Department of Electrical, Computer, and Software Engineering

Part IV Research Project

Final Report

Project Number: 98

Efficiently developing question banks for student learning using Large Language Models

Author: Ojas Madaan

Project Partner: Ben Lowthian

Supervisors:

Paul Denny

Andrew Luxton-Reilly

Juho Leinonen

13/10/2023

Declaration of Originality

This report is my own unaided work and was not copied from nor written in collaboration with any other person.

[illegible]

Name: Ojas Madaan

QuizMate: Using Large Language Models for Question and Distractor Generation Tasks

Ojas Madaan
The University of Auckland
Auckland, New Zealand

ABSTRACT

Multiple-choice questions are extensively used as a method to evaluate students' understanding. Yet, the task of generating multiple-choice questions is a time-consuming process that requires extensive knowledge of the taught content and is often a tedious task due to the requirement of creating plausible distractor answers. This has made automatic question generation a heavily researched field related to education. Similarly, the recent rise of large language models for language-related tasks has prompted their use in a wide range of fields, and their use cases related to the question generation domain have also been explored. This prior literature has shown that the generation of quiz artifacts is certainly possible using large language models, but the existing approaches garnered mixed results, and evaluating items was revealed to be difficult. Our contribution is a novel tool called QuizMate, which is a web-based tool to efficiently generate sets of complete questions or additional distractor choices for existing questions. Through our proposed evaluation metrics, we find that QuizMate can generate high-quality questions and plausible distractors of a level similar to humans and that educators view QuizMate as a helpful tool that they would use as a tool to aid the generation of questions for examination.

CCS CONCEPTS

• **Computing methodologies → Natural language generation**.

KEYWORDS

large language models, question generation, MCQ

ACM Reference Format:
Ojas Madaan. 2023. QuizMate: Using Large Language Models for Question and Distractor Generation Tasks. In *Proceedings of to be determined (Part 4 Project).* ACM, New York, NY, USA, 10 pages. https://doi.org/12345.12345

1 INTRODUCTION

The increasing use of multiple-choice questions (MCQs) for student assessment in an increasingly online learning environment presents several challenges for educators, including high time costs and the need for deep domain knowledge for question creation [1, 26]. Notably, the task of crafting unique distractors to prevent students from guessing accurate answers is particularly demanding [35].

Advancements in artificial intelligence, specifically large language models like GPT-3 and OpenAI Codex, provide exciting new possibilities for addressing these challenges [9, 32]. These models have shown impressive performance in understanding and generating human-like text, making them highly suitable for a wide range of tasks, including explaining complex computer science concepts, producing code examples and providing real-time problem-solving feedback [20]. Their potential usefulness in a computer science education context, therefore, warrants deeper investigation [3, 18].

In response, we introduce 'QuizMate' - a tool designed to leverage the capabilities of large language models for automatic MCQ and distractor generation. Alongside this, we provide empirical findings from the utilisation of QuizMate. The tool operates based on user-provided prompts to generate relevant MCQ sets, facilitating the efficient production of numerous unique distractor choices for each question. Additionally, QuizMate enriches the learning experience by providing brief natural-language explanations for correct answers.

Our contributions in this current paper are three-fold: we first introduce the architecture and operation of QuizMate, then we assess its performance and acceptability among a sample group of educators through a user study and by objectively evaluating generated artifacts, and finally, we propose potential areas for future improvements and developments based on the feedback received.

Our evaluation of QuizMate is driven by the following research questions:

RQ1: How effective are LLMs at generating high-quality MCQs for educational purposes?

RQ2: How effective are LLMs at generating plausible distractor options for MCQs?

RQ3: What are educators perceptions towards using QuizMate as a question and distractor generation tool?

2 RELATED WORK

2.1 Multiple Choice Questions

2.1.1 From a teaching perspective. Many institutions have embraced MCQ testing as a key foundation of their testing systems [22]. This is due to the convenience when it comes to courses where large numbers of candidates are involved. Some other advantages of MCQs are their perceived objectivity and the ability to reduce cheating by having multiple versions of the same test. This provides teachers with better feedback on their teaching and makes it easier to identify gaps in student understanding. However, MCQ test creation can be a difficult task for humans as it requires domain knowledge and an understanding of pedagogical process and context [19]. The number of items in an MCQ test should also be large enough to provide credible evaluation [24], which consequently means that the number of distractor answers required is very large. Distractor generation is a crucial yet time-consuming task for quiz setters, a good distractor should be semantically related to the correct answer and be both semantically consistent and grammatically coherent with the question [15]. Haladyna and Downing [7] emphasised that plausible distractors should be able to attract more than 5% of low-performing students, but based on such criteria, they found that only 8% of the items they evaluated contained 5 effective distractors, this shows how difficult it can be to create meaningful MCQs. Overall, MCQs provide many benefits from a teaching perspective, however, due to the large knowledge and time requirements, they are a very strong candidate for automation.

2.1.2 From a Learning perspective. According to literature, most students prefer MCQs when compared with essay style questions [12, 41]. Abreu et al. [1] conducted a study on the low performance of non-technical students taking programming courses. They found that students had a higher level of motivation related to MCQ questions when compared to development questions (DQs). They had students answer a set of questions where 50% of the questions were MCQs, and the other 50% were DQs, although the students preferred MCQ style questions, their performance was only slightly better in them, so the evidence does not suggest a major discrepancy in the actual difficulty between the two question styles in the study. A drawback of MCQs from a student's perspective is that they can create a false sense of security and knowledge [34, 39], as they can be randomly guessed or deduced through the process of eliminating equivalent distractors [35]. This reiterates the importance of good quality distractors when it comes to MCQs.

2.2 Automatic Generation of Multiple-Choice Questions

Due to the previously mentioned pain points with respect to MCQs, there has been a major rise in research focused on automatically generating them over the years. Automatic Question Generation (AQG) is defined by Rus et al [30] as follows: "Question Generation is the task of automatically generating questions for various inputs such as raw text, database or semantic representation". Based on the definition, the type of input for question generation can vary: it could be, for example, a sentence, a paragraph, or even an entire course syllabus. Question generation is a multiple-step process and as such, the research in this field has typically been divided into two main areas: Question Generation (QG) and Distractor Generation (DG).

2.2.1 Question Generation not using Large Language Models. Several models have shown decent performance in question generation (QG). Question generation methods can be classified as syntax-based, semantic-based, or template-based [40]. Syntax-based doesn't require much understanding of the input. Conversely, semantic-based approaches need a deeper understanding of the input [13].

Many current methods use Natural Language Processing (NLP) and summarisation techniques for QG [21]. For instance, A. Santhanavijayan et al. [31] use a syntax-based tool to generate multiple-choice questions (MCQs). Nwafor and Onyenwe [22] also proposed an NLP-based system for MCQ generation using Term Frequency-Inverse Document Frequency and N-gram. Likewise, Maheen et al. [19] used similar techniques but didn't consider domain-specific terminology.

Mitkov and Ha [21] created an approach by identifying domain-specific terms. They found that their tool reduced the time to generate one test item considerably.

Deep learning-based techniques for QG, such as sequence-to-sequence (Seq2Seq) models [37], are also quite prevalent. Models like those by Kim et al. [10], Zhou et al. [43] and Zhao et al. [42] utilize Seq2Seq approaches. Kriangchaivech and Wangperawong [11] developed an automated model that generated questions from Wikipedia articles with lower training cost and improved performance using transformers.

Semantic-based approaches, however, need additional information usually in the form of ontologies [24]. Such approaches, while domain-specific and not dependent on lexicons, can be time-consuming as they often require extensive input from domain experts [24].

2.2.2 Question Generation approaches using Large Language Models. Large language models have made significant improvements in NLP in recent years [9]. They are trained on massive amounts of text data and therefore are able to generate human-like text, answer questions and complete other language tasks with high accuracy. This has therefore led to new research into the field of QG which is based on using LLMs.

Lopez et al. [17] explored using the pretrained language model GPT-2. They noted that while previous approaches using Transformers and Reinforcement Learning had produced robust QG models, they all required extra mechanisms, complex models and additional features that made them harder to train and expensive to reproduce [17]. They finetuned the GPT-2 model using a formatted version of the previously mentioned SQuAD. They evaluated their results using metrics such as BLEU 1, BLEU 2, BLEU 3, BLEU 4 [25] ROUGE L [16] and METEOR [2]. They found that a simple single transformer-based QG model was able to outperform many of the more complex Seq2Seq models without the need for advanced features and training steps [17].

More recently, the GPT-3 LLM was used by Dijkstra et al. [4] to create an end-to-end quiz generator. They utilised the Curie version of GPT-3 as their testing showed that similar results were achieved with a smaller model when fine-tuning. They introduced the tasks of Step-Wise Quiz Generation (SWQG) and End-to-End Quiz Generation (EEQC). EEQC simply takes context as input, and outputs a question, answer, and distractors. SWQG, outputted a question given context, an answer when given context and a question and outputted distractors when provided with context, the question, and the answer [4]. They used the previously mentioned BLEU 4, ROUGE-L and METEOR metrics to evaluate the performance of their quiz generator. They also compared human evaluation scores to compare the performance of their model with other existing ones based on metrics like fluency, correctness, and distracting ability. When compared with a more general-purpose model MACAW-11b, their tool strongly outperformed it on all of the metrics mentioned. This was expected, however as the GPT model was fine-tuned to this task and GPT-3 is a larger language model when compared to T5, which MACAW is based on, the MACAW model used was also off the shelf and not fine-tuned, which could have led to an unfair comparison [4]. Overall, the approach by Dijkstra et al. was mostly successful, however they noted that sometimes the quality of quizzes was lower than human-generated quizzes and that it also had relative weaknesses in the areas of answer validity and distractor generation, meaning that the tool was still not a complete replacement for manual QG.

The previously mentioned MACAW model was introduced by Tafjord and Clark [38] as a versatile generative question-answering (QA) system. Although it is primarily a QA system, it allows for different angles of its inputs and outputs to be used, meaning that it could take a question as input and provide an answer, but also take an answer and output a question. Another possible permutation is that a question and answer are provided, and MACAW produces multiple-choice options [38]. The model was used in a study [4] with very little to no fine-tuning and was able to still complete end-to-end quiz generation tasks. MACAW is therefore a viable option when it comes to the task of QG.

Gabajiwala et al. [5] proposed a solution to QG which uses an NLP pipeline involving BERT and T5 transformer models. GPT-2 was used to perform the NLP-related tasks as part of the pipeline. BERT was used for the task of keyword extraction and text summarisation. The T5 transformer was also used for the formation of Wh-type questions. To evaluate their tool, they proposed their own method involving human reference. They provided humans with two sets of questions; one set was machine-generated and the other set was human-generated and asked them to identify the automatically generated questions. Whenever they were not able to correctly identify the automatically generated questions, they deemed it as the tool being on par with humans [5]. The findings suggested that around 60% of the automatically generated questions were incorrectly identified as human-generated. This suggests that a combination of different LLMs can be used to improve the quality of QG when compared to existing methods [5].

One of the identified downsides to QG using LLMS is that fine-tuned models are very costly. Furthermore, the models are often domain-specific which hinders their use in a wide range of domains. Dijkstra et al. [4] suggest two solutions to this issue. The first would be to train their model on a very broad domain, so that QG would be possible for a wide range of domains. The other option is to create domain-specific quiz generators, this solution would likely lead to better results for individual domains but would have a much higher cost.

2.2.3 Distractor Generation not using Large Language Models. There have been many attempts to create models capable of DG for MCQs. These include selecting distractors based on the frequency of a word appearing in a corpus [6], using POS tagging [31, 36] or co-occurrence with the correct answer [8]. A more dominant approach seen is the selection of distractors based on their similarity to the key [13, 31], where the similarities could be, but are not limited to domain ontology based, syntactic based or even context-based. The distractors generated by these similarity based methods are selected from a set of possible candidates based on a weighted combination of similarities where weights are determined using specific heuristics [13].

Mitkov and Ha [21] employed WordNet to retrieve hypernyms, hyponyms of the key, and if applicable filtered the returned values to those that appeared in the corpus. They evaluated the generated distractors by comparing the number of students in upper and lower groups who selected each incorrect alternative, as a good distractor should attract more students from a lower group than those with a better understanding of the taught content [21]. A similar approach to DG using WordNet was proposed by Maheen et al. [19] as part of their MCQ generation tool, but they also included google search results and wiktionary to create a set of synonyms and similar words. 3 distractors were then chosen at random from the candidates. To evaluate their findings, they had domain experts assess the generated MCQs with distractor generation as one of the metrics. The distractors received an 80% score, however it should be noted that the questions generated by their tool were mostly fill-in-the-blank style MCQs, so the assessment of distractor usefulness may vary when compared to more complex MCQs.

Similar to the task of QG, DG approaches have also utilised neural networks to generate better distractors [27]. For example, in the end-to-end framework proposed by Gao et al. [6], they employed a hierarchical encoder-decoder network as the base model and used a dynamic attention mechanism to generate distractors. When compared to existing basic encoder-decoder methods like Seq2Seq [11, 37], using the BLEU (1-4) and ROUGE (1, 2, L) metrics, the proposed model achieved the best performance across the board [6]. Human evaluation was also conducted on the generated distractors which found that this model outperformed the other automatic DG methods in question, but human-generated distractors like those found in the RACE dataset still had the strongest capability to confuse the evaluators. Overall, the classical encoder-decoder frameworks are said to be poor at the task of high quality and therefore diverse DG as they are based on the beam search method [39].

The work by Xie et al [39] focuses on generating diverse distractors, as this ensures a higher quality MCQ [35]. Their approach uses multiple key sentences to aid generation and promote distractor diversity. They also introduce question and answer aware mechanisms to help with distractors being coherent and prevent them from being semantically equivalent to the answer. They evaluated their model (MSG-Net) against existing state-of-the-art models including Seq2Seq and the results showed that MSG-Net performed better across automatic and human evaluation. The metrics used to evaluate the distractors automatically were once again BLEU (1-4), ROUGE-L and METEOR. Human evaluation was conducted by volunteers with rich educational experience and was based on fluency, relevance, diversity, and difference. MSG-Net however, was not measured against human-generated distractors, so therefore we do not have a clear understanding of its ability to replace manual DG.

2.2.4 Distractor Generation approaches using Large Language Models. As mentioned in Section 2.2.2, LLMs have shown massive advancements in a range of language processing and QG related tasks. For example, Dijkstra et al. [4] included DG as part of their research into an end-to-end quiz generator which made use of the huge GPT-3 LLM. They found that almost all of the generated distractors were coherent with the text and contained fluent language, however there was room for improvement in their distracting ability [4]. They also observed that the generation of high-quality distractors was more challenging than other tasks like QG and QA. They made use of the BLEU-4, ROUGE-L, and METEOR metrics for automatic evaluation of generated items. The automatic evaluation proved to be problematic in this study as the BLEU-4 scores were uninterruptable [4], therefore more detailed analysis would be required to conclude which model in their study had the best performance.

Offerijins et al. [23] suggested a fine-tuned GPT-2 model for the task of DG, which was trained on the RACE dataset. This dataset was used to train their model for DG tasks rather than SQuAD as it includes distractors along with the questions [14, 23]. They used the BLEU and ROUGE metrics for evaluation, but discussed their shortcomings and the need for human evaluation when it comes to a task like DG. Overall, the results of their study suggested that the quality of generated distractors was substantially higher than previous works and outperformed the start-of-the-art baselines on all BLEU metrics.

Other approaches using transformer models like BERT and T-5 have also found success at the task of DG [29, 39]. Rodriguez-Torrealba et al. [29] tried to evaluate the quality of generated distractors using human evaluation, ROUGE, BLEU, and a distractor score which is based on the similarity between the word vectors of the distractor and the correct answer. The results were overall favourable to their model and highlighted that automatic DG is indeed viable, but that there is a lack of adequate metrics for DG [23, 29]. Another important factor to consider is that none of the methods discussed (LLM and non-LLM based) for DG consider explaining why the distractors are incorrect, which would be crucial in terms of formative feedback and an area where more research should be undertaken with regard to education.

3 METHODS

3.1 A tool for efficiently generating MCQs: QuizMate

To aid us in answering our research questions as outlined in Section 1, we designed a web-based tool called QuizMate. The tool has two key purposes: Question Generation and Distractor Generation.

With regard to RQ1, the Question Generation mode as shown in Figure 1 allows users to generate MC questions based on a natural language prompt about the topic, difficulty, learning outcomes to assess, and programming language if necessary. Additionally, they can decide on the number of questions to generate and how many answer choices each generated question should have.

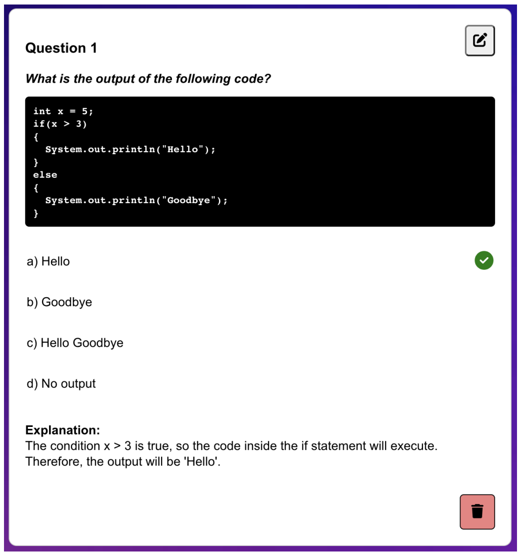
To address RQ2, we provide users with the ability to generate distractors for questions where the correct answer is already known. In this scenario, no additional context is required as input, which greatly reduces the time required for the task of DG, which is otherwise a very tedious task [15].

3.1.1 Implementation. Our implementation of the tool utilises Node.js and Express.js for the backend, React for the frontend, and gpt-3.5-turbo-16k as the language model. Firebase is used to store all the generated artifacts and user interactions, providing a reliable and scalable NoSQL database solution. The website hosting is done through Heroku, ensuring accessibility and availability for users. This tech stack enables seamless communication between the frontend and backend, facilitates interaction with the language model, and ensures secure and robust data storage.

3.1.2 Question Generation. Once the user has input their desired parameters for QG into the page shown in Figure 1 and clicked the generate button, they are taken to a page where they can review the generated questions and download the questions.

The review page, as shown in Figure 2, allows users to modify the generated questions or reject the question completely. If they opt to reject a question, the reasons for doing so are gathered through a form. Within this, the user can tick options for rejecting the question, such as "Too Easy", "Not Relevant", or "Not Unique", and provide additional feedback through text entry. Once they are satisfied with the questions, they can complete the review. The data collected through these user interactions is stored in the Firebase NoSQL database and will aid in answering RQ1, RQ2 and RQ3.

When a download is triggered within the tool, two PDF files are generated. In one of the files, the correct answers and explanations for the generated questions are visible, and in the other, they are hidden. This makes the tool suitable for educators who would like to be able to administer the questions to a class of students without having to manually type out the generated questions. Alternatively, this could benefit students using the tool by providing them with practice material they have never seen before, which is tailored to their specific needs through the prompt. It also removes the barrier of not having access to the correct answer, which could otherwise leave students unsure about their knowledge.


Question 1

What is the output of the following code?

```
int x = 5;
if(x > 3)
{
  System.out.println("Hello");
}
else
{
  System.out.println("Goodbye");
}
```

a) Hello ✓

b) Goodbye

c) Hello Goodbye

d) No output

Explanation:
The condition x > 3 is true, so the code inside the if statement will execute.
Therefore, the output will be 'Hello'.

Figure 2: Generated Question Review Interface

3.1.3 Distractor Generation. For the task of distractor generation, users are asked to upload a CSV file containing questions and their corresponding correct answers. An example of an input file suitable for DG is shown in Figure 3.

The output is very similar to the output of the QG task described in Section 3.1.2, where users can review the generated distractors as shown in Figure 2 and also export the now complete questions to pdf. In this case, the user interactions during the review stage are recorded to aid in answering RQ2 and RQ3.

In addition to generating distractor choices, an explanation of the correct answer is also provided. This, once again could help save educators time when creating MCQs if they do not already

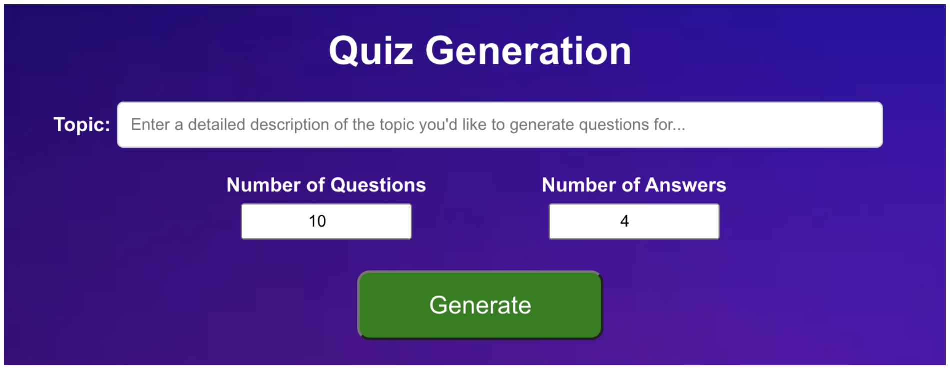
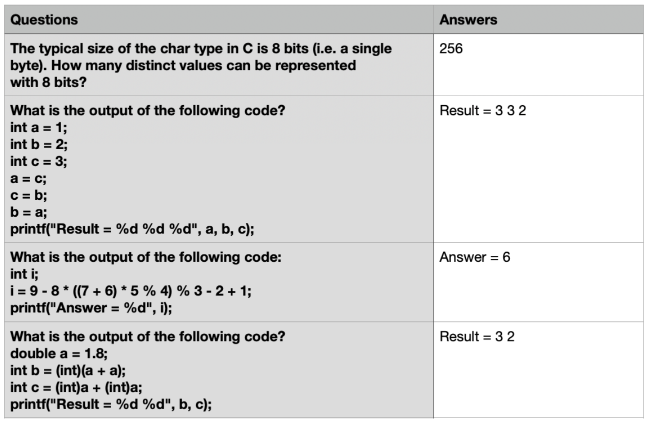

Quiz Generation
Topic:
Enter a detailed description of the topic you'd like to generate questions for...
Number of Questions
10
Number of Answers
4
Generate

Figure 1: Inputs required for automatic question generation

have explanations written, and would be especially useful for code-tracing style questions where step-by-step explanations are time-consuming to create manually.



Questions	Answers
The typical size of the char type in C is 8 bits (i.e. a single byte). How many distinct values can be represented with 8 bits?	256

What is the output of the following code?
int a = 1;
int b = 2;
int c = 3;
a = c;
c = b;
b = a;
printf("Result = %d %d %d", a, b, c);
	Result = 3 3 2

What is the output of the following code:
int i;
i = 9 - 8 * ((7 + 6) * 5 % 4) % 3 - 2 + 1;
printf("Answer = %d", i);
	Answer = 6

What is the output of the following code?
double a = 1.8;
int b = (int)(a + a);
int c = (int)a + (int)a;
printf("Result = %d %d", b, c);
	Result = 3 2

Figure 3: Example format of a csv file input for distractor generation

3.1.4 Prompting. For our tool, we had to design two key prompts, these were for the tasks of QG and DG. In both cases, the language model being used, gpt-3.5-turbo-16k, was also primed with a system message stating, "You are an educator in the field of computer science.". This was done to ensure the model would provide relevant responses and reduce ambiguity.

Given the requirements of the tool, where the language models response was being displayed via a react application, we had to utilise one-shot prompting in order to ensure a consistent data format. During early development and research, we found that one-shot was more than sufficient for the language model to understand the requirements, and that few-shot prompting could actually under-perform when compared to zero-shot or one-shot prompts [28].

For question generation, the model is asked to generate MCQs based on a prompt that the user inputs. This is passed into the LLM verbatim to ensure that the tool has the best chance of generating questions that are related to the learning outcomes and requirements mentioned in the user prompt. Initially, we had planned to extract keywords from the prompt and generate questions based on keywords, which is an approach that had been explored in previous research [5, 22]. However, after early experimentation through OpenAI's playground environment, we discovered that the model was producing better results when provided with the entire user prompt. Another iteration that had to be made was the formatting of code that the model generated as part of the questions. Early feedback from users indicated that the readability of code segments was quite poor, so an additional instruction was added to the QG prompt to wrap any code in an appropriate HTML tag so that the tool could display it correctly, especially since indentation is an important aspect for programming languages like python.

For the task of distractor generation, we found that providing the model with a concrete input and output example was beneficial. This was made possible through the OpenAI chat completions API, where you can provide a multiple-turn conversation between the user and assistant roles. In our case, we first provided an example input, then responded as the assistant with the expected response, and finally provided the model with the actual input for DG. One of the key challenges when developing the prompt for DG was the model hallucinating and creating its own version of the correct answer, even though the correct answer was already provided. To try and counter this, some adjustments were made where it was reiterated to the model that the answer provided was indeed correct. However, the problem still persists at times, and this is discussed further in section 4.1.2.

The final prompts developed for both QG and DG can be found in Table 1.

3.2 User Study

To gauge the practicality and usability of QuizMate, we conducted a user study with a group of computer science educators from the University of Auckland (n=3). The study was divided into two key parts: A hands-on interaction with QuizMate and the completion of post-use surveys on their experiences completing the tasks of QG and DG. The key sentiments we aimed to gather were if, overall, the generated quiz artifacts were relevant, unique and of reasonable difficulty.

The participants were provided with unambiguous instructions detailing how to use the tool, guidance on prompts and parameters, and the question review process. For ease of comparison and to reduce the number of variables to consider, we asked that they

Table 1: Final Prompts Developed

Task	Prompt
QG	Based on this prompt: "USER INPUT PROMPT", Generate X multiple choice questions. Each multiple choice question should have Y options associated with it. Return your answer entirely in the form of a JSON object. The JSON object should have a key named "questions" which is an array of the generated questions. Each quiz question should include the choices, the answer, and a brief explanation of why the answer is correct. The JSON properties of each question should be "query" (which is the question), "choices", "answer", and "explanation". The choices should not have any ordinal value like A, B, C, D or a number like 1, 2, 3, 4. The answer should be the 0-indexed number of the correct choice. The explanation should contain only plain text. Indicate the start of any code with an opening html pre tag and the end of any code with a closing pre tag. Here is an example JSON structure of a single question: SAMPLE JSON
DG	Given an array of questions: "QUESTIONS$_{ARR}$" and the corresponding correct answers: "ANSWERS$_{ARR}$". Generate exactly X additional plausible wrong choices for each question in the question array. Return your answer entirely in the form of a JSON object. The JSON object should have a key named "questions" which is an array of the questions. Each quiz question should include the choices you generated, the original answer that was provided, and a brief explanation of why the answer is correct. Don't include anything other than the JSON. The JSON properties of each question should be "query" (which is the question), "choices", "answer", and "explanation". The choices should not have any ordinal value like A, B, C, D or a number like 1, 2, 3, 4. The answer should be the 0-indexed number of the correct choice. Indicate the start of any code with an opening html pre tag and the end of any code with a closing pre tag.

leave the number of questions to generate and the number of answers as the default options for QG. They were free however, to enter any topic or prompt they saw fit. At the completion of the generation tasks, they were asked to fill out a short questionnaire hosted through Google Forms.

The formulation of the survey questions was designed to evaluate the quality of the originated questions and distractors, gauge the inclination of educators towards future usage, and whether they would consider administering the generated questions in their own assessments. These survey inquiries aid in answering our research questions, especially offering insights into the relevance, level of difficulty, and overall quality of the generated content (RQ1 and RQ2), along with user perceptions of the tool (RQ3).

The selection of the participants was intentional due to their expert background in Computer Science and their experience creating MCQ assessments in an academic setting. This ensured that the user study was contextual, practical, and reflective of realistic usage scenarios or other potential users of the tool.

The data collected through Google Forms and through the user interactions stored in Firebase will be used in the quantitative and qualitative analysis of QuizMate as a tool for QG and DG, with the user study hopefully proving valuable in informing future research, areas for improvement, and refining the tool for more widespread use.

3.3 Evaluation Metrics

In order to quantitatively analyse the MCQs and distractors generated by QuizMate, we first had to decide upon a set of metrics for each task (QG and DG). These metrics were then used to conduct an objective self-evaluation. This evaluation was designed to address our research questions by examining the effectiveness of both the question generation and the distractor generation capabilities of the LLM.

For QG and DG, many previous efforts of evaluating generated artifacts have utilised metrics such as BLEU, ROUGE L and METEOR [4, 5, 17, 39], however these have shortcomings as they are mostly related to text similarity [23], and therefore may not be a good indicator of question quality, and whether the distractors are plausible or not. We therefore opted not to utilise the aforementioned automated metrics.

Instead, we devised our own metrics as shown in Table 2, such that they could be objectively answered using "yes", "no", or "somewhat". For the task of QG, we chose metrics such that we could investigate the quality of questions as a whole. These metrics included relevancy to the topic, difficulty level, provision of a helpful explanation, uniqueness of distractors, correctness of the answer, relevance of the distractors to the question and grammatical coherence. Together, these metrics provided us with a good indication as to how LLMs perform at creating high-quality MCQs, which is in alignment with RQ1.

To evaluate DG, we used a different set of metrics to investigate the plausibility and effectiveness of the generated distractors. The metrics, as seen in Table 3 assess semantic similarity, syntactic similarity, relevance, uniqueness, and whether the model has correctly identified and justified the correct answer. These metrics were chosen to evaluate the LLMs ability to comprise the integral attributes of an effective distractor as highlighted in previous research [15, 27, 35, 39], thus providing insights into RQ2. The metric related to answer validity was included for DG as it was highlighted by Dijsktra et al. as an area of weakness during their evaluation on an LLM based Quiz Generator [4], this was reaffirmed during our initial experimentation as we noted that the model was prone to hallucinating and believing its own answer over the provided correct answer, this is discussed further in Section 4.1.2. Aside from the quality metrics, we also wanted to address a gap identified by Xie et al. and therefore also performed a similarity-based comparison of generated distractors against the human-generated distractors provided [39], the questions used for DG were retrieved from a test given to students taking a first year programming course at the University of Auckland.

Table 2: Metrics used to objectively evaluate generated Multiple-Choice Questions

Metric	Answers
Relevant to topic?	
Suitable Difficulty?	
Helpful Explanation?	
Unique Distractors?	Yes, No, Somewhat
Relevant Distractors?	
Correct Answer?	
Grammatical Sense?	

Table 3: Metrics used to objectively evaluate generated Distractors

Metric	Answers
Semantically Similar?	
Syntactically Similar?	
Relevant?	Yes, No, Somewhat
Unique?	
Correct Answer?	

4 RESULTS AND DISCUSSION

4.1 User Study

4.1.1 Question Generation. The results of the User Study, as presented in Figure 4 and Table 4, offer important insights into the hands-on effectiveness of utilising an LLM for the task of QG. Overall, 75.3% of the generated questions were accepted, with only 2.2% requiring modification and 22.6% being rejected. These findings imply that a relatively large amount of questions generated were of a sufficient quality from a user perspective, which is an encouraging sign with regard to RQ1, which questions if LLMs can generate high-quality MCQs.

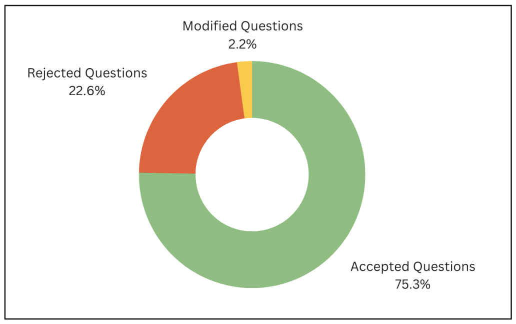


Figure 4: Percentage of user generated questions that were accepted, accepted with modification, or rejected

We further analyse the reasons behind questions being rejected, as observed in Table 4, in order to determine key areas for improvement within the tool and LLMs in general for QG. We note that 5% of the rejected questions were not perceived as relevant to the user prompt, which reaffirms previous GQ efforts and the strength of LLMs at generating relevant content [4]. However, a large portion of the rejected questions were either "too easy" (24%) or "not unique" (33%). By looking deeper into the collected user interactions, we note that "not unique" cases tended to relate to ambiguity within the answer choices where multiple answers could be correct, depending on context or programming language if it wasn't specified in the prompt. Furthermore, we found that by providing a difficulty level in the prompt i.e. "... for a university level CS course", the issue of questions being too easy could mostly be resolved. In both scenarios, the rejection rate could therefore be greatly reduced by providing better guidance for user prompts, as the model performed better when the prompts were more explicit with the learning outcomes, programming language and difficulty level. This mimics the sentiments of one user who stated that the biggest challenge they encountered "was not knowing the appropriate format, or level of detail" with respect to the input prompt they had to provide, and that they would suggest "more scaffolding for the user" in order to help them craft better input prompts for QG.

Upon closer inspection into the 38% of rejections due to miscellaneous reasons, we note that the majority of these are due to the LLM generated answer being incorrect. This is significant as it would severely impact user perceptions and trust (RQ3) when using a tool to generate questions. It could instead lead to more time required for the manual modification and checking of questions, which would be counteractive to the key purpose of the tool. This phenomena however, is a known limitation of LLMs, which are known to "hallucinate" and make up information, in this case incorrect answers [4]. OpenAI has stated that their newest model GPT-4, is less prone to hallucinations, however this model was not available to access via an API, and therefore we did not utilise it. Overall, as LLMs progress and more advanced models become more available, answer validity should in theory increase, which would be a much needed improvement in the context of a QG tool.

The results from the user study relating to QG present a large step in understanding educators perceptions towards utilising LLM generated questions. While the high acceptance rate is encouraging, the inherent downfalls related to ambiguity, answer validity must be addressed to improve user sentiment and confidence. The insights gained from user interactions show that the tool "clearly has excellent potential" when it comes to the task of QG, but that as it stands, they would rather use the tool as a way to brainstorm question ideas.

Table 4: Percentage Occurrence of Each Reason for a Generated Question to be Rejected

Rejection Reason	Occurrence
Too Easy	24%
Not Relevant	5%
Not Unique	33%
Miscellaneous	38%

4.1.2 Distractor Generation. In terms of the DG functionality, the feedback received was generally positive, with one user commenting that QuizMate was a "very interesting tool" and that it has the "potential for helping save time when constructing good MCQs". However, there were some recurring issues which were noted by users.

When questions of high complexity or length were inputted, one educator found that parts of questions and distractors were contaminating other generated distractors. This could imply that the model was either hallucinating, as discussed previously, or alternatively that the input format of the data was not ideal for DG and therefore hindered the models ability to generate effective distractors. However, questions of shorter length, albeit still testing similar learning outcomes and of a similar structure, the model was overall successful in the task of DG. The same educator found that the generators were actually quite effective, and in some cases matched the distractors for the same questions in assessments they had created. This suggests that the LLMs can indeed create plausible distractors under the right circumstances, as evidenced by the ability to match distractors created by experienced computer science educators, which is a positive outcome for RQ2.

Based on survey responses, all of the generated distractors were unique, and most were relevant logically correlated with the questions. This consistency and ability to generate relevant content matches the user sentiments discussed in Section 4.1.1. The acceptance, rejection, and modification rates were also very similar to the QG user study, which a majority of the generated output being accepted by educators. However, there was instances during the user study where the tool would output the incorrect answer alongside generated distractors, despite being provided the correct answer. We speculate that this could once again be a case of the model hallucinating, and choosing to generate an output that is not completely based on the input provided. However, another likely scenario is that the model is choosing to try and solve the question it has been given, and falling for the traps laid by the question creators. Previously, gpt-3 was found to fail university level programming courses without human intervention [33], and given that the questions utilised in the user study were of a similar difficulty, a similar scenario could occur. The use of gpt-4 for a task like this would likely improve the answer validity and subsequently distractor quality, as one of the key factors of generating good distractors is arguably understanding how the correct answer is determined, which gpt-4 is more likely to be able to do [33].

Overall, the user study indicates a lot of potential for the use of LLMs for the task of DG (RQ2), with our tool showing a clear ability to generate relevant and plausible distractors. However, improvements could be made in terms of LLM hallucinations. Educators in general seemed to be satisfied with the tool and answered that they would recommend it to other others, which is encouraging with regard to answering RQ3.

4.2 Manual Evaluation

4.2.1 Question Generation. The data collected from the manual evaluation of MCQs generated (n=99) using QuizMate is summarised in Table 5. This data provides valuable insights into the capabilities and limitations of LLMs when it comes to the task of QG.

Table 5: Results of evaluating QuizMate generated questions against our proposed metrics

Metric	Yes	Somewhat	No
Question is Relevant	95%	5%	0%
Suitable Difficulty	60%	19%	21%
Helpful Explanation	70%	23%	7%
Correct Answer	91%	3%	6%
Unique Distractors	95%	4%	1%
Relevant Distractors	97%	3%	0%
Makes grammatical sense?	99%	1%	0%

An overwhelming 95% of the generated questions were relevant and 5% were somewhat relevant. This data mimics the results we observed during the user study which is discussed in Section 4.1.1, further corroborating the effectiveness of LLMs in generating relevant and contextually pertinent questions.

Given that creating MCQs inherently involves the creation of distractor choices, the results of evaluating the generated distractors would also be crucial with regard to answering our research questions. A key observation here is the high proportion of questions with relevant distractors (97%) and unique distractors (95%). This, along with the 99% acceptance rate for the grammar-related metric, showcases the LLMs strengths in the area of natural language processing.

In terms of QG, the model provided the correct answer for 91% of the generated artifacts, which is a large improvement over the results we discussed in Section 4.1.1. However, 6% of the questions were identified as having an incorrect answer, which is concerning given that the language model had full control over the generation of the questions. On closer inspection, we found these incorrect answers often resulted from more complex questions. While it is known that models can hallucinate at times, it was a surprising outcome as the questions were unlikely to have the previously mentioned "traps" that educators would create. The remaining 3% of the generated answers were evaluated as being somewhat correct, this was due to ambiguity or in cases where multiple options could have technically been correct, however this is not a significant statistic, and in terms of answer validity, our tool outperformed previous LLM-based QG attempts [4].

In terms of difficulty, only 60% of the questions generated were of a suitable level, in the context of early-stage CS students. We previously discuss that users were much more positive about the difficulty of questions when they explicitly mentioned it in the prompt. If we narrow down the questions to those created with a difficulty level in the prompt, we see that 53% of the remaining questions are of a good difficulty. However, the results from one quiz generation session should be excluded here, as all of the questions were marked as "no" for this metric, due to function names being including in "what does the function do" style questions. If we further exclude these questions, then 64% of the questions are of acceptable difficulty. This comparison is shown in Table 6 and illustrates the relatively impressive improvements made when the prompt was more detailed.

Table 6: Comparing the Evaluated Difficulty of Generated Questions

	Suitable Difficulty?		
Difficulty Included in Prompt?	Yes	Somewhat	No
Yes	60%	19%	21%
No	64%	26%	10%

Providing helpful explanations had been previously identified as an area that requires further research. We found that while the model was in most cases (70%) able to provide helpful explanations for the correct answer, it often did not explain why the incorrect answers were not correct. This would arguably be more important from a student perception. This could perhaps be made more explicit to the model within the prompt and would as a result enhance the value of the tool for both educators and students.

Overall, the results of our manual evaluation of QuizMate-generated MCQs are quite encouraging and highlight the significant potential LLMs possess for QG tasks. We find that in the majority of cases, the model can generate a correct MCQ end-to-end without the need for any human intervention, which aids us in assessing RQ1. Yet, there are also some areas that should be improved, in particular the difficulty, quality of explanations and answer validity, which is most important.

4.2.2 Distractor Generation. The manual evaluation of the DG feature helped us to assess the effectiveness of QuizMate and addressed our research questions related to the quality of LLM-generated distractors (RQ2 and RQ3). The results of our evaluation are presented in Table 7.

Table 7: Results of evaluating QuizMate generated distractors against our proposed metrics

Metric	Yes	No
Semantically Similar	100%	0%
Syntactically Similar	100%	0%
Relevant	100%	0%
Unique	100%	0%
Answer is Correct	54%	46%

For the purpose of our evaluation, we utilised questions that were administered to a first-year C programming course at the University of Auckland. This allowed us to compare the LLM-generated distractors with those created by experienced educators. We discovered that nearly 30% of the generated distractors matched the human-generated distractors for the same set of questions. This suggests that LLMs can, to some extent, mimic human thinking in the context of DG.

Another positive aspect to note was the excellent level of uniqueness and relevancy of generated distractors (100% for both), which highlights the LLMs capability to produce contextually appropriate content, which aligns with the other results and given that uniqueness and relevancy are important factors when it comes to distractor quality, reaffirms of the premise of RQ2.

We noted that compared to the user study, the answer validity for the task of DG was worse, as nearly half of the generated answers were actually incorrect. This is likely due to the more complex questions used for our manual evaluation and a similar, previously discussed effect occurring, where the model would "convince" itself of its own incorrect answer instead of the correct answer provided. This area is in need of major improvement if the tool was to be seriously considered for DG in an academic setting. Potential mitigations could be made to the prompt, where instead of providing an array of input questions and answers, we instead provide it with a single question and answer pair at a time. This would likely reduce the LLMs "confusion" and contamination effect experienced by one of the users discussed in Section 4.1.2. Another possible improvement to this metric could be made by utilising a more advanced LLM, such as gpt-4, which has been shown to far outperform earlier OpenAI models at programming assignments [33] and would be less likely to convince itself of incorrect answers.

The manual evaluation of generated distractors showed that LLMs could indeed be used to create plausible distractor choices for MCQs (RQ2). The tool was successful in all cases of creating unique distractors, which is a key measure in order to deter students from guessing the correct answer [35]. Results from our attempt at DG using an LLM are similar to those reported by previous studies [4], where grammar and coherence are of a very high level, but the distracting ability and answer validity could be improved. This further reiterates the need for research in this area, and the previously mentioned notion of DG being a more complex task than QG.

5 CONCLUSION

This study assesses the effectiveness of QuizMate, a tool for educators that utilises large language models (LLMs) to generate meaningful MCQs and distractors. We found that QuizMate, when given sufficient context, is proficient in producing high-quality questions and distractor items, with educators open to incorporating the tool into their own workflows and willing to recommend the tool to their peers.

However, areas for improvement were identified, such as generating suitably difficult questions and maintaining answer validity. The automatic generation of explanations was also explored, albeit not as a primary objective, and should be researched further.

We make three primary contributions in this paper. The first is a comprehensive review of existing automated question and distractor generation techniques. The second is the design and implementation of a novel tool, QuizMate, for generating multiple-choice question banks and distractors efficiently. The third contribution is an empirical evaluation of QuizMate conducted through a user study and manual evaluation of generated artifacts.

Future work should investigate how changes to our approach impact the quality of generated questions and distractors, including the language model used, prompting style and scaffolding for user inputs. It would also be essential to consider the implications of employing a tool like QuizMate in an educational setting in the future.

REFERENCES

[1] PH Abreu, DC Silva, and A Gomes. 2018. Multiple-Choice Questions in Programming Courses: Can We Use Them and Are Students Motivated by Them? *ACM Trans. Comput. Educ.* 19, 1 (Nov 2018), 6.1–6.16. https://doi.org/10.1145/3243137

[2] M. Denkowski and A. Lavie. 2014. Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation* (Baltimore, Maryland, USA). Association for Computational Linguistics, 376–380. https://doi.org/10.3115/v1/W14-3348

[3] Paul Denny, James Prather, Brett A. Becker, James Finnie-Ansley, Arto Hellas, Juho Leinonen, Andrew Luxton-Reilly, Brent N. Reeves, Eddie Antonio Santos, and Sami Sarsa. 2023. Computing Education in the Era of Generative AI. https://doi.org/arXiv:2306.02608 arXiv:2306.02608 [cs.CY]

[4] R Dijkstra, Z Genç, S Kayal, and J Kamps. 2022. Reading Comprehension Quiz Generation using Generative Pretrained Transformers. https://dare.uva.nl/search?identifier=a1109043-92d4-4c63-be33-6e238780d3b7. Accessed: Apr. 28, 2023.

[5] E. Gabajiwala, P. Mehta, R. Singh, and R. Koshy. 2022. Quiz Maker: Automatic Quiz Generation from Text Using NLP. In *Futuristic Trends in Networks and Computing Technologies* (Singapore). Springer Nature Singapore, 523–533. https://doi.org/10.1007/978-981-19-5037-7_37

[6] Y. Gao, L. Bing, P. Li, I. King, and M. R. Lyu. 2018. Generating Distractors for Reading Comprehension Questions from Real Examinations. (2018). https://doi.org/10.48550/ARXIV.1809.02768

[7] T. M Haladyna and S. M Downing. 1993. How many options is enough for a multiple-choice test item? *Educational and Psychological Measurement* 53, 4 (Dec 1993), 999–1010. https://doi.org/10.1177/0013164493053004013

[8] S. Jiang and J. Lee. 2017. Distractor Generation for Chinese Fill-in-the-blank Items. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications* (Copenhagen, Denmark). Association for Computational Linguistics, 143–148. https://doi.org/10.18653/v1/W17-5015

[9] E Kasneci et al. 2023. ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences* 103 (Apr 2023), 102274. https://doi.org/10.1016/j.lindif.2023.102274

[10] Y. Kim, H. Lee, J. Shin, and K. Jung. 2018. Improving Neural Question Generation using Answer Separation. (2018). https://doi.org/10.48550/ARXIV.1809.02393

[11] K. Kriangchaivech and A. Wangperawong. 2019. Question Generation by Transformers. (2019). https://doi.org/10.48550/ARXIV.1909.05017

[12] William L. Kuechler and Mark G. Simkin. 2003. How well do multiple choice tests evaluate student understanding in computer programming classes. *Journal of Information Systems Education* 14, 4 (2003), 389–399.

[13] G Kurdi, J Leo, B Parsia, et al. 2020. A Systematic Review of Automatic Question Generation for Educational Purposes. *Int J Artif Intell Educ* 30 (2020), 121–204. https://doi.org/10.1007/s40593-019-00186-y

[14] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy. 2017. RACE: Large-scale Reading Comprehension Dataset From Examinations. (2017). https://doi.org/10.48550/ARXIV.1704.04683

[15] C Liang, X Yang, N Dave, D Wham, B Pursel, and CL Giles. 2018. Distractor Generation for Multiple Choice Questions Using Learning to Rank. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics, New Orleans, Louisiana, 284–290. https://doi.org/10.18653/v1/W18-0533

[16] C.-Y. Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out* (Barcelona, Spain). Association for Computational Linguistics, 74–81.

[17] Luis Lopez, Diane Cruz, Jan Cruz, and Charibeth Cheng. 2020. Transformer-based End-to-End Question Generation. (2020).

[18] Stephen MacNeil, Joanne Kim, Juho Leinonen, Paul Denny, Seth Bernstein, Brett A. Becker, Michel Wermelinger, Arto Hellas, Andrew Tran, Sami Sarsa, James Prather, and Viraj Kumar. 2023. The Implications of Large Language Models for CS Teachers and Students. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2* (Toronto ON, Canada) *(SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 1255. https://doi.org/10.1145/3545947.3573358

[19] F Maheen et al. 2022. Automatic computer science domain multiple-choice questions generation based on informative sentences. *PeerJ Computer Science* 8 (Aug 2022), e1010. https://doi.org/10.7717/peerj-cs.1010

[20] Kamil Malinka, Martin Peresíni, Anton Firc, Ondrej Hujnák, and Filip Janus. 2023. On the Educational Impact of ChatGPT: Is Artificial Intelligence Ready to Obtain a University Degree?. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1* (Turku, Finland) *(ITiCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 47–53. https://doi.org/10.1145/3587102.3588827

[21] R Mitkov and L. A Ha. 2003. Computer-aided generation of multiple-choice tests. In *Proceedings of the HLT-NAACL 03 workshop on Building educational applications using natural language processing*. Association for Computational Linguistics, 17–22. https://doi.org/10.3115/1118894.1118897

[22] C. A. Nwafor and I. E. Onyenwe. 2021. An Automated Multiple-Choice Question Generation using Natural Language Processing Techniques. *IJNLC* 10, 02 (Apr 2021), 1–10. https://doi.org/10.5121/ijnlc.2021.10201

[23] J. Offerijns, S. Verberne, and T. Verhoef. 2020. Better Distractions: Transformer-based Distractor Generation and Multiple Choice Question Filtering. (2020). https://doi.org/10.48550/ARXIV.2010.09598

[24] Andreas Papasalouros, Konstantinos Kanaris, and Konstantinos Kotis. 2008. Automatic Generation Of Multiple Choice Questions From Domain Ontologies. In *Proceedings of e-Learning*. Amsterdam, 427–434.

[25] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 2001. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02* (Philadelphia, Pennsylvania). Association for Computational Linguistics, 311. https://doi.org/10.3115/1073083.1073135

[26] A Petersen, M Craig, and P Denny. 2016. Employing Multiple-Answer Multiple Choice Questions. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '16*. Association for Computing Machinery, New York, NY, USA, 252–253. https://doi.org/10.1145/2899415.2925503

[27] Z. Qiu, X. Wu, and W. Fan. 2020. Automatic Distractor Generation for Multiple Choice Questions in Standard Tests. In *Proceedings of the 28th International Conference on Computational Linguistics* (Barcelona, Spain (Online)). International Committee on Computational Linguistics, 2096–2106. https://doi.org/10.18653/v1/2020.colingmain.189

[28] Laria Reynolds and Kyle McDonell. 2021. Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI EA '21)*. Association for Computing Machinery, New York, NY, USA, Article 314, 7 pages. https://doi.org/10.1145/3411763.3451760

[29] R. Rodriguez-Torrealba, E. Garcia-Lopez, and A. Garcia-Cabot. 2022. End-to-End generation of Multiple-Choice questions using Text-to-Text transfer Transformer models. *Expert Systems with Applications* 208 (2022), 118258. https://doi.org/10.1016/j.eswa.2022.118258

[30] Vasile Rus, Zhiqiang Cai, and Art Graesser. 2008. Question Generation: Example of A Multi-year Evaluation Campaign. (2008).

[31] A Santhanavijayan, S. R. Balasundaram, S. H. Narayanan, S. V. Kumar, and V. V. Prasad. 2017. Automatic generation of multiple choice questions for e-assessment. *IJSISE* 10, 1/2 (2017). https://doi.org/10.1504/IJSISE.2017.084571

[32] S Sarsa, P Denny, A Hellas, and J Leinonen. 2022. Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models. In *Proceedings of the 2022 ACM Conference on International Computing Education Research V.1*. ACM, Lugano and Virtual Event Switzerland, 27–43. https://doi.org/10.1145/3501385.3543957

[33] Jaromir Savelka, Arav Agarwal, Marshall An, Chris Bogart, and Majd Sakr. 2023. Thrilled by Your Progress! Large Language Models (GPT-4) No Longer Struggle to Pass Assessments in Higher Education Programming Courses. In *Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 1* (Chicago, IL, USA) *(ICER '23)*. Association for Computing Machinery, New York, NY, USA, 78–92. https://doi.org/10.1145/3568813.3600142

[34] J Shin, Q Guo, and MJ Gierl. 2019. Multiple-Choice Item Distractor Development Using Topic Modeling Approaches. *Front. Psychol.* 10 (Apr 2019). https://doi.org/10.3389/fpsyg.2019.00825

[35] M Sinha, T Dasgupta, and J Mandav. 2020. Ranking Multiple Choice Question Distractors using Semantically Informed Neural Networks. In *Proceedings of the 29th ACM International Conference on Information Knowledge Management*. ACM, Virtual Event Ireland, 3329–3332. https://doi.org/10.1145/3340531.3417468

[36] T. Soonklang, S. Pongpinigpinyo, W. Muangon, and S. Kaewjamnong. 2017. Automatic Question Generation System for English Exercise for Secondary Students. (2017).

[37] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[38] O. Tafjord and P. Clark. 2021. General-Purpose Question-Answering with Macaw. (2021). https://doi.org/10.48550/ARXIV.2109.02593

[39] J Xie, N Peng, Y Cai, T Wang, and Q Huang. 2022. Diverse Distractor Generation for Constructing High-Quality Multiple Choice Questions. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 30 (2022), 280–291. https://doi.org/10.1109/TASLP.2021.3138706

[40] X Yao, G Bouma, and Y Zhang. 2012. Semantics-based Question Generation and Implementation. *Dialogue & Discourse* 3, 2 (Mar 2012), 11–42. https://doi.org/10.5087/dad.2012.202

[41] Moshe Zeidner. 1987. *Essay versus multiple-choice type classroom exams: The student's perspective.*

[42] Y. Zhao, X. Ni, Y. Ding, and Q. Ke. 2018. Paragraph-level Neural Question Generation with Maxout Pointer and Gated Self-attention Networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (Brussels, Belgium). Association for Computational Linguistics, 3901–3910. https://doi.org/10.18653/v1/D18-1424

[43] Q. Zhou, N. Yang, F. Wei, C. Tan, H. Bao, and M. Zhou. 2017. Neural Question Generation from Text: A Preliminary Study. (2017). https://doi.org/10.48550/ARXIV.1704.01792