Automatic Generation of Test Tasks Using ChatGPT API*

Andriy Melnyk [1,†], Vasyl Tymchyshyn[1,*,†], Andriy Pukas [1,†] , Liubomyr Matiichuk[2,†] , Iurii Shcherbiak [3,†] ,Tetiana Yurchyshyn [1,†] and Tiande Pan[1,†]

[1] *West Ukrainian National University, 11 Lvivska Street, Ternopil, 46001, Ukraine*
[2] *Ternopil Ivan Puluj National Technical University, 11 Lvivska Street, Ternopil, 46001, Ukraine*
[3] *Catholic University in Ruzomberok Ruzomberok, Hrabovská cesta 1A, Ruzomberok, Slovakia*

Abstract
The development of effective and personalized assessment tools is a crucial challenge in contemporary education, particularly in the context of large-scale and digital learning environments. This paper presents a method for the automatic generation of test tasks using the ChatGPT API, a state-of-the-art language model developed by OpenAI. The proposed approach aims to reduce the workload of educators by automating the creation of diverse and pedagogically sound test items. The system enables the generation of questions tailored to various difficulty levels, ensuring alignment with students' individual knowledge profiles. Additionally, the method guarantees content uniqueness through dynamic question phrasing, thereby enhancing academic integrity and minimizing the reuse of static test materials. The flexibility of the ChatGPT API allows for the generation of multiple types of test questions, including multiple choice, short answer, and open-ended tasks across different subject areas. Experimental implementation demonstrates the feasibility of integrating this technology into modern educational platforms, resulting in improved efficiency, scalability, and personalization in the assessment process.

Keywords
Education system, ChatGPT API, test, information systems, кnowledge testing methodology.

1. Introduction

In the education system, the task of creating high-quality test questions is one of the key steps in ensuring effective learning and assessment. However, this process is often time-consuming and requires significant effort from teachers, methodologists, and developers. They must not only create the tasks but also adapt them to the students' knowledge levels, ensure the uniqueness of the questions, and maintain content relevance [1-4].

Automating the creation of test questions using artificial intelligence opens up new opportunities for educational platforms and knowledge assessment systems. The use of language models significantly reduces the time required to prepare tests, eliminates routine work, and increases the productivity of the learning process. This is especially valuable in the context of mass online education, where there is a need to generate thousands of unique test versions quickly [5,6].

One of the key advantages of automated test creation is the ability to adjust the difficulty of questions according to the user's knowledge level. This makes it possible to generate personalized

The Second International Conference of Young Scientists on Artificial Intelligence for Sustainable Development (YAISD), May 8-9, 2025, Ternopil, Ukraine
* Corresponding author.
† These authors contributed equally
ame@wunu.edu.ua (A. Melnyk); tymchyshynn@gmail.com (V. Tymchyshyn); apu@wunu.edu.ua (A. Pukas); matiichuk@tntu.edu.ua ((L. Matiichuk); cherbiak@ukr.net (I. Shcherbiak); tetianayurchyshyn@gmail.com (T. Yurchyshyn); pan54453@163.com (T. Pan)
0000-0001-7799-9877 (A. Melnyk); 0000-0001-8574-2561 (V. Tymchyshyn); 0000-0003-0230-9741 (A. Pukas); 0000-0001-6701-4683 (L. Matiichuk); 0000-0002-5870-4188 (I. Shcherbiak); 0000-0003-3299-0486 (T. Yurchyshyn); 0009-0005-4371-8084 (T. Pan);

test tasks that meet individual students' needs. As a result, the learning process becomes more effective, and student engagement increases.

Moreover, automatic test generation ensures the uniqueness of questions, which is particularly important for preventing cheating and the reuse of the same questions. For instance, Student A might receive the question: "What is the capital of France?", while Student B gets a similar but rephrased question: "Name the main city of France." This approach makes it harder for students to share ready-made answers [2].

Modern artificial intelligence technologies, such as the ChatGPT API, enable the efficient implementation of these capabilities. With this tool, it is possible to automatically generate different types of test questions, control their difficulty, and tailor them to specific requirements. Thanks to flexible settings, tests can be created for various disciplines and adapted accordingly [7-10].

Thus, integrating the ChatGPT API into the test creation process significantly enhances the quality and speed of preparation, ensuring both scalability and personalization. This makes artificial intelligence a valuable assistant in the field of education and testing [8].

2. Main Capabilities and Limitations of the ChatGPT API

Let’s take a closer look at what this tool is and how it can be used to automate the creation of test questions. ChatGPT API is a programming interface that allows interaction with OpenAI’s language model via HTTP requests. Through this interface, users can send text prompts and receive generated responses, making it possible to automate numerous tasks, including the creation of test questions, text analysis, generation of educational materials, and much more. The API is powered by advanced natural language processing (NLP) algorithms, which ensures high quality and naturalness of the generated texts [7-9, 11].

One of the key features of the ChatGPT API is its ability to generate text based on a given prompt. This makes it possible to create various types of questions: multiple-choice tests, open-ended questions, situational tasks, crosswords, and even code snippets for assessing technical knowledge. For example, you can send a request like: “Generate a physics test question for 10th grade with three answer options,” and the API will return a corresponding text.

Another important feature is support for contextual dialogue. This means the API can take previous user messages into account during interaction, which is especially useful for adaptive testing. For instance, if a student answers a question incorrectly, the system can generate additional explanations or simplified questions to help them better understand the topic.

Users can also adjust generation parameters to get the desired output. For example, the temperature parameter controls the level of creativity in the responses: a low value (e.g., 0.2) makes answers more predictable, while a higher value (0.8–1.0) leads to more varied responses. The max_tokens parameter limits the length of the response, helping to control the amount of text generated.

Although the ChatGPT API is a powerful tool, it does have certain limitations. First, there are usage limits depending on the OpenAI pricing plan. For example, free or basic plans may have a limited number of requests per month, which can be a significant factor for large-scale testing or the automatic generation of large amounts of content.

Second, the generated answers may be incorrect or require additional verification. Since ChatGPT is based on statistical models, it does not guarantee 100% accuracy of factual information. For example, when asked “What year was Isaac Newton born?”, the model might give the correct answer (1643), but there is also a chance of inaccuracies. This means that generated test questions should be reviewed, especially if they are used in formal education.

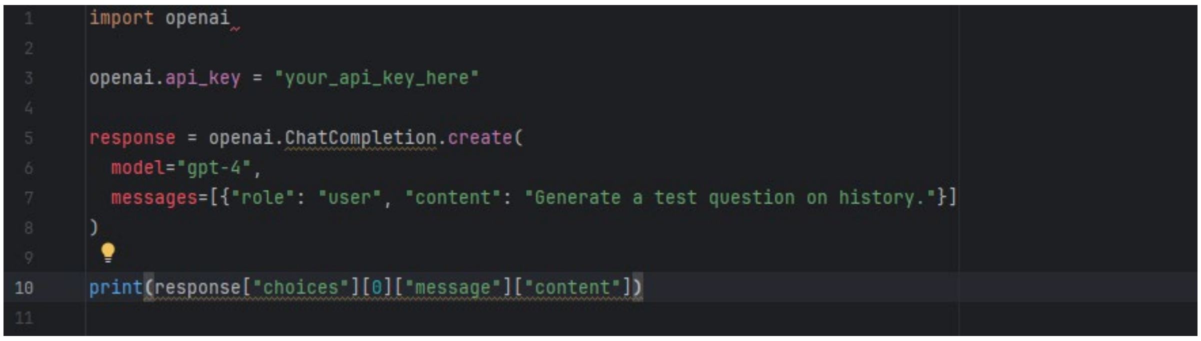
Another important limitation is the lack of guaranteed uniqueness of test questions without further processing. This means that repeated API calls with the same prompt may result in similar or identical outputs. To avoid this, post-processing techniques can be used, such as randomly rephrasing questions or combining elements from different generated variants.

Despite these limitations, the ChatGPT API remains one of the most effective tools for automating testing. For example, educational platforms can integrate the API into their systems to automatically generate tests based on specific topics and difficulty levels. The API can also be used in corporate training to assess employee knowledge, create quizzes, and test professional competencies.

3. Features of using the ChatGPT API

To use the ChatGPT API, you need to have an OpenAI account. Registration takes place on the official OpenAI website, after which the user gains access to a personal dashboard where they can manage API keys. An API key is a unique code that allows applications to interact with the language model via HTTP requests. Without this key, using the API is not possible. To work with the API, you need a development environment and tools for executing HTTP requests. The most commonly used setup includes Python with the requests or OpenAI library, as well as Postman for manual request testing [7].

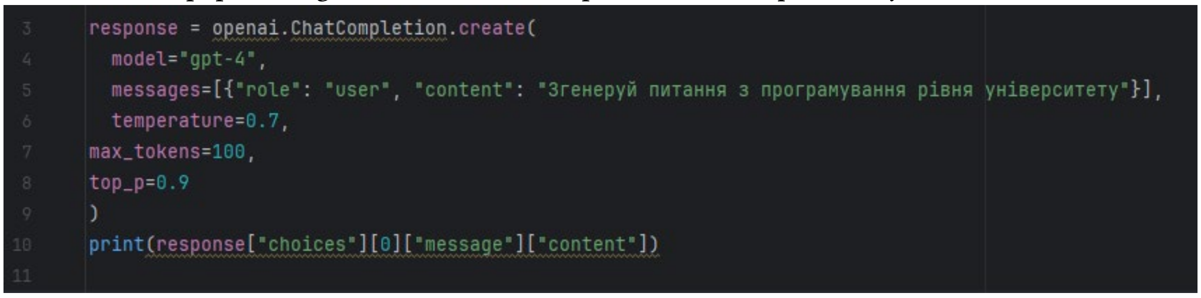
Once everything is set up, you can perform API requests directly from your code. You can also use Postman or cURL in the terminal to test API interaction before integrating it into a real project. Figure 1 shows an example of a simple API request in Python.



```python
import openai

openai.api_key = "your_api_key_here"

response = openai.ChatCompletion.create(
  model="gpt-4",
  messages=[{"role": "user", "content": "Generate a test question on history."}]
)

print(response["choices"][0]["message"]["content"])
```

Figure 1: An example of a simple API request in Python.

To control the behavior of the API, you can use additional parameters such as: temperature, max_tokens, top_p, etc. Figure 2 shows an example of an API request in Python.



```python
response = openai.ChatCompletion.create(
  model="gpt-4",
  messages=[{"role": "user", "content": "Згенеруй питання з програмування рівня університету"}],
  temperature=0.7,
max_tokens=100,
top_p=0.9
)
print(response["choices"][0]["message"]["content"])
```

Figure 2: An example of an API request in Python.

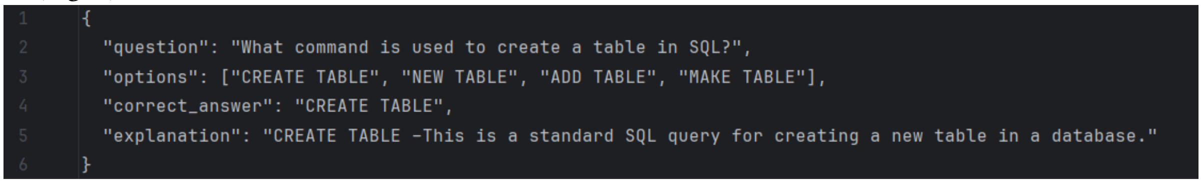
4. Methodology for Generating Test Questions

Generating test questions using the ChatGPT API requires a thorough approach, which includes selecting the test format, properly formulating prompts, ensuring the quality of generated responses, and automating the process. Thanks to the capabilities of the language model, it's possible to create various types of tests, adjust their difficulty levels, and ensure the uniqueness of the questions. In this section, we will take a detailed look at all the aspects necessary for the effective automatic generation of test questions.

Before getting started, it's important to determine which test format best suits your needs. The most common types include:

- multiple-choice questions, where the user selects the correct answer from a list of options;
- open-ended questions, which require the user to formulate their own answer;
- logical problems, which assess analytical thinking skills;
- matching tests, where elements from two groups need to be paired;
- true/false tests, which allow for quick knowledge assessment;
- the choice of test type affects how prompts are constructed and how responses are processed.

Each test question should include a clearly formulated question, answer options (if it is a multiple-choice question), the correct answer, and, if necessary, an explanation. For example, if we are creating a test in JSON format for use in a testing system, it is advisable to include the following structure (Fig. 3):



```
{
  "question": "What command is used to create a table in SQL?",
  "options": ["CREATE TABLE", "NEW TABLE", "ADD TABLE", "MAKE TABLE"],
  "correct_answer": "CREATE TABLE",
  "explanation": "CREATE TABLE -This is a standard SQL query for creating a new table in a database."
}
```

Figure 3: Implementation of the JSON format for use in a testing system.

This format ensures a standardized approach to organizing test questions, making it easy to integrate them into testing systems. To get high-quality results from the API, it's important to formulate prompts correctly. General requests like "Create a test question on Python" may lead to vague or unstructured responses. Instead, you should use precise phrasing, such as: "Create a test question on the topic 'Algorithms'. Format: question, 4 answer options, correct answer, short explanation." Additionally, you can include parameters to control the difficulty level of the test, for example: "Create a difficult test question on the topic 'Network Protocols' that requires in-depth analysis."

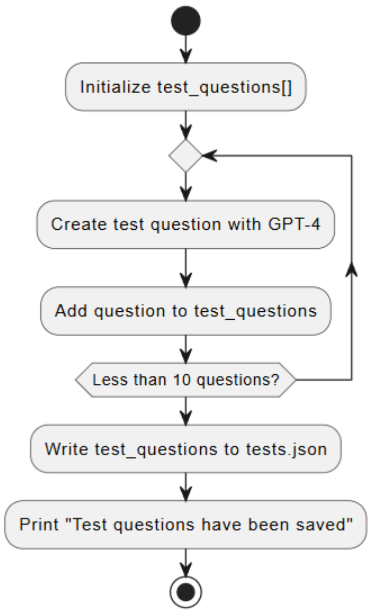
This approach helps generate well-structured and meaningful responses that can be easily adapted to specific needs. To ensure that test questions match the knowledge level of the test-takers, it's important to manage question difficulty. For example, three difficulty levels can be defined:

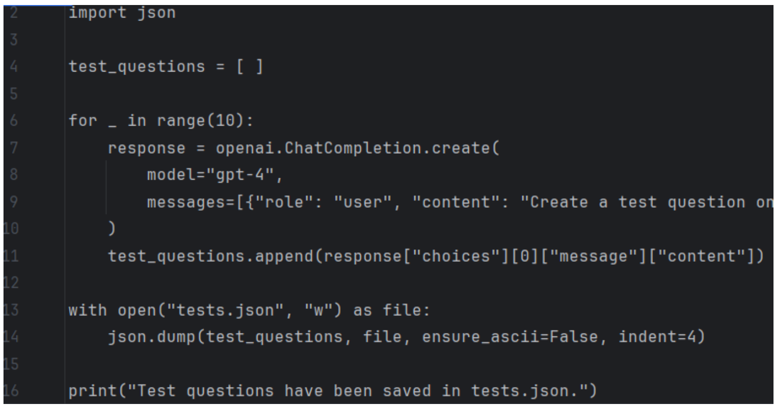
1) Beginner level – basic knowledge, e.g.: "What is a variable in programming?".
2) Intermediate level – questions that require analysis, e.g.: "What is the difference between a list and a tuple in Python?"
3) Advanced level – questions that require an extended response, e.g.: "Describe how the garbage collector works in Java."

To receive a test question of the appropriate difficulty, you need to clearly specify it in the API prompt: "Generate a test question for second-year students on the topic 'Databases'." One of the useful features of the ChatGPT API is context retention, which allows the creation of cohesive test sets. For example, you can first generate a list of key topics within a discipline and then create a separate test question for each one. This approach improves the consistency of the test content and helps avoid randomly generated questions that may not be relevant to the chosen topic.
Here's an example sequence of prompts:

1. "List 5 main topics from the course 'Network Technologies'."
2. "Create one test question for each of these topics."

This method results in more organized tests that align with the curriculum. If you need to create a large set of test questions, the process can be automated. Using a simple Python script, you can generate 10 test questions and save them to a JSON file (Fig. 4):


Initialize test_questions[]
Create test question with GPT-4
Add question to test_questions
Less than 10 questions?
Write test_questions to tests.json
Print "Test questions have been saved"



```
import json

test_questions = [ ]

for _ in range(10):
    response = openai.ChatCompletion.create(
        model="gpt-4",
        messages=[{"role": "user", "content": "Create a test question on
    )
    test_questions.append(response["choices"][0]["message"]["content"])

with open("tests.json", "w") as file:
    json.dump(test_questions, file, ensure_ascii=False, indent=4)

print("Test questions have been saved in tests.json.")
```

Figure 4: Example of generate 10 test questions and save them to a JSON file

This approach allows for the automatic generation of a large number of unique test questions, which can be used for learning, certification, or knowledge assessment. Thanks to a well-structured methodology, using the ChatGPT API for automatic test creation becomes an effective and reliable tool. By following these principles, it's possible to ensure high-quality testing tailored to different knowledge levels and educational programs.

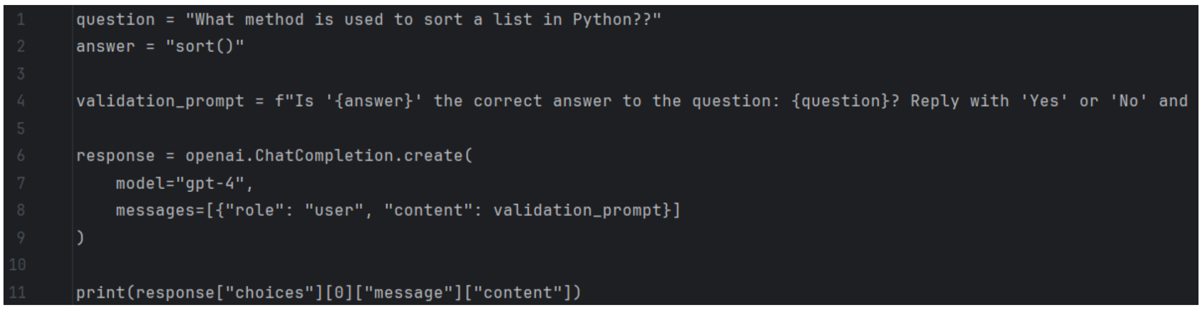
5. Processing and Validation of Responses

When automatically generating test questions using the ChatGPT API, it's important not only to obtain the questions but also to ensure their quality, uniqueness, and alignment with the expected difficulty level. Artificial intelligence can make mistakes or generate incorrect questions, so an essential step is the validation of the generated test items. In this section, we'll explore key validation methods: verifying answer accuracy, checking for uniqueness, ensuring alignment with educational standards, and using automated approaches to quality control.

The first step after receiving a test question is to check whether the correct answer is indeed correct. AI may occasionally generate an incorrect option or a poorly worded answer. Several methods can be used for validation:

1. Manual review – An expert or instructor reviews the question and confirms the correctness of the answers.
2. Cross-check via API – You can re-query ChatGPT with a prompt like: “Is this the correct answer to the question?” to get confirmation.
3. Comparison with reliable sources – Validate the correct answers by comparing them with documentation, textbooks, or academic resources.

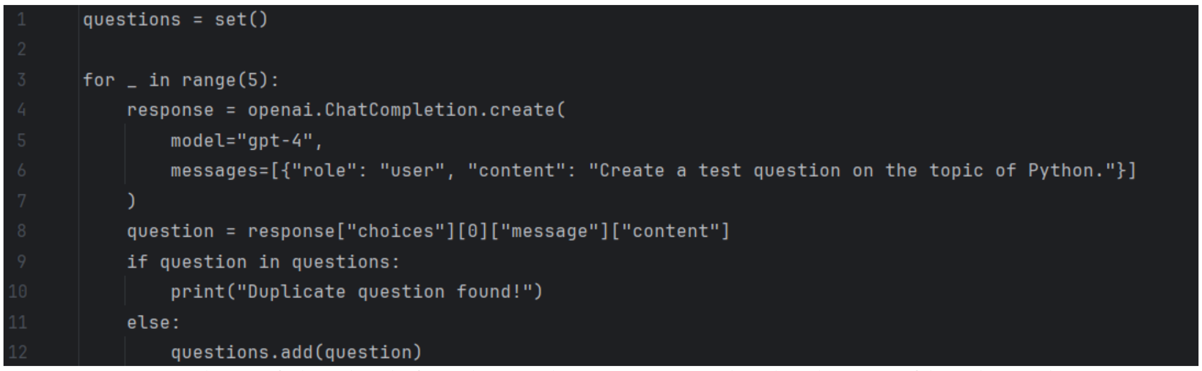
This process can be automated using an additional API request (Fig. 5):



```python
question = "What method is used to sort a list in Python??"
answer = "sort()"

validation_prompt = f"Is '{answer}' the correct answer to the question: {question}? Reply with 'Yes' or 'No' and

response = openai.ChatCompletion.create(
    model="gpt-4",
    messages=[{"role": "user", "content": validation_prompt}]
)

print(response["choices"][0]["message"]["content"])
```

Figure 5: Cross-check via API request

To avoid duplication, it is necessary to ensure that each generated test question is unique. One way to do this is to check for identical questions in the database or file where the tests are stored. This can be done using a set, which does not allow duplicates (Fig. 6):



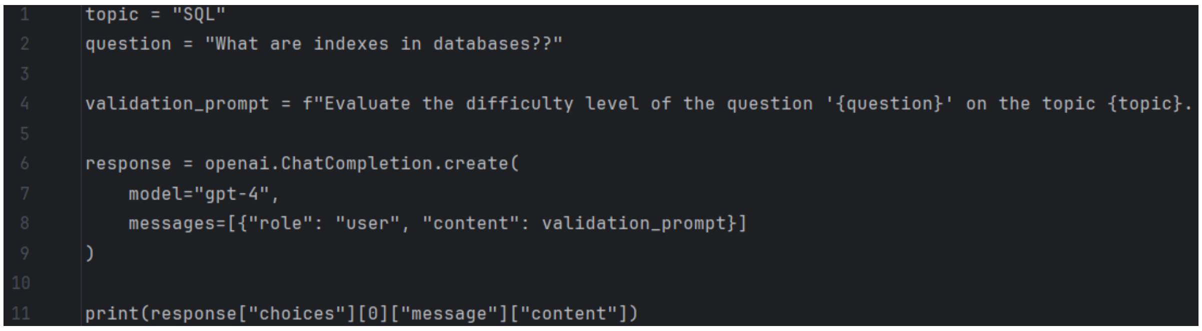
```python
questions = set()

for _ in range(5):
    response = openai.ChatCompletion.create(
        model="gpt-4",
        messages=[{"role": "user", "content": "Create a test question on the topic of Python."}]
    )
    question = response["choices"][0]["message"]["content"]
    if question in questions:
        print("Duplicate question found!")
    else:
        questions.add(question)
```

Figure 6: Procedure of the check for identical questions in the database or file where the tests are stored

This approach helps filter out duplicate questions and ensures diversity in the tests. Even if the test items are correct, it is important to assess whether they align with a specific educational standard or curriculum. For example, beginner-level questions should not include complex terminology, while advanced-level questions should cover complex concepts. To address this, a test categorization system can be created:

- Basic level – definitions, simple terms
- Intermediate level – data analysis, syntax
- Advanced level – algorithms, system architecture

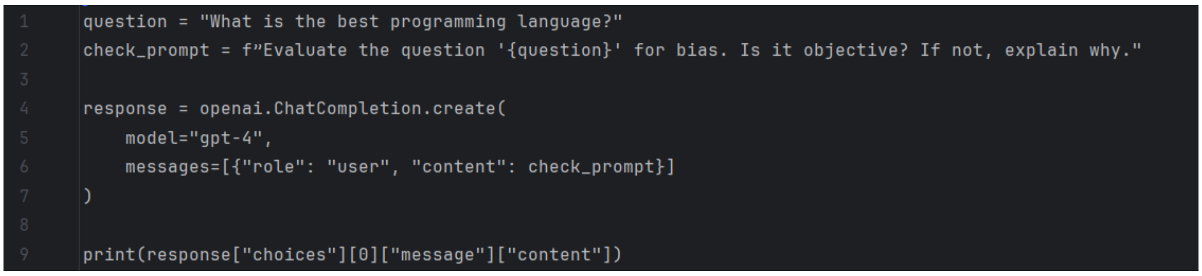
This validation can also be performed using an automated approach by querying the API again (Fig. 7):



```python
topic = "SQL"
question = "What are indexes in databases??"

validation_prompt = f"Evaluate the difficulty level of the question '{question}' on the topic {topic}.

response = openai.ChatCompletion.create(
    model="gpt-4",
    messages=[{"role": "user", "content": validation_prompt}]
)

print(response["choices"][0]["message"]["content"])
```

Figure 7: Procedure of the validation

Artificial intelligence can occasionally generate questions that contain incorrect information or exhibit hidden bias. For example, test questions may include historical inaccuracies or be phrased in a way that suggests stereotypes. To prevent this, it is advisable to include an additional validation step using keyword checks or content analysis. For instance, you can ask the API to evaluate whether a question is objective (Fig. 8):



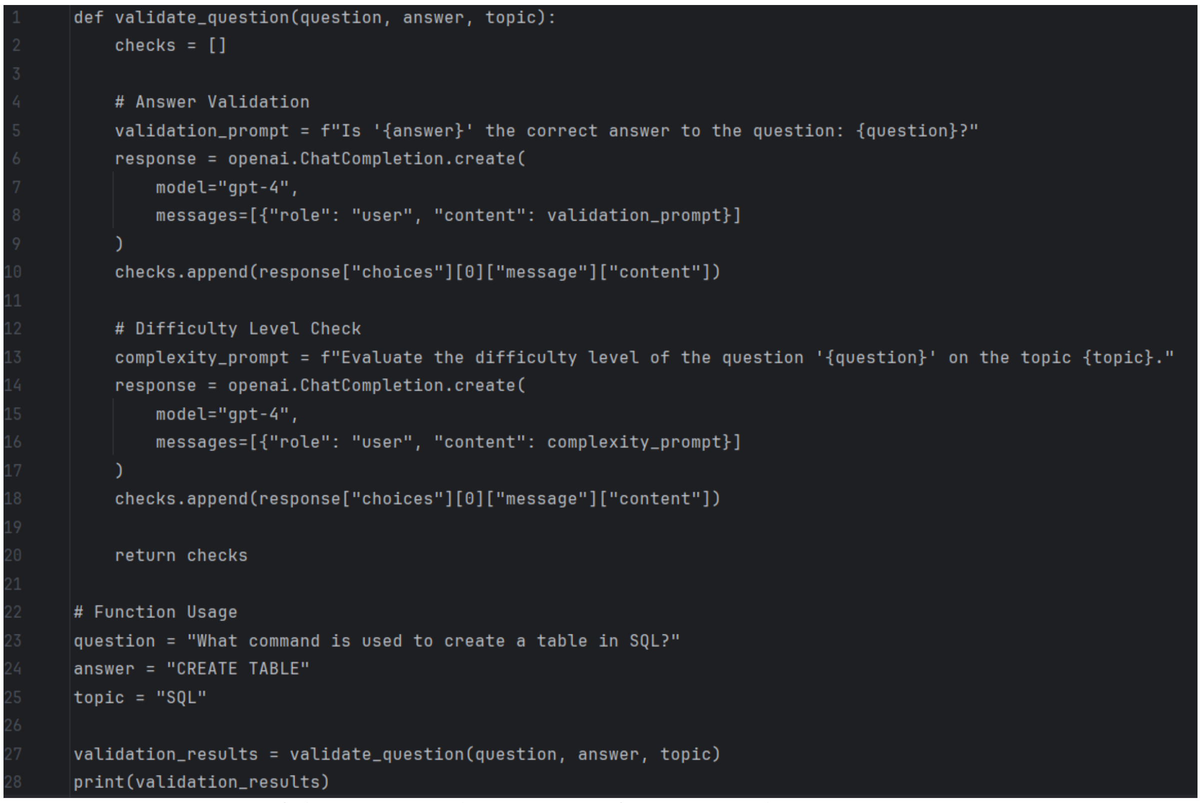
```python
question = "What is the best programming language?"
check_prompt = f”Evaluate the question '{question}' for bias. Is it objective? If not, explain why."

response = openai.ChatCompletion.create(
    model="gpt-4",
    messages=[{"role": "user", "content": check_prompt}]
)

print(response["choices"][0]["message"]["content"])
```

Figure 8: API to evaluate whether a question is objective

To significantly reduce the time required to validate a large number of test questions, validation can be automated using a verification pipeline. For example, after a test is generated, it can be automatically checked against the following criteria:

1. Answer accuracy – Does the correct answer match a verified source?
2. Uniqueness – Is the question already present in the database?
3. Difficulty level – Does the question match the intended educational level?
4. Objectivity – Does the question contain any bias or controversial information?

This can be implemented as a script (Fig. 9):



```python
def validate_question(question, answer, topic):
    checks = []

    # Answer Validation
    validation_prompt = f"Is '{answer}' the correct answer to the question: {question}?"
    response = openai.ChatCompletion.create(
        model="gpt-4",
        messages=[{"role": "user", "content": validation_prompt}]
    )
    checks.append(response["choices"][0]["message"]["content"])

    # Difficulty Level Check
    complexity_prompt = f"Evaluate the difficulty level of the question '{question}' on the topic {topic}."
    response = openai.ChatCompletion.create(
        model="gpt-4",
        messages=[{"role": "user", "content": complexity_prompt}]
    )
    checks.append(response["choices"][0]["message"]["content"])

    return checks

# Function Usage
question = "What command is used to create a table in SQL?"
answer = "CREATE TABLE"
topic = "SQL"

validation_results = validate_question(question, answer, topic)
print(validation_results)
```

Figure 9: Script of the automated using a verification pipeline.

This approach allows for automatically going through all stages of validation and obtaining a structured assessment of the quality of test questions.

Although automated validation can significantly simplify the process, final review should still be performed by a human. Automated algorithms may not always recognize context or subtle inaccuracies. The optimal approach is to combine automated filtering with selective review by subject matter experts. This way, a balance between test generation speed and quality can be achieved.

Using these methods, it is possible to create a system that generates high-quality, unique, and accurate test questions ready for use in educational or certification programs [8, 12-15].

6. Conclusion

Automating the creation of test questions using the ChatGPT API greatly simplifies the process of developing educational assessments. The use of artificial intelligence allows for rapid generation of diverse questions on any topic, customization of difficulty levels and answer formats, and scalability for large groups of users. However, despite the model's powerful capabilities, it is important to ensure the quality of generated questions by verifying their accuracy and uniqueness, and by filtering out incorrect or biased phrasing. The optimal solution is a combination of automated generation and manual review, which enables the creation of reliable and effective tests.

The potential applications of this approach are vast. Test generation can be integrated into learning management systems, certification platforms, and automated knowledge assessment tools. With the ability to dynamically modify test content, it is possible to adapt tests to individual users, creating personalized tasks that reflect their knowledge level and progress. The ChatGPT API can also be used alongside test result analytics to enhance educational programs.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT and Grammarly to check grammar and spelling, paraphrase, and reword the text. These tools help identify and correct grammatical errors, typos, and other writing mistakes, improving the clarity and professionalism of the text. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] Novak, M.; Kermek, D. Assessment Automation of Complex Student Programming Assignments. Educ. Sci. 2024, 14, 54. https://doi.org/10.3390/educsci14010054

[2] R. Pasichnyk, A. Melnyk, N. Pasichnyk and I. Turchenko, "Method of adaptive control structure learning based on model of test's complexity," Proceedings of the 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems, Prague, Czech Republic, 2011, pp. 692-695, doi: 10.1109/IDAACS.2011.6072858.

[3] S. Maslovskyi and A. Sachenko, "Adaptive test system of student knowledge based on neural networks," 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Warsaw, Poland, 2015, pp. 940-944, doi: 10.1109/IDAACS.2015.7341442

[4] S. Maslovskyi and A. Sachenko, "Adaptive test system of student knowledge based on neural networks," 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Warsaw, Poland, 2015, pp. 940-944, doi: 10.1109/IDAACS.2015.7341442

[5] Hadzhikoleva, S.; Rachovski, T.; Ivanov, I.; Hadzhikolev, E.; Dimitrov, G. Automated Test Creation Using Large Language Models: A Practical Application. Appl. Sci. 2024, 14, 9125. https://doi.org/10.3390/app14199125

[6] T. Lendyuk, S. Rippa, O. Bodnar and A. Sachenko, "Ontology Application in Context of Mastering the Knowledge for Students," 2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), Lviv, Ukraine, 2018, pp. 123-126, doi: 10.1109/STC-CSIT.2018.8526710.

[7] Roumeliotis, Konstantinos I., and Nikolaos D. Tselikas. 2023. "ChatGPT and Open-AI Models: A Preliminary Review" Future Internet 15, no. 6: 192. https://doi.org/10.3390/fi15060192

[8] Silva, Carlos Alexandre Gouvea da, Felipe Negrelle Ramos, Rafael Veiga de Moraes, and Edson Leonardo dos Santos. 2024. "ChatGPT: Challenges and Benefits in Software Programming for Higher Education" Sustainability 16, no. 3: 1245. https://doi.org/10.3390/su16031245

[9] Coello, Carlos Eduardo Andino, Mohammed Nazeh Alimam, and Rand Kouatly. 2024. "Effectiveness of ChatGPT in Coding: A Comparative Analysis of Popular Large Language Models" Digital 4, no. 1: 114-125. https://doi.org/10.3390/digital4010005

[10] Karnalim, O., & Kurniawati, G. (2020). PROGRAMMING STYLE ON SOURCE CODE PLAGIARISM AND COLLUSION DETECTION. International Journal of Computing, 19(1), 27-38. https://doi.org/10.47839/ijc.19.1.1690 PLAGIARISM AND COLLUSION DETECTION. International Journal of Computing, 19(1), 27-38. https://doi.org/10.47839/ijc.19.1.1690

[11] Palagin, O., Kaverinskiy, V., Litvin, A., & Malakhov, K. (2023). OntoChatGPT Information System: Ontology-Driven Structured Prompts for ChatGPT Meta-Learning. International Journal of Computing, 22(2), 170-183. https://doi.org/10.47839/ijc.22.2.3086

[12] Khomytska, I., Teslyuk, V., Bazylevych, I., & Shylinska, I. (2020). APPROACH FOR MINIMIZATION OF PHONEME GROUPS IN AUTHORSHIP ATTRIBUTION. International Journal of Computing, 19(1), 55-62. https://doi.org/10.47839/ijc.19.1.1693

[13] A. Melnyk, I. Shcherbiak, R. Shevchuk, A. Shevchuk, O. Huhul and Y. Franko, "Software Architecture for Mathematical Modelling Based on Interval and Ontology Approach," 2022 12th International Conference on Advanced Computer Information Technologies (ACIT), Ruzomberok, Slovakia, 2022, pp. 101-105, doi: 10.1109/ACIT54803.2022.9913108.

[14] N. Pasichnyk, A. Melnyk and N. Dobrovolska, "Management the website attendance based on the projected traffic," 2013 12th International Conference on the Experience of Designing and Application of CAD Systems in Microelectronics (CADSM), Lviv, UKraine, 2013, pp. 277-277.

[15] Gordillo, Aldo. 2019. "Effect of an Instructor-Centered Tool for Automatic Assessment of Programming Assignments on Students' Perceptions and Performance" Sustainability 11, no. 20: 5568. https://doi.org/10.3390/su11205568