Arabic NLP: Generating Multiple Choice Questions From Saudi SchoolBooks

Yazan Alshuabi
Yalialshaebi@stu.kau.edu.sa

Mohanad Al Dakheel
mohanad.dakheel@kaust.edu.sa

Moayad Alghamdi
Moayadtalalgh@hotmail.com

Zyad Alzahrani
zyadalzhranie@gmail.con

Abstract

Arabic, a language renowned for its rich morphology and complex linguistic structures, presents unique challenges for Natural Language Processing (NLP). This project tackles this challenge by the power of Retrieval-Augmented Generation (RAG) and Large Language Models (LLMs) to generate Multiple-Choice Questions (MCQs) from Arabic texts found in Saudi schoolbooks. By employing LLM, we extract key concepts, relationships, and facts suitable for MCQ formulation. This project represents a significant step forward in the application of Arabic NLP to educational technologies, with the potential to streamline the creation of assessments, personalize learning experiences, and ultimately empower students in their educational journey.

Keywords: Arabic NLP, Multiple Choice Question Generation, Saudi Schoolbooks, Large Language Models, Retrieval-Augmented Generation (RAG), Few-shot Learning, Optical Character Recognition (OCR), ArabicMMLU

1. Introduction

The demand for innovative educational technologies is growing rapidly. As the Saudi educational system undergoes modernization, there is a critical need for digitalizing the entire curriculum, making it more efficient and accessible for software and AI solutions. In this context, the ability to generate high-quality Multiple-Choice Questions (MCQs) from existing Primary and Secondary educational materials is of great importance. Such a tool not only saves educators time but also ensures that assessments stay consistent and can be easily updated to reflect curriculum changes. Also, it sparks new ideas such as making competitions amongst the students for a more enjoyable learning experience. The implementation of such a project tackles Natural Language Processing (NLP) challenges specific to the Arabic language, such as its rich morphology and complex syntax. By leveraging advanced NLP models and machine learning techniques, we aim to use AI to generate MCQs from Arabic school textbooks, ensuring both accuracy and relevance to the curriculum. This involves a combination of data extraction, preprocessing, model selection, and evaluation of generated questions to meet educational standards. In this report, we will discuss how we built such a project, what challenges we faced, how we overcame those challenges, and what is our project's future plan.

2. Objectives

1. Assisting schoolteachers by providing support in formulating exams to **reduce their workload**.
2. Build a **repository** of high-quality, AI generated questions for easy access by teachers and students.
3. Develop an AI system that adapts to student performance when generating questions.
4. Contribute to Arabic NLP advancements in support of **Saudi Vision 2030**.

3. Challenges

1. **Data Limitations:** Limited availability of high-quality annotated datasets for Arabic MCQ generation poses challenges for model training and performance.
2. **Data Collection & Preprocessing:** The tedious process of extracting relevant data from books.
3. **Limited Advanced Models:** Current models struggle with generating high-quality distractors.

4. Data Acquisition

In the field of AI, data serves as a foundational component. Our initial objective was to obtain Saudi schoolbooks in a machine-readable format, along with corresponding multiple-choice questions (MCQs) for each book. Although we initially anticipated this task to be relatively simple, given the widespread availability of educational materials, we encountered significant challenges. Many schoolbooks were available in an unreadable format, with no comprehensive dataset existed specifically for exam questions based on these books. To address these issues, we undertook the manual extraction of content from the book "Biology 1". Then, we managed to get a knowledge-based dataset, called ArabicMMLU. ArabicMMLU is a Modern Standard Arabic(MSA) dataset, sourced from school exams across diverse educational levels in different countries spanning North Africa, the Levant, and the Gulf regions [1]. We will utilize this dataset as an alternative, and by applying Retrieval Augmented Generation (RAG) technique, we will see how we overcame the challenge of not finding a dataset specific to the Saudi curriculum.

5. Data Preprocessing

As we stated before, Saudi Schoolbooks are not machine-readable, and extracting the data manually is not efficient. Instead, we trained a YOLOv8 model to detect paragraphs from the pages, as they contain the most important information for question generation. Thus, when we apply Optical Character Recognition (OCR) techniques, we do not detect irrelevant text from the book. Applying that, we will have our Biology book transcripted into a corpus of text. Next, We will divide the text into chunks. see **Figure 1**.

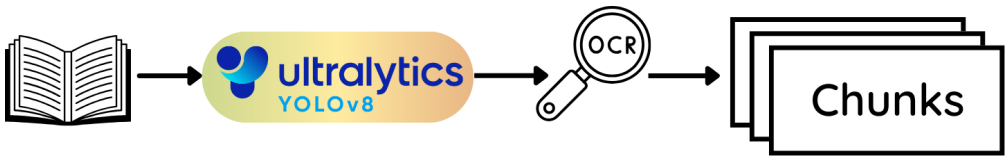

Figure 1: *Data Preprocessing block (partial)*

In other cases, some instructors use their own material, written in .docx or .txt format. Thus, there will be no need to use YOLOv8 nor OCR, because the material is already in a readable format. We considered both cases in our pipeline, see **Figure 2**

.

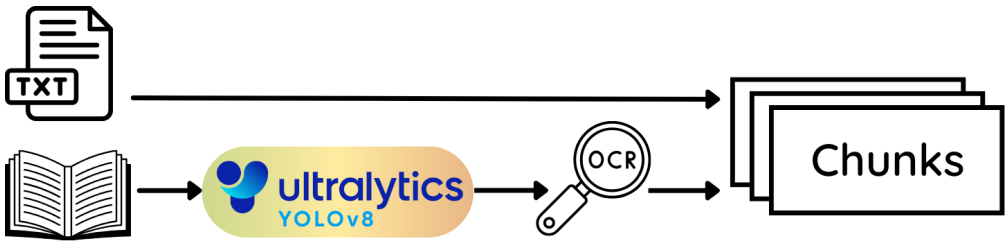

Figure 2: *Entire Preprocessing Block*

Chunking allows us to focus on smaller segments of text while still maintaining some level of contextual relevance within each chunk. Moreover, it sets our generated questions to be more in depth through the topic. The chunking process is simple [2].

We enumerated through the cleaned lines of text–outputted by OCR– and divided them into manageable segments based on their length. Specifically, we defined chunks to be text segments with a minimum length of 200 characters to ensure each chunk contained enough context for meaningful questions generation. see **Figure 3.**

```
#Removing small chunks
chunks = [chunk for chunk in cleaned_lines if len(chunk) >= 200]

for i, chunk in enumerate(chunks):
    print(f"Chunk {i+1}:")
    print(chunk)
    print("\n" + "="*50 + "\n")
```

Figure 3: *Chunking Code snippet*

Now that we have our text divided into chunks, our data is ready to be analyzed for MCQs generation.

6. Text Similarity Analysis

In this part, we want to analyze our chunks and try to find which items of our general knowledge dataset align with a specific chunk, then, we are going to use that information in a prompt which will generate the questions. This approach is called RAG, or Retrieval Augmented Generation.

6.1 ArabicMMLU Dataset

We mentioned before how this dataset is outsourced from various school exams across diverse countries and web. However, some of those data are not of our interest at the moment. There exists questions on Chemistry, Physics, etc... while we only care about biology. Thus, we will drop those columns. Moreover, we deleted 5-option questions, as we want to focus solely on 4-option questions since it adheres to the format we want to generate. Some of the data contains NaN values, so, we divided the data into sub-datasets based on how many NaN options it possesses.

6.2 TF-IDF and Cosine Similarity

In our project, we used TF-IDF (Term Frequency-Inverse Document Frequency) and cosine similarity to match content between chunks and the ArabicMMLU dataset. These techniques were employed to enhance the relevance of the automatically generated Multiple-Choice Questions (MCQs) and ensure alignment with educational material. TF-IDF is a statistical measure used to evaluate the importance of a word in a document relative to a corpus. This approach helps in transforming text into numerical vectors that capture the significance of terms within a given context. To measure the similarity between chunks from the Saudi schoolbooks and questions from the ArabicMMLU dataset, cosine similarity was employed. Cosine similarity quantifies the cosine of the angle between two non-zero vectors in a multi-dimensional space. This metric is useful for determining the degree of similarity between a chunk and dataset questions based on their TF-IDF vectors. Thus, the output of our Text Similarity Analysis is a chunk, and the corresponding three most similar questions see **Figure 4**. In the following Section, we will elaborate on why we took the three most similar questions instead of one similar question.

7. Choosing a suitable LLM

To identify the most effective model for generating high-quality distractor options, we evaluated several Large Language Models. Given the need for sophisticated reasoning to handle the complexity of our problem, we considered the following models:

1. AraGPT2
2. Arabic T5
3. ChatGPT 4
4. ChatGPT 4o
5. ChatGPT 4o-mini

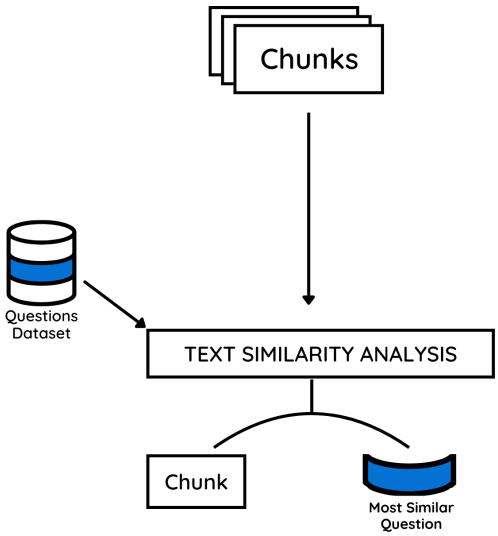

Figure 4: *Text Similarity Analysis Phase*

The evaluation was based on the models' ability to understand and contextualize the data from Saudi schoolbooks and the ArabicMMLU dataset. ChatGPT 4o-mini emerged as the most suitable choice due to its advanced embeddings designed to handle non-english languages better than ChatGPT 4, with its compact and light size which makes it faster than the regular ChatGPT 4o and other LLMs. Our qualitative assessment showed that ChatGPT 4o-mini provided the most coherent and contextually appropriate distractors.

8. MCQ Generation

What is a prompt? A prompt is a textual input designed to elicit a specific type of response from a language model. It can be a question, statement, or instruction that directs the model to generate relevant content. The effectiveness of the prompt largely depends on its clarity and specificity, as it helps the model understand the context and requirements of the task. All the steps we followed in the similarity analysis part were to collect information that enhances the quality of our prompt. The way we write the prompt, the information and examples we include, and the specificity of the required task are all part of a technique called Prompt Engineering. In our prompt engineering step, we wrote a message which provides clear and specific instructions, including:

1. The role of the model as a biology teacher.
2. The task of correcting and summarizing the text.
3. The requirement to generate MCQs with contextual distractors.
4. Avoidance of questions that rely on specific text or images.

Then, we integrated the chunk and three most similar questions outputted from our Text Similarity Analysis phase as an example, see **Figure 5**.

```python
# Function to generate MCQs from a data chunk with a system message
def generate_mcqs(data_chunk, questions):

    system_message = '''You are a biology teacher,
    You will be provided with a chunk of text and three questions,
    The text is extracted from a biology book using OCR. It has some mispronunciation and mistakes, I want you first to fix it,
    then summarize it into a correct paragraph based on the context, and then generate the multiple-choice questions based on it,
    Your task is to generate a multiple-choice questions in Arabic with an answer and 3 distractors,
    The multiple-choice question should be related to the chunk of text; the provided questions are just to help you understand the structure of the questions,
    Make sure that the 3 distractors are related to the context of the answer and try to manipulate students,
    The answer key should contains only the option key, for example: C,
    Avoid questions that need the text or some photo. for example it shouldn't contain "3-19 الشكل" or "بالاصفارة بالسهم" ,
    Take your time understanding the chunk of text, generating the multiple-choice question, and the distractors.'''

    # Build the prompt by iterating over the questions list
    prompt = f"Text:\n{data_chunk}\n\n"
    for idx, question in enumerate(questions, 1):
        prompt += f'''Example Question {idx}: {question['question']}
    A. {question['options'][0]}
    B. {question['options'][1]}
    C. {question['options'][2]}
    D. {question['options'][3]}
    Answer: {question['answer']}\n\n'''
```

Figure 5: *Prompt Engineering Code snippet*

But the question is, why did we give our model three most similar questions as an example? Generating MCQs is not an easy task. It requires understanding the context, the model will not only have to figure out the answer, but also generate other incorrect options that are closely related to the answer (which are called, distractors). In that case, the best approach we tried is using few-shot learning. Unlike Zero-shot and One-Shot learning, few-shot learning gives the language model a few examples within the prompt, given enough examples; The language model will learn and understand the context more and comes with not only high-quality questions, but also higher quality distractors. We tried giving ChatGPT 4o-MINI from 0 to 3 examples and noticed that it performed better when we give it three examples. Now, we have our Generated Multiple Choice Questions. In the following section, we will discuss how we evaluated their quality, and see the results. for the full project pipeline, see **Figure 6**.

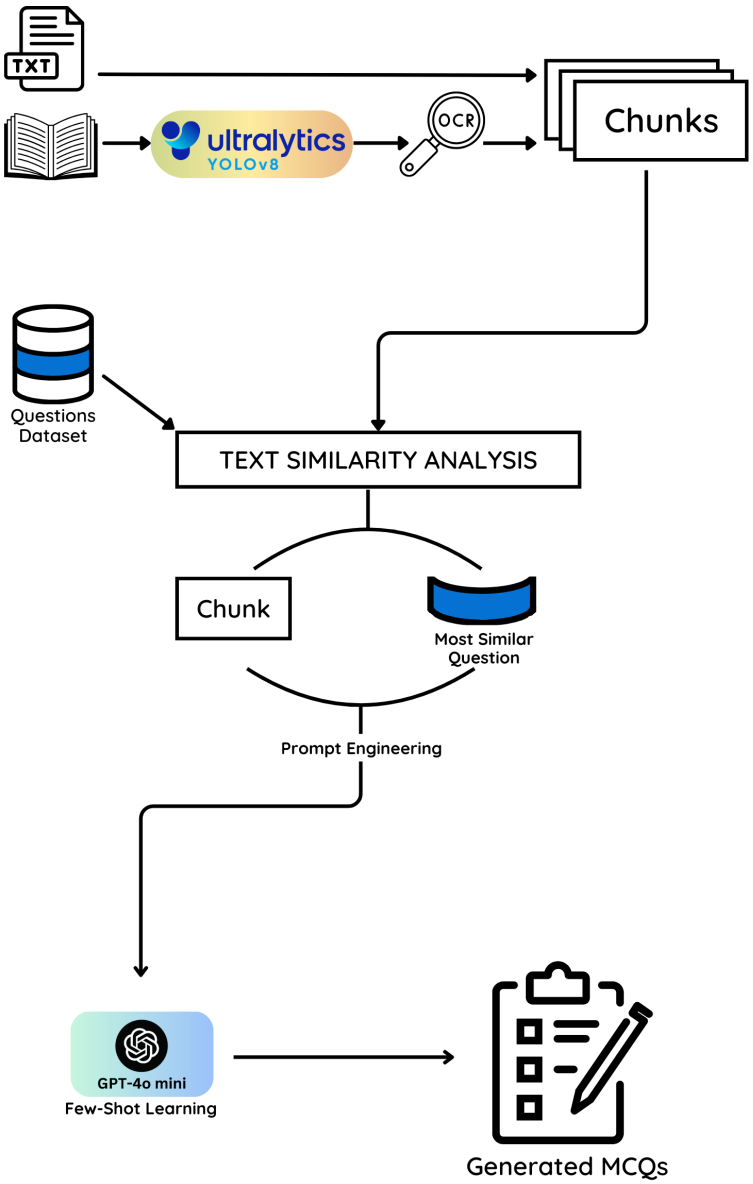

Figure 6: *Complete pipeline of the project*

9. Results

The evaluation part was challenging, because none of the evaluations techniques we came across seemed suitable. Thus, we concluded that the most appropriate approach is Human-Evaluation. we took a sample of 20% of our generated MCQs, then, we divided the sample to 10 different people to evaluate. We made sure

that our evaluators are familiar with Biology to assure discarding any evaluation inaccuracies. The evaluation consists of five criterias:

1. **Coherence Quality:** measures how closely netted the question is, how much readable it is**(rated from 0 to 5, where 5 is the highest score).**
2. **Distractors Quality:** measures how close the incorrect options are to the correct one **(rated from 0 to 5, where 5 is the highest score).**
3. **Question Quality:** measures how much the question catches the crucial information from the chunk**(rated from 0 to 5, where 5 is the highest score).**
4. **Relevance to the text:** measures how close the question is to given chunk, to insure that the generated questions don't get outside the scope**(rated from 0 to 5, where 5 is the highest score).**
5. **Correctness:** classify if the given answer key is correct or incorrect **(0 for incorrect, 1 for correct).**

Figure 7 depicts the mean and standard deviation of each criteria. The model performed relatively well and achieved higher than 4 out of 5 in all criteria except the distractors quality which scored 3.96. On the other hand, the standard deviation measures the scatter that exists in the distribution. We notice that the distractors achieved the highest standard deviation as well. That indicates that there were some questions scored 5 and other scored 0, that is because there is a ratio of 4.8% of incorrect answer keys, see **Figure 8**. Those questions with incorrect answer keys will score a zero on distractors quality because the question itself doesn't have a present correct answer. However, the overall results of our model are promising, with more than 95% of correct questions, and a mean of a minimum close to 4 out of 5 is definitely a milestone for us that we can continue on enhancing it in the future.

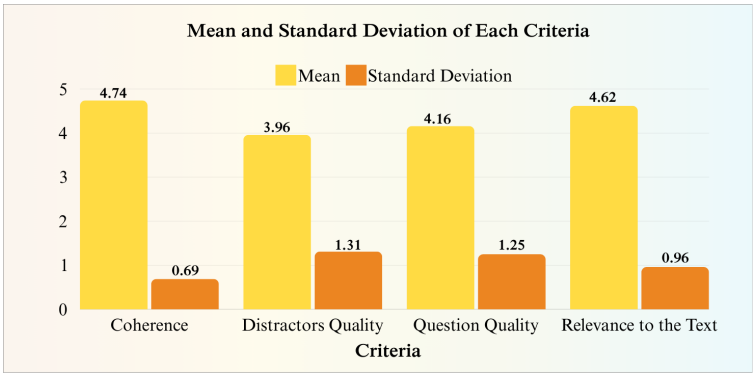

Figure 7: *Bar chart representing the mean and standard deviation(out of 5) of Coherence, Distractors, Question quality and relevance*

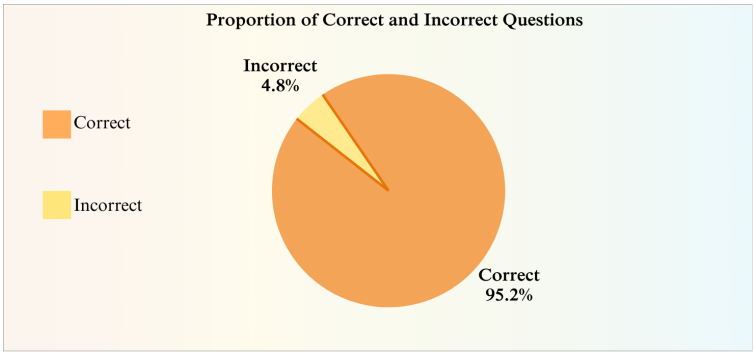

Figure 8: *Pie chart representing correct and incorrect questions*

10. Conclusion

To sum up, we presented a comprehensive approach to the generation of Multiple-Choice Questions (MCQs) from Saudi schoolbooks. By employing Natural Language Processing (NLP) techniques such as Term Frequency-Inverse Document Frequency (TF-IDF), cosine similarity, and Retrieval-Augmented Generation (RAG), we ensured the relevance of the questions from our knowledge base dataset to the specific educational content. Then, Through prompt engineering process and the application of few-shot learning with the ChatGPT 4o-mini model, we achieved high-quality MCQ generation with contextually appropriate distractors.

The evaluation of the generated questions showed strong performance, with most criteria scoring above 4 out of 5. However, some inconsistencies in distractor quality point to areas for further refinement.

This project represents a notable contribution to Arabic NLP and educational technology, assisting teachers in exam creation while improving the student learning experience. Future work will focus on building our own dataset, expanding the scope of our project to cover more subjects, and building an independent platform where students can compete, and teachers can create exams! We aspire to support Saudi Vision 2030's goals for digital innovation in education.

References

1. *Koto, F., Li, H., Shatnawi, S., Doughman, J., Sadallah, A. B., Alraeesi, A., Almubarak, K., Alyafeai, Z., Sengupta, N., Shehata, S., Habash, N., Nakov, P., & Baldwin, T. (2024, July 30). Arabicmmlu: Assessing massive multitask language understanding in Arabic. arXiv.org.* `https://arxiv.org/abs/2402.12840`
2. Manning, C., Jurafsky, D., et al. (2019). *CS224n: Natural Language Processing with Deep Learning.* Stanford University. Retrieved from `https://web.stanford.edu/class/cs224n/`