QUESTION CONTEXT RETRIEVAL, BASED ON TOC LOOKUP IN QUESTION GENERATION AND TESTING SYSTEMS

Simeon Monov, Emre Myumyun,
Nikolay Pavlov, Andrey Nikolov

Abstract. *Despite the advancements in large language models (LLM), question generation (QG) remains a difficult task to manage. Modern Generative LLMs can perform well on QG on pretrained data [1] but QG on domain-specific data requires pretraining and fine-tuning custom models on the domain-specific contexts and question-answer pairs [2]. Another approach is to select a smaller question context from a domain-specific document and provide it as part of the prompt to the LLM. Selecting the correct context is challenging especially when working with large documents.*

In this paper we propose a method of retrieval and selection of question context, based on the document table of contents (TOC) using LLM. The method is very effective on tasks, where a sequence of questions must be generated, so that each question depends on the previous ones. The method can be applied in automated testing for job interviews or student examinations.

Key words: Question Context, Question Generation, LLM, Context Retrieval.

Introduction

Information Extraction (IE), which turns plain text into structured information, is a big step forward in Natural Language Processing (NLP). Traditional IE methods like Named Entity Recognition, Relation Extraction, and Event Extraction are useful, but they face challenges due to their dependence on domain-specific models [3]. New improvements in Large Language Models (LLMs), like GPT-4, have changed NLP by using a lot of pretraining on different datasets. Zero-shot and few-shot learning are made possible for a variety of applications by LLMs' production of human-like text. This has paved the way for generative approaches to IE, which create structured information directly from text while handling large and complex datasets.

This study describes a unique approach for automating question generation and structured document analysis utilizing generally available LLM. In recent years, a variety of innovative methods have been proposed to improve information extraction and automate question generation [7, 8]. Our approach addresses challenges such as domain-specific limitations and redundancy. This work demonstrates how LLMs can be guided with prompt engineering to generate diverse question types.

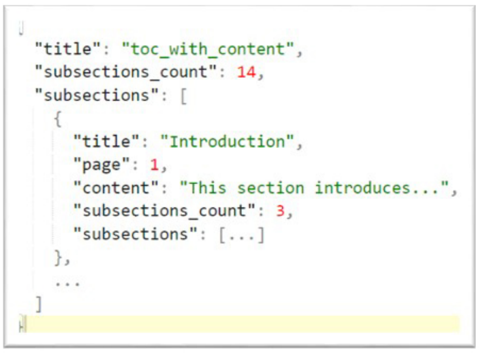
Solution Overview

The purpose of this research is to develop an automated workflow for the development of adaptive questions utilizing LLMs. By leveraging Large Language Models (LLMs) like GPT-4o and GPT-4o-mini [4], this method transforms documents into structured data, dynamically selects relevant sections, and generates context-aware questions. It adjusts question difficulty based on user performance, providing personalized assessments and enhancing the efficiency and scalability of question generation.

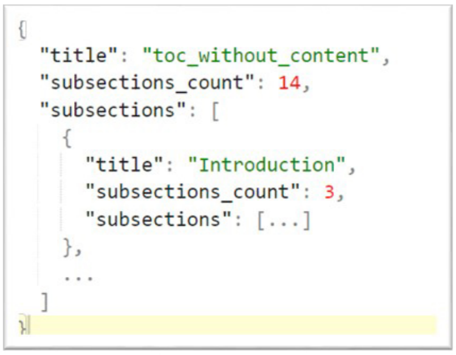
1. **Document Processing:** Extracts and organizes all textual content in a tree.
2. **Section Selection:** The most appropriate section is selected dynamically by providing previously answered questions and TOC content.
3. **Context Preparation:** Combines all content from selected sections and their subsections.
4. **Question Generation:** Utilizes the prepared context and predefined prompts to generate a question.

Document Processing

To transform the content of a document into a structured and analyzable format, document processing serves as the initial step in the question creation workflow. This process ensures that all relevant information is collected and organized, which is crucial for stages like **section selection** and **context preparation**. The primary goal of this stage is extracting two Table of Content (TOC) representations. Each one has a title in it, and just one has content, which is used later for text collecting. They both are stored in a tree data structure in the same hierarchical structure as the TOC of the document.



```
{
  "title": "toc_with_content",
  "subsections_count": 14,
  "subsections": [
    {
      "title": "Introduction",
      "page": 1,
      "content": "This section introduces...",
      "subsections_count": 3,
      "subsections": [...]
    },
    ...
  ]
}
```

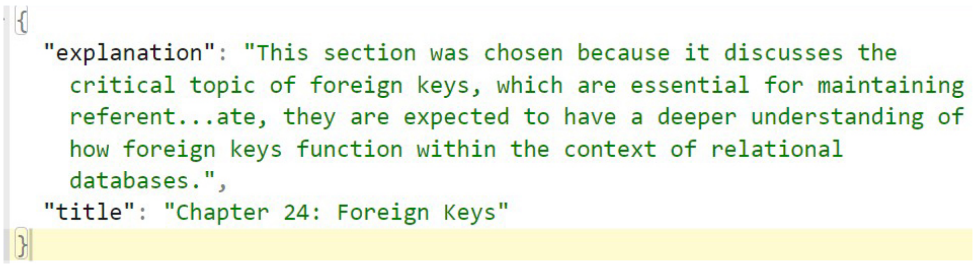


```
{
  "title": "toc_without_content",
  "subsections_count": 14,
  "subsections": [
    {
      "title": "Introduction",
      "subsections_count": 3,
      "subsections": [...]
    },
    ...
  ]
}
```

Figure 1. TOC representation examples

Section Selection

Section selection is a step for identifying the most relevant section for the next question to generate meaningful and precise questions. This process involves analyzing the TOC without content and referencing previously answered questions. Using appropriate prompts, the system guides the LLM to retrieve contextually relevant sections while avoiding asking the same question twice and selecting increasingly complex content. The selected sections are then returned in structured JSON format, ready for integration into the next steps in the workflow.



```
{
  "explanation": "This section was chosen because it discusses the
    critical topic of foreign keys, which are essential for maintaining
    referent...ate, they are expected to have a deeper understanding of
    how foreign keys function within the context of relational
    databases.",
  "title": "Chapter 24: Foreign Keys"
}
```

Figure 2. Example response when SQL related document is provided

Context Preparation

This step focuses on extracting all the relevant information about the chosen section in the document so that the LLM has a set of data to work with. The main goal is to ensure that the model receives a well-structured and meaningful context to generate precise and contextually appropriate questions. This process contains 3 steps until everything is extracted properly:

1. **Hash Table Storage:** All sections of the document, including with their subsections, are pre-stored in a hash table during the

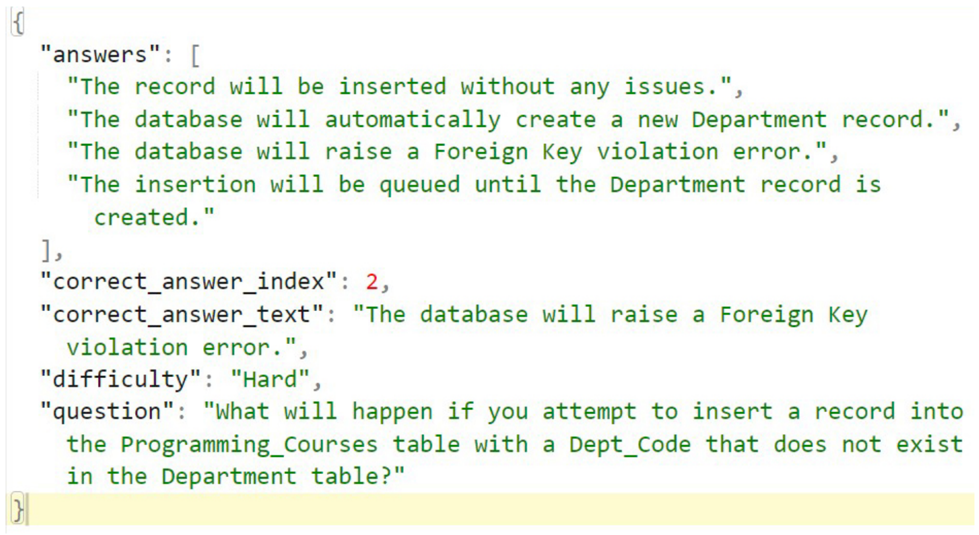
document processing phase. In this structure, each section title serves as a key, and the corresponding content is the value. This approach allows for fast retrieval of relevant sections without the need for repetitive operations for every question.

2. **Content Extraction:** Once the target section is found, a recursive depth-first search (DFS) algorithm is used to go through the section and all of its subsections. During this traversal, the textual content of each visited node is extracted and concatenated into a single string. This ensures that no relevant information from the chosen section hierarchy is left out.

3. **Context Assembly:** The aggregated content from the DFS traversal is stored as one object. This assembled content includes all pertinent information, ensuring that the LLM receives sufficient detail to generate questions while avoiding irrelevant or redundant material.

Question Generation

This is the final step in the workflow, where the prepared context is used to create relevant question. This step relies on LLMs and carefully crafted prompts to guide the question-generation process. The input consists of the prepared context and previous questions, which help ensure logical progression. The LLM generates questions in single-choice format based to the content of the selected section.



```
{
  "answers": [
    "The record will be inserted without any issues.",
    "The database will automatically create a new Department record.",
    "The database will raise a Foreign Key violation error.",
    "The insertion will be queued until the Department record is
      created."
  ],
  "correct_answer_index": 2,
  "correct_answer_text": "The database will raise a Foreign Key
    violation error.",
  "difficulty": "Hard",
  "question": "What will happen if you attempt to insert a record into
    the Programming_Courses table with a Dept_Code that does not exist
    in the Department table?"
}
```

Figure 3. Generated Single-Choice Question with Correct Answer and Metadata

The output is formatted in a predefined structure in JSON to facilitate easy integration with other systems. The phase concludes with the delivery of questions ready for use, demonstrating how automation enhances both productivity and quality in question generation. This approach ensures precision and adaptability while reducing time compared to human question development.

Prompt Engineering

Designing effective prompts has quickly become crucial for getting the best results from large language models. We used a zero-shot learning approach with carefully crafted prompts to guide the LLM through each step of the process [5]. The input prompts serve as instructions that influence the model's output, but they don't change how the model's internal settings or "weights" work. In-context learning can be done in different ways (zero-shot, one-shot, or few-shot learning) depending on how much information you include in the prompt [6].

In our approach, we provide instructions without giving specific examples, allowing the model to generate the desired outputs based on these instructions. Below are snippets from our system and user prompts for the different steps.

System message prompt for choosing a new section:

You are conducting interviews based on specific documents. Your task is to choose the next section for the interview using seniority level, the document's TOC in JSON format (with section titles) and all previous questions. Follow the TOC order, focusing on related questions before moving to new topics and limit to three questions per topic. Cover the whole document with specific subtopic questions, avoiding overly general ones. Return answers in the specified JSON format only: `{JSON_EXAMPLE}`.

User message prompt for choosing a new section:

Using seniority: `{seniority_level}`, *previous questions:* `{answered_questions}`, *and document TOC:* `{toc_without_content}`, *choose the next section. Return answer in the specified JSON format only:* `{JSON_EXAMPLE}`.

System message prompt for generating a new question:

You are a senior technical engineer conducting technical interviews. Based on the question context, candidate seniority and previous answers, generate tailored interview questions. Questions must strictly align with the context and adjust difficulty based on answers. For unclear contexts, return no questions. Ensure questions are clear, self-contained, relevant to the context, and not repeated. Return answers in the specified JSON format only: `{JSON_EXAMPLE}`.

User message prompt for generating a new question:

Previous questions: `{answered_questions}`. *Using the context, candidate seniority:* `{seniority_level}` *and no prior knowledge, generate one question with one correct and three wrong answers. Ensure no repetition. Return answers in the specified JSON format only:* `{JSON_EXAMPLE}`.

The JSON example provides a structured template for generating single-choice questions. It includes fields for the question text, possible answers, the correct answer (both its index and text), and the question's difficulty level. This format ensures consistency and enables easier integration with other components of the system.

Test Experiments

We conducted various test experiments to evaluate how the LLM models handled different tasks. For this purpose, we created an application that implements the different steps of our approach. This application functions as an interview platform based on specific technologies provided as PDF documents.

Beginner candidate with provided SQL Document and answering correctly to questions continuously:

- **First Question:** The question was CRUD-related with difficulty "Easy".

  Explanation: "This section was chosen because it serves as an introduction to SQL, which is fundamental for a junior candidate.

It allowed for basic questions that assessed the candidate's understanding of SQL concepts before moving on to more complex topics."

- **Second Question:** Upon correctly answering the first question, the next question was related to "ALTER TABLE" with a difficulty level of "Medium."

  Explanation: "This section was chosen as a logical next step after discussing basic SQL concepts. 'ALTER TABLE' introduces more complex SQL operations suitable for a beginner candidate to understand, especially after their initial grasp of CRUD operations. The questions delved into specific examples and syntax related to altering tables, which is crucial for database management."

- **Subsequent Questions:** After correctly answering the second and third questions with medium difficulty (the model avoided asking hard questions to a beginner candidate until a sequence of correct answers was achieved), the next question was "Hard" and related to Cascade Deleting.

  Explanation: "This section was chosen because it introduces the concept of cascading deletes, an essential aspect of relational database design. Since the candidate had already answered questions about basic SQL operations, moving to the implications of operations like delete, especially in the context of referential integrity, challenged their understanding further."

- **Adaptive Focus:** When the model detected correct answers on specific sections, it continued asking more precise questions, focusing on specific subtopics.

Experienced candidate with provided Java Spring Framework Document and answering correctly to questions continuously:

- **First Question:** The question was related to Spring Beans Autoconfiguration with a difficulty of "Hard".

  Explanation: "This section was chosen because it delves into core concepts suitable for an experienced candidate, assessing familiarity with Spring's foundational principles."

- **Adaptive Difficulty:** The model continued asking "Hard" questions until a wrong answer was provided.

Beginner candidate with provided Python Document and answering incorrectly after a sequence of correct answers:

- **Adjustment After Incorrect Answer:** Answering a "Hard" question incorrectly after a streak of five correct answers resulted in decreasing the difficulty level to "Easy" and changing the topic from infinite loops to mathematical operations. The section with the wrong answer was excluded from future questions, and the model adjusted accordingly.

Conclusion

We found that Large Language Models (LLMs) like GPT-4o and GPT-4o-mini can be effectively used with zero-shot learning to automate question generation and structured document analysis from unstructured texts. Our approach successfully integrated document processing, section selection, context preparation, and question generation to address challenges inherent in traditional information extraction methods, such as domain-specific limitations and redundancy.

The experiments demonstrated that our method could adapt question difficulty based on candidate performance, providing personalized assessments across various domains. However, we observed that the model does not always adapt the difficulty level properly, which is a challenge to be addressed in future work. Additionally, on rare occasions (2–3 times), the model generated questions where the provided answer was not correct. Resolving this issue is also part of our future plans.

In our future experiments, we plan to enhance the types of questions the model can generate by adding open-ended questions. We will also explore few-shot learning approaches to further improve the model's performance and adaptability. By addressing these challenges, we aim to refine our question-generation system, paving the way for more advanced and efficient NLP automation.

Acknowledgments

This study is partially supported by the project FP23-FMI-002 "Intelligent software tools and applications in research in mathematics, informatics, and teaching pedagogy" at the Plovdiv University "Paisii Hilendarski".

References

[1] X. Yuan, T. Wang, Y. Wang, E. Fine, R. Abdelghani, P. Lucas, H. Sauzéon, P. Oudeyer, Selecting better samples from pre-trained LLMs: A case study on question generation, *arXiv preprint*, 2022, https://doi.org/10.48550/arXiv.2209.11000.

[2] A. Hasan, M. Ehsan, K. Shahnoor, S. Tasneem, *Automatic question & answer generation using generative Large Language Model (LLM)*, Doctoral dissertation, Brac University.

[3] D. Xu, W. Chen, W. Peng, C. Zhang, T. Xu, X. Zhao, X. Wu, Y. Zheng, Y. Wang, E. Chen, Large language models for generative information extraction: a survey, *arXiv preprint*, 2023, https://doi.org/10.48550/arXiv.2312.17617.

[4] OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, et al. GPT-4 technical report, *arXiv preprint*, 2023, https://doi.org/10.48550/arXiv.2303.08774.

[5] X. Amartiain, Prompt Design and Engineering: Introduction and Advanced Methods, *arXiv preprint*, 2024, https://doi.org/10.48550/arXiv.2401.14423.

[6] Microsoft, Getting Started with LLM Prompt Engineering, 2024.

[7] A. Malinova, O. Rahneva, A. Golev, Automatic generation of English language test questions on parts of speech, *International Journal of Pure and Applied Mathematics*, Vol. 111, No. 3, 2016, DOI: 10.12732/ijpam.v111i3.14.

[8] A. Malinova, V. Kyurkchiev, G. Spasov, Parameterized examination in Econometrics, *AIP Conference Proceedings*, 1926, 020028, 2018, https://doi.org/10.1063/1.5020477.

Simeon Monov[1], Emre Myumyun[2], Nikolay Pavlov[3], Andrey Nikolov[4],
[1,2,3,4] Paisii Hilendarski University of Plovdiv,
Faculty of Mathematics and Informatics,
236 Bulgaria Blvd., 4027 Plovdiv, Bulgaria
Corresponding author: smonov@uni-plovdiv.bg