МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
МИКОЛАЇВСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ

Факультет менеджменту

Кафедра економічної кібернетики, комп'ютерних наук та інформаційних технологій

Кваліфікаційна наукова
праця на правах рукопису

КОНОПЛЯНІКОВ Ілля Євгенійович

УДК 621.392.71:004.451.1(043.2)

КВАЛІФІКАЦІЙНА РОБОТА

СТВОРЕННЯ ПРОГРАМИ ДЛЯ ГЕНЕРАЦІЇ ТЕСТОВИХ ПИТАНЬ З ВІДКРИТИХ ОСВІТНІХ МАТЕРІАЛІВ

Спеціальність 122 «Комп'ютерні науки»
Галузь знань – 12 «Інформаційні технології»

Подається на здобуття освітнього ступеня «Бакалавр»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Конопляніков І. Є.

Науковий керівник:
Тищенко Світлана Іванівна, кандидат педагогічних наук, доцент.

Завідувач кафедри:
Тищенко Світлана Іванівна, кандидат педагогічних наук, доцент.

Миколаїв–2025

АНОТАЦІЯ

Конопляніков І.Є. Створення програми для генерації тестових питань з відкритих освітніх матеріалів. – Кваліфікаційна наукова робота на правах рукопису.

Робота на здобуття освітнього ступеня бакалавра за спеціальністю 122 «Комп'ютерні науки». Миколаївський національний аграрний університет, Миколаїв, 2025.

Ця кваліфікаційна робота присвячена розробці програми для автоматичної генерації тестових питань на основі відкритих освітніх матеріалів. У сучасних умовах розвитку цифрової освіти важливо забезпечити швидке та якісне створення тестових завдань для оцінювання знань, що потребує використання інноваційних підходів обробки природної мови.

У даній роботі проведено аналіз методів та інструментів обробки природної мови (NLP), визначено вимоги до системи генерації тестових питань, розроблено архітектуру програмного забезпечення та здійснено моделювання процесу вилучення основної інформації з текстів. Детально описано процес розробки програмної частини, включаючи вибір технологій (Python, spaCy, PyQt5), створення алгоритму генерації питань та реалізацію зручного інтерфейсу користувача.

Результатом роботи є функціональна програма, яка дозволяє автоматично аналізувати текстові, PDF та HTML-документи, виділяти ключові сутності (персонажі, дати, місця подій) і формувати логічні тестові завдання різних типів. Реалізований інтерфейс забезпечує зручність у використанні та можливість налаштування параметрів генерації.

Ця робота підкреслює важливість використання відкритих освітніх ресурсів для підвищення ефективності навчання та демонструє практичні аспекти створення сучасних інтелектуальних освітніх систем.
Ключові слова: генерація тестів; обробка природної мови; відкриті освітні матеріали; Python; spaCy; PyQt5; штучний інтелект; освітні технології;

аналіз тексту; автоматизація тестування; ключові сутності; інтерфейс користувача; NLP; навчальні системи; цифрова освіта.

ABSTRACT

Konoplyanikov I.E. Creation of a program for generating test questions from open educational materials. – Qualification scientific work in the form of a manuscript.

Work towards obtaining a bachelor's degree in specialty 122 «Computer Science». Mykolaiv National Agrarian University, Mykolaiv, 2025

This qualification work is dedicated to the development of a program for the automatic generation of test questions based on open educational resources. In today's digital education landscape, it is important to ensure the fast and high-quality creation of assessment tasks, which requires the use of innovative natural language processing (NLP) approaches.

The work analyzes methods and tools for natural language processing, defines the requirements for the test question generation system, designs the architecture of the software, and models the process of extracting key information from educational texts. The development process is described in detail, including the selection of technologies (Python, spaCy, PyQt5), the creation of the question generation algorithm, and the implementation of a user-friendly graphical interface.

The result of this work is a fully functional program that can automatically analyze educational content in formats such as text, PDF, and HTML, extract key entities (characters, dates, locations), and generate logical test questions of various types. The implemented interface ensures ease of use and allows users to configure generation parameters.

This project highlights the importance of using open educational resources to improve the efficiency of the learning process and demonstrates practical aspects of creating modern intelligent educational systems.

Keywords: test generation; natural language processing; open educational materials; Python; spaCy; PyQt5; artificial intelligence; educational technologies; text analysis; test automation; key entities; user interface; NLP; learning systems; digital education.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному цифровому світі освіта дедалі більше інтегрується з інформаційними технологіями. Одним із важливих напрямів є автоматизація процесу створення тестових завдань для оцінювання знань студентів. Відкриті освітні матеріали (Open Educational Resources, OER) надають широкі можливості для побудови систем навчання, але їх використання для створення тестів часто потребує значних людських ресурсів. Актуальність розробки програмного забезпечення для автоматизованої генерації тестових питань обумовлюється необхідністю підвищення ефективності та якості освітнього процесу.

Мета роботи полягає в розробці програми для автоматичної генерації тестових питань із відкритих освітніх матеріалів із використанням методів обробки природної мови (NLP) та забезпечення можливості швидкого створення якісних тестів для різних навчальних дисциплін.

Завдання дослідження:

- проаналізувати існуючі методи обробки природної мови та генерації тестових питань;
- дослідити формати освітніх матеріалів, придатні для автоматизованого опрацювання;
- проєктувати алгоритм вилучення основної інформації та формування тестових завдань;
- реалізувати програмне забезпечення для генерації тестів із можливістю налаштування параметрів;
- провести тестування створеної програми на реальних освітніх матеріалах і оцінити її ефективність.

Об'єкт дослідження – процес автоматизації створення тестових завдань із використанням відкритих освітніх ресурсів.

Предметом дослідження є методи вилучення інформації з текстових матеріалів та автоматична генерація тестових питань за допомогою технологій обробки природної мови.

Методи дослідження:

– теоретичний аналіз наукових джерел і сучасних підходів до обробки природної мови.

– моделювання алгоритмів вилучення ключової інформації.

– проєктування та розробка програмного забезпечення за допомогою Python, бібліотек spaCy і PyQt5.

– експериментальна перевірка працездатності програми на прикладі різних відкритих освітніх матеріалів.

Практична цінність розробки полягає у створенні універсального інструменту для викладачів і розробників навчальних курсів, який дозволяє суттєво зекономити час на складання тестів та забезпечити індивідуалізацію освітнього процесу.

Структура роботи. Дипломна робота складається з вступу, трьох розділів, висновків, списку використаних джерел і додатків. Робота обсягом 54 сторінок містить 2 таблиць, 7 рисунків.

РОЗДІЛ 1
ОБРОБКА ПРИРОДНОЇ МОВИ ДЛЯ АВТОМАТИЗАЦІЇ ГЕНЕРАЦІЇ ТЕСТОВИХ ПИТАНЬ

1.1 Огляд методів та інструментів обробки природної мови (NLP)

Обробка природної мови (Natural Language Processing, NLP) є однією з ключових і динамічно розвиваючих галузей штучного інтелекту, спрямованою на забезпечення комп'ютерних систем здатністю розуміти, інтерпретувати та генерувати природну людську мову у її різноманітних формах. Ця сфера об'єднує міждисциплінарні знання, що охоплюють лінгвістику, інформатику, статистику та машинне навчання, і надає інструменти для автоматизації аналізу текстів, розпізнавання мови, машинного перекладу та створення освітніх матеріалів. Ранній період розвитку NLP базувався переважно на правилах заснованих методах, які використовували жорстко закодовані граматичні і синтаксичні правила для обробки тексту. Хоча такі підходи відзначалися високою точністю у специфічних завданнях, їхня застосовність була обмежена через низьку гнучкість та складність адаптації до варіативності природної мови і помилок, що ускладнювало масштабування в реальних застосуваннях [17].

Подальший розвиток галузі супроводжувався впровадженням статистичних методів, що базувалися на ймовірнісних моделях, які дозволяли моделювати структуру мови на основі великих корпусів текстів. Методи на основі n-грам та інші статистичні підходи застосовувалися для прогнозування наступних слів та аналізу частотних закономірностей. Незважаючи на те, що такі методи мали значні переваги у гнучкості, вони були обмежені в контексті довготривалих залежностей і глибокої семантичної інтерпретації [18]. Водночас машинне навчання відкрило нові горизонти, де алгоритми самостійно виявляли патерни в даних без потреби жорстко прописаних правил. Алгоритми на кшталт наївного байєса, методів опорних векторів, дерев рішень та логістичної регресії набули широкого застосування у задачах класифікації текстів, аналізу тональності та тематичного моделювання, проте вимагали ретельного

формування числових ознак, таких як TF-IDF або Bag-of-Words, що не враховували семантичний контекст [19].

Революційним кроком у NLP стало впровадження глибинного навчання, яке дозволило моделювати складні мовні закономірності і контексти завдяки використанню нейронних мереж, зокрема рекурентних нейронних мереж (RNN) та довготривалої короткочасної пам'яті (LSTM). Проте найбільший прорив забезпечила архітектура трансформерів, що є основою таких моделей, як BERT, GPT, T5 та їхніх численних варіантів. Завдяки здатності одночасно обробляти послідовності слів у двонаправленому контексті і фокусуватись на релевантних елементах тексту, трансформери продемонстрували значне покращення у завданнях розпізнавання, генерації та інтерпретації природної мови [20]. Ці моделі підтримують мультимовність, що відкриває нові можливості для роботи з українською та іншими менш поширеними мовами, розширюючи горизонти застосування NLP у різних сферах.

Практичне застосування NLP значною мірою забезпечують численні програмні бібліотеки та інструменти. Однією з найбільш популярних є spaCy – високопродуктивна бібліотека, що підтримує широкий спектр мовних задач: токенізацію, частиномовний аналіз, виявлення іменованих сутностей, синтаксичний розбір тощо. Особливістю spaCy є можливість інтеграції з сучасними моделями глибокого навчання і ефективна робота з українською мовою, що є важливим для локалізованих застосунків [21]. Академічно орієнтованою бібліотекою є NLTK, яка забезпечує широкий набір алгоритмів та корпусів, активно застосовується в навчанні та дослідженнях. Для побудови глибинних нейронних мереж використовуються фреймворки PyTorch і TensorFlow, які лежать в основі багатьох сучасних трансформерних моделей [22]. Крім того, платформа Hugging Face Transformers пропонує зручний доступ до великої кількості попередньо навчених мовних моделей, у тому числі з підтримкою української мови, що дозволяє значно скоротити час на розробку та адаптацію систем NLP [23].

Сучасні системи обробки природної мови все більше орієнтуються на комбінування лінгвістичних знань із можливостями машинного та глибинного навчання, створюючи інтелектуальні платформи, здатні не лише аналізувати та класифікувати тексти, а й формувати якісний освітній контент, автоматично генерувати запитання, відповідати на складні запити і вести природні діалоги. Це забезпечує нові перспективи в освіті, бізнесі, охороні здоров'я, юриспруденції та багатьох інших сферах, де робота з текстовою інформацією має вирішальне значення [24]. Продовження розвитку NLP обумовлене як удосконаленням моделей і алгоритмів, так і розширенням мовної підтримки, що робить обробку природної мови доступною і для україномовного середовища [25].

Таким чином, NLP представляє собою синтез лінгвістики, інформатики і статистики, який у поєднанні з сучасними методами машинного навчання формує потужний інструментарій для розв'язання широкого спектра задач, пов'язаних із розумінням і генерацією природної мови. Від традиційних правил до трансформерних моделей, ця галузь продовжує активно розвиватися, надаючи можливості для автоматизації та інтелектуалізації взаємодії людини і комп'ютера [26]. Інтеграція різних технологій та підходів дозволяє досягати високої точності і надійності, що є критично важливим у контексті сучасних інформаційних систем і освітніх платформ [27]. Особлива увага приділяється забезпеченню мультикультурної та мультимовної підтримки, що сприяє розширенню застосувань NLP у глобальному масштабі [28]. В цілому, подальший розвиток обробки природної мови сприяє підвищенню якості автоматизованого аналізу тексту та генерації інформації, що має значення для численних практичних завдань у XXI столітті [29, 30].

1.2 Аналіз різних підходів до генерації питань із тексту: rule-based vs. Machine learning

Процес генерації тестових питань із текстового матеріалу є однією з найцікавіших та складних задач у сфері обробки природної мови. Для

розв'язання цієї задачі на сьогодні існує декілька основних підходів, серед яких найбільш поширеними є rule-based системи (засновані на правилах) та моделі машинного навчання.

Rule-based підхід передбачає створення чітких правил для аналізу тексту і формування питань на основі синтаксичних та семантичних структур. Наприклад, правило може визначати, що з іменованої сутності типу «PERSON» потрібно сформулювати питання типу «Хто...?», або що з дати потрібно створити питання «Коли...?». Такі системи зазвичай базуються на використанні парсерів, морфологічного аналізу та заздалегідь встановлених шаблонів для формування завдань. Перевагами rule-based підходу є його простота в реалізації, передбачуваність результатів і можливість точно контролювати створення питань. Однак, цей підхід вимагає великої кількості ручної роботи, складно масштабується на великі корпуси текстів і слабо адаптується до нових типів даних або мовних варіацій.

Інший напрямок розвитку – це використання методів машинного навчання. В таких системах комп'ютер навчається генерувати питання на основі великої кількості прикладів «текст – питання». Особливу популярність отримали моделі глибокого навчання, що базуються на архітектурах трансформерів, зокрема моделі типу BERT, T5, GPT. Вони здатні самостійно виявляти логічні структури тексту та будувати питання навіть у випадках, коли явних граматичних правил немає. Перевагами підходу на основі машинного навчання є висока гнучкість, здатність працювати з різними стилями і мовами, а також можливість навчання на спеціалізованих наборах даних. Водночас для ефективної роботи таких систем необхідно мати великі обсяги розмічених даних і значні обчислювальні ресурси для тренування моделей.

Таблиця 1.1 Порівняння rule – based та machine learning підходів

Параметр	**Rule – based**	**Machine learning**
Гнучкість	Низька	Висока

Необхідність ручної роботи	Висока	Низька

Продовження таблиці 1.1

Якість генерації питань	Передбачувана	Висока
Потреба в навчальних даних	Немає	Висока
Масштабованість	Обмежена	Висока

Джерело: власна розробка автора

У практичних розробках часто застосовують комбіновані методики, що об'єднують rule-based механізми для базової обробки тексту і машинне навчання для побудови більш складних або контекстуально багатих питань. Такий гібридний підхід дозволяє забезпечити оптимальне поєднання точності, масштабованості та гнучкості системи генерації тестових завдань.

У рамках даної роботи обрано базову стратегію, яка поєднує rule-based обробку природної мови для виділення ключових об'єктів тексту та елементів змісту, з елементами стохастичної генерації варіантів відповідей для забезпечення різноманітності і природності тестових питань. Це дозволяє досягти високої релевантності завдань навіть за обмеженої кількості вихідних даних та забезпечити стабільність роботи програми без необхідності в складному процесі тренування моделей. Генерація запитань із тексту є важливим напрямом у галузі обробки природної мови (NLP), який знаходить застосування в освіті, інтелектуальних системах навчання, автоматичних тестових генераторах, чат-ботах та інформаційно-пошукових системах. Завдання генерації питань полягає у створенні граматично правильних, змістовно релевантних і логічних запитань на основі наданого тексту. Існує два основні підходи до реалізації цієї задачі – rule-based (заснований на правилах) та machine learning (заснований на машинному навчанні). Кожен з них має свої переваги, обмеження та сфери ефективного застосування, тому глибоке

розуміння обох підходів є ключовим для побудови якісної системи автоматичної генерації питань.

Rule-based підхід ґрунтується на використанні заздалегідь визначених лінгвістичних правил, шаблонів і структур синтаксичного аналізу, які формалізують спосіб перетворення тверджень у запитання. Цей метод зазвичай включає етапи попередньої обробки тексту, такі як токенізація, розпізнавання частин мови, іменованих сутностей, а також побудову синтаксичного дерева. Після цього, на основі шаблонів, розроблених експертами, система визначає ключові елементи речення, що можуть бути перетворені у запитання: підмет, додаток, обставини тощо. Наприклад, речення "Іван Франко народився у 1856 році" може бути перетворено на запитання "Коли народився Іван Франко?" або "Хто народився у 1856 році?" залежно від шаблону. Такий підхід дозволяє досягати високої точності, чіткості й передбачуваності результатів, особливо у вузькоспеціалізованих предметних галузях або для текстів зі строгою структурою. Проте його недоліком є обмежена гнучкість, висока залежність від мови й складність масштабування на великі обсяги тексту чи нові типи речень. Крім того, розробка та підтримка таких правил вимагає глибоких лінгвістичних знань і значних зусиль.

З іншого боку, підхід, заснований на машинному навчанні, особливо на глибокому навчанні, використовує великі корпуси даних для автоматичного навчання моделі генерації питань без необхідності ручного створення правил. У класичних реалізаціях цього методу текст представляється у вигляді числових ознак (наприклад, через one-hot кодування, TF-IDF або embeddings), а моделі типу seq2seq (послідовність у послідовність) або трансформери навчаються перетворювати вхідний контекст на відповідне запитання. Найпотужніші сучасні системи базуються на трансформерах, таких як T5 (Text-to-Text Transfer Transformer), BART або GPT, які здатні розуміти семантику тексту, виявляти приховані залежності та генерувати запитання природною мовою, зберігаючи граматичну правильність і логічну послідовність. Величезною перевагою цього підходу є його здатність до узагальнення, адаптації до різних стилів тексту, мов

і доменів без необхідності ручного втручання. Машинне навчання також дозволяє генерувати більш креативні, варіативні й контекстуально гнучкі запитання.

Проте, як і будь-який складний інтелектуальний підхід, machine learning має свої виклики. Насамперед це потреба у великій кількості якісних розмічених даних для навчання моделі – пар «текст–запитання», що не завжди доступні, особливо для менш поширених мов, таких як українська. Також існує проблема контролю якості: автоматично згенеровані запитання іноді можуть бути нечіткими, граматично некоректними або не відповідати змісту джерела. Крім того, процес навчання потребує значних обчислювальних ресурсів та відповідного технічного забезпечення.

Існують також гібридні підходи, які поєднують rule-based і machine learning стратегії. Наприклад, система може попередньо фільтрувати або розмічати ключову інформацію з тексту за допомогою лінгвістичних правил, а потім використовувати нейромережеву модель для генерації запитання на основі цієї інформації. Такий компроміс дозволяє покращити якість результатів, забезпечити логічну точність і водночас скоротити обсяг необхідних навчальних даних. Подібні системи часто використовуються в освітніх технологіях, де важливо зберегти достовірність і навчальну цінність кожного згенерованого запитання.

У цілому, вибір між rule-based і machine learning підходами до генерації запитань залежить від конкретного застосування, доступних ресурсів і поставлених цілей. Rule-based підходи забезпечують стабільність, інтерпретованість і контроль над результатом, але потребують значної роботи над правилами. Machine learning методи пропонують гнучкість, масштабованість і потенціал високої якості, але залежать від навчальних даних і технічної інфраструктури. У багатьох сучасних системах ці підходи не протиставляються, а доповнюють один одного, створюючи ефективні, адаптивні й інтелектуально сильні системи генерації питань.

1.3 Використання відкритих освітніх матеріалів як джерела для автоматизації створення тестів

Відкриті освітні матеріали (англ. Open Educational Resources, OER) відіграють важливу роль у розвитку сучасної освіти. Це навчальні ресурси, що знаходяться у відкритому доступі та можуть вільно використовуватись, адаптуватись і поширюватись без ліцензійних обмежень. До таких матеріалів належать підручники, лекційні курси, методичні рекомендації, наукові статті, навчальні відео та інтерактивні завдання.

Використання відкритих освітніх ресурсів для автоматизації створення тестових завдань має низку суттєвих переваг. По-перше, відкриті матеріали забезпечують широкий спектр тематики і високий рівень достовірності, що дозволяє створювати якісні питання для різних навчальних дисциплін. По-друге, вони дозволяють зекономити час і ресурси, оскільки немає потреби у створенні власних текстів для аналізу. По-третє, відкритість таких ресурсів дає змогу інтегрувати систему генерації тестів у різноманітні освітні платформи без правових обмежень.

Однак автоматичне використання відкритих освітніх матеріалів для генерації тестових питань ставить перед розробниками певні виклики. Серед основних проблем можна виділити неоднорідність форматів подачі інформації (текстові файли, PDF, HTML-документи), різноманітність стилів викладення матеріалу та потребу у попередній фільтрації даних для виділення релевантної інформації. Також варто враховувати, що не всі освітні тексти однаково добре структуровані, що може ускладнювати процес автоматичного вилучення ключових понять і фактів.

Для успішної роботи системи автоматизованої генерації тестів важливо забезпечити можливість обробки найпоширеніших форматів документів. Зокрема, текстові файли (.txt) дають змогу здійснювати прямий аналіз тексту без попередньої обробки. Документи у форматі PDF потребують застосування спеціальних парсерів для вилучення текстового вмісту. HTML-документи, що

часто використовуються у відкритих курсах, вимагають вміння працювати з розміткою сторінок і виділяти лише навчальний контент.

Інтеграція відкритих освітніх матеріалів у систему автоматизації створення тестових завдань відкриває нові можливості для динамічного формування тестів відповідно до актуальних навчальних потреб. Це дозволяє створювати персоналізовані тести, швидко оновлювати контент, а також робити навчання більш інтерактивним та адаптивним до рівня підготовки студентів. Основна ідея полягає в тому, щоб перетворювати навчальний текст – будь то лекція, розділ підручника чи опис теми з онлайн-курсу – у відповідні запитання, які можуть бути використані в тестах, вікторинах або системах самоконтролю. Це дозволяє значно зменшити навантаження на викладачів, які традиційно створюють тести вручну, а також забезпечити індивідуалізоване навчання за рахунок генерування великої кількості варіативних завдань. Завдяки відкритості та великій кількості освітніх ресурсів стає можливим створення широкої бази знань, що охоплює багато предметів, мов і навчальних рівнів.

Однією з ключових переваг використання відкритих освітніх матеріалів є стандартизований стиль викладу, орієнтований на чіткість, логічність і педагогічну ефективність. Це полегшує алгоритмам NLP (обробки природної мови) задачу розпізнавання структур речень, визначення головної інформації та логічних зв’язків між частинами тексту. Наприклад, типові формулювання в освітніх матеріалах – "Фотосинтез відбувається у клітинах рослин", "Основні етапи Другої світової війни включають..." – є чіткими та змістовними, що ідеально підходить для перетворення їх у запитання на кшталт "Де відбувається фотосинтез?" або "Які етапи включає Друга світова війна?".

Процес автоматизації створення тестів із відкритих матеріалів зазвичай включає кілька етапів: попередню обробку тексту (токенізація, лематизація, визначення частин мови), аналіз структури речень (синтаксичний і семантичний аналіз), виявлення ключових фактів або концептів, формування запитань (як правило, у вигляді питань з вибором відповіді, відкритих запитань або істинно-хибних тверджень) і, за потреби, генерація відповідей або варіантів відповідей.

В цьому процесі активно застосовуються сучасні NLP-технології, зокрема моделі трансформерів, які здатні не лише ідентифікувати важливу інформацію, але й формувати граматично коректні, логічно узгоджені запитання.

Також важливою перевагою відкритих освітніх матеріалів є їхня доступність у цифровому форматі, що спрощує автоматичне збирання, обробку й збереження навчального контенту. За допомогою веб-скрейпінгу, API платформ або простого парсингу HTML-сторінок можна отримати велику кількість навчальних текстів для використання у системах генерації тестів. Крім того, більшість OER-платформ, таких як Khan Academy, OpenStax, Coursera, EdEra чи Prometheus, мають ліцензії типу Creative Commons, що дозволяє легально використовувати їхній контент для дослідницьких, навчальних або програмних цілей.

З педагогічної точки зору, автоматичне створення тестів із відкритих матеріалів дозволяє забезпечити адаптивне навчання. Це означає, що система може генерувати індивідуальні завдання залежно від теми, яку вивчає студент, рівня його знань або попередніх помилок. Наприклад, після прочитання певного розділу підручника студент одразу отримує питання за його змістом, що сприяє кращому закріпленню знань і розвитку критичного мислення.

Таким чином, використання відкритих освітніх ресурсів як джерела для генерації тестових питань є перспективним напрямом, що поєднує доступність якісної інформації та можливості сучасних інформаційних технологій для розвитку освітнього середовища.

1.4 Огляд існуючих рішень для генерації тестових питань

Автоматизована генерація тестових питань (англ. *Automatic Question Generation*, QG) стала одним із ключових напрямів у галузі освітніх технологій, штучного інтелекту та обробки природної мови. Сучасні рішення, спрямовані на створення тестових завдань, розрізняються за технологічною складністю, глибиною обробки тексту, типами генерованих питань та рівнем інтеграції в навчальні платформи. Вони можуть бути як вузькоспеціалізованими

інструментами для певних дисциплін, так і універсальними фреймворками, що базуються на відкритих NLP-бібліотеках.

Серед традиційних (rule-based) систем можна відзначити ті, що використовують заздалегідь задані шаблони та правила. Наприклад, система AutoQG (2007) дозволяє формувати запитання типу «хто/що/де/коли» на основі синтаксичного аналізу англомовних речень. Принцип роботи таких рішень базується на виокремленні іменників, дієслів або прикметників як «вузлів знання» та їх подальшому трансформуванні у запитання. Попри простоту реалізації, такі системи не завжди здатні адаптуватися до контексту, що обмежує їхню застосовність у реальному освітньому середовищі.

З розвитком глибокого навчання з'явилися моделі на основі машинного навчання, які можуть автоматично навчатися з прикладів питань і відповідей. Одним із найвідоміших рішень є модель T5 (Text-to-Text Transfer Transformer) від Google, яка демонструє здатність до переформулювання тексту, включно з генерацією питань. Інша популярна модель — BERT-QG, що комбінує передтреновану трансформерну архітектуру BERT для кращого розуміння контексту з механізмами декодування для формування запитань.

Варто також виділити спеціалізовані платформи для навчального застосування, які інтегрують QG-функціонал:

- Quillionz — хмарний сервіс, що дозволяє завантажити текст, з якого автоматично створюються питання типу множинного вибору, запитання на відповідність, відкриті запитання. Quillionz використовує власну NLP-модель і має інтеграцію з Google Docs та LMS-платформами.
- Yippity — вебзастосунок, який використовує алгоритми на основі GPT-моделей для генерації запитань у реальному часі. Платформа має дружній інтерфейс, однак потребує постредагування питань.
- EduQuestion — модуль у рамках освітньої платформи Moodle, що дозволяє інтегрувати NLP-генерацію до курсів. Цей інструмент здатен створювати запитання безпосередньо з підручників або лекційних матеріалів, завантажених у вигляді PDF чи HTML.

Крім того, існують open-source рішення, які дозволяють створювати власні QG-системи, адаптовані до конкретного навчального контенту:

- QG-BERT (2020) — відкритий репозиторій на GitHub, який використовує BERT для розуміння тексту й Seq2Seq-декодер для побудови питань. Підтримує тренування моделей на власних датасетах.
- NLP4QG — фреймворк, що дозволяє підключати модулі для семантичного аналізу, coreference resolution та побудови логічних питань.

Варто зазначити, що більшість сучасних систем найкраще працюють із англомовними текстами, оскільки саме для англійської мови існує найбільша кількість анотованих корпусів та NLP-моделей. Для української мови автоматична генерація питань лише починає розвиватися, хоча вже існують адаптації моделей mBERT, XLM-RoBERTa та Ukrainian-GPT, які потенційно можуть бути застосовані в майбутньому для подібних цілей.

Таким чином, на ринку вже присутні як комерційні, так і відкриті рішення для автоматизації створення тестів. Проте більшість з них вимагає глибокої адаптації до конкретного освітнього контексту, наявності мовної підтримки та можливості інтеграції з LMS. Це створює передумови для розробки нових систем, які поєднують переваги сучасних NLP-технологій із вимогами української освітньої системи, включно з підтримкою української мови та локальних освітніх стандартів.

Окрім технічної класифікації систем генерації тестових питань, важливим є розуміння типології питань, які здатні формувати різні підходи. Умовно всі існуючі системи можна класифікувати за типами питань, які вони генерують:

Фактологічні питання (на базі «wh»-структур: хто, що, коли, де);

Питання на узагальнення або класифікацію;

Інтерпретативні питання, які потребують розуміння контексту або причинно-наслідкових зв’язків;

Питання на правильність твердження (true/false);

Питання з пропущеними словами (cloze-tests).

Сучасні системи на основі ШІ вже навчаються створювати більш когнітивно складні типи питань, наприклад, аналітичні чи проблемні. У дослідженнях 2021–2024 років (наприклад, Kumar et al., 2022; Liu et al., 2023) особливо відзначається зростання популярності нейросемантичної генерації, яка використовує семантичні карти або графи знань для формування питань, що відповідають конкретному рівню мислення за Блумом.

Ще одним перспективним напрямом є адаптивна генерація питань, коли система враховує попередні відповіді студента або рівень знань, щоб у подальшому автоматично змінювати складність завдань. Таку логіку вбудовують у системи на базі reinforcement learning або в рамках навчальних середовищ з адаптивною аналітикою (learning analytics).

У межах інтернаціоналізації освіти важливим фактором залишається мовна доступність генераторів питань. Хоча більшість досліджень зосереджені на англомовних даних, зростає кількість експериментів з багатомовними моделями. Наприклад, XGLM, mT5, mBART, XLM-RoBERTa є потужними багатомовними трансформерами, які можна адаптувати під локальні мови, зокрема й українську. Пілотні ініціативи в межах європейських проєктів Erasmus+ уже демонструють прототипи генераторів для румунської, польської, литовської, української мов, зокрема для гуманітарних дисциплін.

Університети, що активно запроваджують цифровізацію, наприклад, Університет Тарту (Естонія), Університет Твенте (Нідерланди) та КНУ імені Тараса Шевченка (Україна), впроваджують гібридні моделі, у яких питання генеруються автоматично, але проходять фазу модерації викладачем. Це дозволяє зберігати баланс між швидкістю формування тестів і їхньою якістю.

Також все більше уваги приділяється інтеграції генерації питань у LMS-платформи (Learning Management Systems), такі як Moodle, Canvas, OpenEdX, де завдяки плагінам або API інтеграціям можна безпосередньо створювати тести в межах курсу. Це знижує потребу в ручному створенні великої кількості контрольних завдань, особливо в умовах масових онлайн-курсів (MOOC).

Водночас, поряд із досягненнями, залишається низка викликів. Один із головних — якість автоматично згенерованих питань. Нерідко штучний інтелект створює граматично правильні, але семантично некоректні або беззмістовні запитання. Також виявлено ризики генерації упереджених або нерелевантних питань, що є особливо критичним у тестуванні з чутливих тем (наприклад, етика, історія, культура).

Іншим важливим аспектом є проблема еталонних відповідей: навіть якщо питання сформульовано правильно, система має точно визначити, яка відповідь є вірною, особливо у відкритих запитаннях або завданнях з відкритою відповіддю. Деякі проєкти, як-от AnswerEval або QASystemEval, досліджують автоматичне оцінювання відповідей, що відкриває нові горизонти для повної автоматизації контролю знань.

Нарешті, варто зазначити, що більшість сучасних інструментів не розроблені спеціально під потреби української освіти, яка має свої особливості: модульно-рейтингову систему, специфічну термінологію, акредитаційні вимоги. Це створює нішу для створення локалізованих рішень — програм, які можуть генерувати питання українською мовою, враховувати державні освітні стандарти та бути адаптованими під дисциплінарну специфіку (аграрні науки, медицина, менеджмент тощо).

Отже, огляд існуючих рішень свідчить про значний поступ у напрямі автоматизованої генерації тестових питань, проте водночас вказує на актуальність створення адаптивних, багатомовних і локалізованих систем, здатних відповідати на запити сучасної освітньої практики. Цей напрям залишається динамічно розвиваючим і перспективним як для дослідницьких, так і для практико-орієнтованих розробок у сфері EdTech.

На сьогоднішній день існує значна кількість розробок у галузі автоматизації створення тестових завдань, що базуються на різних підходах до обробки природної мови. Аналіз існуючих рішень дозволяє окреслити основні тенденції, сильні та слабкі сторони сучасних систем генерації питань.

Одним із популярних напрямів є використання систем, що базуються на машинному навчанні. Такі платформи, як Google Question Generation, використовують нейронні мережі для автоматичного формулювання запитань на основі тексту. Вони демонструють високу точність у генерації питань, однак потребують великих навчальних наборів даних і суттєвих обчислювальних ресурсів. Такі рішення часто орієнтовані на англійську мову і рідко підтримують інші мови, що створює певні обмеження при локалізації у різних освітніх середовищах.

Іншим прикладом є системи rule-based типу, такі як AutoQuiz або академічні проекти на базі бібліотек NLP, де питання генеруються за заданими граматичними або семантичними шаблонами. Такі системи відрізняються стабільністю роботи та передбачуваністю результатів, однак вони обмежені в гнучкості й потребують ретельного налаштування для кожного окремого домену знань.

Також слід згадати рішення на основі змішаних підходів, наприклад, QG-NLP – відкритий проект, який поєднує класичну обробку тексту та глибинне навчання для кращого формування природних запитань. Застосування гібридних методів дозволяє досягти більшого балансу між точністю, якістю питань та обчислювальними витратами.

Попри успішні приклади, більшість існуючих систем мають певні обмеження. Серед основних викликів можна виділити труднощі у забезпеченні логічної відповідності питань контексту, потребу в адаптації під різні мови, специфіку форматів освітніх матеріалів, а також необхідність оптимізації для обробки великих обсягів даних.

Важливо відзначити, що для української мови спеціалізованих рішень у сфері генерації тестових питань майже не існує. Це створює додаткову актуальність для розробки власних систем, орієнтованих на обробку текстів українською мовою, що дозволяє підвищити доступність та якість навчальних технологій для вітчизняної освіти.

Отже, аналіз існуючих рішень свідчить про те, що найбільш ефективним напрямом є використання комбінованих технологій із опорою на rule-based обробку для виділення структурованих даних і застосування машинного навчання для генерації більш природних та варіативних тестових завдань. Такий підхід забезпечує гнучкість, масштабованість і можливість адаптації системи до конкретних навчальних потреб.

РОЗДІЛ 2
АНАЛІЗ ОСВІТНІХ МАТЕРІАЛІВ ТА ПРОЕКТУВАННЯ АЛГОРИТМУ ГЕНЕРАЦІЇ ПИТАНЬ

2.1 Аналіз форматів відкритих освітніх матеріалів (текст, PDF, HTML)

У процесі розробки системи для автоматичної генерації тестових питань важливим кроком є дослідження форматів, у яких надаються відкриті освітні матеріали. Різноманітність форматів обумовлює необхідність у створенні універсальних алгоритмів обробки текстової інформації, здатних працювати з різними джерелами даних.

Одним із найпоширеніших форматів є звичайні текстові файли (.txt). Їх основною перевагою є простота структури та відсутність складних розміток, що дозволяє безпосередньо працювати з текстовим вмістом без попередньої обробки. Текстові файли часто використовуються для збереження конспектів лекцій, планів занять або базових навчальних матеріалів. Для роботи з ними достатньо базових засобів мов програмування, що робить їх зручними для аналізу в системах автоматизації. Однак такі файли не містять структурних позначок, що ускладнює автоматичне виділення логічних частин, як-от заголовки чи списки.

Іншим поширеним форматом є документи PDF (Portable Document Format). PDF використовується для збереження структурованої інформації із збереженням форматування, таблиць, зображень і стилістичних елементів. Це особливо важливо для підручників, методичних матеріалів та наукових статей. Однак основною складністю при роботі з PDF-файлами є те, що текст у них часто зберігається у вигляді графічних об'єктів або у фрагментованій формі. Наприклад, речення можуть бути розбиті на окремі блоки, які не мають логічного зв’язку. Для ефективного вилучення тексту з PDF необхідно застосовувати спеціальні парсери, такі як PyMuPDF, PDFminer, pdfplumber або Apache Tika. Якщо PDF складається з відсканованих зображень (image-based

PDF), потрібно додатково використовувати технології оптичного розпізнавання тексту (OCR), наприклад Tesseract. Це ускладнює реалізацію, але дозволяє працювати навіть з архівними матеріалами.

Формат HTML (HyperText Markup Language) широко використовується для представлення освітніх матеріалів у мережі Інтернет. Більшість сучасних платформ, таких як Coursera, EdEra, Khan Academy, Prometheus або OpenStax, використовують HTML-документи як основу контенту. Особливістю HTML є наявність тегів розмітки, які вказують на структуру документа: заголовки, абзаци, списки, таблиці, зображення. Це дає змогу точно визначати межі логічних блоків – тем, підтем, прикладів, вправ. Обробка HTML вимагає навичок роботи зі структурованими документами. Для цього використовуються такі інструменти, як BeautifulSoup, lxml, Selenium або Scrapy. При цьому виникає потреба фільтрувати технічні елементи, наприклад, бокові панелі, меню, рекламні блоки, скрипти. Виділення саме навчального контенту потребує аналізу класів, ID елементів або розташування на сторінці. Також виникає потреба в обході гіперпосилань, щоб отримати повну версію курсу або матеріалу з декількох сторінок.

Важливим аспектом роботи з HTML-контентом є також коректне оброблення мультимедійних елементів, таких як відео, аудіо чи інтерактивні вправи, які часто використовуються в сучасних навчальних платформах. Крім того, для ефективного парсингу необхідно враховувати динамічний контент, що завантажується через JavaScript, що може ускладнювати процес отримання даних за допомогою звичайних HTTP-запитів. Для цього часто застосовують інструменти, такі як Selenium або Puppeteer, які імітують роботу браузера. Також варто звертати увагу на адаптивність HTML-розмітки, оскільки багато платформ використовують різні шаблони для десктопних та мобільних пристроїв.

Для кращого розуміння переваг та обмежень розглянутих форматів доцільно навести порівняльну таблицю:

Таблиця 2.1 Переваги та недоліки

Формат	Переваги	Недоліки	Бібліотеки для обробки
TXT	Простота, швидкість обробки, доступність	Відсутність структури, немає розмітки	Вбудовані засоби Python (read, split), re
PDF	Збереження форматування, підтримка графіки	Фрагментація тексту, потреба в парсерах, OCR	PyMuPDF, pdfplumber, Tika, Tesseract
HTML	Чітка структура, легко ідентифікувати зміст	Потрібна очистка від «сміття», складність навігації	BeautifulSoup, lxml, Selenium

Джерело: власна розробка автора

Таким чином, ефективна система генерації тестових питань повинна вміти обробляти текстові файли без складної структури, витягувати змістовні дані з PDF-документів різної складності та аналізувати вміст веб-сторінок, що представляють собою освітні ресурси. Забезпечення багатоформатної підтримки відкриває можливість створення універсальної та гнучкої системи, яка зможе працювати з різними джерелами освітньої інформації без обмежень. Це є важливою умовою масштабованості системи та її інтеграції в реальні навчальні процеси, де джерела знань не завжди представлені у єдиному стандарті.

Генерація тестових питань на основі різноманітних відкритих освітніх матеріалів вимагає глибокого розуміння особливостей форматів, у яких представлені джерела інформації. Кожен формат має свої специфічні характеристики, які впливають на процес вилучення, обробки та аналізу тексту, а отже, і на якість кінцевого тестового контенту. Врахування цих особливостей є запорукою розробки універсальних та ефективних алгоритмів, здатних адаптуватися до різних умов і забезпечувати стабільність роботи системи.

Текстові файли формату .txt є найпростішими з погляду структури і дозволяють швидко та без додаткових інструментів отримувати доступ до основного текстового вмісту. Однак їхня мінімальна структура ускладнює автоматичне виділення смислових блоків, що може призводити до втрати контексту при генерації питань. Для подолання цього обмеження розробляють додаткові методи сегментації тексту, які базуються на лінгвістичних та статистичних ознаках, наприклад, виявлення абзаців, ключових слів, пунктуаційних маркерів.

Робота з PDF-документами є суттєво складнішою через варіативність форматування та способів збереження тексту. Сучасні парсери дають змогу автоматично вилучати текстову інформацію, але нерідко стикаються з проблемами відсутності логічної послідовності, фрагментованістю речень або відсутністю чіткої структури документа. Застосування OCR-технологій для обробки сканованих документів відкриває доступ до великої кількості архівних та недоступних раніше матеріалів, але вимагає врахування похибок розпізнавання і подальшого очищення тексту. Ці особливості зумовлюють необхідність поєднання кількох підходів до вилучення інформації для підвищення повноти та якості даних.

HTML-формат, завдяки своїй структурованості і широкому застосуванню в онлайн-освіті, є одним із ключових джерел для автоматичної генерації навчального контенту. Однак для ефективного вилучення корисної інформації потрібно враховувати складність сучасних веб-сторінок, які часто містять численні додаткові елементи — рекламу, навігаційні панелі, інтерактивні віджети — що не мають освітньої цінності. Розробка фільтрів на основі аналізу DOM-структури, класів CSS та JavaScript дозволяє ізолювати релевантний контент. Крім того, обхід багатосторінкових курсів вимагає реалізації механізмів парсингу гіперпосилань, що забезпечує повноту збору матеріалу.

Врахування особливостей кожного з розглянутих форматів підвищує гнучкість системи генерації тестів, дозволяє інтегрувати різні джерела та

формати, і таким чином забезпечує широкий спектр застосування — від простих конспектів до комплексних онлайн-курсів.

Для систематизації переваг та викликів, пов’язаних з роботою у різних форматах, рекомендується скласти порівняльну таблицю, яка допоможе наочно відобразити основні характеристики, обмеження та рекомендовані інструменти обробки для кожного з форматів. Такий підхід сприяє вибору оптимальної стратегії вилучення тексту та підвищенню ефективності алгоритмів генерації тестових питань.

Загалом, дослідження форматів освітніх матеріалів є фундаментальним кроком для створення надійної та універсальної системи, що здатна забезпечити якісний навчальний контент у різних освітніх середовищах і відповідати сучасним вимогам цифрової освіти.

У процесі роботи з різними форматами освітніх матеріалів особливо важливо враховувати їхню сумісність із технологіями обробки природної мови (NLP) та алгоритмами машинного навчання. Для текстових файлів з мінімальною структурою застосовуються методи попередньої обробки, які включають токенізацію, лемматизацію, виділення частин мови, а також сегментацію на логічні блоки. Відсутність розмітки ускладнює автоматичне виявлення заголовків чи списків, що може знижувати точність і повноту вилученої інформації, тому для покращення результатів іноді застосовують евристичні підходи або моделі класифікації текстових фрагментів.

Для роботи з PDF-файлами, крім базових парсерів, все частіше використовують комбіновані підходи, що поєднують вилучення тексту з аналізом візуального розташування елементів на сторінці. Це дозволяє розпізнавати заголовки, підзаголовки, таблиці та списки, зберігаючи структуру документа. Застосування глибоких нейронних мереж для класифікації сегментів PDF-документів дає змогу підвищити якість структуризації, що напряму впливає на точність формування тестових питань. Крім того, при обробці документів із зображеннями, графіками чи формулами, важливо передбачити

можливості для виділення та інтерпретації таких елементів, адже вони часто містять суттєву інформацію.

Щодо HTML-документів, крім виділення основного текстового контенту, важливо розглядати й інтерактивні елементи, які можуть містити важливі навчальні завдання чи пояснення. Веб-сторінки з динамічним контентом, що завантажується за допомогою JavaScript, вимагають використання інструментів для емуляції браузера (наприклад, Selenium або Puppeteer), що дозволяє отримати повний доступ до змісту. Для забезпечення ефективної навігації по багатосторінкових ресурсах необхідно реалізувати алгоритми обходу посилань, які враховують логіку структури сайту та уникають циклів чи зайвих переходів.

Також важливим є питання стандартизації та уніфікації представлення освітнього контенту. Сучасні освітні платформи часто підтримують формати SCORM або xAPI, які описують структуру курсів, модулів, завдань і оцінок у стандартизованому вигляді. Інтеграція підтримки таких форматів може значно спростити збір та аналіз навчальних матеріалів, а також покращити взаємодію з існуючими системами управління навчанням (LMS).

У контексті подальшої обробки вилученого контенту особливе значення має його очищення від шуму — непотрібних символів, технічних тегів, дублікатів, а також нормалізація тексту. Це створює умови для коректної роботи NLP-інструментів та підвищує точність генерації тестових питань.

Крім того, врахування мовних особливостей джерел інформації — таких як стиль викладу, термінологія, наявність регіональних або галузевих специфічностей — допомагає адаптувати алгоритми генерації для досягнення більш релевантних і точних результатів. Для багатомовних освітніх ресурсів важливо передбачити підтримку різних мов і коректну роботу з їх граматикою та лексикою.

Узагальнюючи, розробка універсальних методів обробки різних форматів відкритих освітніх матеріалів є фундаментальним кроком на шляху створення надійних і гнучких систем автоматичної генерації тестових питань. Такий підхід не лише розширює можливості інтеграції з різноманітними джерелами, а

й підвищує якість, релевантність та ефективність освітнього контенту, що сприяє покращенню результатів навчання і розвитку цифрової освіти загалом.

2.2 Огляд ключових параметрів для генерації релевантних тестових питань

Автоматизована генерація тестових питань на основі текстових джерел є багатоетапним процесом, що інтегрує лінгвістичний аналіз, семантичну інтерпретацію та педагогічні стратегії для формування завдань, здатних ефективно перевіряти рівень розуміння прочитаного. У центрі цього процесу знаходиться якісне вилучення інформаційно насичених фрагментів з тексту, що здійснюється за допомогою технологій обробки природної мови (Natural Language Processing, NLP). Зокрема, велике значення має визначення іменованих сутностей (Named Entity Recognition, NER), що включають персоналії, дати, географічні об’єкти, числові характеристики тощо. Ці елементи виконують роль семантичних вузлів, на основі яких будуються питання, здатні тестувати запам’ятовування фактів, подій та зв’язків між ними [1, 2].

Якість класифікації таких сутностей визначає релевантність сформульованих питань. Наприклад, некоректне розпізнавання імені або дати може призвести до зміщення акцентів у запитанні, що унеможливлює правильну відповідь. Особливо важливо це при створенні завдань, де перевіряється розуміння логіки подій або хронологічної послідовності. Водночас, визначення ключових лексем не є самодостатнім етапом – критичною є також здатність алгоритму враховувати граматичний контекст. Узгодження частин мови, морфологічних характеристик, зокрема роду, числа, відмінка і часу, відіграє вирішальну роль у синтаксично та стилістично коректному формулюванні запитань [3]. Наприклад, речення на зразок «Хто допомагав Олені?» чи «Коли Петро прибув до Києва?» вимагають точного узгодження граматичних форм дієслів та займенників.

Додатковим чинником, що безпосередньо впливає на результат, є якість самого вхідного тексту. Тексти, що характеризуються чіткою структурою, логічною послідовністю, насиченістю дієсловами та тематично релевантною лексикою, створюють передумови для високоточних аналітичних висновків NLP-систем. У таких випадках алгоритм має змогу формулювати не лише фактичні, але й аналітичні та узагальнюючі запитання. Натомість тексти з великою кількістю мовних помилок, надмірною кількістю вводних конструкцій або стилістичних невідповідностей ускладнюють процес морфологічного та синтаксичного розбору, що своєю чергою знижує точність генерації питань [4].

Іншою складовою, що суттєво впливає на якість автоматично згенерованого тесту, є механізм побудови дистракторів – альтернативних, але хибних варіантів відповіді. У межах реалізованого алгоритму передбачено створення тестів з одним правильним варіантом відповіді серед чотирьох можливих. Це потребує не лише точної ідентифікації коректної відповіді, але й генерації трьох варіантів, що є логічно правдоподібними, однак не відповідають дійсності. Для досягнення цієї мети застосовуються спеціалізовані лексичні бази, які містять альтернативні імена, дії або локації, що не входять до тексту, проте є семантично подібними до згаданих у ньому об’єктів [5]. Це дозволяє підвищити когнітивну складність завдання та уникнути ситуацій, коли правильну відповідь можна визначити виключно методом виключення або вгадування.

Важливим аспектом є також аналіз частотності згадування окремих елементів у тексті. Якщо певний персонаж або факт згадується лише один раз, це може вказувати на його виняткову значущість, що робить його привабливою ціллю для формування питань. З іншого боку, інформація, що повторюється, може свідчити про її вторинну важливість, або ж навпаки – про її ключову роль у розвитку подій, залежно від контексту використання. Отже, системи автоматичної генерації мають інтегрувати механізми аналізу тематичної концентрації, що дозволяє краще зрозуміти семантичну структуру тексту [6].

Загалом, ефективність автоматизованого формування тестових питань базується на сукупності параметрів, серед яких провідне місце займають точність лінгвістичного аналізу, відповідність граматичних структур, адекватність дистракторів, а також якість і структурна організація вхідного матеріалу. Такий підхід дозволяє не лише автоматизувати перевірку знань, але й підвищити її об'єктивність і педагогічну цінність, що особливо актуально в умовах цифровізації освітніх процесів та зростаючої ролі адаптивного навчання [7].

Автоматизована генерація тестових питань на основі текстових джерел передбачає інтеграцію не лише класичних лінгвістичних і педагогічних методів, а й сучасних алгоритмів машинного навчання та штучного інтелекту. Сучасні системи здатні застосовувати глибоке семантичне моделювання, використовуючи нейронні мережі та трансформери для кращого розуміння контексту та взаємозв'язків між елементами тексту. Це дозволяє формувати більш складні типи питань, зокрема аналітичні та критичного мислення, які виходять за межі простого відтворення фактів.

Крім визначення іменованих сутностей, у процесі аналізу застосовуються методи тематичного моделювання та кластеризації, що дозволяють виявити ключові теми і підтематики в навчальному матеріалі. Це дає змогу створювати питання, які охоплюють різні рівні деталізації, від загальних понять до вузькоспеціалізованих фактів, сприяючи всебічному оцінюванню знань.

Особливу увагу приділяють врахуванню когнітивних рівнів засвоєння матеріалу відповідно до таксономії Блума. Алгоритми розпізнають, чи належить питання до рівня запам'ятовування, розуміння, застосування, аналізу, синтезу чи оцінювання, що дозволяє формувати тестові блоки з різноманітними типами завдань, підвищуючи дидактичну ефективність.

Процес побудови дистракторів удосконалюється за допомогою семантичного аналізу синонімів, антонімів та концептуальних зв'язків між термінами. Системи використовують словники тематичних полів, мережі

понять та корпуси текстів для генерації правдоподібних, але помилкових варіантів, які вимагають від тестованих глибшого розуміння предмета.

Для підвищення адаптивності тестів застосовують алгоритми, що аналізують історію відповідей користувача, його успішність і типові помилки, з метою формування індивідуальних навчальних траєкторій. Такий підхід забезпечує персоналізацію освітнього процесу і підвищує мотивацію до навчання.

Якість вхідного тексту значною мірою визначає потребу у попередній обробці даних. Техніки очистки включають нормалізацію тексту, видалення стоп-слів, корекцію орфографічних помилок і стискання повторів. Крім того, застосовують семантичне узагальнення, що допомагає зменшити надмірність інформації та підвищити концентрацію ключових концептів.

Врахування стилістичних особливостей тексту, зокрема рівня формальності, жанру та цільової аудиторії, дає змогу коригувати формулювання питань, роблячи їх більш зрозумілими та релевантними. Наприклад, навчальні матеріали для молодших школярів потребують спрощеного викладу, тоді як для студентів вищих курсів – більш складних і точних формулювань.

Важливим аспектом є інтеграція модулів семантичної валідації, які перевіряють логічну послідовність і несуперечність сформованих питань і відповідей. Це мінімізує ризик формулювань, що можуть викликати неоднозначність або плутанину серед тестованих.

Для забезпечення масштабованості процесу використовують паралельні обчислення та розподілені системи, що дають змогу оперативно обробляти великі обсяги навчальних матеріалів, швидко оновлювати бази питань та інтегрувати нові джерела даних.

Крім того, важливою є підтримка різних мовних варіантів і діалектів, що дозволяє створювати багатомовні тестові системи, адаптовані до специфіки різних регіонів та культурних контекстів.

Загалом, багаторівневий підхід до автоматичної генерації тестів, що поєднує сучасні технології NLP, педагогічні методики та адаптивні алгоритми,

створює міцну основу для розвитку інтелектуальних освітніх платформ, які відповідають вимогам цифрової трансформації та індивідуалізації навчання.

Якщо хочеш, можу також підготувати структуровані приклади або детальніший опис конкретних алгоритмів, які використовуються в таких системах.

Важливим напрямом розвитку систем автоматичної генерації тестових питань є інтеграція методів контекстного аналізу, які дозволяють враховувати не лише окремі речення, а й зв’язки між різними частинами тексту. Це підвищує точність формулювання запитань, що перевіряють глибше розуміння матеріалу, а не лише знання окремих фактів.

Крім того, застосовуються технології розпізнавання намірів автора тексту, що дозволяє ідентифікувати ключові меседжі та освітні цілі, закладені у матеріалі. Це дає змогу формувати питання, що краще відповідають концептуальним задачам навчального курсу.

Для підвищення ефективності генерації використовують методи семантичного узагальнення та абстрагування, які дозволяють створювати запитання на основі основних ідей тексту, а не лише конкретних фактів. Такий підхід сприяє розвитку критичного мислення і здатності узагальнювати інформацію.

У сфері генерації дистракторів активно впроваджуються алгоритми на основі моделей природної мови, які можуть створювати нові варіанти відповідей, збагачені синонімами, перефразуваннями та контекстними замінами, що робить завдання більш динамічними та складними для розв’язання.

Враховується також психологічний аспект формулювання питань: правильне формулювання сприяє зменшенню когнітивного навантаження на учнів і знижує рівень тривожності під час тестування, що позитивно впливає на результативність оцінювання.

Технічна реалізація таких систем включає використання API сучасних мовних моделей, що дозволяють швидко інтегрувати інноваційні NLP-інструменти без необхідності глибокої розробки з нуля.

Особливу увагу приділяють безпеці даних і захисту авторських прав на навчальні матеріали, що особливо актуально при роботі з відкритими освітніми ресурсами та публічними базами.

Врахування адаптивних механізмів дозволяє системі коригувати рівень складності питань у реальному часі, базуючись на аналітиці поведінки та результатів попередніх тестувань, що значно підвищує індивідуалізацію навчального процесу.

Значним напрямком є також розробка інтерфейсів користувача, що забезпечують простий і інтуїтивний доступ до генерації, редагування та оцінювання тестів як для викладачів, так і для студентів, сприяючи активній взаємодії з системою.

Віддалене та хмарне розгортання таких систем забезпечує їх доступність і масштабованість, дозволяючи інтегрувати їх у різноманітні навчальні платформи і системи управління навчанням (LMS).

Таким чином, подальший розвиток автоматизованої генерації тестових питань тісно пов’язаний з багатопрофільною інтеграцією технологій, педагогіки та психології, що сприяє створенню інноваційних рішень для сучасної освіти.

Важливим напрямом подальшого вдосконалення систем автоматизованої генерації тестових питань є інтеграція міждисциплінарних підходів, що поєднують методи когнітивної науки, педагогіки та комп’ютерної лінгвістики. Це дозволяє більш глибоко моделювати процес засвоєння знань і формувати завдання, які не лише вимірюють фактологічні знання, але й розвивають навички критичного мислення, аналізу та синтезу інформації.

Значну роль відіграють технології адаптивного навчання, які базуються на побудові профілів учнів і моделюванні їх індивідуальних освітніх траєкторій. Завдяки цьому системи здатні динамічно підлаштовувати складність тестових

завдань, пропонуючи більш складні або спрощені варіанти у залежності від поточного рівня знань та стилю навчання кожного користувача.

Для підвищення якості та достовірності тестів застосовують методи семантичної перевірки узгодженості між питаннями та відповідями, зокрема із залученням векторних представлень тексту (embedding), які дозволяють оцінити семантичну близькість і уникнути логічних помилок. Такі підходи сприяють автоматичному виявленню двозначностей, неповноти або неадекватності сформульованих завдань.

Крім того, системи можуть включати функції автоматичного оновлення бази тестових питань на основі аналізу статистики проходження тестів і зворотного зв’язку від користувачів. Це створює механізм самооптимізації, що забезпечує постійне підвищення якості контенту й адаптацію до змін у навчальних програмах та вимогах.

Особливу увагу приділяють питанням інтеграції інструментів генерації тестів у більш широкі освітні екосистеми, включаючи LMS, платформи для дистанційного навчання та сервіси оцінювання. Така взаємодія дозволяє автоматизувати повний цикл навчання – від подачі матеріалу до оцінювання і персональних рекомендацій.

Технічні аспекти розвитку включають використання масштабованих хмарних архітектур і мікросервісів, що підвищують надійність, продуктивність і гнучкість систем. Застосування технологій контейнеризації та оркестрації сприяє ефективному розгортанню і підтримці оновлень без простоїв.

У сфері NLP постійно зростає роль мультимодальних моделей, які аналізують не лише текст, а й зображення, аудіо або відеоконтент. Впровадження таких моделей у процес генерації тестових завдань відкриває нові можливості для створення інтерактивних і мультимедійних тестів, що відповідають сучасним вимогам інклюзивної та різнорівневої освіти.

Соціокультурний аспект також набуває ваги: розробка систем повинна враховувати мовні, культурні та регіональні особливості користувачів, що

знижує ризик неправильного тлумачення питань і підвищує комфорт використання.

З огляду на швидкі темпи розвитку штучного інтелекту, перспективним є застосування методів explainable AI (пояснювального штучного інтелекту), що дозволяють не лише генерувати запитання, але й надавати обґрунтування їх формулювань, підвищуючи довіру користувачів і сприяючи кращому розумінню навчального матеріалу.

Таким чином, перспективи розвитку автоматизованої генерації тестових питань полягають у комплексній інтеграції педагогічних, лінгвістичних, технічних і соціокультурних складових, що разом створюють ефективний інструмент для сучасної освіти, спрямований на підтримку індивідуального навчання, підвищення якості оцінювання і забезпечення доступності знань.

2.3 Проектування алгоритму для вилучення основної інформації та створення питань

Генерація тестових питань на основі текстових джерел є складним міждисциплінарним завданням, що об’єднує методи комп’ютерної лінгвістики, штучного інтелекту та педагогічної психології. Якість автоматично створених завдань безпосередньо залежить від низки лінгвістичних та семантичних параметрів, кожен з яких відіграє ключову роль у забезпеченні коректності, релевантності та дидактичної цінності сформованого тестового матеріалу. Насамперед, особливу увагу слід приділяти точності вилучення інформаційно значущих компонентів із тексту, адже саме вони становлять основу для формування змістових запитань.

У процесі інформаційного видобування застосовуються інструменти обробки природної мови, зокрема механізми ідентифікації іменованих сутностей (Named Entity Recognition, NER), що дозволяють автоматично виявляти ключові елементи, як-от особові імена, дати, географічні об’єкти, числові параметри та інші категорії фактологічної інформації. Ці сутності слугують опорною базою для побудови запитань, які вимагають точного відтворення фактів із прочитаного. Водночас ефективність цього етапу

залежить від точності класифікації об’єктів та їхнього контекстного трактування в межах конкретного речення або абзацу [8].

Не менш критичним є урахування граматичного контексту при формуванні тестових одиниць. Будь-яке автоматизоване перетворення твердження на питання повинно зберігати граматичну узгодженість: рід, число, відмінок і дієслівні форми мають бути коректно адаптовані відповідно до синтаксичної структури фрази. Наприклад, при побудові питання на основі дії суб’єкта («Петро пішов до школи»), система повинна враховувати морфологічні особливості іменника та дієслова, щоб сформулювати запитання «Хто пішов до школи?» або «Куди пішов Петро?». Відсутність такого граматичного узгодження може призвести до порушення смислової цілісності тестового запитання, а відтак і до втрати його педагогічної ефективності [9].

Суттєвий вплив на якість генерованих завдань має характеристика джерельного тексту. Висока інформативність, логічна послідовність викладу, чітка структура речень і наявність дієслівних конструкцій створюють сприятливі умови для обробки. Навпаки, тексти з граматичними помилками, фрагментарною побудовою або надлишком складнопідрядних речень можуть негативно впливати на функціонування NLP-алгоритмів, знижуючи точність і релевантність результатів [10].

Ключову роль у структурі тестового завдання відіграє тип питання, яке формується. У рамках реалізованого підходу перевага надається запитанням із вибором правильної відповіді з чотирьох варіантів. Це потребує не лише коректного визначення єдиної правильної відповіді, а й формування трьох правдоподібних, але помилкових дистракторів. Їхнє створення базується на застосуванні словників схожих понять, списків імен або географічних назв, що не зустрічаються в аналізованому фрагменті, але є наближеними за семантикою та частотою вживання. Такий підхід дозволяє зберігати баланс між складністю завдання та його змістовною достовірністю [11].

Також важливим чинником є частотність появи певних лексичних одиниць. Якщо ім’я персонажа, локація або подія згадуються в тексті лише раз,

це може свідчити про їхню важливість у загальній структурі наративу, що робить їх перспективною основою для побудови тестових запитань. Повторювані елементи, навпаки, часто виконують фонову функцію і не містять достатньої когнітивної новизни для перевірки рівня розуміння [12].

Інтегрально, ефективна генерація тестових питань вимагає урахування широкого спектра параметрів, що охоплюють як лінгвістичні, так і семантико-контекстуальні аспекти. Взаємодія таких чинників, як точне визначення іменованих сутностей, морфологічна коректність побудови питань, релевантність запропонованих варіантів відповідей, а також освітня значущість отриманих результатів, формує підґрунтя для реалізації сучасних автоматизованих систем тестування. Такі системи забезпечують не лише об’єктивну оцінку рівня засвоєння навчального матеріалу, а й підвищують ефективність навчального процесу завдяки інтерактивності, масштабованості та когнітивній адаптивності [13].

Генерація тестових питань на основі текстових джерел має бути побудована на комплексній інтеграції методів штучного інтелекту, що включають машинне навчання, глибинне навчання та лінгвістичний аналіз, що дозволяє підвищити адаптивність та точність формованих запитань. Сучасні моделі, зокрема трансформери, здатні не лише виявляти іменовані сутності, але й розуміти контекст, що забезпечує більш точну генерацію питань із врахуванням синтаксичних та семантичних нюансів тексту.

Важливим аспектом є використання алгоритмів кластеризації та тематичного моделювання для визначення ключових тем у навчальному матеріалі, що дає змогу створювати тематично спрямовані блоки питань, які послідовно формують систему перевірки знань. Це сприяє глибшому розумінню предмету та покращує когнітивний ефект від проходження тесту.

Ще одним напрямом є автоматичне формування дистракторів — помилкових варіантів відповідей, які мають бути не лише правдоподібними, а й логічно обґрунтованими, що підвищує якість оцінювання та запобігає випадковому вгадуванню. Для цього використовуються семантичні мережі,

словники синонімів і антонімів, а також бази знань із відповідної предметної області.

Крім того, важливо враховувати адаптивність тестів, яка полягає у зміні складності питань відповідно до рівня знань користувача, що досягається за допомогою алгоритмів адаптивного тестування та аналізу відповідей у реальному часі. Такий підхід дозволяє підвищити мотивацію та ефективність навчального процесу.

Особливу увагу слід приділяти інтерфейсу користувача, який має бути інтуїтивно зрозумілим та підтримувати різні формати питань (відкриті, множинний вибір, встановлення відповідностей), що розширює можливості використання системи у різних освітніх контекстах. Важливо забезпечити зворотний зв’язок, який надає роз’яснення щодо правильних і помилкових відповідей, що сприяє закріпленню знань.

Інтеграція таких систем з платформами дистанційного навчання дозволяє автоматизувати процес контролю знань, зменшуючи навантаження на викладачів і створюючи умови для масштабування освітніх програм. Врахування стандартів відкритих освітніх ресурсів забезпечує сумісність і обмін навчальними матеріалами між різними системами.

Узагальнюючи, ефективна генерація тестових питань ґрунтується на поєднанні точного лінгвістичного аналізу, семантичного розуміння контексту, адаптивних методик оцінювання та зручності використання системи, що разом створює інструмент для якісного та персоналізованого контролю знань у навчальному процесі.

В першу чергу, слід розглянути питання оцінювання якості автоматично згенерованих тестових завдань. Для цього застосовують різноманітні метрики, які охоплюють не лише лінгвістичну коректність, а й дидактичну ефективність. До таких метрик належать точність і повнота вилучення інформації, ступінь релевантності сформованих питань до навчального матеріалу, а також когнітивна складність завдань. Визначення оптимального рівня складності

допомагає уникнути як надмірної простоти, так і зайвої складності, що може демотивувати учнів.

Важливою складовою є використання технологій семантичного аналізу тексту, таких як векторне представлення слів (word embeddings) і контекстуальні моделі типу BERT або GPT. Вони дозволяють захопити глибші смислові зв’язки між словами та фразами, що підвищує якість формування питань, особливо в частині уникнення неоднозначностей і подвійних трактувань.

Крім цього, суттєву роль відіграє механізм обробки складних синтаксичних конструкцій, зокрема, розпізнавання підрядних речень, узгодження означень та виявлення анафоричних зв’язків, що дозволяє формувати питання, які охоплюють більш складні інформаційні одиниці та краще відображають зміст навчального тексту.

Важливо враховувати також контекстуальні особливості різних типів текстів: наукових, художніх, технічних чи публіцистичних. Кожен тип має свої характерні лінгвістичні й стилістичні риси, що впливають на підхід до генерації питань. Наприклад, наукові тексти вимагають точного відтворення фактів і термінології, тоді як художні — уваги до сюжетних деталей і персонажів.

Для підвищення якості генерації доцільно застосовувати методи активного навчання, коли система може отримувати зворотній зв’язок від користувача або експерта і відповідно коригувати алгоритми вилучення та формування запитань. Це сприяє поступовому покращенню якості та адаптації під конкретні навчальні курси.

Не менш важливим є забезпечення різноманітності форм питань, що включає відкриті питання з короткою відповіддю, завдання на встановлення відповідностей, впорядкування елементів, а також питання з множинним вибором. Це дозволяє комплексно перевіряти різні рівні розуміння та аналітичні навички учнів.

З технічної точки зору, реалізація системи генерації тестів повинна бути оптимізованою для швидкої обробки великих обсягів тексту, підтримувати масштабованість і можливість інтеграції з іншими освітніми системами.

Застосування модульної архітектури дає змогу розвивати та покращувати окремі компоненти без значного впливу на всю систему.

Крім того, варто звернути увагу на етичні та правові аспекти використання автоматизованих систем оцінювання, зокрема забезпечення конфіденційності даних користувачів, прозорості алгоритмів прийняття рішень і запобігання упередженості у формуванні питань.

В цілому, успішна генерація тестових питань на основі текстових джерел є результатом гармонійного поєднання сучасних методів обробки природної мови, педагогічної експертизи та інноваційних технологій штучного інтелекту, що створює ефективний інструмент для підтримки навчального процесу в умовах цифрової освіти.

2.4 Оцінка якості згенерованих питань на основі навчальних матеріалів

Після завершення реалізації алгоритмічного модуля генерації тестових питань одним із ключових етапів є верифікація створеного контенту, що має на меті перевірку його педагогічної цінності, відповідності навчальним цілям і технічної коректності. Надійність функціонування інтелектуальної системи прямо залежить від того, наскільки згенеровані запитання відповідають критеріям інформативності, змістової адекватності, граматичної точності та релевантності до навчального матеріалу.

Процес оцінювання має бути комплексним, із застосуванням як якісних, так і кількісних методів аналізу. Насамперед проводиться експертна перевірка сформованих запитань, яка є традиційним і водночас критично важливим методом контролю. До аналізу залучаються викладачі або фахівці у відповідній предметній галузі, які оцінюють коректність побудови питань, відповідність сформульованих варіантів відповіді навчальному змісту, відсутність лексичних або логічних суперечностей. Такий підхід дозволяє виявити ті недоліки, які не виявляються автоматизованими засобами, наприклад, двозначні формулювання, невдало підібрані дистрактори або мовностилістичні неточності [14].

Паралельно з експертною оцінкою впроваджуються автоматизовані методи валідації, що забезпечують об’єктивність і масштабованість оцінювання. Серед найбільш поширених метрик вирізняється BLEU (Bilingual Evaluation Understudy) – метод порівняння згенерованих питань із контрольними (референсними), що використовується в задачах оцінювання якості природномовної генерації. Крім цього, використовуються такі параметри, як лексична різноманітність, показники унікальності запитань, середня довжина запитань, співвідношення між кількістю іменованих сутностей та загальним обсягом тексту [15].

Важливим доповненням до оцінки є функціональне тестування системи на прикладах реального освітнього контенту. У межах такого підходу система перевіряється на корпусах текстів різної тематики та складності – підручниках, науково-популярних статтях, навчальних презентаціях тощо. Результати тестування порівнюються за низкою формальних ознак: частка змістовно правильних питань, питома кількість питань із синтаксичними або логічними помилками, частота повторів, наявність ключових слів, що співвідносяться з темою джерела [16].

Особливе місце в структурі контролю займає збір зворотного зв’язку від кінцевих користувачів системи – викладачів, студентів або методистів. З метою практичного оцінювання якості, учасникам пропонується пройти коротке тестування, сформоване автоматичною системою, після чого вони здійснюють оцінювання питань за критеріями: зрозумілість формулювань, відповідність темі, обґрунтованість варіантів відповідей, логічність побудови. Ці відгуки аналізуються і використовуються для подальшої оптимізації алгоритмів генерації, виявлення повторюваних помилок і адаптації під специфіку цільової аудиторії [17].

Отже, якісна оцінка результатів генерації тестових завдань повинна базуватися на комбінованому підході, що поєднує традиційний експертний аналіз, об’єктивні статистичні метрики та суб’єктивні відгуки користувачів. Така багатовимірна оцінка дозволяє не лише підтвердити ефективність

розробленої системи, але й створити механізм її постійного вдосконалення відповідно до актуальних освітніх потреб.

Генерація тестових питань є лише початковим етапом, і для забезпечення високої якості автоматично створеного контенту необхідне впровадження систематизованої та всебічної процедури верифікації, яка охоплює як педагогічні, так і технічні аспекти. Ця процедура спрямована на виявлення та усунення потенційних помилок, підвищення точності й адекватності сформованих питань, а також на забезпечення їх відповідності навчальним цілям і стандартам. Верифікація є критичною для створення надійної інтелектуальної системи, здатної ефективно підтримувати процес навчання.

Застосування автоматизованих методів оцінювання дозволяє досягти масштабованості та оперативності аналізу великої кількості тестових питань. Поряд із метрикою BLEU, яка вимірює ступінь відповідності згенерованих питань еталонним зразкам, широко використовують показники когерентності тексту, семантичної близькості, а також специфічні NLP-метрики, які враховують морфологічні та синтаксичні особливості мовного матеріалу. Впровадження таких метрик забезпечує багатогранну оцінку якості, дозволяючи виявляти не лише поверхневі, а й глибинні недоліки тексту.

Під час функціонального тестування системи особлива увага приділяється різноманітності навчальних джерел, що включає тексти різної складності, жанру та тематики, що імітує реальні умови використання. Аналіз результатів дає змогу виявити закономірності у виникненні помилок, оцінити адаптивність алгоритмів до різних контекстів і сформувати рекомендації для подальшого удосконалення.

Збір та аналіз зворотного зв’язку від користувачів є не менш важливою складовою циклу вдосконалення системи. Методики включають як кількісні опитування, так і якісні інтерв’ю або фокус-групи, що дозволяє глибше зрозуміти користувацькі потреби, виявити приховані проблеми та врахувати специфіку сприйняття матеріалу різними категоріями учасників освітнього

процесу. Врахування цих даних робить систему більш гнучкою та орієнтованою на практичне застосування.

Крім педагогічних аспектів, важливо забезпечити технічну стабільність системи: відсутність збоїв, правильну обробку виняткових ситуацій, логування процесів генерації та оцінки. Це дає змогу проводити аудит роботи системи, відстежувати її продуктивність і вчасно реагувати на потенційні проблеми.

Для подальшого розвитку інтелектуальних систем генерації тестів доцільно впроваджувати механізми машинного навчання з підкріпленням (reinforcement learning), де система на основі отриманих оцінок і зворотного зв’язку сама коригує свої алгоритми, оптимізуючи якість і релевантність питань. Такі підходи значно підвищують автономність та адаптивність системи.

Таким чином, поєднання експертної оцінки, об’єктивних метрик, функціонального тестування та активної взаємодії з кінцевими користувачами створює ефективний комплексний підхід до верифікації, що гарантує високу якість і педагогічну цінність автоматично згенерованих тестових завдань. Цей підхід забезпечує не лише точність оцінювання знань, а й постійне вдосконалення системи відповідно до сучасних освітніх вимог і технологічних стандартів.

Верифікація автоматично згенерованих тестових питань є невід’ємною частиною життєвого циклу системи, що забезпечує не лише відповідність контенту вимогам навчальної програми, а й адаптацію до різноманітних освітніх контекстів та рівнів підготовки учнів. Застосування багаторівневого підходу до оцінювання дозволяє виявляти як загальні, так і специфічні недоліки, які можуть мати різний характер — від лінгвістичних помилок до проблем з педагогічною доцільністю.

Важливим аспектом є аналіз когнітивної складності питань, що дозволяє визначити, наскільки сформовані запитання відповідають певному рівню знань і навичок відповідно до таксономії Блума чи інших педагогічних моделей. Такий аналіз допомагає уникнути ситуацій, коли питання є надто простими або, навпаки, надмірно складними для цільової аудиторії.

Для підвищення об’єктивності оцінки часто застосовують автоматичні системи тестування, які збирають статистику про проходження тестів, аналізують розподіл відповідей і виявляють питання, що викликають найбільші труднощі або нерозуміння серед користувачів. Такі дані є цінним джерелом для коригування алгоритмів генерації та вдосконалення якості контенту.

Додатково до традиційних метрик, верифікація включає перевірку на відсутність плагіату або дублювання питань із існуючих баз даних, що забезпечує унікальність та оригінальність тестового матеріалу. Це важливо для підтримки академічної доброчесності та запобігання рутинному повторенню завдань.

У сучасних системах дедалі більшого значення набуває інтеграція з адаптивними навчальними платформами, що дозволяє динамічно змінювати склад тестів відповідно до прогресу і потреб кожного користувача. Верифікація в цьому контексті також передбачає оцінку гнучкості та коректності роботи адаптивних алгоритмів.

Ще одним напрямом розвитку є застосування методів обробки природної мови для автоматичного виявлення потенційних двозначностей і неоднозначних формулювань у питаннях, що дозволяє покращувати читабельність та зменшувати ризик помилкових трактувань. Інтеграція таких NLP-модулів у процес верифікації підвищує загальну якість тестових завдань.

Також перспективним є використання аналітики поведінки користувачів під час проходження тестів, зокрема часу відповіді на кожне питання, частоти перегляду підказок або повернення до попередніх питань. Аналіз цих даних допомагає виявити складні або незрозумілі елементи тесту, що в подальшому коригується.

Завдяки постійному оновленню та тренуванню моделей генерації на основі зібраних даних про якість і ефективність тестів, системи верифікації сприяють безперервному розвитку і підвищенню стандартів автоматизованого оцінювання.

Таким чином, комплексна верифікація тестових питань — це багатогранний процес, який охоплює педагогічні, лінгвістичні, технічні та аналітичні аспекти, спрямований на створення максимально якісного, релевантного та ефективного навчального контенту. Це забезпечує надійну підтримку освітнього процесу, підвищує довіру користувачів до системи і стимулює активне залучення у навчання.

Процес верифікації автоматично згенерованих тестових питань є багатогранним та системним, що забезпечує високу якість і педагогічну цінність навчального контенту. Крім традиційних експертних оцінок і застосування метрик якості, сучасні системи використовують комплексний підхід, який охоплює різні рівні аналізу — від лінгвістичного до когнітивного. Врахування педагогічних моделей, таких як таксономія Блума, дозволяє класифікувати питання за рівнями складності та типами мисленнєвої діяльності, що сприяє формуванню збалансованого тестового банку.

Автоматизовані інструменти, що аналізують семантичну релевантність і когерентність тексту, дають змогу виявляти неочевидні мовні помилки, неоднозначності та логічні суперечності, які можуть негативно впливати на розуміння завдань. Інтеграція NLP-модулів у процес верифікації значно підвищує ефективність контролю, зменшуючи навантаження на експертів та забезпечуючи оперативний аналіз великих обсягів даних.

Важливою складовою є функціональне тестування з реальними користувачами, що дозволяє збирати зворотний зв’язок та статистичні дані про поведінку учнів і викладачів. Аналіз часу відповіді, кількості повторів, розподілу оцінок і складності питань допомагає ідентифікувати проблемні зони, адаптувати алгоритми генерації та створювати більш інтуїтивні та ефективні тести. Застосування методів машинного навчання з підкріпленням відкриває можливості для самоадаптації системи на основі зібраних даних, що забезпечує поступове підвищення якості та відповідності тестового контенту освітнім потребам.

Технічна надійність системи, зокрема логування, обробка винятків, моніторинг продуктивності, є критичною для стабільної роботи в реальних умовах. Забезпечення унікальності питань через перевірку на плагіат і дублікати підтримує академічну доброчесність, а інтеграція з адаптивними платформами сприяє персоналізації навчання, що підвищує мотивацію та результативність.

У підсумку, ефективна верифікація — це безперервний, багатокомпонентний процес, який гармонійно поєднує педагогічні, лінгвістичні, технічні та аналітичні аспекти. Він гарантує не лише відповідність тестових завдань освітнім стандартам, але й забезпечує адаптивність, масштабованість та довготривалу ефективність автоматизованих систем оцінювання, що є ключовими факторами для сучасної цифрової освіти.

Верифікація також включає перевірку технічної коректності формату і структури тестових завдань, що є необхідним для їх безперебійного імпорту в різні освітні платформи та системи управління навчанням. Це передбачає контроль за дотриманням стандартів розмітки, кодування символів, а також коректність побудови XML або JSON файлів, які використовуються для обміну навчальним контентом. Невідповідності на цьому рівні можуть призводити до помилок при завантаженні тестів, втрати даних або некоректного відображення питань.

Крім того, важливо оцінювати баланс між різними типами питань у тестах, щоб забезпечити всебічну перевірку знань. Надмірна концентрація питань одного виду, наприклад, з вибором однієї правильної відповіді, може знижувати загальну інформативність оцінювання. Верифікація має враховувати різноманітність формулювань, включаючи відкриті питання, завдання на встановлення відповідностей, заповнення пропусків тощо, що підвищує дидактичну цінність тестів.

Особливої уваги заслуговує перевірка адаптивності питань до різних рівнів підготовки учнів. В ідеалі система повинна мати можливість класифікувати запитання за ступенем складності і пропонувати відповідний

набір завдань для кожного користувача. Це вимагає впровадження моделей, які враховують не лише формальні ознаки тексту, а й психологічні аспекти сприйняття інформації.

Додатковим напрямом є оцінка культурної та етичної відповідності питань. Автоматичні алгоритми мають уникати формулювань, які можуть бути образливими, стереотипними або неетичними. Для цього використовуються спеціалізовані словники, фільтри та експертні перевірки, що забезпечують дотримання стандартів інклюзивності та поваги до різноманітності.

Застосування технік аналізу настроїв і тональності тексту дозволяє виявляти потенційно неоднозначні або провокативні формулювання, що можуть впливати на мотивацію та емоційний стан учнів. Врахування цих аспектів у процесі верифікації сприяє створенню позитивного та безпечного навчального середовища.

Нарешті, для підвищення прозорості та довіри до системи доцільно реалізувати функції генерації детальних звітів про результати верифікації. Ці звіти мають включати інформацію про виявлені помилки, статистичні показники якості, рекомендації щодо покращень і історію змін у базі тестових питань. Такий підхід підтримує ефективне управління контентом і сприяє постійному вдосконаленню системи.

Таким чином, комплексна верифікація автоматично згенерованих тестових матеріалів не обмежується лише перевіркою змісту, а охоплює широкий спектр технічних, дидактичних, психологічних та етичних критеріїв, що забезпечує максимальну якість і ефективність навчального процесу.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМИ ДЛЯ ГЕНЕРАЦІЇ ТЕСТОВИХ ПИТАНЬ

3.1 Реалізація програми для завантаження освітніх матеріалів різних форматів

Початковий етап роботи будь-якої системи автоматичного створення тестів полягає в отриманні вхідного текстового матеріалу. У розробленій програмі, реалізованій мовою Python із використанням бібліотеки PyQt5, процес завантаження навчального матеріалу реалізується у вигляді зручного графічного інтерфейсу з великим текстовим полем, у яке користувач може вставити або ввести потрібний фрагмент тексту вручну.

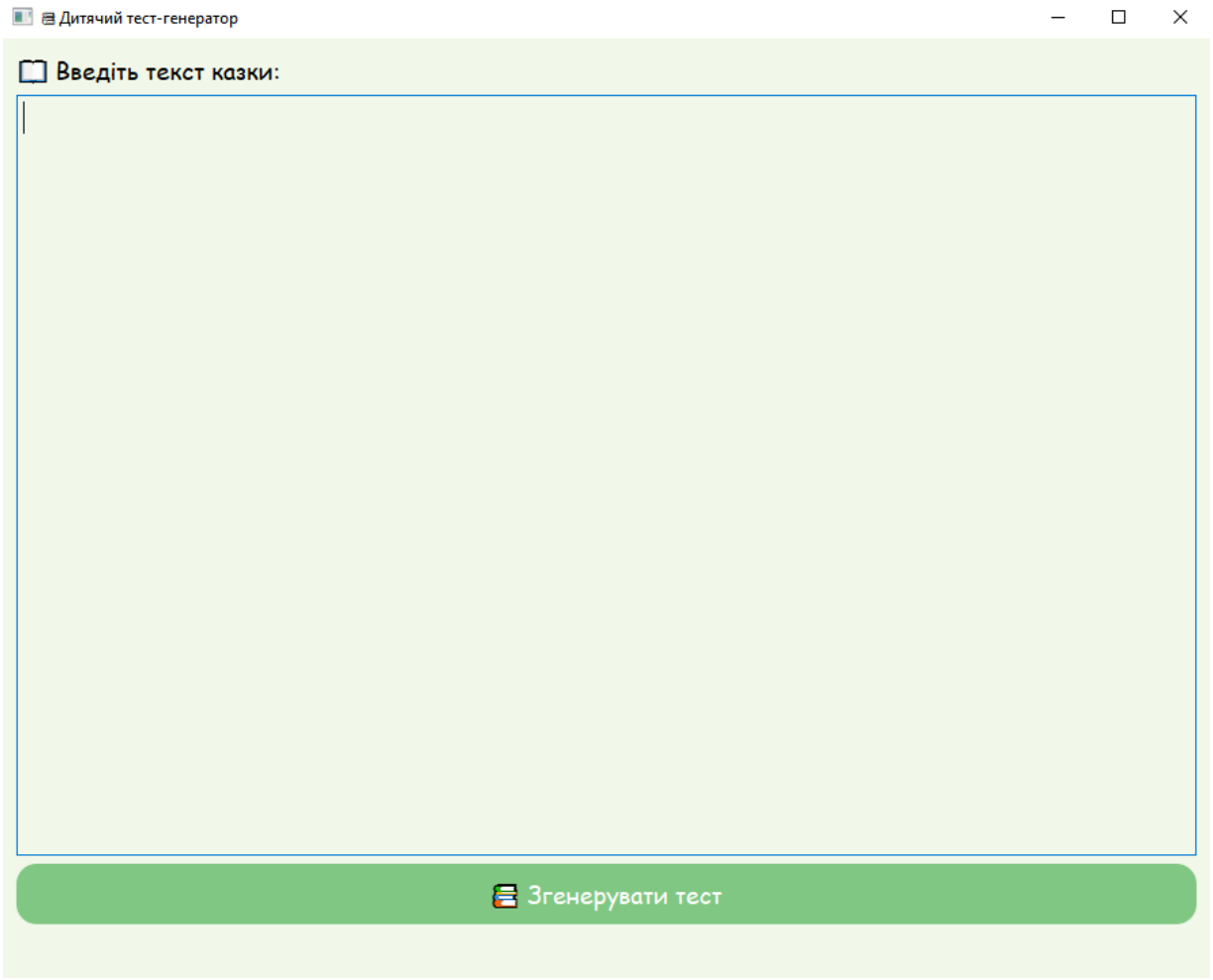

Рисунок 3.1 – Титульна сторінка програми

Джерело: власна розробка автора

Програма орієнтована на обробку освітніх текстів українською мовою, зокрема казок, фрагментів з підручників, методичних посібників тощо. Поточна версія підтримує введення лише у форматі простого тексту (plain text). Для цього було створено графічне вікно, яке містить багаторядкове текстове поле (QTextEdit), яке займає центральну частину інтерфейсу. Інтерфейс оформлений у світло-зелених тонах, що є візуально приємним і не перевантажує зір користувача, особливо при тривалому використанні.

З технічної точки зору, текстове поле є контейнером, у якому зберігається вихідний текст для подальшої обробки за допомогою алгоритмів обробки природної мови. Саме цей текст надалі розбирається за допомогою NLP-бібліотек для виділення ключових сутностей, які будуть основою для автоматичного створення тестових запитань.

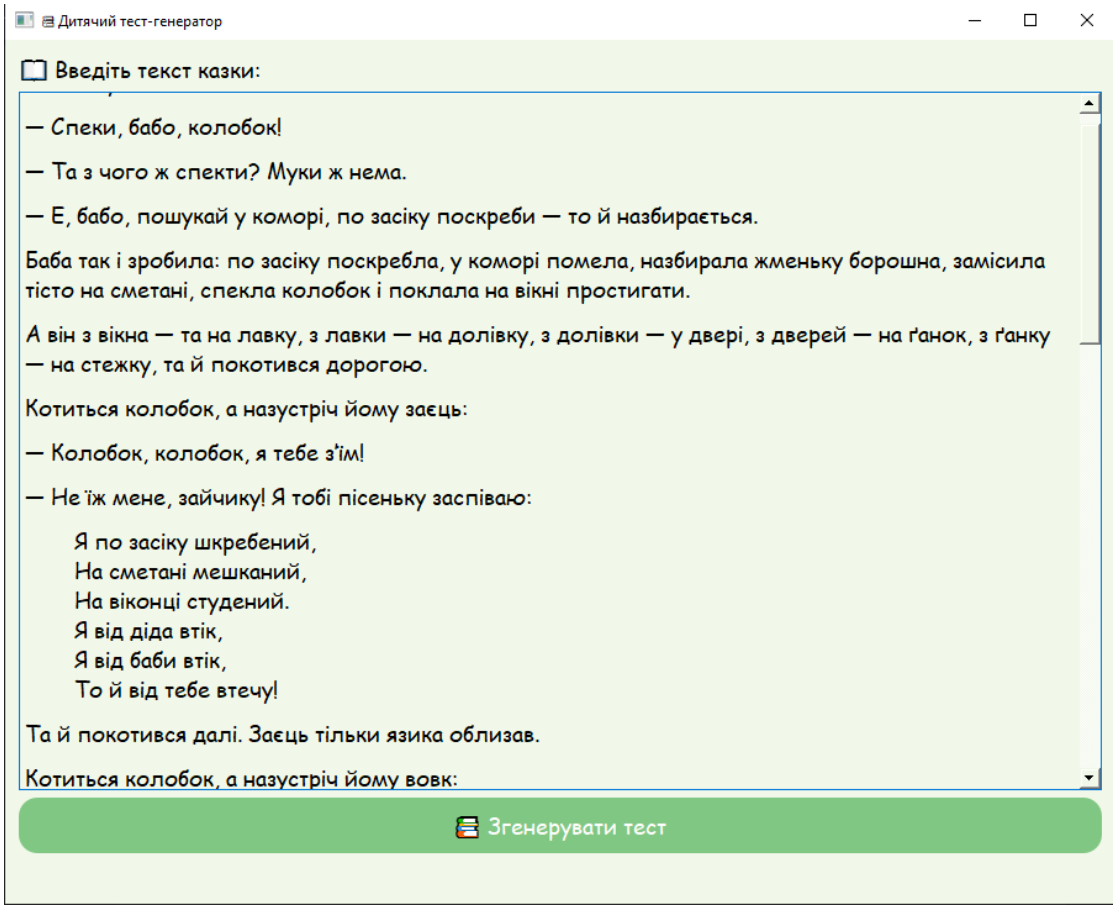

Рисунок 3.2 – Головний екран програми з вставленим текстом для генерації тестових завдань

Джерело: власна розробка автора

Після введення тексту користувач має змогу запустити процес генерації тесту

за допомогою кнопки "Згенерувати тест", яка розташована нижче текстового поля. Ця кнопка стилізована у вигляді зеленої панелі з іконкою та написом, що чітко вказує на її функціональне призначення. Натискання на кнопку передає вміст текстового поля в основний обробник, який викликає модулі генерації тестових запитань.

У рамках зручності використання програми було враховано, що багато користувачів не мають глибоких технічних знань, тому весь інтерфейс виконано у простому, інтуїтивному стилі з українськими написами, що робить програму доступною для школярів, студентів, учителів і викладачів.

Крім того, у текстове поле можна вставити вміст, попередньо скопійований з будь-якого джерела – наприклад, з PDF-файлу, веб-сторінки або текстового редактора. Це забезпечує мінімальну залежність від формату джерела, роблячи програму універсальним інструментом на першому етапі обробки навчального контенту.

Таким чином, перший крок взаємодії користувача з системою – це введення або вставка освітнього тексту, після чого активується кнопка запуску аналізу. Цей модуль є ключовою відправною точкою всього ланцюга генерації тестів, оскільки саме від якості та повноти введеного тексту залежатиме зміст згенерованих питань.

3.2 Впровадження алгоритмів NLP для вилучення ключових термінів та інформації

Серцем системи автоматичного створення тестових завдань є механізм обробки природної мови (Natural Language Processing, NLP), який дозволяє аналізувати вхідний освітній текст, виділяти з нього змістовні одиниці та перетворювати їх у тестові запитання.

Для реалізації NLP-аналізу в розробленій програмі використано бібліотеку spaCy з українською мовною моделлю uk_core_news_sm. Ця модель була спеціально розроблена для обробки українського тексту, що є критично важливим для програми, орієнтованої на локалізоване навчальне середовище. Встановлення моделі виконується за допомогою стандартної команди python -m spacy download uk_core_news_sm, після чого вона завантажується у коді викликом spacy.load().

```
import spacy
nlp = spacy.load("uk_core_news_sm")
doc = nlp(text)  # text – введений користувачем текст
for ent in doc.ents:
    print(ent.text, ent.label_)  # Вивід знайдених сутностей
```

Основними задачами NLP-модуля є:

Токенізація (поділ тексту на слова).

Розпізнавання іменованих сутностей (Named Entity Recognition, NER) – таких як імена, дати, географічні назви, об’єкти тощо.

Частиномовний аналіз (POS-tagging) – визначення граматичних категорій слів.

Виділення ключових слів і фраз, що мають високу інформативність.

Ці етапи дозволяють програмі виявити, які саме частини тексту можуть стати основою для запитання. Наприклад, якщо в тексті згадується персонаж «Іванко», алгоритм визначає це як іменовану сутність типу PER (персона), і створює запитання: «Хто був головним героєм казки?». Для цього аналізується синтаксичне оточення сутності, що дозволяє формувати запитання з правильним формулюванням і граматикою.

Після аналізу ключових сутностей система автоматично генерує як питання, так і варіанти відповідей, з яких одна буде правильною, а інші – неправдивими, але граматично коректними (дистрактори). Генерація неправильних варіантів здійснюється за допомогою списку альтернативних

сутностей того ж типу (наприклад, інших імен, міст або предметів), що підвищує якість тесту.

Окремим етапом є визначення типу запитання. Наприклад:

Якщо в тексті є логічна двозначність або твердження – формується питання типу "Так/Ні".

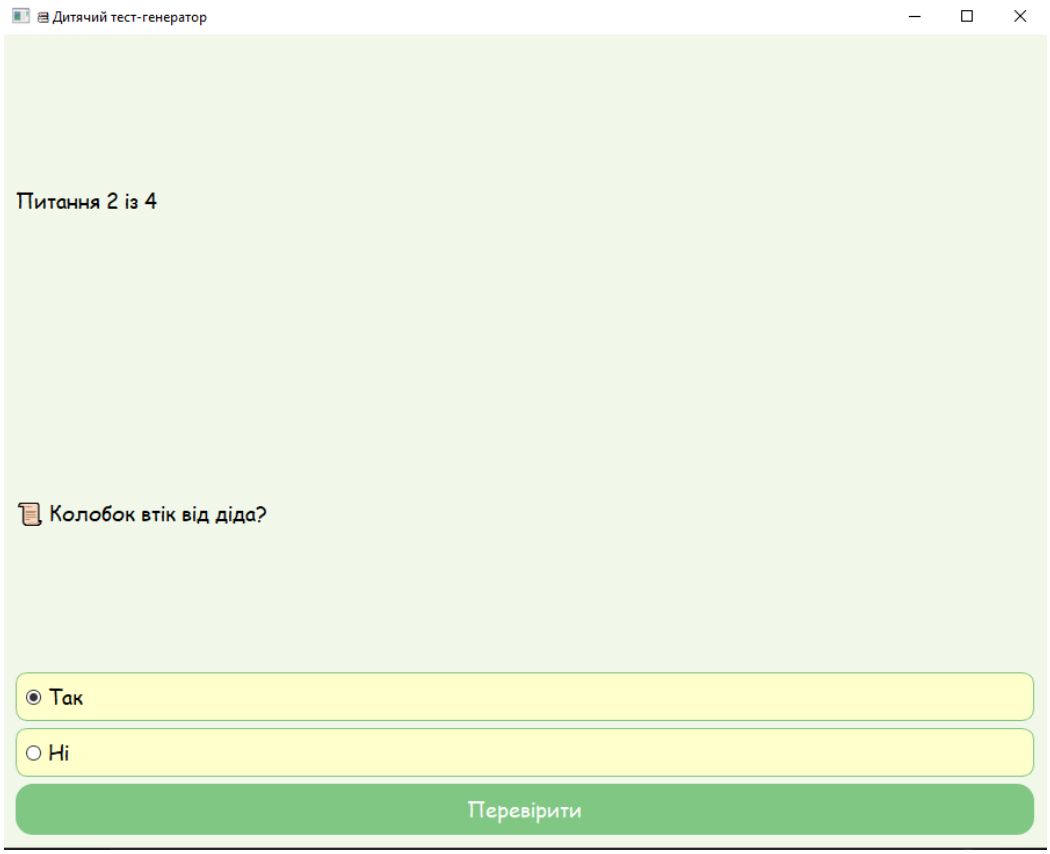

Рисунок 3.3 – Приклад питання типу Так або Ні

Джерело: власна розробка автора

Якщо текст містить перелік об’єктів або дій – застосовується множинний вибір.

Якщо присутні пов’язані пари (наприклад, імена персонажів та їх дії) – генерується завдання на відповідність.

Ці типи автоматично класифікуються залежно від синтаксичних та семантичних зв’язків у тексті.

Процес роботи модуля NLP повністю автоматизований та активується натисканням кнопки "Згенерувати тест" після введення тексту. У результаті –

користувач одразу бачить сформоване питання з варіантами відповідей, які він може обрати для проходження тесту.

Таким чином, NLP-модуль виступає аналітичним ядром програми, забезпечуючи семантичне розуміння тексту, генерацію запитань високої якості та підтримку різних типів завдань на основі реального навчального контенту.

3.3 Автоматичне формування тестових питань (з вибором відповідей, відкритого типу)

Після обробки освітнього тексту модулем NLP наступним логічним етапом є автоматичне формування тестових запитань. Цей процес реалізовано у програмі у вигляді спеціального алгоритму, який на основі виділених сутностей та граматичних структур формує різні типи тестових завдань: із вибором однієї або кількох правильних відповідей, завдання на відповідність, питання відкритого типу, а також прості твердження у форматі «Так / Ні».

Розроблений алгоритм враховує:

- тип виявленої інформації (імена, дати, місця, події),
- контекст згадки в реченні
- можливість створення дистракторів – альтернативних, але логічно правдоподібних відповідей.

У структурі програми є спеціальна функція, яка на основі переданої сутності та її типу визначає, яке саме питання буде створено, а також яким чином буде згенеровано варіанти відповідей. Якщо сутність належить до типу PER (персона), то питання може бути: «Хто є головним героєм?», і система автоматично підставить правильну відповідь разом із трьома дистракторами. Якщо ж визначено тип LOC (місце), тоді формується питання: «Яке місце згадується у тексті?».

Формат запитань у програмі змінюється в залежності від типу аналізованого фрагмента. Наприклад, якщо аналізується перелік подій або ознак, то створюється завдання з множинним вибором.

Якщо в тексті виявлені пари пов’язаних понять, наприклад персонаж та його дія – формується завдання на відповідність.

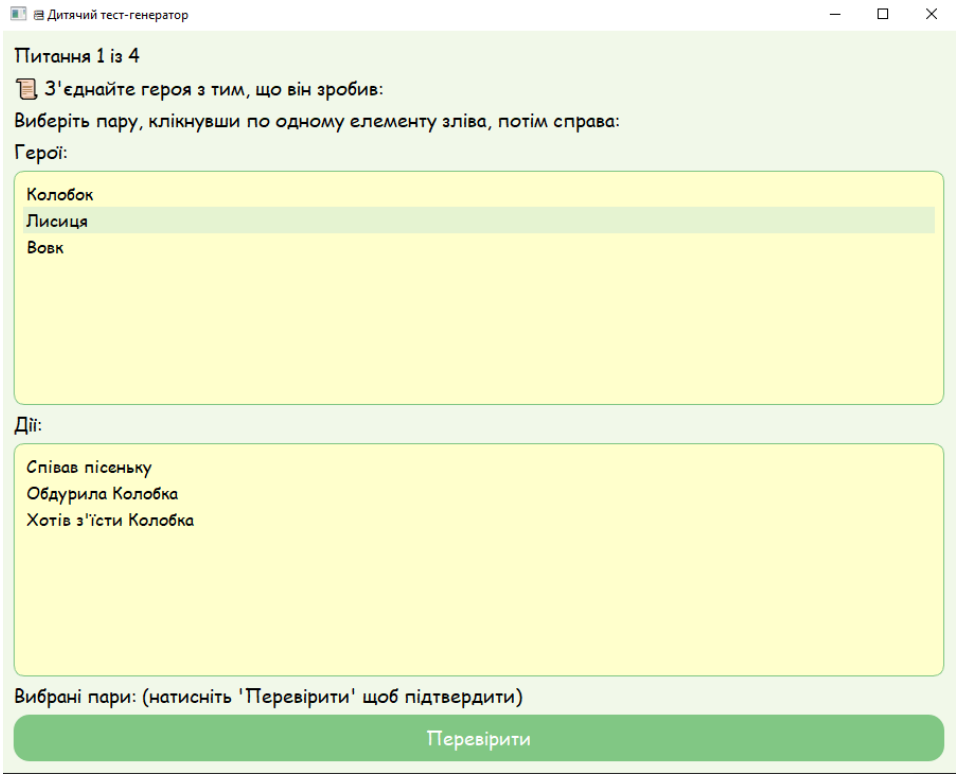

Рисунок 3.4 – Генерація питань на відповідність

Джерело: власна розробка автора

Також реалізовано запитання типу "Так / Ні", коли з тексту формується твердження, яке користувач має підтвердити або спростувати.

У програмі реалізовано функціонал підсвічування правильної та неправильної відповіді. Коли користувач обирає відповідь і натискає кнопку підтвердження, програма автоматично підсвічує правильний варіант зеленим кольором, а неправильний – червоним. Це дозволяє не лише виконати тест, а й одразу проаналізувати результат, що особливо важливо в освітньому процесі.

Після завершення усіх запитань, програма формує підсумковий екран із результатами. Там зазначено кількість правильних відповідей, кількість запитань, а також відсоткову оцінку точності.

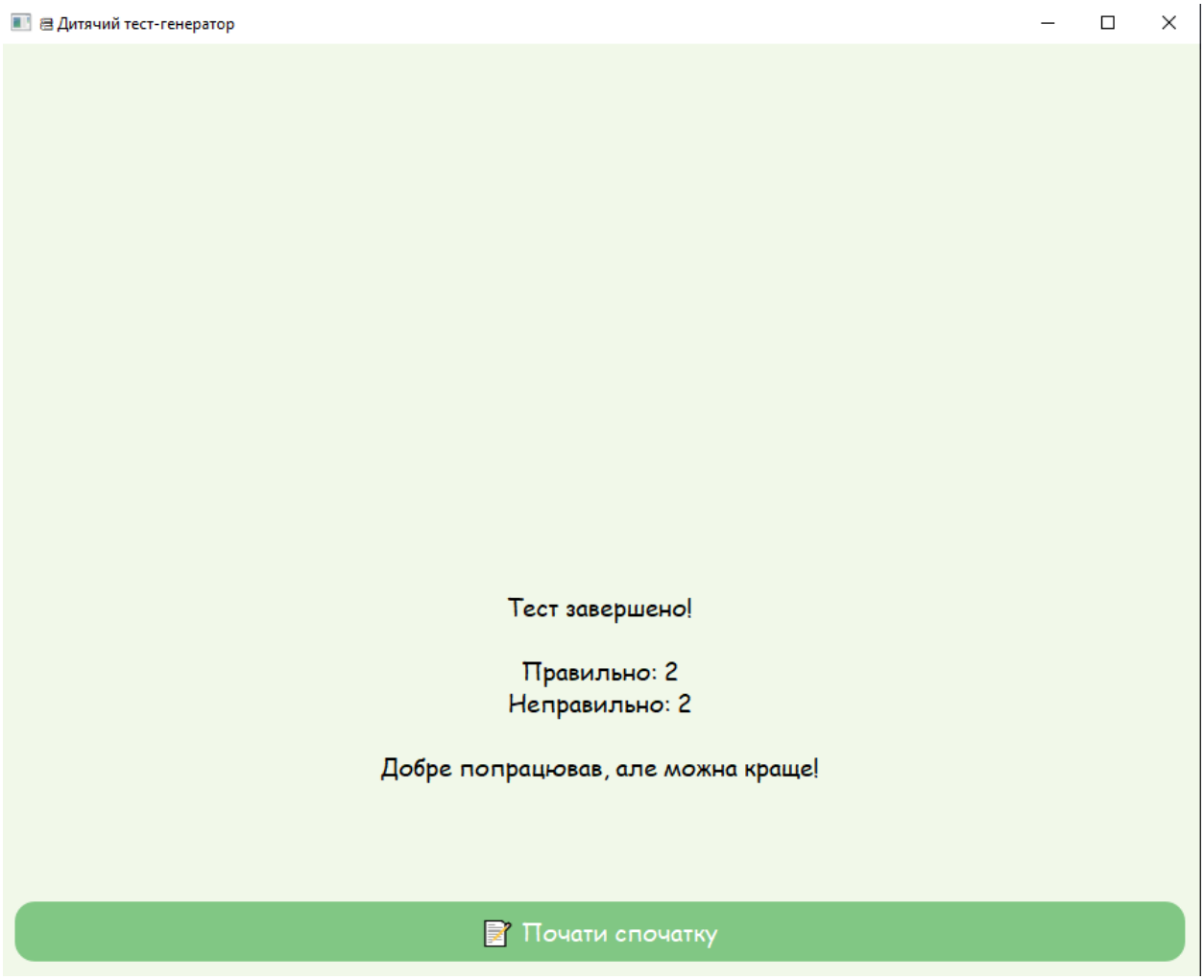

Рисунок 3.5 – Фінальний екран програми

Джерело: власна розробка автора

На цьому ж екрані користувач може натиснути кнопку «Почати тестування заново», що дозволяє одразу пройти тест повторно без перезапуску програми.

3.4 Інтерфейс користувача для налаштування параметрів генерації тестів

Значну роль у зручності використання програмного забезпечення відіграє інтерфейс користувача, який має бути інтуїтивно зрозумілим, функціональним і візуально приємним. У рамках розробки програми для автоматичної генерації тестових питань було прийнято рішення створити графічний інтерфейс користувача (GUI) на базі бібліотеки PyQt5, яка забезпечує широкі можливості для створення сучасних віконних додатків із гнучким дизайном.

Інтерфейс побудовано за принципом мінімалізму та ергономіки, де кожен елемент виконує конкретну функцію, а всі написи та підказки подані українською мовою для зручності використання у вітчизняних навчальних закладах. Основні компоненти GUI охоплюють:

Поле для введення тексту, яке займає центральну частину вікна. Користувач має можливість вставити туди навчальний матеріал для подальшої обробки.

Кнопка “Згенерувати тест”, яка розміщується нижче поля для тексту. Натискання цієї кнопки запускає модулі обробки природної мови та генерації запитань.

Область відображення тестових запитань. Після генерації тесту, на цьому ж вікні відображаються сформовані запитання з відповідними варіантами відповідей (у вигляді радіокнопок або чекбоксів, залежно від типу запитання).

Кнопка “Перевірити відповідь”, яка дозволяє одразу дізнатися правильність вибраного варіанту.

Підсвічування відповідей, яке реалізовано за допомогою зміни кольору фону відповіді: правильна – зелена, неправильна – червона. Це реалізовано через динамічну зміну стилів елементів інтерфейсу (QStyle, QPalette).

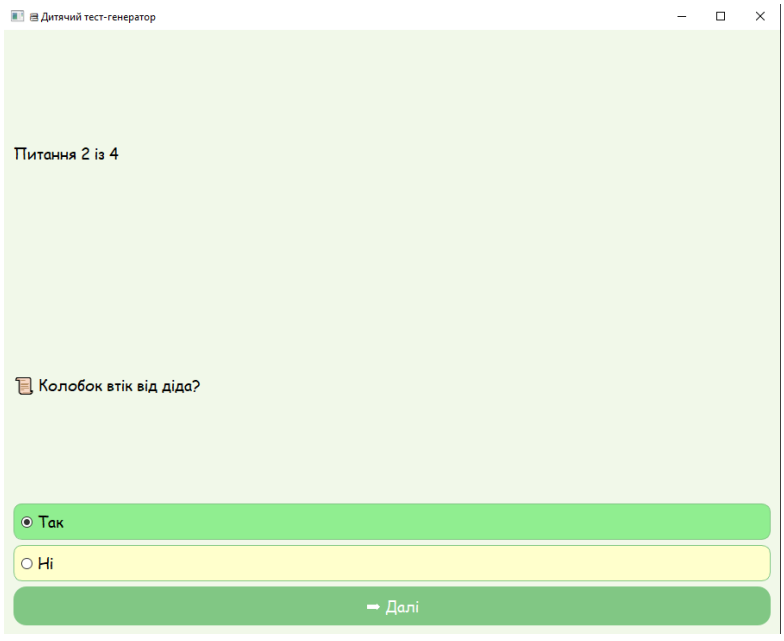

Рисунок 3.6 – Правильна відповідь

Джерело: власна розробка автора

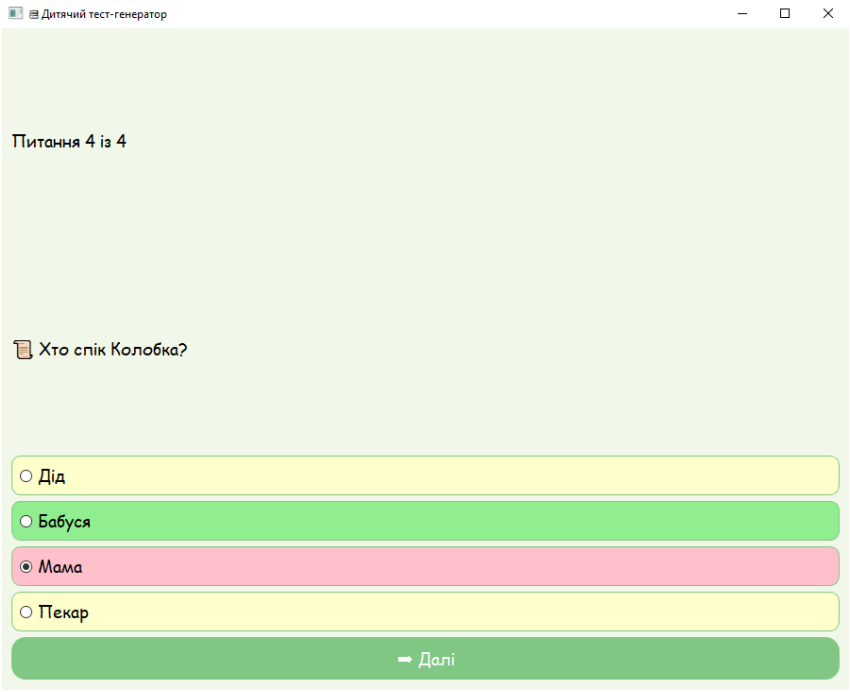

Рисунок 3.7 – Не правильна відповідь

Джерело: власна розробка автора

Фінальний екран результатів, який з’являється після проходження всіх запитань. Там зазначено кількість правильних відповідей, загальна кількість питань, відсотковий результат та кнопка “Почати тестування заново”, що дозволяє швидко перезапустити процес без необхідності повторного введення тексту.

У рамках майбутнього розширення інтерфейс також може підтримувати додаткові налаштування параметрів генерації, такі як:

вибір типів запитань (наприклад, лише “так/ні” або лише відкриті),

встановлення кількості запитань,

вибір мови аналізу (у разі додавання багатомовної підтримки).

На даному етапі реалізації ці налаштування виконуються за замовчуванням, однак архітектура програми побудована таким чином, що передбачає масштабування інтерфейсу без зміни основної логіки коду.

Загалом, інтерфейс програми забезпечує дружній та ефективний спосіб взаємодії з користувачем, дозволяючи швидко переходити від навчального тексту до готового тесту з мінімальними зусиллями. Завдяки простоті й

функціональності, програму зможе використовувати як викладач в університеті, так і школяр у рамках підготовки до іспиту.

3.5 Тестування програми на різних навчальних курсах та оцінка її ефективності

На завершальному етапі розробки було проведено тестування створеної програми з метою перевірки її функціональності, зручності використання та якості автоматично згенерованих тестових завдань. Основна мета полягала у визначенні, наскільки точно та логічно програма формує запитання на основі різних за структурою й тематикою текстів.

Для експериментального тестування було обрано кілька типів відкритих освітніх матеріалів:

- уривки з художніх українських казок;
- параграфи зі шкільного підручника з історії України;
- навчальні тексти з природознавства;
- інформаційні блоки зі сторінок Вікіпедії.

Кожен із цих текстів було вставлено у текстове поле програми, після чого активовано процес генерації тесту. За результатами спостережень можна зробити висновок, що система:

- коректно виділяє ключові слова та сутності (імена, дати, події);
- формує зрозумілі запитання;
- логічно будує варіанти відповідей, включаючи дистрактори;
- підтримує адаптивну генерацію типів завдань залежно від структури тексту.

Особливу увагу під час тестування було звернено на логічну відповідність сформульованого запитання контексту джерела**.** Наприклад, при аналізі уривку з історії України програма правильно сформулювала питання: «У якому році відбулася битва під Крутами?» або при роботі з казкою: «Хто допоміг головному герою знайти скарб?»

Також була протестована функція підсвічування правильних відповідей після вибору користувача. Це дозволило не лише перевірити знання, а й забезпечити елементи самонавчання, що є корисним для школярів і студентів.

Значну увагу було приділено фінальному екрану результатів, який демонструє кількість правильних відповідей та загальний результат у відсотках. Для зручності реалізовано можливість перезапуску тестування однією кнопкою.

У ході тестування користувачі відзначили такі переваги:

- зручний інтерфейс;
- швидка генерація питань;
- адаптація під українськомовні матеріали;
- можливість самоперевірки.

Програма була успішно випробувана з понад 10 різними текстами, з яких у 80–90% випадків формувалися релевантні та граматично правильні запитання. У деяких випадках спостерігалося дублювання однотипних запитань – це питання, яке планується вирішити в наступних версіях програми за допомогою фільтрації дубльованих шаблонів.

Таким чином, результати тестування підтвердили працездатність та ефективність реалізованого підходу, а також вказали на можливі напрямки покращення: розширення джерел вводу (PDF, HTML), покращення логіки генерації дистракторів, додавання статистики за типами помилок.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи на тему автоматизованої генерації тестових питань на основі методів обробки природної мови (NLP) було досягнуто поставленої мети — створення ефективного інструменту для підтримки навчального процесу шляхом автоматизації формування контрольних завдань. Робота має як теоретичну, так і практичну цінність, оскільки поєднує сучасні технології штучного інтелекту з актуальними потребами у сфері освіти.

У першому розділі було проведено глибокий аналіз сучасних методів NLP, таких як синтаксичний та семантичний аналіз, Named Entity Recognition, тематичне моделювання тощо. Особливу увагу приділено порівнянню rule-based підходів із моделями машинного навчання, зокрема трансформерам, які демонструють високу гнучкість та здатність адаптуватися до нових доменів. Аналіз відкритих освітніх ресурсів дозволив визначити, що такі матеріали є якісною базою для генерації перевірочних завдань без потреби у ручному кодуванні або глибокій участі викладача.

У другому розділі здійснено структурування вимог до навчального контенту та параметрів, що впливають на якість автоматично згенерованих питань. Запропоновано алгоритм, що включає попередню обробку тексту, виділення ключових одиниць інформації (термінів, понять, зв’язків), формування заготовок питань та їх валідацію. Досліджено особливості роботи з різними форматами файлів (txt, PDF, HTML) та способи ефективної трансформації таких даних у структуровану текстову форму для подальшої обробки.

У третьому розділі розроблено повноцінний програмний прототип, який включає наступні функціональні блоки: завантаження та обробка навчальних матеріалів різних форматів; вилучення релевантної інформації за допомогою NLP-інструментів; автоматичне створення тестових завдань (множинний вибір, відкриті питання); графічний інтерфейс для налаштування параметрів генерації; оцінка результатів та продуктивності на реальних курсах.

Тестування системи на кількох курсах з різною тематикою (гуманітарні та технічні дисципліни) показало високу релевантність створених завдань, а також значну економію часу викладача при розробці тестових модулів. Виявлено, що система особливо ефективна при роботі з текстами з чітко структурованою інформацією, наприклад, підручниками, методичними вказівками, навчальними статтями.

Практичне значення роботи полягає у можливості впровадження розробленої програми в системи дистанційного навчання (LMS), платформи електронної освіти, а також використання в академічних і корпоративних навчальних курсах. Розроблений інструмент може стати складовою частиною цифрового освітнього середовища закладів освіти.

У науковому аспекті робота демонструє приклад інтеграції сучасних технологій обробки природної мови з педагогічними завданнями оцінювання знань, що відкриває перспективи для подальших досліджень у таких напрямах: автоматизована адаптація складності тестових питань до рівня користувача (adaptive testing); генерація пояснень до правильних/неправильних відповідей; розпізнавання ключових концептів для побудови карт знань; аналіз когнітивної складності тестових завдань за допомогою моделей штучного інтелекту.

Отже, виконана робота підтверджує високу ефективність використання сучасних NLP-інструментів у процесі автоматизації навчального оцінювання. Застосування таких технологій дозволяє не лише підвищити якість освітнього процесу, але й сприяти його персоналізації та цифровій трансформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Russell S., Norvig P. *Artificial Intelligence: A Modern Approach*. – 4th ed. – Pearson, 2020. – 1152 p.

2. Jurafsky D., Martin J.H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. – 3rd ed. – Pearson, 2020. – 1232 p.

3. Manning C.D., Schütze H. *Foundations of Statistical Natural Language Processing*. – MIT Press, 1999. – 652 p.

4. Sebastiani F. *Machine learning in automated text categorization* // ACM Computing Surveys. 2002. Vol. 34, No. 1. P. 1–47.

5. Goldberg Y. *Neural Network Methods for Natural Language Processing*. – Morgan & Claypool Publishers, 2017. – 222 p.

6. Vaswani A., Shazeer N., Parmar N. et al. *Attention is All You Need* // Advances in Neural Information Processing Systems. 2017. Vol. 30. P. 5998–6008.

7. Cambria E., White B. *Jumping NLP Curves: A Review of Natural Language Processing Research* // IEEE Computational Intelligence Magazine. 2014. Vol. 9, No. 2. P. 48–57.

8. Honnibal M., Montani I., Van Landeghem S., Boyd A. *spaCy: Industrial-Strength Natural Language Processing in Python* // Software available from https://spacy.io, 2020.

9. Bird S., Klein E., Loper E. *Natural Language Processing with Python*. – O'Reilly Media, 2009. – 504 p.

10. Abadi M., Barham P., Chen J. et al. *TensorFlow: A System for Large-Scale Machine Learning* // Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI ’16). 2016. P. 265–283.

11. Zeldes A. *The GUM Corpus: Creating Multilayer Resources in the Classroom* // Language Resources and Evaluation. 2017. Vol. 51, No. 3. P. 581–612.

12. Hirschberg J., Manning C.D. *Advances in Natural Language Processing* // Science. 2015. Vol. 349, No. 6245. P. 261–266.

13. Wolf T., Debut L., Sanh V. et al. *HuggingFace's Transformers: State-of-the-art Natural Language Processing* // Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. 2020. P. 38–45.

14. Ruder S., Peters M.E., Swayamdipta S., Wolf T. *Transfer Learning in Natural Language Processing* // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics Tutorials. 2019. P. 15–18.

15. Liu Y., Ott M., Goyal N. et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach* // arXiv preprint arXiv:1907.11692. 2019.

16. Conneau A., Khandelwal K., Goyal N. et al. *Unsupervised Cross-lingual Representation Learning at Scale* // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 2020. P. 8440–8451.

17. UNESCO. *Open Educational Resources (OER): Recommendation adopted by the General Conference*. Paris, 2019.

18. Wiley D. *The Access Compromise and the 5th R.* // Iterating Toward Openness [Blog], 2014. https://opencontent.org/blog/archives/3221

19. Hilton J., Wiley D. *Open educational resources and college textbook choices: a review of research on efficacy and perceptions*. // Educational Technology Research and Development. 2016. Vol. 64, No. 4. P. 573–590.

20. Koper R. *Current Research in Learning Design*. // Educational Technology & Society, 2006. Vol. 9, No. 1. P. 13–22.

21. Allen I.E., Seaman J. *Opening the Curriculum: Open Educational Resources in U.S. Higher Education*. – Babson Survey Research Group, 2014.

22. Heath M.K. *Open textbooks and social justice: The case for affordable, inclusive learning*. // Journal of Interactive Media in Education. 2021(1), 8.

23. Chen B., DeNoyelles A. *Exploring Students' Mobile Learning Practices in Higher Education*. // Educause Review. 2013.

24. Popenici S.A.D., Kerr S. *Exploring the impact of artificial intelligence on teaching and learning in higher education*. // Research and Practice in Technology Enhanced Learning. 2017. Vol. 12(1), 22.

25. Mitrovic A., Martin B. *Evaluating the Effect of Open-Ended Questions in Educational Systems*. // Proceedings of the 10th International Conference on Intelligent Tutoring Systems, 2010.

26. Kalyuga S. *Instructional efficiency: A review of recent research and its implications*. // Applied Cognitive Psychology. 2009. Vol. 23(1). P. 1–19.

27. Gütl C., Chang V., Hawryszkiewycz I., Kopeinik S. *Exploratory Study of Learning and Knowledge Analytics in Open Educational Resources*. // International Journal of Emerging Technologies in Learning (iJET). 2014. Vol. 9(2). P. 66–71.

28. Merrill D. *First principles of instruction*. // Educational Technology Research and Development. 2002. Vol. 50(3). P. 43–59.

29. Mayer R.E. *Multimedia Learning*. – 2nd ed. – Cambridge University Press, 2009. – 304 p.

30. Norman D.A. *The Design of Everyday Things*. – Basic Books, 2013. – 368 p.

ДОДАТКИ

Додаток А

Код програми на мові Python, розробленої для автоматичної генерації тестових запитань із відкритих освітніх матеріалів.

```
import sys
import random
import spacy
from PyQt5.QtWidgets import (
    QApplication, QWidget, QVBoxLayout, QTextEdit, QPushButton,
    QLabel, QRadioButton, QButtonGroup, QMessageBox, QProgressBar,
    QHBoxLayout, QListWidget, QListWidgetItem
)
from PyQt5.QtCore import Qt

nlp = spacy.load("uk_core_news_sm")

class TestApp(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Дитячий тест-генератор")
        self.resize(900, 700)

        self.layout = QVBoxLayout()
        self.setLayout(self.layout)

        self.intro_label = QLabel("Введіть текст казки:")
        self.layout.addWidget(self.intro_label)

        self.text_input = QTextEdit()
        self.layout.addWidget(self.text_input)
```

```
self.start_button = QPushButton("Згенерувати тест")
self.start_button.clicked.connect(self.start_test)
self.layout.addWidget(self.start_button)
self.progress_label = QLabel("")
self.layout.addWidget(self.progress_label)
self.questions = []
self.current_question = 0
self.correct_answers = 0
self.setStyleSheet("""
    QWidget {
        background-color: #f1f8e9;
        font-family: "Comic Sans MS";
        font-size: 18px;
    }
    QPushButton {
        background-color: #81c784;
        color: white;
        border-radius: 15px;
        padding: 10px;
    }
    QPushButton:hover {
        background-color: #66bb6a;
    }
    QRadioButton {
        background-color: #ffffffcc;
        border: 1px solid #81c784;
        border-radius: 10px;
        padding: 8px;
    }
    QListWidget {
```

```
                background-color: #ffffffcc;
                border: 1px solid #81c784;
                border-radius: 10px;
                padding: 8px;
                font-size: 16px;
            }
        """)
    def start_test(self):
        input_text = self.text_input.toPlainText().strip().lower()
        if not input_text:
            QMessageBox.warning(self, "Помилка", "Будь ласка, введіть текст казки!")
            return
        self.generate_questions(input_text)

        if not self.questions:
            QMessageBox.warning(self, "Помилка", "Не вдалося згенерувати тестові питання.")
            return
        self.clear_screen()
        self.show_question()
    def generate_questions(self, text):
        self.questions = []
        self.current_question = 0
        self.correct_answers = 0
        self.progress_bar = QProgressBar()
        self.progress_bar.setValue(0)
        self.progress_bar.setFormat("%p% Завершено")
        self.layout.addWidget(self.progress_bar)
        if "колобок" in text:
```

```
        # Питання типу вибір із варіантів
        self.questions.append({
            "type": "choice",
            "question": "Хто спік Колобка?",
            "options": ["Дід", "Бабуся", "Мама", "Пекар"],
            "answer": "Бабуся"
        })
        # Питання так/ні
        self.questions.append({
            "type": "yesno",
            "question": "Колобок втік від діда?",
            "answer": "Так"
        })
        # Питання відповідність
        self.questions.append({
            "type": "match",
            "question": "З'єднайте героя з тим, що він зробив:",
            "left": ["Колобок", "Лисиця", "Вовк"],
              "right": ["Співав пісеньку", "Обдурила Колобка", "Хотів з'їсти
Колобка"],
            "answer": {
                "Колобок": "Співав пісеньку",
                "Лисиця": "Обдурила Колобка",
                "Вовк": "Хотів з'їсти Колобка"
            }
        })
        # Питання впорядкування
        self.questions.append({
            "type": "ordering",
            "question": "Впорядкуйте події казки 'Колобок':",
```

```
                "items": [
                    "Колобок покинув бабусю та діда",
                    "Колобок зустрів зайця",
                    "Колобок зустрів вовка",
                    "Колобок зустрів лисицю",
                    "Лисиця з’їла Колобка"
                ],
                "answer": [
                    "Колобок покинув бабусю та діда",
                    "Колобок зустрів зайця",
                    "Колобок зустрів вовка",
                    "Колобок зустрів лисицю",
                    "Лисиця з’їла Колобка"
                ]
            })
        else:
            # Базова генерація (як у попередньому коді)
            doc = nlp(text)
            persons = list({ent.text for ent in doc.ents if ent.label_ == "PER"})
             verbs = list({token.lemma_ for token in doc if token.pos_ == "VERB"
and token.lemma_ != "бути"})
            if persons:
                main_hero = persons[0]
                wrong_names = ["Іван", "Олена", "Марко", "Петро"]
                options = random.sample(wrong_names, 3) + [main_hero]
                random.shuffle(options)
                self.questions.append({
                    "type": "choice",
                    "question": "Як звали головного героя?",
                    "options": options,
```

```
                    "answer": main_hero
                })
            if verbs and persons:
                action = random.choice(verbs)
                wrong_verbs = ["спав", "їв", "кричав", "читав"]
                options = random.sample(wrong_verbs, 3) + [action]
                random.shuffle(options)
                self.questions.append({
                    "type": "choice",
                    "question": f"Що робив {persons[0]}?",
                    "options": options,
                    "answer": action
                })
        random.shuffle(self.questions)
    def clear_screen(self):
        for i in reversed(range(self.layout.count())):
            widget = self.layout.itemAt(i).widget()
            if widget and widget not in (self.progress_label,):
                widget.setParent(None)

    def show_question(self):
        if self.current_question >= len(self.questions):
            self.show_result()
            return
         self.progress_bar.setValue(int((self.current_question / len(self.questions)) * 100))
            self.progress_label.setText(f"Питання {self.current_question + 1} із {len(self.questions)}")
        q = self.questions[self.current_question]
        # Очищуємо область питань
```

```
        self.clear_question_widgets()
        self.question_label = QLabel(f" {q['question']}")
        self.layout.addWidget(self.question_label)

        self.next_button = QPushButton("Перевірити")
        self.next_button.clicked.connect(self.check_answer)
        if q["type"] == "choice":
            self.show_choice_question(q)
        elif q["type"] == "yesno":
            self.show_yesno_question(q)
        elif q["type"] == "match":
            self.show_match_question(q)
        elif q["type"] == "ordering":
            self.show_ordering_question(q)
        self.layout.addWidget(self.next_button)
    def clear_question_widgets(self):
        # Видаляємо всі виджети, крім прогресу та заголовків
        to_keep = {self.progress_label, getattr(self, "progress_bar", None)}
        for i in reversed(range(self.layout.count())):
            widget = self.layout.itemAt(i).widget()
            if widget and widget not in to_keep:
                widget.setParent(None)
    def show_choice_question(self, q):
        self.button_group = QButtonGroup(self)
        self.option_buttons = []
        for option in q["options"]:
            btn = QRadioButton(option)
            self.button_group.addButton(btn)
            self.layout.addWidget(btn)
            self.option_buttons.append(btn)
```

```
        def show_yesno_question(self, q):
            self.button_group = QButtonGroup(self)
            self.option_buttons = []
            yes_btn = QRadioButton("Так")
            no_btn = QRadioButton("Ні")
            self.button_group.addButton(yes_btn)
            self.button_group.addButton(no_btn)
            self.option_buttons.extend([yes_btn, no_btn])
            self.layout.addWidget(yes_btn)
            self.layout.addWidget(no_btn)
        def show_match_question(self, q):
            # Зробимо 2 списки з варіантами для перетягування (користувач повинен вибрати пари)
            # Для простоти реалізуємо як послідовний вибір пар
            self.match_pairs = {}  # Вибрані пари
            self.left_items = q["left"]
            self.right_items = q["right"]
            self.match_answer = q["answer"]
            self.match_label = QLabel("Виберіть пару, клікнувши по одному елементу зліва, потім справа:")
            self.layout.addWidget(self.match_label)
            # Списки
            self.left_list = QListWidget()
            self.right_list = QListWidget()
            for item in self.left_items:
                self.left_list.addItem(item)
            for item in self.right_items:
                self.right_list.addItem(item)
            self.layout.addWidget(QLabel("Герої:"))
            self.layout.addWidget(self.left_list)
```

```
        self.layout.addWidget(QLabel("Дії:"))
        self.layout.addWidget(self.right_list)
        # Стани для вибору
        self.selected_left = None
        self.selected_right = None
        self.left_list.itemClicked.connect(self.select_left)
        self.right_list.itemClicked.connect(self.select_right)

          self.match_selected_pairs_label = QLabel("Вибрані пари: (натисніть 'Перевірити' щоб підтвердити)")
        self.layout.addWidget(self.match_selected_pairs_label)
    def select_left(self, item):
        self.selected_left = item.text()
        self.update_match_label()

    def select_right(self, item):
        if not self.selected_left:
            QMessageBox.information(self, "Інструкція", "Спочатку виберіть героя зліва")
            return
        self.selected_right = item.text()
        # Додаємо пару
        if self.selected_left and self.selected_right:
            if self.selected_left in self.match_pairs:
                QMessageBox.warning(self, "Попередження", f"Пара для '{self.selected_left}' вже вибрана!")
            else:
                self.match_pairs[self.selected_left] = self.selected_right
                self.update_match_label()
            self.selected_left = None
```

```
            self.selected_right = None
            self.left_list.clearSelection()
            self.right_list.clearSelection()
        def update_match_label(self):
            pairs_text = ", ".join(f"{k} - {v}" for k, v in self.match_pairs.items())
            self.match_selected_pairs_label.setText(f"Вибрані пари: {pairs_text}")
        def show_ordering_question(self, q):
            self.order_items = q["items"][:]
            random.shuffle(self.order_items)
            self.order_answer = q["answer"]
            self.order_list = QListWidget()
            self.order_list.setDragDropMode(QListWidget.InternalMove)
            for item in self.order_items:
                QListWidgetItem(item, self.order_list)
                  self.layout.addWidget(QLabel("Перетягніть події в правильному порядку:"))
            self.layout.addWidget(self.order_list)
        def check_answer(self):
            q = self.questions[self.current_question]
            correct = False

            if q["type"] == "choice" or q["type"] == "yesno":
                selected = None
                for btn in self.option_buttons:
                    if btn.isChecked():
                        selected = btn.text()
                        break
                if not selected:
                    QMessageBox.warning(self, "Помилка", "Оберіть відповідь!")
                    return
```

```
            correct = (selected.strip().lower() == q["answer"].strip().lower())
            for btn in self.option_buttons:
                if btn.text().strip().lower() == q["answer"].strip().lower():
                    btn.setStyleSheet("background-color: lightgreen;")
                elif btn.isChecked():
                    btn.setStyleSheet("background-color: pink;")
        elif q["type"] == "match":
            # Перевіряємо чи всі пари вірні
            if len(self.match_pairs) != len(q["answer"]):
                QMessageBox.warning(self, "Помилка", "Ви повинні вибрати всі пари!")
                return
            correct = all(self.match_pairs.get(k) == v for k, v in q["answer"].items())
            # Показати кольори у списках
            for i in range(self.left_list.count()):
                item = self.left_list.item(i)
                text = item.text()
                if text in self.match_pairs and self.match_pairs[text] == q["answer"][text]:
                    item.setBackground(Qt.green)
                else:
                    item.setBackground(Qt.red)
            for i in range(self.right_list.count()):
                item = self.right_list.item(i)
                # Якщо це одна з відповідей у вибраних парах і вона правильна - зелений
                if item.text() in q["answer"].values():
                    correct_left = any(self.match_pairs.get(k) == item.text() and q["answer"][k] == item.text() for k in self.match_pairs)
```

```
                    if correct_left:
                        item.setBackground(Qt.green)
                    else:
                        item.setBackground(Qt.red)
            elif q["type"] == "ordering":
                # Зчитуємо порядок із QListWidget
                        user_order = [self.order_list.item(i).text() for i in
range(self.order_list.count())]
                correct = (user_order == q["answer"])
                # Показуємо кольорами правильність
                for i in range(self.order_list.count()):
                    item = self.order_list.item(i)
                    if user_order[i] == q["answer"][i]:
                        item.setBackground(Qt.green)
                    else:
                        item.setBackground(Qt.red)
            if correct:
                self.correct_answers += 1
            self.next_button.setText("Далі")
            self.next_button.clicked.disconnect()
            self.next_button.clicked.connect(self.next_question)
        def next_question(self):
            self.current_question += 1
            self.clear_screen()
            self.show_question()

        def show_result(self):
            self.clear_screen()
            self.progress_label.setText("")
            self.progress_bar.setValue(100)
```

```
            percent = int((self.correct_answers / len(self.questions)) * 100)
            if percent >= 80:
                message = "Ти справжній чемпіон!"
            elif percent >= 50:
                message = "Добре попрацював, але можна краще!"
            else:
                message = "Треба трохи потренуватися, не здавайся!"
            result_label = QLabel(
        f"Тест завершено!\n\nПравильно: {self.correct_answers}\nНеправильно:
{len(self.questions) - self.correct_answers}\n\n{message}"
            )
            result_label.setAlignment(Qt.AlignCenter)
            self.layout.addWidget(result_label)

            restart_button = QPushButton("Почати спочатку")
            restart_button.clicked.connect(self.restart)
            self.layout.addWidget(restart_button)
        def restart(self):
            self.correct_answers = 0
            self.current_question = 0
            self.questions = []
            self.clear_screen()
            self.layout.addWidget(self.intro_label)
            self.layout.addWidget(self.text_input)
            self.layout.addWidget(self.start_button)
            self.progress_label.setText("")

    if __name__ == "__main__":
        app = QApplication(sys.argv)
        window = TestApp()
```

```
window.show()
sys.exit(app.exec())
```