A Review of Automatic Question Generation in Teaching Programming

Jawad Alshboul, Erika Baksa-Varga
University of Miskolc, Faculty of Mechanical Engineering and Informatics, Miskolc, Hungary

***Abstract*—Computer programming is a complex field that requires rigorous practice in programming code writing and learning skills, which can be one of the critical challenges in learning and teaching programming. The complicated nature of computer programming requires an instructor to manage its learning resources and diligently generate programming-related questions for students that need conceptual programming and procedural programming rules. In this regard, automatic question generation techniques help teachers carefully align their learning objectives with the question designs in terms of relevancy and complexity. This also helps in reducing the costs linked with the manual generation of questions and fulfills the need of supplying new questions through automatic question techniques. This paper presents a theoretical review of automatic question generation (AQG) techniques, particularly related to computer programming languages from the year 2017 till 2022. A total of 18 papers are included in this study. one of the goals is to analyze and compare the state of the field in question generation before COVID-19 and after the COVID-19 period, and to summarize the challenges and future directions in the field. In congruence to previous studies, there is little focus given in the existing literature on generating questions related to learning programming languages through different techniques. Our findings show that there is a need to further enhance experimental studies in implementing automatic question generation especially in the field of programming. Also, there is a need to implement an authoring tool that can demonstrate designing more practical evaluation metrics for students.**

Keywords*—Question generation; question generation techniques; automatic question generation; teaching programming

I. Introduction

A. Background

In recent years, a number of researchers have been attracted from different disciplines toward automatic question generation for educational purposes [1] [2] [3]. The researcher in [4] defined question generation as "It is an activity of automatically generating questions for different inputs like raw text through semantic illustration". The definition shows that the generated question type can differ like being a sentence, a semantic map, or a paragraph. The educational question generation is not a new concept, but it has a long history and can be traced back to the use of logic in questions [2], [4], [5]. Cohen is the pioneer of this research area who initially proposed the content for generating questions through an open formula and used one or more unbound variables [1]. Whilst research on generating questions is carried out for a long time, using techniques for automatic question generation for teaching computer programming has raised interest only recently amongst various research communities, because it requires inclusion of cognitive science, natural language processing and human interaction with computers. Intelligent Tutoring System (ITS) is the recently proposed computer-based teaching context to help students learn programing languages but to the best of our knowledge, computer-based applications for teaching programming are not widely implemented [2]. With rising novice computer scientists, specific questions are generated which can address knowledge gaps that were often negligible in the manual process of articulating questions [6]. In support of enhancing the metacognitive skills of students, asking students to generate questions can be a constructive process but the use of various metacognitive skills can be time-consuming and needs extended knowledge of various metacognitive strategies.

B. Problem Statement

The success of a rule-based question generation (QG) method depends on the quantity and quality of teachers' domain knowledge, language knowledge and the amount of time spent on it. On the other hand, data-driven techniques have recently emerged such as deep neural network-based methods that are considered a promising approach for different tasks like recognizing entities, and sentiment categorization. In particular, teaching programing skills to students require extensive motivation which cannot be an easy task from the teachers' perspective. A key issue is, that students interact with a program that has limited, syntactic and sematic level knowledge of a programming language (for example a compiler or interpreter), while teachers know and therefore can teach also the logic of programming. This requires teachers' knowledge of how to apply the key concepts of computer programming and teach them to students with appropriate models of Automatic Question Generation (AQG).

The research in [7] provided a question generation method considering a generative encoder-decoder model. The researchers in [5] mentioned that although there are advances made in the neural models for automatic question generation, there is still a gap indicating a comprehensive survey on how different learning paradigms can present improvements in automatic question generation that broadens the input spectrum of instructors for teaching programming. This shows that various studies have been conducted on automatic question generation process for educational purposes, but to the best of our knowledge, there are only a few studies reporting on the state-of-the-art techniques used for automatic question generation in teaching programming languages, especially for the previous six years i.e. 2017 till 2022. Thus, the objective of this study is to review recent studies that provided different

techniques for automatic question generation with reference to various methods required for teaching programming.

This review extends the systematic review previously conducted on automatic question generation by Kurdi et al. [8] that covered literature from 2014 and up to 2019, respectively, by focusing on the domain of computer programming. The extensive amount of research that has been published since Kurdi et al, an extension of these studies is reasonable at this stage as it helps in understanding recent developments in the given field. The present article uses the previous article by Kurdi et al. [8] as a starting point and extending the review in a number of ways like additional review questions and criteria for inclusion/exclusion, which is discussed in the methodology section.

C. Research Questions

The questions that will be addressed in this research comes as follows:

- What are the recent developments made in the techniques for automatic question generation in teaching programming?
- What is the difference in the state of the field focusing on before COVID-19 and after COVID-19 period?
- What can be the future directions based on identified gaps in the field of automatic question generation in teaching programming?

D. Research Objectives

To address research questions the following objectives are considered for this work:

- To review recent literature on the approaches and techniques used for automatic question generation in teaching programming.
- To analyze and compare the state of the field before COVID-19 and after COVID-19 period.
- To provide summary of the challenges and future directions in the field of automatic question generation in teaching programming.

The remaining parts of the paper is structured as follows: Section II represent a literature review which will address previous work on the field. Section III describes the methodology used to conduct this review. Section IV shows the analysis and findings for the papers considered. Section V concludes the paper. Finally, Section IV describes future directions of this work.

II. Literature Review

Manual question creation consumes much time and labor. The concept of question generation was envisioned back in 1960s [4]. It is generally estimated that learning scientists believe that the provision of high-quality learning questions must use the basics of language knowledge and domain knowledge and so approach the activity using “rule-based” technique [9]. The given approach used syntactic changes to transform declarative sentences into questions. For example, multiple choice questions were generated using rules of term extraction. Next, questions were sometimes generated using over-generate-and-rank manner that helped in ranking questions [7]. The given methods were limited in their wider applications and rules were also based on a few subjects, like English language. They were not easily applied to other domains, since defining rules and procedures needed considerable efforts from expertise. The given methods are also very limited and do not offer high quality questions, thereby limiting the implementation of rule-based generators. However, entering into digital landscape required wider-scale online learning; hence, the demand for automatic question generation is also increased along with massive number of online courses available to learners. In order to address this need, various computational techniques like deep neural network-based state-of-the-art techniques were proposed [7] [6]. Du et al [4] is pioneered in providing encoder-decoder sequence learning that was later on used for automatic question generation. The given model automatically captured question-asking patterns without taking any help from hand-crafted rules, indicating supreme performance over previous rule-based methods. However, it had a major gap identified by scholars that this technique was helpful in collecting data for machine reading comprehension tasks. Notably, such datasets included a very limited number of useful questions for learning, indicating that extended research is needed to offer question generations for complex learning documents to facilitate teachers.

Assessing students’ answers manually is also a time-consuming mission. Depending on the type of the question, this action can also be automatized more or less. Question types of quizzes are generally categorized into two groups: objective questions and subjective questions. Objective means that there is only one correct answer, which can be checked automatically. The objective questions request the learners to select the correct answer from several options or offer a number of words/short sentences as options or to complete a sentence. Multiple-choice, matching, true-false, and fill-in-the-blank are examples of objective questions, and they are considered the most popular ones, since they provide automatic assessment methods due to its their unambiguous, quick and trusted evaluation process. On the other hand, the subjective questions ask for a written answer composed by the learners and such answer might be short or long. A short answer could be up to three sentences while a long answer might be as long as an essay. The subjective questions should have an increased attention by the teachers to assess learner’s deep knowledge and understanding of the topics, similarly to the traditional education system that have been employed for centuries [10].

Most researchers have focused on generating objective-type questions, automatically or semi-automatically, while limited focus has put on subjective question generation because scoring is a difficult challenge in the case of subjective assessment. Comparing the level of difficulty to generate questions and in assessing the learner’s answers, learner's assessment of the textual question types is very easy for close questions and multiple-choice questions, easy for open-close questions, and difficult for subjective questions. Recent advances in deep learning-based natural language processing (NLP) offer promising solutions in answer assessment of objective-type questions [10].

A. Automatic Question Generation Process

In recent years, there has been an emerging interest in AQG from the text, which is explained as an activity to generate a question from a passage and optionally an answer. According to [11], AQG helps in curating question-answer datasets and improves our experience in Artificial Intelligence (AI) systems and helps in designing educational materials. But for this purpose, it is equally important for the questions to be grammatically error-free and answerable. Pertinent approaches emphasize on encoding the whole passage as the relationship is shown between them using complicated operations and then questions are generated in one single passage [12]. However, empirical studies show that if careful analysis is carried out for question generation, there are some approaches that probably miss out one or more of the essential aspects of the questions. For example, as mentioned in Table I, the question is generated by the single-pass baseline model that is grammatically correct but it does not fit to the answer [3].

TABLE I. THE PASSAGES AND THE QUESTIONS GENERATED FROM THEM USING REFNET [8]

Passage 1: Functioned by Napoleon army in 1800, Warsaw was given the authority of the newly established Duchy of Warsaw
Questions: Baseline: What was the governor of the newly established Warsaw? *RefNet: Who gave freedom to Warsaw?* *Reward-RefNet: Whose army was given freedom in 1800?*
Passage 2: In order to fix the process of carbon dioxide into the sugar molecules, the enzyme called rubisco is used by chloroplasts.
Questions: Baseline: What is used by chloroplasts? *RefNet: What is used by chloroplasts to fix carbon dioxide?* *Reward RefNet: What is used by chloroplasts to fix carbon dioxide into the sugar molecules?*

Table I shows that there is a visible scope of enhancing the general quality of the question generation process in contemporary field of teaching [6]. In the literature, scholars argue that one way is to approach this by constantly re-visiting the passage and then answer them with the goal to refine the preliminary draft by producing a better question in the second stage and then bringing further enhancements in the further stages [1][8]. This can be done using a comparison between human process of generating questions and with the computer-generated models [3]. In the examples above RefNet (Refine Network) was used as the model to evaluate the initial questions generated and then carried out a second test to generate revised questions. The Reward RefNet used explicit reward signals to attain the refined questions with two attributes: fluency and answering. This RefNet is a sequence-to-sequence model that includes two decoders called Initial and Refined decoder. According to [13], the proposed dual model helped generating the final question by revisiting the adequate parts of the input passage and preliminary draft.

B. Automatic Question Generation in Teaching Programming

The Literature exhibited various approaches for automated generation of questions, based on the extraction of featured words given on the topic and established variations on the same question for object-focused programming quizzes. For example, [13] used programming-by-example for code snippets that was a test regulated by synthesis. Recently, the research article [15] provided facts for generating questions for teaching programming languages. The study reported in [16] found that students are taught by teachers to produce a program but they are not fully sure about the use of their own codes as they failed to grasp basic concepts. But if students are taught using Quality License Scheme (QLS,) it can prompt them to have a deeper level of understanding of given concepts. Schulte's Block Model of Program is another model used by scholars to attain better understanding of programming that needs different levels of knowledge about the program, and its purpose. However, experts argue that such models need knowledge at different levels ranging from individual building blocks to the integration of constructs to complete deficient Research shows that we cannot evaluate how coding process works as it requires skills and debugging process and the aptitude of learners to trace code that helps them write automatically testable code [18]. It is hypothesized that writing codes in a programming language requires preceding abilities of learners to write comparable codes that can be better understood by questions about learner's code (QLCs) approach that helps in learning programming comprehensions. On the other hand, in 2019, scholars of the George Washington University created a software that produced program-tracing questions for introducing programming content to students [12]. But this system also failed as the majority of questions were not applicable due to their complex nature. Later, scholars used Turing test as the tool to determine algorithm-based questions and identify the extent to which they could be helpful for teachers to generate questions that analyze students' learning abilities [14]. But this Turing test also did not pass as it failed to identify the particular issue in the algorithm when generating questions. Later and to extend this work, the researchers in [3] proposed a system called QuizJET, as another tool for generating questions, especially for teaching Java. This QuizJET was based on template-based questions that was linked to different concepts of Java and was actively used for generating quizzes; however, the research conducted in [16] did not find any correlation between the success rates of QuizJET and students' home assignment work, because QuizJET was more focused on understanding programs while home assignments were focused on writing skills of computer programming learners. Despite all these, QuizJET was identified as a valuable source in preparing students for exams.

III. METHODOLOGY

This study uses systematic literature review as the methodological approach to assess and understand the pertinent literature linked with research questions. From a number of approaches available in the literature for systematic review of studies, this study uses the approach provided by the researchers in [16] who provided detailed guidelines for conducting reviews. The methodological review consists of three key stages which in turn included ten sub activities. The details are showcased in Fig. 1. In the preliminary stage, the given questions were addressed; what are the key areas for which question generation is being designed; what are the different techniques and tools used for question generation process in teaching programing; what are the different modes of delivery for the creation of question generation in teaching programming; and how authors validated such techniques.

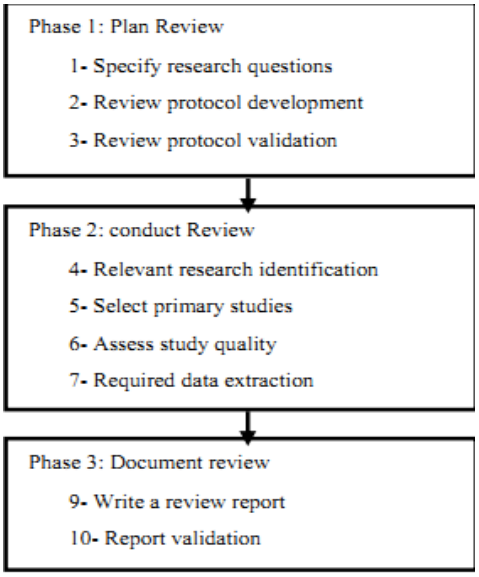

Fig. 1. Phases of Review Adopted from [16].

The questions were generated to complete the preliminary stages and then the sub-activity was carried out by reviewing protocols for the study. The review included determined time span for the published papers, and keywords along with the sources of publications to find most accurate data. The focus of research articles spanned for the last six years i.e. 2017 till 2022 and the sources were obtained from IEEE Xplorer, science direct, Google scholar, and SCOPUS. Table II shows the sources and keywords used to search the materials. Whereas Table III presents the considered papers in each year.

The search process offered a set of total 100 articles and they were then screened on the basis of their titles and abstracts. After filtration, only 18 articles were left that met the inclusion criteria of this study. In order to strengthen the search linked with the articles, backward snowballing provided by [19] was also adopted to identify the most cited articles. This step was ensured to address questions of the study in an effective manner so that no question was left unanswered. The inclusion criteria of this study are based on the research questions of this study and in the end, this study only included those articles that were accurately linked to the research questions of this study.

TABLE II. REVIEW TABLE

Year	Sources	Key words
2017-2022	IEEE Xplorer, Science direct, Google Scholar, SCOPUS	question generation techniques, teaching programming, automatic question generation techniques

TABLE III. YEAR WISE SEARCH RESULTS

Year	Number of papers
2017	4
2018	2
2019	3
2020	3
2021	5
2022	1

IV. RESEARCH ANALYSIS AND DISCUSSION

This study has reviewed a total of 18 papers for AQG in teaching programming in the past six years from 2017 to 2022. Older papers are excluded from the search criteria. Most of them used automated evaluation tools [4][5] [6] [14] [17] or they used automated contexts for programming languages [20] [21] [15]. It was also identified that generating feedback for the questions produced to teach programming languages are equally important. Most of the tools were used to grade student solutions but there were some that offered extended feedback and could be used to support learning process of students.

The study conducted in [21] described the attributes of the tools and identified challenges and some future directions. However, a study by [19] selected some papers and mentioned qualitative elements at the time of evaluating tools for question generation. It was also found that the majority of the studies i.e. 12 out of 18 studies lacked comprehensiveness and the scope of the tools varied immensely. Tools are mostly grouped but there is no such agreement on the naming of different groups. Eight papers discussed technical aspects of teaching tools used for programming languages.

As observed from Table IV, studies [1][3][4][5][14][21] used web-based solution to generate questions for students on the online platform. Whereas some of them proposed a theoretical framework like [7][13] [15][22] and the rest of the studies proposed it as a computer application [12] [16][19] [23] [24]. The main observation on the literature is that most of the applied applications use web-based solution to generate general questions from existing materials using different techniques like Artificial Intelligence, Machine Learning, Deep Learning, and traditional custom-made algorithms.

This review paper is different from other reviews as it focused on the dimensions of generating questions and the feedback in teaching programming tools by closely evaluating the various types of feedback provided on the techniques.

A. *Comparison of State-of-the-Field before Covid-19 and after Covid-19*

The findings of the study show that teachers require various kinds of tools and techniques when they generate new programming questions. In order to find out the state of the field before COVID-19 and after COVID-19, a review of the techniques used for generating questions by programming teachers shows that varying techniques have been employed to address certain needs of the instructors. The analysis also showed that prior to COVID-19, scholars have been involved in generating questions for teaching programming using Bayesian Network [5], ITSB tool (Delphi IDE) [2], Artificial Neural Network (ANN) based technique [13][17] [9], and ANN combined with Vortex Optimization Algorithm [7]. All these tools were either based on web-based or experimental or theoretical framework that does not exhibit strength of one technique over another technique for generating questions. Some interesting facts were revealed during the analysis. These are the following: 1) after COVID-19, the instructors valued code-writing question generation techniques like Junit that is a web-based tool and used user feedback as a validating tool. 2) Next, ITA [12], JAVA software, C programming for generating

questions, CodeTraining [15], dynamic codes [16] and PQG model [23] were employed by instructors for generating questions that were identified as the key source of positive ratings in the assessment of students during exams. 3) The analysis of articles published after COVID-19, i.e. from 2020 till 2022, valued code-tracing procedures as compared to pre-COVID-19 that seems a major difference between the two periods. 4) Before COVID-19 the focus was on supporting students in learning (finding answers for questions from long texts). Whereas after COVID-19 the focus has shifted to supporting teachers in generating online quizzes and assessing student assignments automatically. This difference also exhibits that the extensibility of tools used for generating questions got significantly positive ratings from users as compared to pre-COVID-19 tools [23]. This finding is interesting as it identified that the models used after COVID-19 address the needs of particular users and provided only extensible questions to the instructors to be used after edits in practice. On the other hand, the analysis also shows that the automatic question generation process through code-tracing method is very limited and can generate questions when the topic is wide and it is difficult for instructor to cover all ideas when making the questions. One plausible explanation could be, that instructors now anticipate the questions to be more in line with the content they teach in the class.

TABLE IV. ANALYSIS OF ARTICLES

Year of publication	Title of the study	Domain	Aim	Tool for generating questions	Mode of delivery	Validating study
2017	[5]	Computer Science	Teaching programming language	Bayesian Network	Web-based	Experimental
2017	[2]	Computer Science	Teaching programming language	ITSB tool (Delphi IDE)	Web-based	User feedback
2017	[7]	Computer Science	Teaching programming language	Artificial Neural Network based technique & Vortex Optimization Algorithm	Theoretical framework	Experimental
2017	[13]	Computer Science	Teaching programming language	Artificial Neural Network	Theoretical framework	User feedback
2018	[4]	Computer Science	Teaching programming language	Bayesian Network	Web-based	Student feedback
2018	[18]	Computer Science	Teaching programming language	Automatic Item Generation (AIG)	test-item templates	User feedback
2019	[1]	Computer Science	Teaching programming language	ITA	Web-based	Teacher feedback
2019	[23]	Computer Science	Teaching Programming Language	SQL question Generation (DB-Learn)	Computer based application	User feedback
2021	[19]	Computer Science	Teaching programming language and question generation	Question Similarity mechanism.	Application	Teacher feedback and user response
2019	[21]	Computer Science	Teaching programming language	ITA	Web-based	Experimental
2020	[3]	Computer Science	Teaching programming language	JUnit	Web-based	User feedback
2020	[12]	Computer Science	Teaching programming language	ITA	Computer based application	Student performance
2020	[22]	Computer Science	Teaching programming language	AA	Theoretical framework	Teacher feedback
2021	[6]	Computer Science	Teaching programming language	JAVA software	Web-based	Student performance
2021	[14]	Computer Science	Teaching programming language	Java Programming Course	Web-based	Student performance
2021	[15]	Computer Science	Teaching programming language	CodeTraining	Theoretical framework	User feedback
2021	[16]	Computer Science	Teaching programming language	Dynamic codes	Computer based application	Student performance
2022	[24]	Computer Science	Teaching programming language	PQG model	Computer based application	Student performance

B. Techniques of Automatic Question Generation in Teaching Programming

Intelligent Tutoring Systems (ITS) are mainly identified as major systems for teaching programming domain. The analysis of articles showed that there are some general ITS techniques such as the tools that use model tracing for analyzing the process followed by students to solve problems in programming languages. The authors contrasted production rules and buggy rules in [16] while other researchers in [19] used constraint-based modeling techniques that is based on three levels, logical, empirical and data-based constraints. This model has a major limitation for how to conduct loop and eliminate the chances of generating error messages that could affect teachers' ability to generate questions. This study identified theoretical and practical gaps in the literature that could be used as future direction by scholars.

Dynamic code analysis through automated testing is another technique used by [13] as the way to teach programing and then generate questions for analyzing students' abilities and knowledge skills. According to [25], this was identified as the major type of automated testing but it lacked modern techniques necessary for unit testing and property-based testing, mostly executed through pertinent test frameworks like JUnit. In order to address this limitation, the research in [19] used another technique, called basic static analysis and identified the misunderstood concepts and the inadequacy of code structures. However, recently the article [15] also concluded that Program Transformations are another language processing tool that reduces the syntactical complications and help instructors to produce same level of abstraction when designing questions. Notably, all the tools identified during the analysis of articles showed that they fall within two major categories; automated assessment (AA) and intelligent tutoring system (ITS). Automated assessment focused on the use of tools that evaluate students' abilities to solve questions with a feedback report, but ITS helps students to reach solutions with novice feedback that is helpful for teachers to generate questions [17][18].

A recent study conducted by [15] mentioned that previous meta-analysis methods have been used to recognize the factors affecting students' performance but this study is pioneered in evaluating students' computer programming skills through Educational Data mining approaches. The authors stated that there are various categorizations of algorithms for identifying student's performance studying computer programming and the most effective is to use pooled approach to estimate students' performance progress in programming as their educational domain. The authors tried to identify the probable sources of heterogeneity by using subgroup analysis and sensitivity analysis; however, a major gap is identified in defining the sources of variability as authors were unable to establish any in their cases. It is highly likely that the key reason for this colossal heterogeneity was linked with some studies that were obtained from the varying sample sizes; however, it needed further research that could help in adopting different algorithms for assessing student performance.

A study by [16] provided important contribution in generating quizzes for C programming language. The key contribution of this study is that authors solved the programming questions and generated questions by following an entity discovery approach. It was estimated during the analysis of the question generation process that teachers and students can use them for solving quizzes and attain concepts in a better way. However, this study has a major limitation of practical knowledge related to precision and inclusion of more features like mining answers from the posts and grouping the questions into different levels of their difficulty. On the other hand, an article by [12] generated questions for teaching C programming through JAVA software application that illustrated the likelihood to produce automatic quizzes. It is estimated that if students have acquired knowledge for C programming language, students are able to learn other programming languages in a better way. This paper has a key strength to test the students' knowledge about how to define the C language functions; however, knowledge gap is identified about how students can enhance their aptitude skills for programming in other languages. This restricts the generalizability of the study on learning other programming languages. Although an article by [12] used educational software for studying JAVA language and then assessed their skills towards basics of object-focused programming. The authors provided very important contribution by generating automatic quizzes for JAVA programming through six different types of parametrized questions. This particular technique had major implication in theory as every time the test is conducted to test comprehensive skills of learners, new questions are generated. However, this article did not specify how learners can answer accurate questions based on their programming knowledge. CodeTraining is a very new approach that used an authoring tool for Gamified Programming Learning Environment; however, this particular tool lacks approaches on how teachers can create questions through the integration of different resources [25]. It is also suggested that future scholars can extend knowledge based on Gamification Programming Learning Environment by conducting experimental studies and through the use of an authoring tool that demonstrates how to design questions relevant to the course.

V. Conclusion

The objectives of this study are attained by offering a picture of the existing state of teaching programming and the tools used for generating questions. We have discussed the gaps in each article and the limitations of each article that could be used as future research directions. The article also addressed the objective related to comparing state of the field situation in the pre-COVID-19 and during COVID-19 periods. During the review of the studies, various effective methods were identified to that can be helpful in generating questions. A total of 18 papers were analyzed from the year 2017 up to 2022 that were relevant to the given topic of the study. The key techniques included AA and ITS and in particular, Dynamic code analysis, JUnit tests, JAVA programming software, CodeTraining, Program Transformations, PQG model and Educational Data mining approaches. Some gaps are related to the inefficacy linked with the models and the techniques adopted by scholars for generating questions. The inability to properly evaluate students' performance and abilities is related to the quality of the data used for that evaluation. Future studies may seek to

focus on the development of techniques based on key approaches identified in this study to improve the applicability of techniques on a wider scale and in practical context. Furthermore, interested researchers can work on the identified promising research topics in question generation for educational purposes.

VI. Future Research Directions

Based on the literature, the following points represent promising research topics for the interested researchers in question generation for educational purposes:

- Implementing question generation approaches to generate questions on programming languages topics.
- Enhancing experimental reporting, standardizing evaluation metrics, and studying and developing more practical evaluation metrics.
- Extracting informative sentences from existing sentences need to be improved.
- Generating questions from several sentences and summarizing sentences based on their relations need to be explored.

Acknowledgment

The authors gratefully acknowledge the financial assistance from the Institute of Information Science, Faculty of Mechanical Engineering and Informatics, University of Miskolc.

References

[1] A. A. Soofi and M. Uddin, "A Systematic Review of Domains, Techniques, Delivery Modes and Validation Methods for Intelligent Tutoring Systems," International Journal of Advanced Computer Science and Applications, vol. 10, no. 3, 2019, doi: 10.14569/IJACSA.2019.0100312.

[2] C.-P. Wu and S.-L. Wu, "Development of a Web-Based Learning System to Engage Students in Question Generation Activities," International Journal of Future Computer and Communication, vol. 6, no. 3, pp. 119–122, Sep. 2017, doi: 10.18178/ijfcc.2017.6.3.502.

[3] K. Cunningham, R. A. Bejarano, M. Guzdial, and B. Ericson, "'Im not a computer': How identity informs value and expectancy during a programming activity," in 14th International Conference of the Learning Sciences, 2020, pp. 705–708.

[4] A. L. Santos, "Enhancing Visualizations in Pedagogical Debuggers by Leveraging on Code Analysis," in Proceedings of the 18th Koli Calling International Conference on Computing Education Research, Nov. 2018, pp. 1–9. doi: 10.1145/3279720.3279732.

[5] C. Vieira, A. J. Magana, M. L. Falk, and R. E. Garcia, "Writing In-Code Comments to Self-Explain in Computational Science and Engineering Education," ACM Transactions on Computing Education, vol. 17, no. 4, pp. 1–21, Sep. 2017, doi: 10.1145/3058751.

[6] D. Moonsamy, N. Naicker, T. T., and R. E., "A Meta-analysis of Educational Data Mining for Predicting Students Performance in Programming," International Journal of Advanced Computer Science and Applications, vol. 12, no. 2, 2021, doi: 10.14569/IJACSA.2021.0120213.

[7] E. de Angelis, F. Fioravanti, A. Pettorossi, and M. Proietti, "Semantics-based generation of verification conditions via program specialization," Sci Comput Program, vol. 147, pp. 78–108, Nov. 2017, doi: 10.1016/j.scico.2016.11.002.

[8] G. Kurdi, J. Leo, B. Parsia, U. Sattler, and S. Al-Emari, "A Systematic Review of Automatic Question Generation for Educational Purposes," Int J Artif Intell Educ, vol. 30, no. 1, pp. 121–204, Mar. 2020, doi: 10.1007/s40593-019-00186-y.

[9] H. Keuning, J. Jeuring, and B. Heeren, "A Systematic Literature Review of Automated Feedback Generation for Programming Exercises," ACM Transactions on Computing Education, vol. 19, no. 1, pp. 1–43, Jan. 2019, doi: 10.1145/3231711.

[10] B. Das, M. Majumder, S. Phadikar, and A. A. Sekh, "Automatic question generation and answer assessment: a survey," Res Pract Technol Enhanc Learn, vol. 16, no. 1, p. 5, Dec. 2021, doi: 10.1186/s41039-021-00151-1.

[11] J. Laine, T. Lindqvist, T. Korhonen, and K. Hakkarainen, "Systematic Review of Intelligent Tutoring Systems for Hard Skills Training in Virtual Reality Environments," International Journal of Technology in Education and Science, vol. 6, no. 2, pp. 178–203, May 2022, doi: 10.46328/ijtes.348.

[12] N. A. B. Aziz, "Choosing Appropriate Retrieval based Learning Elements among Students in Java Programming Course," International Journal of Psychosocial Rehabilitation, vol. 24, no. 5, pp. 5448–5455, Apr. 2020, doi: 10.37200/IJPR/V24I5/PR2020251.

[13] L. Zavala and B. Mendoza, "On the Use of Semantic-Based AIG to Automatically Generate Programming Exercises," in Proceedings of the 49th ACM Technical Symposium on Computer Science Education, Feb. 2018, pp. 14–19. doi: 10.1145/3159450.3159608.

[14] R. Garcia, K. Falkner, and R. Vivian, "Instructional Framework for CS1 Question Activities," in Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education, Jul. 2019, pp. 189–195. doi: 10.1145/3304221.3319732.

[15] R. Rodriguez-Torrealba, E. Garcia-Lopez, and A. Garcia-Cabot, "End-to-End generation of Multiple-Choice questions using Text-to-Text transfer Transformer models," Expert Syst Appl, vol. 208, p. 118258, Dec. 2022, doi: 10.1016/j.eswa.2022.118258.

[16] S. G. Aithal, A. B. Rao, and S. Singh, "Automatic question-answer pairs generation and question similarity mechanism in question answering system," Applied Intelligence, vol. 51, no. 11, pp. 8484–8497, Nov. 2021, doi: 10.1007/s10489-021-02348-9.

[17] R. Layona, B. Yulianto, and Y. Tunardi, "Authoring Tool for Interactive Video Content for Learning Programming," Procedia Comput Sci, vol. 116, pp. 37–44, 2017, doi: 10.1016/j.procs.2017.10.006.

[18] M. Divate and A. Salgaonkar, "Automatic Question Generation Approaches and Evaluation Techniques," Curr Sci, vol. 113, no. 09, p. 1683, Nov. 2017, doi: 10.18520/cs/v113/i09/1683-1691.

[19] S. S. Alanazi, N. Elfadil, M. Jarajreh, and S. Algarni, "Question Answering Systems: A Systematic Literature Review," International Journal of Advanced Computer Science and Applications, vol. 12, no. 3, 2021, doi: 10.14569/IJACSA.2021.0120359.

[20] J. Salac and D. Franklin, "If They Build It, Will They Understand It? Exploring the Relationship between Student Code and Performance," in Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education, Jun. 2020, pp. 473–479. doi: 10.1145/3341525.3387379.

[21] Ms. R. S. M. Sc. MPhil and Ganesh. K, "Automatic Question Paper Generator System," International Journal of Trend in Scientific Research and Development, vol. Volume-3, no. Issue-3, pp. 138–139, Apr. 2019, doi: 10.31142/ijtsrd21646.

[22] Y. Choi and C. McClenen, "Development of Adaptive Formative Assessment System Using Computerized Adaptive Testing and Dynamic Bayesian Networks," Applied Sciences, vol. 10, no. 22, p. 8196, Nov. 2020, doi: 10.3390/app10228196.

[23] K. Atchariyachanvanich, S. Nalintippayawong, and T. Julavanich, "Reverse SQL Question Generation Algorithm in the DBLearn Adaptive E-Learning System," IEEE Access, vol. 7, pp. 54993–55004, 2019, doi: 10.1109/ACCESS.2019.2912522.

[24] H. Roitman, U. Singer, Y. Eshel, A. Nus, and E. Kiperwasser, "Learning to Diversify for Product Question Generation," Jul. 2022.

[25] W. He, J. Shi, T. Su, Z. Lu, L. Hao, and Y. Huang, "Automated test generation for IEC 61131-3 ST programs via dynamic symbolic execution," Sci Comput Program, vol. 206, p. 102608, Jun. 2021, doi: 10.1016/j.scico.2021.102608.