Automated Objective-Type Question Generation: A ML Model for Diverse Assessments

Syed Rayan, Nameera Nousheen, Maria Ashfaq Hussain

[1,2,3] B.E CSE(AI&DS) Student
[1,2,3] Computer Science & Artificial Intelligence Department
[1,2,3] Muffakham Jah College of Engineering & Technology, Hyderabad, Telangana State, India – 500 034

Abstract - This machine learning model aims to automate the generation of various objective-type questions, including, but not limited to, multiple-choice questions (MCQs), True/Yes or False/No questions, Assertion and Reasoning questions, and fill in the blank questions, based on given input text and subsequently organize them into a CSV file. The working of this idea is initiated by creating a model that is focused upon the generation of MCQs, which is now working efficiently. The process involves parsing the input text to extract relevant information for question generation. The model utilizes the OpenAI library and various Python modules such as json, re, and csv for processing and manipulation of data. Key functionalities include handling different formats of input text, converting responses into a structured format, and generating MCQs while considering Bloom's Taxonomy levels. Alongside this, the other columns include some metrices for giving reliable scores of the generated content such as Difficulty Level, Accuracy Score, Distractors Quality Score, Taxonomy Alignment Score, and Clarity Score. Additionally, it filters out irrelevant content such as references to figures. The generated questions are then stored in a JSON format and subsequently converted into a CSV file. The model provides error handling mechanisms and flags to monitor the execution flow. Overall, this tool streamlines the process of generating objective-type questions from text input, facilitating educational content creation and assessment development, that can also be integrated into the existing e-learning platforms that are available in the market.

Index Terms - Corpus, Objective-Type Questions, OpenAI, JSON, CSV, Bloom’s Taxonomy Levels, API Call, Question Generation, Educational Assessment, E-learning Integration.

I. INTRODUCTION

Frequent assessments are essential for evaluating student knowledge and understanding, but the manual generation of questions for these assessments has become increasingly exhausting. Educators often resort to copying questions from various resources, leading to redundancy and a lack of originality in the questions posed to students. To address this challenge, a machine learning model is created to automate the generation of various objective-type questions. This model can create multiple-choice questions (MCQs), True/Yes or False/No questions, Assertion and Reasoning questions, and fill in the blanks questions based on input text and organize them into a CSV file.

The project began with a focus on generating MCQs, a task the model now performs efficiently. It parses input text to extract relevant information for question generation, leveraging the OpenAI library and various Python modules such as json, re, and csv for data processing and manipulation. Key functionalities include handling different text formats, converting responses into structured formats, and generating MCQs while considering Bloom's Taxonomy levels. Additionally, the model incorporates metrics such as Difficulty Level, Accuracy Score, Distractors Quality Score, Taxonomy Alignment Score, and Clarity Score to ensure the quality and reliability of the generated questions.

Irrelevant content, such as references to figures, is filtered out to maintain the clarity and relevance of the questions. The generated questions are stored in a JSON format and subsequently converted into a CSV file. The model includes error handling mechanisms and flags to monitor the execution flow, streamlining the process of generating objective-type questions from text input. This tool facilitates educational content creation and assessment development and can be integrated into existing e-learning platforms, enhancing their capabilities, and providing educators with a valuable resource for crafting original and effective assessments.

II. AIM & OBJECTIVE

The primary aim of this project is to develop an automated machine learning model that generates various objective-type questions, including MCQs, True/False questions, Assertion and Reasoning questions, and fill in the blanks. This model seeks to streamline the assessment process, enhance educational content creation, and integrate seamlessly with existing e-learning platforms. Key objectives include improving efficiency by addressing the manual exhaustion and redundancy in question generation, ensuring the quality and reliability of generated questions through specific metrics, and leveraging advanced technologies such as the OpenAI library and Python modules for data processing. Additionally, the model aims to handle diverse input formats, support educational development, and implement robust error handling and monitoring mechanisms.

III. LITERATURE SURVEY

Bolanle Ojokoh and Emmanuel Adebisi [4]: The authors provide a comprehensive overview and analysis of question answering (QA) systems, proposing a new classification scheme based on various criteria such as domain, question type, data source, answer form, language paradigm, and approaches.
Qingyu Zhou et al. [2]: This study proposes a neural network approach for automatic question generation from text passages. Experiments on the SQuAD dataset show that neural encoder-decoder models can produce meaningful questions, although they struggle with certain question types due to insufficient training examples.
Xinya Du, Junru Shao, and Claire Cardie [3]: The authors tackle question generation for reading comprehension using neural networks, avoiding hand-crafted rules. Their encoder-decoder architecture with an attention mechanism highlights the importance of context but requires a large amount of high-quality training data.

Sanjay K Dwivedi and Vaishali Singh [1]: This paper proposes a taxonomy for characterizing QA systems and provides a qualitative analysis of major QA systems in literature, highlighting limitations in linguistic approaches and the early stages of advanced reasoning.

K.S.D. Ishwari et al.[5]: The authors review the progression of QA systems from rule-based to deep learning approaches, discussing state-of-the-art models like Dynamic Memory Networks and Neural Turing Machines. They highlight challenges in domain independence and the integration of real-world knowledge.

IV. Hardware & Software Requirements

Hardware Requirement

1. Hard Disk: 256 SSD (Minimum)
2. RAM: 4 GB (Minimum)

Software Requirement

1. Operating system: Windows 10 & Above
2. Platform: Jupyter Notebook/Google Collab, Python 3.10

V. Existing System

The Neural Question Generation from Text framework utilizes a bidirectional GRU encoder and an attention-based GRU decoder to automatically generate questions from input text passages. It incorporates lexical and answer position features to focus on generating relevant questions, alongside a copy mechanism for handling rare words. Training on question-answer pairs from the SQuAD dataset, evaluations demonstrate superior performance over rule-based approaches, with enhancements from the addition of a copy mechanism and pre-trained word embeddings.

Human ratings affirm the system's ability to generate questions more pertinent to input sentences and answers compared to rule-based approaches. Analysis reveals high precision and recall for common question types like WHAT, WHO, HOW, and WHEN, while encountering more difficulty with less frequent types such as WHICH and WHY. Overall, the neural encoder-decoder framework exhibits promising results for natural language question generation, showcasing its potential for practical applications in text processing tasks.

Existing neural question generation systems can produce high quality questions from text, but struggle with less common question types like WHICH and WHY. Performance on rare questions needs improvement for neural models to robustly handle diverse real-world inputs. Enhancing neural architectures to generate better questions of all types from natural language remains an open research problem.

VI. Proposed System

The proposed system presents a comprehensive solution leveraging OpenAI's ChatCompletion endpoint, focusing on meticulous prompt engineering to convey requirements effectively to the designated model, 'text-davinci-003' (got deprecated on 4th January 2024 [6]). Initially developed for generating multiple-choice questions (MCQs), the system now supports various objective-type questions, including True/False, Assertion and Reasoning, and Fill in the Blanks. By specifying the 'n' value, the system facilitates multiple API calls to ensure thorough processing. The input prompt and subsequent model responses are structured around defined tokens, ensuring precise communication. Following model generation, responses are formatted into JSON for clarity and ease of processing. The system enhances question categorization by compiling lists based on Bloom's Taxonomy Levels and includes difficulty levels for further refinement. Ultimately, the output is organized into a downloadable CSV file, maintaining column order for accessibility and convenience. This approach streamlines the process from request to output, optimizing functionality and usability, and addresses the need for efficient and varied question generation in educational assessments.

VII. Advantages Of Proposed System

1. **Structured Communication:** The system enhances communication with OpenAI's ChatCompletion endpoint by employing prompt engineering techniques, ensuring precise conveyance of requirements to the model.
2. **Efficient Processing:** By facilitating multiple API calls based on the specified 'n' value, the system enables thorough processing of requests, improving the quality and accuracy of generated responses.
3. **Clear Output Formatting:** Responses from the model are structured into JSON format, enhancing clarity and ease of processing, thereby facilitating downstream analysis and utilization.
4. **Enhanced Categorization:** Lists of Bloom's Taxonomy Levels aid in categorizing questions effectively, allowing for better organization and understanding of generated responses.
5. **Objective Question Variety:** The system supports the generation of various types of objective questions, including MCQs, True/False, Assertion and Reasoning, and Fill in the Blanks, making it a versatile tool for educational assessments.
6. **Convenient Output:** The system generates outputs in a downloadable CSV file, maintaining column order for accessibility and convenience, thus simplifying further analysis and integration into existing workflows.
7. **Optimized Workflow:** By integrating these features, the proposed system optimizes the response generation process, improving efficiency and effectiveness in generating responses from OpenAI's model.

VIII. Specifications Of The Proposed System

1. **OpenAI API Integration –**
   i. Utilizes OpenAI's ChatCompletion endpoint 'text-davinci-003' for response generation.
2. **Prompt Engineering Approach –**
   i. Adopts prompt engineering to convey requirements effectively to the model.
   ii. Ensures precise communication by mentioning each detail explicitly.
3. **Multiple API Calls –**
   i. Allows for the specification of the 'n' value to determine the number of API calls.
   ii. Facilitates thorough processing of requests to improve response quality.

4. **Token-Based Input and Response –**
   i. Structures input prompt and model response based on specified tokens.
   ii. Enhances communication and understanding between the user and the model.
5. **JSON Formatting –**
   i. Processes model responses into JSON format for clarity and ease of understanding.
   ii. Facilitates downstream analysis and processing of generated responses.
6. **Bloom's Taxonomy Integration –**
   i. Incorporates lists of various Bloom's Taxonomy Levels to enhance question categorization.
   ii. Improves organization and comprehension of generated responses.
7. **Difficulty Level Specification –**
   i. Mentions the difficulty level of each question based on the given corpus of text.
   ii. Provides additional information for better categorization and refinement of responses.
8. **CSV Output Generation –**
   i. Generates outputs in a downloadable CSV file format.
   ii. Maintains column order for accessibility and ease of integration into existing workflows.
9. **Efficiency and Optimization –**
   i. Optimizes the response generation process, improving efficiency and effectiveness.
   ii. Enhances user experience by streamlining communication and output delivery.

IX. System Architecture Design

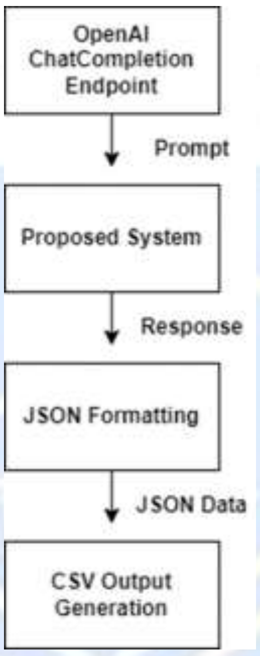


Figure 1: Data Flow Diagram

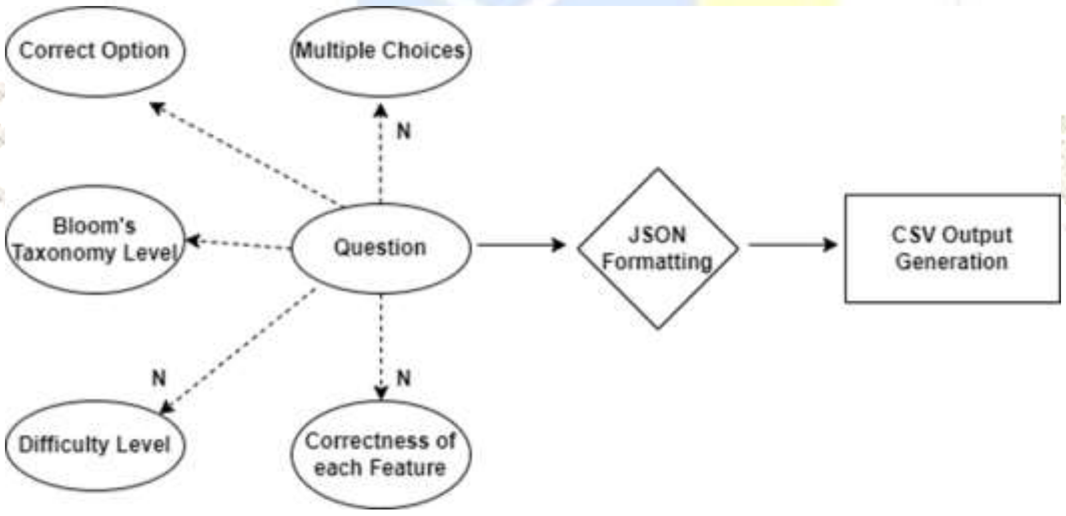


Figure 2: Entity-Relationship Diagram

X. Use Case

The use case involves getting a 'corpus of text' from the user, upon which a Question, Four Choices, Single Correct Option, Bloom's Taxonomy level, Difficulty Level, Accuracy Score, Distractors Quality Score, Taxonomy Alignment Score, Clarity Score are provided for every question that is being generated.

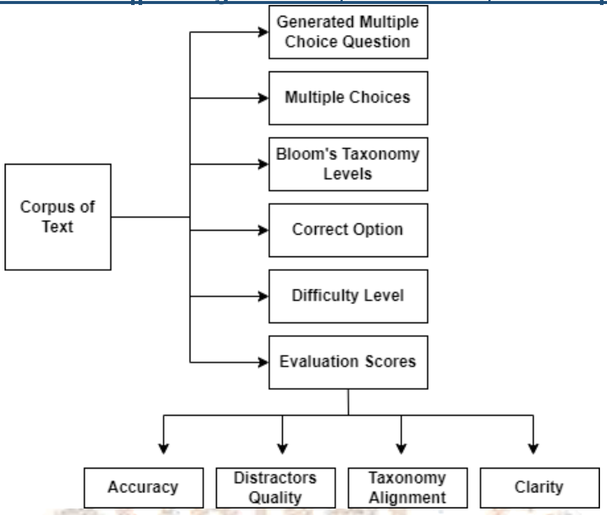


Figure 3: Use Case Diagram

XI. Sequence Diagram

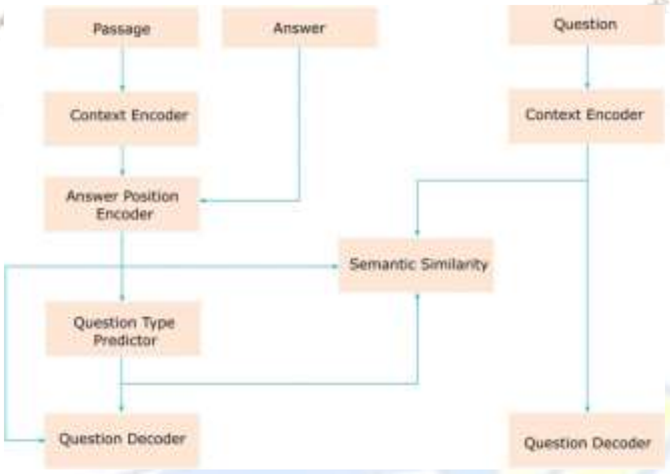
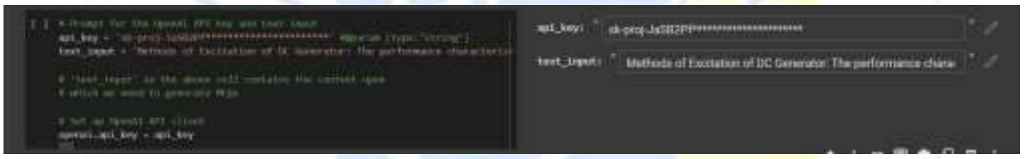


Figure 4: Sequence Diagram

XII. Code Snippets of Execution Flow

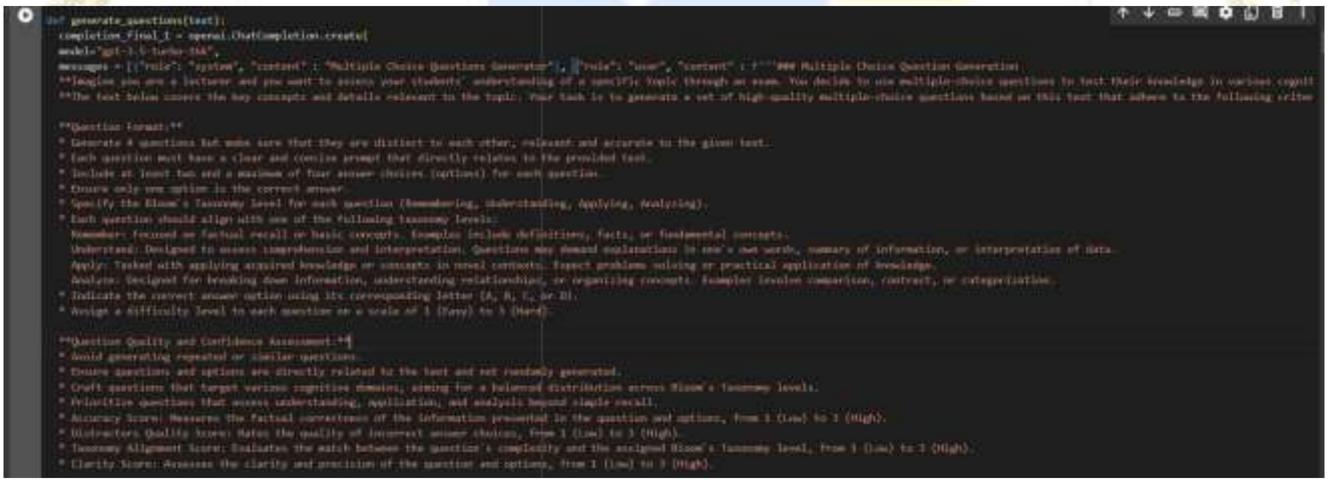


```
[ ] # Prompt for the OpenAI API key and text input
    api_key = 'sk-proj-JaSB2P**************************' #@param {type:"string"}
    text_input = 'Methods of Excitation of DC Generator: The performance characteristic

    # 'text_input' is the above cell contains the content upon
    # which we need to generate MCQs

    # Set up OpenAI API client
    openai.api_key = api_key
```

api_key: sk-proj-JaSB2P************************

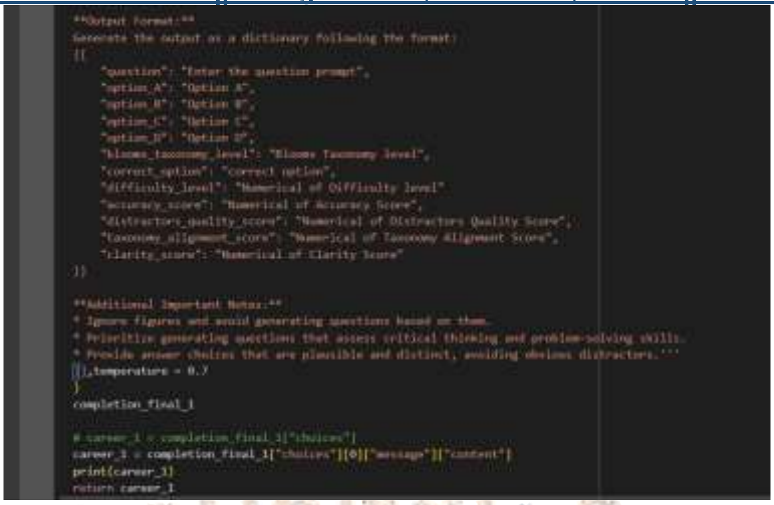
text_input: Methods of Excitation of DC Generator: The performance chara

Figure 5: Input fields of API Key and Corpus of text (input text)



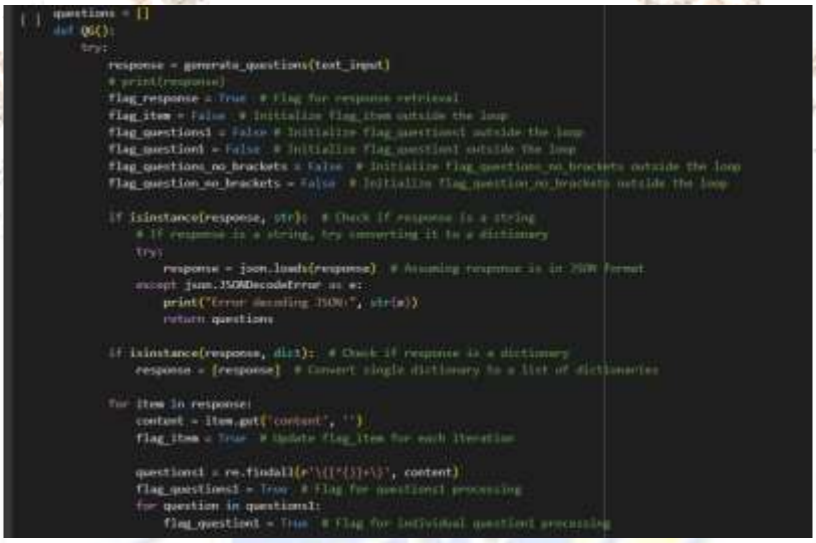
```
def generate_questions(text):
  completion_final_1 = openai.ChatCompletion.create(
  model="gpt-3.5-turbo-16k",
  messages = [{"role": "system", "content" : "Multiple Choice Questions Generator"}, {"role": "user", "content" : f'''### Multiple Choice Question Generation
  **Imagine you are a lecturer and you want to assess your students' understanding of a specific topic through an exam. You decide to use multiple-choice questions to test their knowledge in various cognit
  **The text below covers the key concepts and details relevant to the topic. Your task is to generate a set of high-quality multiple-choice questions based on this text that adhere to the following criter

  **Question Format:**
  * Generate 4 questions but make sure that they are distinct to each other, relevant and accurate to the given text.
  * Each question must have a clear and concise prompt that directly relates to the provided text.
  * Include at least two and a maximum of four answer choices (options) for each question.
  * Ensure only one option is the correct answer.
  * Specify the Bloom's Taxonomy level for each question (Remembering, Understanding, Applying, Analyzing).
  * Each question should align with one of the following taxonomy levels:
    Remember: Focused on factual recall or basic concepts. Examples include definitions, facts, or fundamental concepts.
    Understand: Designed to assess comprehension and interpretation. Questions may demand explanations in one's own words, summary of information, or interpretation of data.
    Apply: Tasked with applying acquired knowledge or concepts in novel contexts. Expect problems solving or practical application of knowledge.
    Analyze: Designed for breaking down information, understanding relationships, or organizing concepts. Examples involve comparison, contrast, or categorization.
  * Indicate the correct answer option using its corresponding letter (A, B, C, or D).
  * Assign a difficulty level to each question on a scale of 1 (Easy) to 5 (Hard).

  **Question Quality and Confidence Assessment:**
  * Avoid generating repeated or similar questions.
  * Ensure questions and options are directly related to the text and not randomly generated.
  * Craft questions that target various cognitive domains, aiming for a balanced distribution across Bloom's Taxonomy levels.
  * Prioritize questions that assess understanding, application, and analysis beyond simple recall.
  * Accuracy Score: Measures the factual correctness of the information presented in the question and options, from 1 (Low) to 3 (High).
  * Distractors Quality Score: Rates the quality of incorrect answer choices, from 1 (Low) to 3 (High).
  * Taxonomy Alignment Score: Evaluates the match between the question's complexity and the assigned Bloom's Taxonomy level, from 1 (Low) to 3 (High).
  * Clarity Score: Assesses the clarity and precision of the question and options, from 1 (Low) to 3 (High).
```

Figure 6: Part-1 of prompt given. Conveys the question format, Quality and Confidence Assessment.



```
**Output Format:**
Generate the output as a dictionary following the format:
[{
    "question": "Enter the question prompt",
    "option_A": "Option A",
    "option_B": "Option B",
    "option_C": "Option C",
    "option_D": "Option D",
    "blooms_taxonomy_level": "Blooms Taxonomy level",
    "correct_option": "correct option",
    "difficulty_level": "Numerical of Difficulty level"
    "accuracy_score": "Numerical of Accuracy Score",
    "distractors_quality_score": "Numerical of Distractors Quality Score",
    "taxonomy_allignment_score": "Numerical of Taxonomy Allignment Score",
    "clarity_score": "Numerical of Clarity Score"
}]

**Additional Important Notes:**
* Ignore figures and avoid generating questions based on them.
* Prioritize generating questions that assess critical thinking and problem-solving skills.
* Provide answer choices that are plausible and distinct, avoiding obvious distractors.'''
}],temperature = 0.7
)
completion_final_1

# career_1 = completion_final_1["choices"]
career_1 = completion_final_1["choices"][0]["message"]["content"]
print(career_1)
return career_1
```

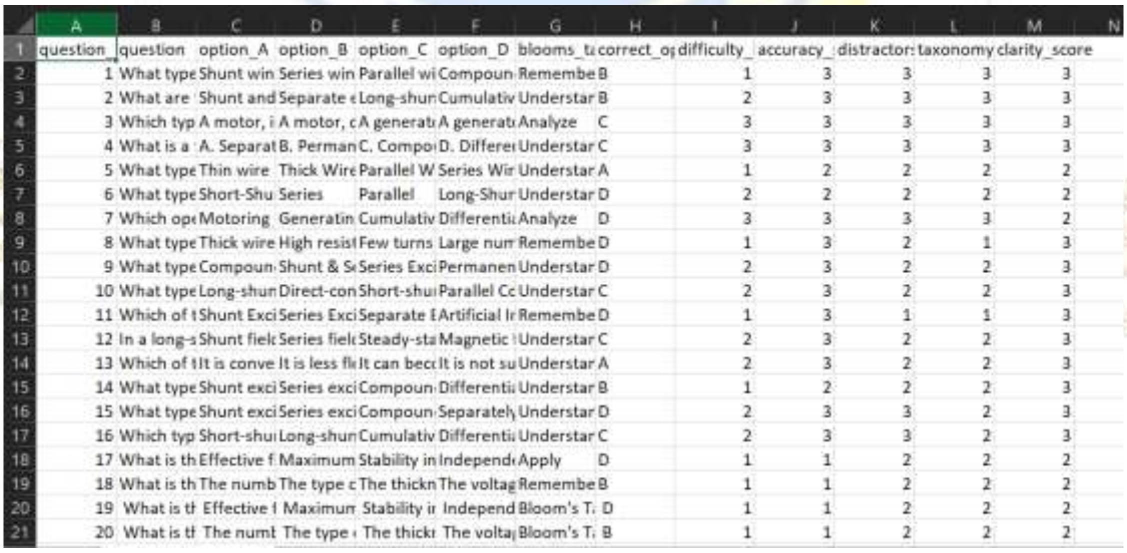
Figure 7: Part-2 of prompt given. Conveys the Output format, additional notes, and other parameters.



```
questions = []
def QG():
    try:
        response = generate_questions(text_input)
        # print(response)
        flag_response = True  # Flag for response retrieval
        flag_item = False  # Initialize flag_item outside the loop
        flag_questions1 = False # Initialize flag_questions1 outside the loop
        flag_question1 = False  # Initialize flag_question1 outside the loop
        flag_questions_no_brackets = False  # Initialize flag_questions_no_brackets outside the loop
        flag_question_no_brackets = False  # Initialize flag_question_no_brackets outside the loop

        if isinstance(response, str):  # Check if response is a string
            # If response is a string, try converting it to a dictionary
            try:
                response = json.loads(response)  # Assuming response is in JSON format
            except json.JSONDecodeError as e:
                print("Error decoding JSON:", str(e))
                return questions

        if isinstance(response, dict):  # Check if response is a dictionary
            response = [response]  # Convert single dictionary to a list of dictionaries

        for item in response:
            content = item.get('content', '')
            flag_item = True  # Update flag_item for each iteration

            questions1 = re.findall(r'\{[^{}]+\}', content)
            flag_questions1 = True  # Flag for questions1 processing
            for question in questions1:
                flag_question1 = True  # Flag for individual question1 processing
```

Figure 8: Processing of the generated 'response' to convert into a CSV file

XIII. OUTPUT



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	question_	question	option_A	option_B	option_C	option_D	blooms_t	correct_o	difficulty_	accuracy_	distractor	taxonomy	clarity_score	
2	1	What type	Shunt win	Series win	Parallel wi	Compoun	Remembe	B	1	3	3	3	3	
3	2	What are	Shunt and	Separate e	Long-shun	Cumulativ	Understar	B	2	3	3	3	3	
4	3	Which typ	A motor, i	A motor, c	A generati	A generati	Analyze	C	3	3	3	3	3	
5	4	What is a	A. Separat	B. Perman	C. Compo	D. Differe	Understar	C	3	3	3	3	3	
6	5	What type	Thin wire	Thick Wire	Parallel W	Series Wir	Understar	A	1	2	2	2	2	
7	6	What type	Short-Shu	Series	Parallel	Long-Shur	Understar	D	2	2	2	2	2	
8	7	Which ope	Motoring	Generatin	Cumulativ	Differenti	Analyze	D	3	3	3	3	2	
9	8	What type	Thick wire	High resist	Few turns	Large nur	Remembe	D	1	3	2	1	3	
10	9	What type	Compoun	Shunt & S	Series Exci	Permanen	Understar	D	2	3	2	2	3	
11	10	What type	Long-shun	Direct-con	Short-shu	Parallel Cc	Understar	C	2	3	2	2	3	
12	11	Which of t	Shunt Exci	Series Exci	Separate E	Artificial Ir	Remembe	D	1	3	1	1	3	
13	12	In a long-s	Shunt fielc	Series fielc	Steady-sta	Magnetic	Understar	C	2	3	2	2	3	
14	13	Which of t	It is conve	It is less fl	It can becc	It is not su	Understar	A	2	3	2	2	3	
15	14	What type	Shunt exci	Series exci	Compoun	Differenti	Understar	B	1	2	2	2	3	
16	15	What type	Shunt exci	Series exci	Compoun	Separately	Understar	D	2	3	3	2	3	
17	16	Which typ	Short-shu	Long-shun	Cumulativ	Differenti	Understar	C	2	3	3	2	3	
18	17	What is th	Effective f	Maximum	Stability in	Independ	Apply	D	1	1	2	2	2	
19	18	What is th	The numb	The type c	The thickn	The voltag	Remembe	B	1	1	2	2	2	
20	19	What is th	Effective f	Maximum	Stability ir	Independ	Bloom's T	D	1	1	2	2	2	
21	20	What is th	The numt	The type	The thickr	The volta	Bloom's T	B	1	1	2	2	2	

Figure 9: View of the output in a CSV File

The generated CSV file is easily downloadable, that contains data structured into columns representing various attributes of the questions generated by the system. Each row corresponds to a single question.

The columns include:

1. **Question Number:** This column serves as a unique identifier for each question, allowing for easy reference and organization of the generated content.
2. **Question:** This column contains the text of the question generated by the system.
3. **Option A, Option B, Option C, Option D:** These consecutive columns contain the four choices provided as options for the question.

4. **Single Correct Option:** This column specifies the correct option among the four choices.
5. **Bloom's Taxonomy Level:** This column indicates the cognitive complexity level of the question based on Bloom's Taxonomy.
6. **Difficulty Level:** This column denotes the perceived difficulty level of the question, determined from the provided corpus of text.
7. **Accuracy Score:** This column provides a numerical score representing the accuracy of the generated question.
8. **Distractors Quality Score:** This column offers a score indicating the quality of the distractors (incorrect options) provided with the question.
9. **Taxonomy Alignment Score:** This column indicates how well the generated question aligns with the specified Bloom's Taxonomy level.
10. **Clarity Score:** This column provides a score representing the clarity and coherence of the question text.

XIV. Output Summary

The generated CSV file provides a comprehensive overview of the questions generated by the system. Each row represents a unique question identified by a question number. The columns include the question text, four choices, the single correct option, Bloom's Taxonomy level, difficulty level, accuracy score, distractors quality score, taxonomy alignment score, and clarity score. This structured output facilitates efficient integration and analysis of the generated questions, aiding in educational content creation and assessment development processes.

XV. Conclusion

In conclusion, the project demonstrates the successful integration of advanced natural language processing techniques, specifically leveraging OpenAI's ChatCompletion endpoint, to streamline the process of generating educational content. Through meticulous prompt engineering and systematic processing, the system effectively generates questions and responses tailored to user specifications. The incorporation of Bloom's Taxonomy levels, difficulty assessments, and quality scoring mechanisms enhances the relevance and quality of the generated content. The resulting output, presented in a downloadable CSV file, provides a structured and comprehensive overview of the generated questions, facilitating seamless integration into educational systems and workflows. Overall, the project underscores the potential of AI-driven solutions to augment and optimize educational content creation processes, ultimately enhancing learning experiences for students and educators alike.

XVI. Future Enhancements

The above proposed system is just the version – 1 of the planned products. The next versions include the corpus of text to not be just in the textual format, but also will be extended to by making the input available with pdf, image format, text files etc, by the help of OCR. The output too could be using GenAI for generating Image or Video-based outputs, case-studies generation. Another cherry-on-the-top feature will be to extend the functionality of the model in such a way that it should be able to query-out the technical information from the given image (example – a labelled electric circuit diagram), whilst the same information might not have been explicitly mentioned in the textual format.

The proposed system, overall, can be integrated with other e-learning platforms or Learning Management Systems, such as Black Box, to conduct assessment quiz, generate progress reports and in other development tracking aspects.

Black Box is a famous software of its kind that particularly needed the input of objective type questions in an Excel or CSV file, and for this it will generate assessment/quizzes and then does further analytical process. The teaching faculty currently must prepare this input file (Excel or CSV) manually which is a hectic process. The above proposed model can be integrated at the initial stage of the software mentioned to generate required type of questions from the given corpus of text, which can then be fed as the input to the software to automate the task of creating assessments.

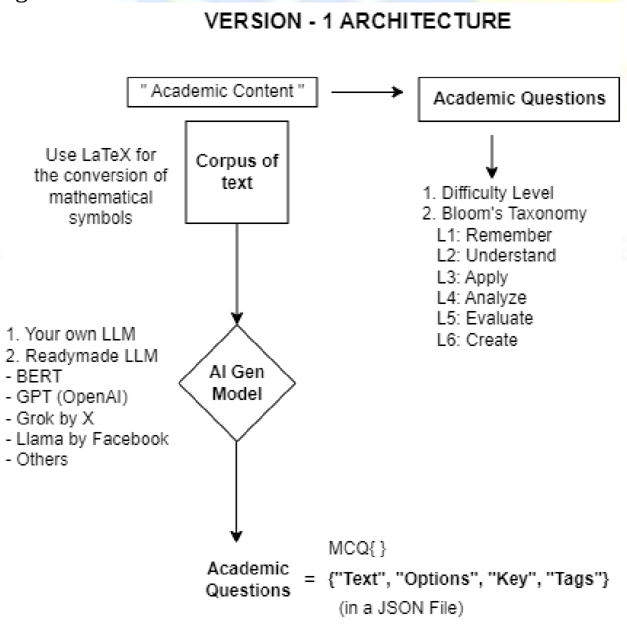


Figure 10: Phase/Version-1 Architecture

XVII. References

[1] Sanjay K Dwivedia, Vaishali Singh, “Research and reviews in question answering system”, International Conference on Computational Intelligence: Modeling Techniques and Applications (CIMTA) 2013, Procedia Technology 10 (2013) 417 – 424.
[2] Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, Ming Zhou, “Neural Question Generation from Text: A Preliminary Study”, arXiv:1704.01792v3 [cs.CL] 18 Apr 2017.
[3] Xinya Du, Junru Shao, Claire Cardie, “Learning to Ask: Neural Question Generation for Reading Comprehension”, arXiv:1705.00106v1 [cs.CL] 29 Apr 2017.
[4] Bolanle Ojokoh, Emmanuel Adebisi, “A Review of Question Answering Systems”, Journal of Web Engineering, Vol. 17 8, 717–758, doi: 10.13052/jwe1540-9589.1785.
[5] K.S.D. Ishwari, A.K.R.R. Aneeze, S. Sudheesan, H.J.D.A. Karunaratne, A. Nugaliyadde, Y. Mallawarrachchi, “Advances in Natural Language Question Answering: A Review”, arxiv/papers/1904/1904.05276.
[6] Deprecated Models of OpenaAI - has the “Legacy Model” and “Legacy Model Pricing” of the used OpenAI ChatCompletion Model “text-davinci-003”.