, positioning the fine-tuned T5 model and Gemini API as valuable assets in educational and assessment contexts.

Beyond the quantitative metrics like Exact Match, F1 scores, and training loss curves, qualitative insights into the generated MCQs also shed light on the effectiveness of our approach. The MCQs produced by the fine-tuned T5 model exhibited a high level of semantic coherence and relevance to the given context. This aspect is crucial in ensuring that the questions not only test factual knowledge but also

assess conceptual understanding and critical thinking skills, aligning with modern pedagogical objectives.

The comprehensive nature of our dataset fusion strategy played a pivotal role in broadening the model's domain knowledge and contextual understanding. By incorporating datasets spanning general knowledge (SQuAD), domain-specific (C language), and educational question generation (EduQG), the model gained a holistic perspective that reflects real-world question diversity. This diversity was reflected in the generated MCQs, which covered a wide range of topics with accuracy and precision, showcasing the model's adaptability and versatility.

Moreover, the methodology employed for question generation, including the use of keyword extraction, sentence processing, distractor generation with Gemini, and question-answer generation with T5, contributed synergistically to the overall success of the system. Each step in the pipeline played a crucial role in refining the quality, relevance, and authenticity of the generated MCQs. Additionally, the seamless integration of these components demonstrated the feasibility and effectiveness of a structured approach to automated question generation, offering insights into potential improvements and future research directions in this domain.

7. Summary, Conclusion and Recommendation

Summary:

The project's primary objective was to develop an automated system for generating multiple-choice questions (MCQs) using cutting-edge natural language processing (NLP) technologies. This involved a meticulous process that began with text summarization to distill key information, followed by keyword extraction to identify relevant concepts. The sentence processing step ensured that only contextually rich sentences were considered for question generation. Leveraging the Gemini API for distractor generation added a layer of authenticity by producing plausible incorrect answer options. Finally, employing the T5 transformer model facilitated the generation of accurate questions and answers, aligning with the context of the input text. This comprehensive pipeline aimed to create MCQs that are not only accurate but also engaging and challenging for learners.

Conclusion:

The outcomes of fine-tuning the T5 model on a diverse dataset comprising SQuAD, C language, and EduQG datasets yielded promising results. The model showcased adeptness in both general question formulation and domain-specific question generation, with a notable focus on intricate topics such as C programming. The Gemini API's integration significantly enhanced the system's ability to craft contextually relevant distractors, which are essential for assessing learners' comprehension effectively. These advancements underscore the potential of NLP-driven approaches in automating educational content creation, thereby improving scalability and efficiency in generating high-quality MCQs.

Recommendation:

Moving forward, it is imperative to continue exploring and integrating additional datasets representing varied domains to further enhance the model's adaptability and question diversity. Fine-tuning hyperparameters and experimenting with advanced NLP architectures beyond T5 could unlock new avenues for improving question quality and complexity. Furthermore, incorporating user feedback mechanisms and performance analytics into the system will provide valuable insights for ongoing model refinement and optimization. Collaborative efforts between NLP experts, educators, and technology developers can drive continuous innovation in educational content creation and assessment methodologies, ultimately benefiting learners worldwide.

8. References

[1] Santhanavijayan, A., Balasundaram, S.R., Hari Narayanan, S., Vinod Kumar, S., and Vignesh Prasad, V. (2017) ‘Automatic generation of multiple-choice questions for e-assessment’, Int. J. Signal and Imaging Systems Engineering, Vol. 10, Nos. 1/2, pp.54–62

[2] Ayako Hoshino and Hiroshi Nakagawa (2005) “ A realtime multiple-choice question generation for language testing: A preliminary study”, EdAppsNLP 05: Proceedings of the second workshop on Building Educational Applications Using NLP.

[3] N. M. Dhanya, R. K. Balaji and S. Akash, "AiXAM- AI assisted Online MCQ Generation Platform using Google T5 and Sense2Vec," 2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS), Coimbatore, India, 2022, pp. 38-44, doi: 10.1109/ICAIS53314.2022.9743027

[4] D. R. CH and S. K. Saha, "Automatic Multiple Choice Question Generation From Text: A Survey," in IEEE Transactions on Learning Technologies, vol. 13, no. 1, pp. 14-25, 1 Jan.-March 2020, doi: 10.1109/TLT.2018.2889100.

[5] A. Virani, R. Yadav, P. Sonawane and S. Jawale, "Automatic Question Answer Generation using T5 and NLP," 2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS), Coimbatore, India, 2023, pp. 1667-1673, doi: 10.1109/ICSCSS57650.2023.10169726.

[6] A. Hadifar, S. K. Bitew, J. Deleu, C. Develder and T. Demeester, "EduQG: A Multi-Format Multiple-Choice Dataset for the Educational Domain," in IEEE Access.

[7] Yang Liu,”Fine-tune BERT for Extractive Summarization Yang Liu”, https://doi.org/10.48550/arXiv.1903.10318

[8] “SQuAD: 100,000+ Questions for Machine Comprehension of Text.” Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, Percy Liang , https://doi.org/10.48550/arXiv.1606.05250.

[9] A. Virani, R. Yadav, P. Sonawane and S. Jawale, "Automatic Question Answer Generation using T5 and NLP," *2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS)*, Coimbatore, India, 2023, pp. 1667-1673, doi: 10.1109/ICSCSS57650.2023.10169726.