EvalQuiz — LLM-based Automated Generation of Self-Assessment Quizzes in Software Engineering Education

Niklas Meißner [1], Sandro Speth [1], Julian Kieslinger [1], Steffen Becker [1]

Abstract: Self-assessment quizzes after lectures, educational videos, or chapters are a commonly used method in software engineering (SE) education to allow students to test the knowledge they have gained. However, creating these quizzes is time-consuming, cognitively exhausting, and complex, as an expert in the field needs to create the quizzes and review the lecture material for validity. Therefore, this paper presents a concept to automatically generate self-assessment quizzes based on lecture material using a large language model (LLM) to reduce lecturers' workload and simplify the general quiz creation process. The developed prototype was handed to experts, who subsequently evaluated the approach. The results show that automatic quiz generation saves time, and the quizzes cover the delivered lecture material well. However, the generated quizzes often lack originality and versatility. Therefore, further prompt engineering might be required to achieve more elaborate results.

Keywords: Self-Assessment; Software Engineering Education; Automatic Question Generation; GPT-4; Prompt Engineering

1 Introduction, Motivation, and Research Questions

In today's higher education, self-assessment quizzes are increasingly used to provide students with tests of what they previously should have learned [FB89]. Also, in the area of software engineering (SE) education at universities, these teaching methods are used and integrated into lectures. Self-assessment quizzes are usually separate, independent questions, such as multiple-choice questions (MCQs), and only offer students a test without any external assessment or feedback [Bo13]. Quizzes are also a growing component of digital education that is increasingly being delivered with course content in learning management systems (LMSs). Many LMSs allow manual insertion of self-assessments in the form of quizzes. This allows instructors to distribute their created self-assessment quizzes in lectures and remotely on the LMSs [Sw17]. However, creating self-assessments is challenging and can be tedious, especially because MCQs are time-consuming to create [Ga19]. It is not easy to come up with quizzes that are relevant, interesting, and of the right difficulty level, as it requires creativity and results in cognitive drain. Lecturers often lack time to create and offer self-assessments. Also, in higher education, too little emphasis is placed on student self-assessment, although it benefits students [An19]. In addition, self-assessment is not considered essential as it is less conventional than traditional forms of teaching. The integration and flexibility of the quizzes in the LMSs are relatively poor.

[1] University of Stuttgart, Institute of Software Engineering, Germany {niklas.meissner,sandro.speth,julian.kieslinger,steffen.becker}@iste.uni-stuttgart.de

doi:10.18420/seuh2024_04

Previous research has shown the potential of LLMs such as *GPT-3.5-turbo* for creating programming exercises in higher education [SMB23]. Based on the positive results of the programming exercise creation using *GPT-3.5-turbo*, the GPT LLM was selected in this paper, and the generation pipeline for creating self-assessment quizzes such as MCQs was developed. Thus, *EvalQuiz* conceptually serves as a pipeline for the automated generation of self-assessment quizzes using OpenAI's *GPT-4* [Op23] API. Therefore, our research questions are the following:

RQ1: *"How can the process of creating self-assessment quizzes in SE courses be automated to support lecturers in their teaching?"*

RQ2: *"How suitable are GPT-generated self-assessment quizzes in the field of SE in higher education?"*

In this paper, we tackle these problems and introduce *EvalQuiz*, a concept for the automated generation of self-assessment quizzes based on lecture material using large language models (LLMs). By answering these research questions, we aim to provide the basis for further research into the automated generation of self-assessment quizzes using LLMs. A demo video of the implemented *EvalQuiz* prototype is available on YouTube[2].

The remainder is structured as follows: In Section 2, we present the concept of quiz generation with the corresponding pipeline, topic modeling, and keyword extraction, prompt engineering and question generation, assembling and validation of generated quizzes, and evaluating their quality. The prototypical implementation of the concept is described in Section 3. Afterward, in Section 4, we examine our evaluation, including the process, results, and discussion. We discuss our threats to validity in Section 5 and outline related work in Section 6. Finally, we conclude the paper in Section 7.

2 Quiz Generation Concept

To ensure that the generation of questions is coherent from a didactic point of view regarding structure, content, and desired capabilities, close cooperation with experts in higher education didactics at the University of Stuttgart was established as part of the research. This involved focusing on the problems described above and deriving methods for the automated generation of MCQs on this basis. This chapter contains the concept of *EvalQuiz*, including the generation pipeline and the individual steps from uploading the lecture material to the final generated MCQs.

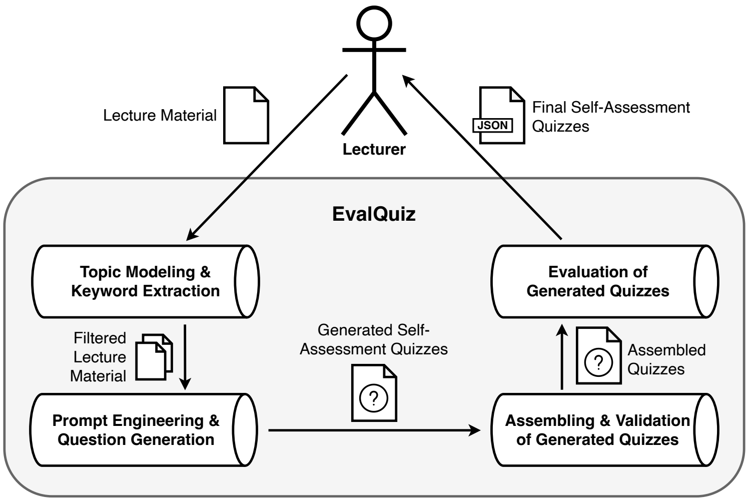
Lecturer
Lecture Material
Final Self-Assessment Quizzes
JSON
EvalQuiz
Topic Modeling & Keyword Extraction
Filtered Lecture Material
Prompt Engineering & Question Generation
Generated Self-Assessment Quizzes
Assembling & Validation of Generated Quizzes
Assembled Quizzes
Evaluation of Generated Quizzes

Fig. 1: Pipeline of quiz generation process.

2.1 Generation Pipeline

Figure 1 shows the conceptual structure of the self-assessment quiz generation pipeline. The lecturer provides lecture material and adds it to the pipeline. Lecturers can also specify configurations regarding how the questions should be generated or what focus they should have based on keywords and their relations and capabilities according to Bloom's taxonomy [BK20]. The first processing step is topic modeling & keyword extraction (Section 2.2), in which keywords for the quizzes are filtered, and texts from the lecture material are summarized. The filtered lecture material is then transferred to the prompt engineering stage (Section 2.3), where a prompt for the LLM is created, and the question generation is started. The generated quizzes are inserted into the assembling & validation stage of the pipeline (Section 2.4), where the generated questions are validated. Subsequently, the quality of the generated questions can be evaluated by lecturers and regenerated in case of poor generation quality (Section 2.5). To ensure a uniform listing and output of the self-assessment quizzes, we use the *JSON* format for the generated quizzes.

2.2 Topic Modeling and Keyword Extraction

LLMs only support prompts up to a particular token size. Unfortunately, lecture materials can exceed these boundaries, so reducing the input as much as possible without losing relevant information is crucial. This stage filters input documents to select parts of specific topics. There are various methods for processing large quantities of data as input. We assume

[2] https://www.youtube.com/watch?v=kC-9e2Bh4nQ

Type	Amount	Example
System message	1	“You are a question-generation assistant that supports generating questions in multiple fixed formats.”
Few-shot examples	n	Configurations & filtered input; “Your goal is to use the given markdown formatted text input to generate a question in the following JSON format: *example*”
Query message	1	“Give your answer in the specified JSON format.”

Tab. 1: Composition of question generation prompts.

a lecture material is topic-wise and conceptionally divided into chapters. One possibility is to split up the chapters and process each part individually. However, cross-references between different chapters become complex. It is essential to maintain the coherence of content with the keywords between chapters, so an algorithm is needed that produces a subset of the input text containing information of the related topics in a reduced form. First, the lecture material is uniformly transformed into *markdown* to simplify the search for keywords. Then, the *Word2Vec* [Ch17] topic modelfilters and summarize by keywords to consider the coherence of sentences, which is a strength of this topic model. *Word2Vec* uses a neural network model to learn word associations, works on a word-vector representation, and encodes words in a multidimensional vector space. This allows for synthesizing information based on sentences containing keywords summarized for a specific context.

2.3 Prompt Engineering & Question Generation

The filtered lecture material from the first stage and the specified configuration of the lecturers serve as input for the prompt engineering and question generation stage. This step composes a prompt for the LLM. In our prototype, we use OpenAI's *GPT-4* [Op23] API to generate the MCQs. The prompts are composed of a *system message*, which provides the frame of the prompt, then *few-shot examples*, which includes the content of the request and contains sample answers and formats, and a *query message*, which gives a concrete instruction for action. The structure of the prompts with examples of the individual components is shown in Table 1. The *system message* contains general instructions for the LLM to show what is important in the following messages and what needs to be done. The *few-shot examples* consist of the lecturer configuration and the filtered lecture material. It also contains the desired competence level [BK20], based on which the quiz will be generated. As the questions ultimately have to be formatted uniformly in *JSON* format, the few-shot examples also contain a *JSON* example that the LLM should be guided by. We set placeholders for the question-and-answer texts and the distractors in the provided *JSON* example for the creation of MCQs, which the LLM must replace with the generated question-and-answer options. The *query message* then emphasizes the generation based on the *JSON* example. Finally, the composed prompt is transmitted isolated to the LLM, i.e., in our prototype, the *GPT-4* API, and thus the generation of questions starts.

```json
{
    "question_type": "multiple_choice",
    "generation_result": {
        "question_text": "What does SDLC stands for?",
        "answer_text": "Software Development Life Cycle",
        "distractor_text": ["System Design Life Cycle", "Software Design Life
             Cycle", "System Development Life Cycle"]
    }
}
```

List. 1: Multiple-choice example as defined by *EvalQuiz* specification (JSON).

2.4 Assembling & Validation of Generated Quizzes

Once the LLM has generated the set of quizzes, they are tested and validated for specification matching in this stage. To use the output directly and deliver it back to the lecturer, the output must match the required specification, i.e., the *JSON* format with the corresponding attributes. The required specification of the output format and attributes used in *EvalQuiz* is shown as an example in Listing 1. The format of the generated questions is checked by an algorithm that matches the *JSON* output with the given specification. For this purpose, the generated LLM output is divided into three categories. (1) The output format matches the specification and can be used directly; (2) the output format roughly matches the specification, but further modifications need to be made, e.g., the attributes are misnamed. An improvement can be achieved, i.e., by re-executing the request so that the LLM generates a new response to the same prompt, as already experienced in previous work [SMB23]. (3) The output format does not match the specification, and the prompt must be rebuilt and revised in the second stage to improve the output.

2.5 Evaluation of Generated Quizzes

The last stage evaluates the quality of the generated and validated quizzes. Here, another prompt engineering process is used to transmit the generated questions back to the LLM with a new prompt and requirements for their evaluation. For instance, the question evaluation could be configured to focus on selected question-wording guidelines and check the generated questions against these. (1) The question uses simple language and is easy to understand. (2) The question should have a simple structure and avoid double negatives. (3) Suggestive questions should be avoided as they may be too obvious. (4) The questions have a temporal reference, and dates and time periods are stated precisely. (5) Use of concise answer categories. Closed questions should have disjunctive (non-overlapping) answer categories. (6) Unclear and necessary terms should be explained. (7) Terms associated with strong opinions and emotions should be avoided. (8) Questions must be unambiguous, leaving no room for individual interpretation. The questions are evaluated on these categories, and based on this, a decision is made on whether a question can be sent to the lecturer or whether it needs to be regenerated. All questions that have been fully evaluated and accepted are then passed on to the lecturer. In the *EvalQuiz* prototype (Section 3), the automated evaluation by the LLM of the generated questions described here is only partially implemented, requiring further specification of the respective criteria (1-8).

3 EvalQuiz Prototype

The prototype developed encompasses the concept described in Section 2 and is used to test and evaluate the concept described in Section 4. The implementation included three areas:

1. *Material server*[3]: handles the storage and management of the lecture material.
2. *Pipeline server*[4]: includes the described concept pipeline. This uses the stored lecture material and generates the respective MCQs.
3. *Frontend*[5]: provides faculty members a user interface (UI) to interact with the system, upload lecture material, and receive the generated questions.

The *frontend* is implemented using *React*, while the *backend* was developed using *Python*. The source code is open-source and available on GitHub and documented[6]. Based on experience in previous work [SMB23], we decided to use OpenAI's *GPT-4* as LLM for this prototype, which we call via its API. For space constraints, this paper will not further discuss our implementation methods. Furthermore, the prototype has not yet fully implemented all the functions described in the pipeline concept, e.g., the question evaluation stage is only partially implemented. However, a demo video of the prototype is available on YouTube[2].

4 Evaluation, Results and Discussion

This chapter outlines the evaluation of the concept. For this purpose, the implemented prototype was used and distributed to faculty members. The participants were obtained based on our professional network, all of them working in the field of SE education. The following subsections describe the evaluation process (Section 4.1), provide demographic information about the participants (Section 4.2), report the results (Section 4.3), and finally discuss the results (Section 4.4).

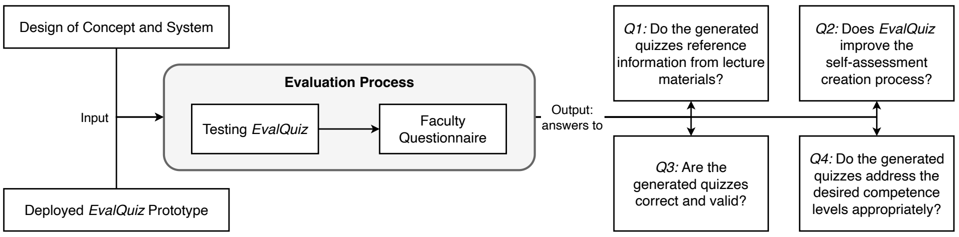
Design of Concept and System
Input
Deployed *EvalQuiz* Prototype
Evaluation Process
Testing *EvalQuiz*
Faculty Questionnaire
Output: answers to
Q1: Do the generated quizzes reference information from lecture materials?
Q2: Does *EvalQuiz* improve the self-assessment creation process?
Q3: Are the generated quizzes correct and valid?
Q4: Do the generated quizzes address the desired competence levels appropriately?

Fig. 2: Evaluation process.

4.1 Evaluation Process

The evaluation process is illustrated in Figure 2. The participants received an instruction, which included a description of the concept (available online[7]) and documentation of the prototype. In addition, we publicly deployed a version of *EvalQuiz* so that the participants could directly focus exclusively on testing the functionalities. They were invited to upload lecture material and generate questions in the deployed version of *EvalQuiz* and were then asked to describe their interaction, experiences, and impressions in the provided questionnaire. The questions and results are available online on Zenodo[8] as well as on the Microsoft Forms summary page[9]. The results of the survey are then used to answer the questions of whether *(Q1)* the generated quizzes reflect the uploaded lecture material, *(Q2) EvalQuiz* improves the self-assessment process, and *(Q3)* the generated quizzes are correct and valid.

4.2 Participants

In September and October 2023, six participants took part in evaluating and completing the questionnaire. The demographic information is included in the questionnaire responses[8]. Three participants (50%) reported being in the role of professor or lecturer, while two participants (33%) represented PhD students and one (17%) PostDoc position. Of the participants, two (33%) are in the research and teaching area of software engineering, two (33%) in computer science, one (17%) in machine learning, and one (17%) in cloud computing and systems architecture. Three participants (50%) indicated, that they are *"very experienced"* in creating assessments, two (33%) answered they are *"experienced"*, and one (17%) chose *"moderate"*.

[3] https://github.com/MEITREX/evalquiz-material-server
[4] https://github.com/MEITREX/evalquiz-pipeline-server
[5] https://github.com/MEITREX/evalquiz-client-react
[6] https://meitrex.github.io/evalquiz-material-server/,
https://meitrex.github.io/evalquiz-pipeline-server
[7] https://demo.hedgedoc.org/s/ZuHxrgpSb
[8] https://zenodo.org/doi/10.5281/zenodo.10040085
[9] https://bit.ly/evalquiz-questionnaire

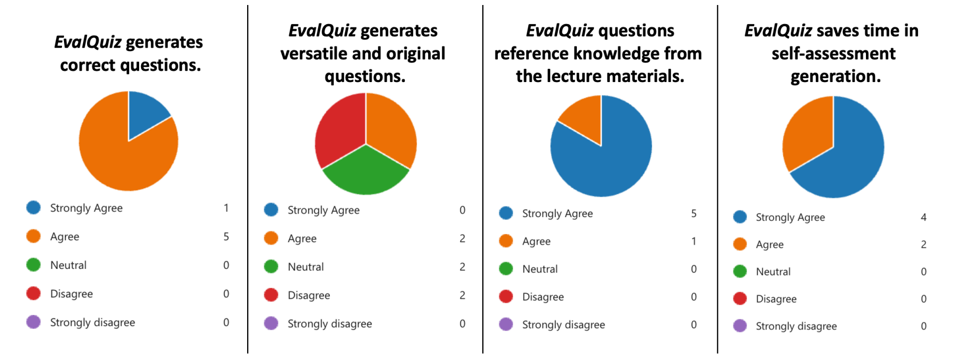

Fig. 3: Questionnaire results.

4.3 Results

Regarding general questions about self-assessment in SE education, the participants indicated that MCQs are most commonly used in their lectures (83%), followed by group discussion exercise slides ("blue slides") (67%), where students can test their knowledge during the lecture, and mock exams (67%). All stated that time constraints are the biggest challenges in self-assessment creation, followed by creativity (33%). 67% of them claimed that there would not be enough self-assessment in their lectures.

All participants responded that they had uploaded lecture slides, although one had also uploaded a paper. Regarding using *EvalQuiz*, all participants indicated that the tool generates correct questions. In addition, they agreed that *EvalQuiz* generates questions based on the uploaded lecture material's content and that using the tool saves time in self-assessment generation. However, participants were divided about whether *EvalQuiz* creates versatile and original questions. There, opinions were split to *"agree"* (33%), *"neutral"* (33%), and *"disagree"* (33%). Figure 3 depicts these results. Nevertheless, half of the participants agreed that the generation of questions is intuitive, and one disagreed (two neutrals). Moreover, half of the participants agreed that *EvalQuiz* gives the possibility to steer the question generation in the desired direction. The other half had a neutral opinion.

In response to the question of where *EvalQuiz* still has difficulties in the generation, the participants answered that some MCQs have incomprehensible or erroneous distractors. There were some suggestions for improvement, like the support of PDF files, which is currently still limited. In addition, further improvements in the generation of distractors and improvements of questions in the application context were requested. Further, future features were proposed, such as more question types, more variety and diversity in the questions, as well as a user intervention in the prompt engineering of the tool. Finally, participants mentioned that the *EvalQuiz* implementation is a good initial minimum viable product and that they would like to use it for their lectures. The detailed questionnaire results are documented on Zenodo[8] and Microsoft Forms[9].

4.4 Discussion

The evaluation shows that *EvalQuiz* improves the biggest bottleneck in self-assessment creation: ***time***. Feedback on the quality of the quiz generation is positive, while there are drawbacks in control and intuitiveness. The participants liked the iterative generation approach but wished they had more control. However, different concepts of each pipeline step influence the generation, which makes it more complex to give more control to the users. Overall, the uploaded lecture materials strongly influence the generation process.

Furthermore, the results from Section 4.3 can be used to answer the questions from the evaluation process, as *(Q1)* the generated questions using *EvalQuiz* address information from the lecture material, *(Q2) EvalQuiz* improves the self-assessment creation process, especially in the time dimension, and *(Q3)* the generated quizzes are correct and valid. The evaluation results provide valuable feedback, even if large-scale evaluations are required for deeper insights. Individual comments recommend possible further improvements. One participant states that he "would actually use [the system] to get some questions". The evaluation indicates that *EvalQuiz* has the potential to be used in education and can support lecturers in creating self-assessment quizzes. However, we expected our participants to use lecture materials besides lecture slides. Thus, books, wikis, exercises, and other lecture materials remain unexplored.

5 Threats to Validity

In this chapter, we discuss potential threats to the validity of our study. First, we discuss the internal validity, then external validity, and finally, the construct validity.

Internal Validity: The evaluation participants could know each other since the tool was only distributed in the authors' professional network. Therefore, participants could have talked to each other before testing *EvalQuiz* and filling out the questionnaire. However, we do not assume that the participants talked to each other beforehand since all of them used different lecture materials for the evaluation, and all of them were approached around the same time, so a short-term exchange is considered unlikely. In addition, the participants have different roles and backgrounds at the universities, so there could be differences in terms and wording. Nonetheless, we attempted to define all the essential terms in the questionnaire and describe them with examples.

External Validity: Performing a powerful evaluation of the concept and the tool with six participants is insufficient to make a representative statement about its general suitability. Nevertheless, we tried asking participants with different roles and backgrounds to get a broad spectrum of feedback and educational content. In general, participant responses may not be complete for our research questions, and there may be other opinions and application areas where *EvalQuiz* performs differently. Nevertheless, our results are valid answers to the two research questions but might require further investigation.

Construct Validity: We prepared the questionnaire questions before distributing them to the participants and ran a test run, which we excluded from the results. While conducting the evaluation of the tool and completing the questionnaire, we did not change any questions or modify any functionalities or attributes of the tool. Therefore, we assume that there is no threat to construct validity.

6 Related Work

Previous works focus on specific parts of the pipeline for the automated generation of self-assessment quizzes described in this paper. This chapter focuses on the related work of the individual pipeline elements and distinguishes them from this work.

Previous work by Majumder and Saha [MS15] focused on sentence selection from the input text, among other things, similar to how it is done in this work. They select sentences for the MCQ based on keywords, grammatical structures, the position and length of sentences, as well as pronouns, the completeness of context, and the similarities of definitions. While the approach is similar to our chosen one, it differs in the use of the tools, as in our approach, the input format is first transformed to markdown source code and then filtered for keywords exclusively with the use of *GPT-4*. In addition to exploring input selection, Majumder and Saha designed an MCQ generation system with topic modeling based on an existing topic modeling tool. For the generation, a reference set of MCQs is first collected from a domain. The reference questions are converted from an interrogative to an assertive form. The parse tree structure of the reference sets is analyzed and matched with the input text to find more candidates for MCQs. However, the authors did not use LLM to create the MCQ, and the approach has a few limitations, e.g., sentences could be selected that are too general to answer or omit information that is required for an explicit answer. Also, the generated distractors could be considered valid answers to the question, making the questions flawed.

Araki et al. [Ar16] present two strategies for generating question sentences from multiple input sentences using semantic text analysis. Their method works with texts annotated by experts. Distractor generation is implemented by searching for annotations similar to the answer. A limitation is that questions and distractors are sometimes prone to grammatical errors [Ar16]. While using a different approach for the generation, the authors also tested their approach only in the field of biology. Generalizability to other domains, such as SE, is unclear. Klein and Nabi [KN19] also developed a system for generating questions and answers using transformer models and argued that the two tasks are closely related and can benefit from each other. The authors use BERT and GPT-2, respectively, and argue that GPT-2 is suitable for generating questions and BERT is suitable for answering questions [KN19]. However, the authors did not investigate any generations of questions based on the lecture material utilized, and furthermore, the chosen LLMs are outdated nowadays.

Tsai et al. [TCY21] present a method for retrieving sentences with relevant keywords in a Python learning course using BERT and creating self-assessment questions using GPT-2. They stated that BERT is fine-tuned for better domain-specific keyword recognition, which improves the performance of correct keyword recognition from 94% to 98% [TCY21]. However, examples of falsely recognized keywords in a sentence are not transparently highlighted. Also, the domain-specific fine-tuning does not apply to lecture materials outside the domain of Python programming. Also, Nguyen et al. [Ng22] present and evaluate an approach for creating MCQs in a data science course using MOOCCubeX as a topic modeling tool [Ng22]. However, another LLM (Google T5) is used, and the authors do not provide a fixed format for the generated questions.

7 Conclusion & Future Work

In this paper, we present a concept for the automated generation of self-assessment quizzes in the software engineering (SE) education domain. For this purpose, we built a pipeline that collects lecture material from lecturers, processes them using topic modeling and keyword extraction, generates a prompt, and thus generates, validates, and evaluates questions about the material. The final generated self-assessment quizzes are then delivered to the lecturer as output. The results of the evaluation conducted on the concept and implemented prototype show that lecturers perceived the generation of questions as time-saving and were able to generate questions quickly. Furthermore, most lecturers stated that the generation of questions is reliable, and all of them stated that *EvalQuiz* generates correct questions. Based on these results, the first research question *(RQ1)* can be answered in a positive way, and *EvalQuiz* supports lecturers in creating self-assessment quizzes. The second research question *(RQ2)* can be answered by indicating that all lecturers stated that the generated questions fit the presented lecture material. Nevertheless, lecturers were undecided whether the generated questions were original and versatile, which could make it difficult to use the generation for many similar SE materials. However, based on the results, *RQ2* can also be considered with a positive result and the generation of self-assessment quizzes using *GPT-4* is suitable in the field of SE in higher education.

In future work, we investigate how students perceive the automated self-assessment quizzes and evaluate them accordingly. We also aim to optimize the generation of self-assessment quizzes to generate more original and versatile quizzes. We also integrate the pipeline into an intelligent tutoring system concept [MSB23] to gain sufficient student analysis items through the automated generation of MCQs without requiring any extra effort from faculty members. Furthermore, we test the performance of other open-source LLMs.

Acknowledgement

We acknowledge the support of the Ministerium für Wissenschaft, Forschung und Kunst Baden-Württemberg (MWK, Ministry of Science, Research and the Arts Baden-Württemberg under Az. 33-7533-9-19/54/5) in Künstliche Intelligenz & Gesellschaft: Reflecting Intelligent Systems for Diversity, Demography and Democracy (IRIS3D) and the support by the Interchange Forum for Reflecting on Intelligent Systems (IRIS) at the University of Stuttgart.

References

[An19] Andrade, H. L.: A Critical Review of Research on Student Self-Assessment. Frontiers in Education 4/, 2019, ISSN: 2504-284X.

[Ar16] Araki, J. et al.: Generating Questions and Multiple-Choice Answers using Semantic Analysis of Texts. In: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers. The COLING 2016 Organizing Committee, Osaka, Japan, pp. 1125–1136, 2016.

[BK20] Bloom, B. S.; Krathwohl, D. R.: Taxonomy of educational objectives: The classification of educational goals. Book 1, Cognitive domain. longman, 2020.

[Bo13] Boud, D.: Enhancing learning through self-assessment. Routledge, 2013.

[Ch17] Church, K. W.: Word2Vec. Natural Language Engineering 23/1, 2017.

[FB89] Falchikov, N.; Boud, D.: Student Self-Assessment in Higher Education: A Meta-Analysis. Review of Educational Research 59/4, pp. 395–430, 1989.

[Ga19] Gamage, S. H. P. W. et al.: Optimising Moodle quizzes for online assessments. International Journal of STEM Education 6/1, p. 27, Aug. 2019.

[KN19] Klein, T.; Nabi, M.: Learning to answer by learning to ask: Getting the best of gpt-2 and bert worlds. arXiv preprint arXiv:1911.02365/, 2019.

[MS15] Majumder, M.; Saha, S. K.: A System for Generating Multiple Choice Questions: With a Novel Approach for Sentence Selection. In: Proceedings of the 2nd NLP-TEA workshop. Pp. 64–72, 2015.

[MSB23] Meißner, N.; Speth, S.; Breitenbücher, U.: An Intelligent Tutoring System Concept for a Gamified e-Learning Platform for Higher Computer Science Education. In: Proceedings of SEUH 2023. GI, Feb. 2023.

[Ng22] Nguyen, H. A. et al.: Towards Generalized Methods for Automatic Question Generation in Educational Domains. In: Educating for a New Future: Making Sense of Technology-Enhanced Learning Adoption. Springer International Publishing, Cham, pp. 272–284, 2022, ISBN: 978-3-031-16290-9.

[Op23] OpenAI: GPT-4 Technical Report, 2023, arXiv: 2303.08774 [cs.CL].

[SMB23] Speth, S.; Meißner, N.; Becker, S.: Investigating the Use of AI-Generated Exercises for Beginner and Intermediate Programming Courses: A ChatGPT Case Study. In: 2023 IEEE 35th International Conference on Software Engineering Education and Training (CSEE&T). Pp. 142–146, 2023.

[Sw17] Swart, A. J.: Using reflective self-assessments in a learning management system to promote student engagement and academic success. In: 2017 IEEE Global Engineering Education Conference (EDUCON). Pp. 175–180, 2017.

[TCY21] Tsai, D. C. L.; Chang, W. J. W.; Yang, S. J. H.: Short Answer Questions Generation by Fine-Tuning BERT and GPT-2. In: Proceedings of the 29th International Conference on Computers in Education Conference, ICCE. 2021.