Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Інформаційних управляючих систем
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів і моделей генерування тестів в освітній платформі

(тема)

Виконав:
здобувач 2 року навчання,
групи ІУСТм-24-1
Дмитро КАМСЮК
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Інформаційні управляючі системи та технології
(повна назва освітньої програми)

Керівник: доц. Тетяна БОРИСЕНКО
(посада, власне ім'я, прізвище)

Допускається до захисту

Зав. кафедри ІУС _____ Костянтин ПЕТРОВ
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Інформаційних управляючих систем

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Інформаційні управляючі системи та технології
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
" 24 " листопада 20 25 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Камсюку Дмитру Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів і моделей генерування тестів в освітній платформі

затверджена наказом по університету від " 24 " листопада 2025 р. № 1055Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії " 16 " грудня 2025 р.

3. Вихідні дані до роботи матеріали передатестаційної практики, науково-технічні публікації та інтернет-джерела з тематики кваліфікаційної роботи

4. Перелік запитань, що потрібно опрацювати у роботі 1) аналіз об'єкту дослідження та постановка задачі дослідження; 2) теоретичні основи великих мовних моделей та особливості їх застосування для генерації тестів; 3) інформаційна технологія генерації генерації тестів в освітній платформі та програмна реалізація; 4) експериментальна перевірка наукових результатів.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз об’єкта дослідження та змістовна постановка задачі дослідження	24.11.2025 - 25.11.2025	Виконано
2	Огляд існуючих підходів та методів автоматичної генерації тестових запитань	26.11.2025 - 27.11.2025	Виконано
3	Формулювання функціональних та нефункціональних вимог до підсистеми генерації тестів	28.11.2025	Виконано
	Дослідження теоретичних основ великих мовних моделей (LLM) та їх архітектури	29.11.2025 - 30.11.2025	Виконано
	Аналіз ризиків використання та вибір оптимальної LLM для генерації тестів	01.12.2025 - 02.12.2025	Виконано
	Розробка підходів і механізмів мінімізації ризиків та забезпечення якості генерації	03.12.2025 - 04.12.2025	Виконано
	Розробка інформаційної технологій генерації навчальних тестів з використанням LLM	05.12.2025 - 06.12.2025	Виконано
	Програмна реалізація модуля генерації тестів	07.12.2025 - 09.12.2025	Виконано
	Експериментальна перевірка: генерація тестових запитань та аналіз результатів	10.12.2025 – 11.12.2025	Виконано
	Оформлення пояснювальної записки кваліфікаційної роботи	12.12.2025 –13.12.2025	Виконано
	Перевірка кваліфікаційної роботи на плагіат	14.12.2025	Виконано
	Попередній захист кваліфікаційної роботи	16.12.2025	Виконано

Дата видачі завдання 24 листопада 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Тетяна БОРИСЕНКО
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 134 с., 26 рис., 6 табл., 2 дод., 22 джерела.

ВЕЛИКА МОВНА МОДЕЛЬ, ГЕНЕРУВАННЯ ТЕСТІВ, ОСВІТНЯ ПЛАТФОРМА, ПРОМПТ, ТРАНСФОРМЕР; API, GPT, JSON, LLM, NLP.

Об’єктом дослідження є процес автоматичного створення тестових запитань на основі навчальних матеріалів у рамках освітньої платформи.

Метою роботи є розробка інформаційної технології (ІТ) генерації тестових запитань в освітній платформі із використанням сучасних методів обробки природної мови.

Під час дослідження застосовувалися такі методи: наукові факти, порівняння, експеримент та аналіз.

Робота містить таке: аналіз об’єкту дослідження та постановку задач дослідження, опис теоретичних основ та методів застосування мовних моделей для генерації тестів; опис розробленої ІТ генерації тестів в освітній платформі та програмної реалізації; опис експериментальної перевірки наукових результатів.

Наукова новизна роботи полягає у запропонованої системі критеріїв вибору нейронних генеративних моделей, в удосконаленні процесу генерації навчальних тестів з використанням великих мовних моделей та в розробленої ІТ генерації навчальних тестів з використанням мовної моделі сімійства GPT.

Отримані результати мають практичне значення для освітніх платформ і можуть бути використані для автоматизації створення навчальних тестів.

Кваліфікаційну роботу виконано згідно методичних вказівок щодо розробки та оформлення кваліфікаційної роботи [1], ДСТУ 3008:2015 [2] та ДСТУ 8302:2015 [3].

ABSTRACT

Explanatory note of the qualification work: 134 pages, 26 figures, 6 tables, 2 appendices, 22 sources.

LARGE LANGUAGE MODEL, TEST GENERATION, EDUCATIONAL PLATFORM, PROMPT, TRANSFORMER; API, GPT, JSON, LLM, NLP.

The object of the research is the process of automatic generation of test questions based on educational materials within an educational platform. The aim of the work is to develop an information technology (IT) for generating test questions in an educational platform using modern natural language processing methods.

The following methods were applied during the research: scientific facts, comparison, experiment, and analysis.

The work contains: analysis of the research object and formulation of research objectives; description of the theoretical foundations and methods of applying language models for test generation; description of the developed IT for test generation in the educational platform and its software implementation; description of the experimental verification of the scientific results.

The scientific novelty of the work lies in the proposed system of criteria for selecting generative neural models, in the improvement of the process for generating educational tests using large language models, and in the developed IT for generating educational tests using a large language model.

The obtained results have practical significance for educational platforms and can be used to automate the creation of educational tests.

The qualification work was completed in accordance with the methodological guidelines for the development and formatting of qualification works [1], DSTU 3008:2015 [2] and DSTU 8302:2015 [3].

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

API-інтеграція – інтеграція через програмний інтерфейс

ІС – інформаційна система

ІТ – інформаційні технології

Промпт – запит або інструкція для мовної моделі

Чанк – фрагмент тексту, частина документа

GPT – Generative Pre-trained Transformer

IDEF0 – методологія функціонального моделювання

JSON – JavaScript Object Notation

LLM – Large Language Model (велика мовна модель)

LLM-генерація – генерація тексту за допомогою великої мовної моделі

QA – Question Answering

QG – Question Generation

RNN – Recurrent Neural Network

Seq2Seq – Sequence-to-Sequence

ВСТУП

Стрімкий розвиток сучасних інформаційних технологій та поширення цифрових освітніх платформ суттєво змінюють підходи до організації навчального процесу, контролю знань та оцінювання компетентностей. В умовах зростаючої потреби у масштабованості та автоматизації навчальних сервісів особливої актуальності набуває проблема створення навчального контенту, зокрема тестових запитань, які традиційно формуються викладачами вручну. Такий підхід є трудомістким, потребує значних часових витрат та не завжди забезпечує оперативне оновлення тестових банків відповідно до змін у навчальних програмах.

Великі мовні моделі (Large Language Model, LLM) та сучасні методи обробки природної мови відкривають нові можливості для автоматизації рутинних процесів у сфері освіти. Зокрема, моделі, побудовані на архітектурі трансформерів, здатні аналізувати великі обсяги тексту, виокремлювати ключові концепти та формувати логічно структуровані тестові запитання різних типів. Це робить їх ефективним інструментом для створення адаптивних освітніх систем, що підтримують гнучке формування запитань та персоналізоване навчання.

Попри значний потенціал мовних моделей, процес автоматичної генерації тестів супроводжується низкою проблем: ризиком фактологічних помилок, втратою контексту при опрацюванні великих документів, дублюванням питань, нечіткістю або надмірною узагальненістю формулювань. Для отримання якісного результату необхідно поєднувати можливості LLM з класичними NLP-методами, застосовувати механізми структурного поділу тексту (chunking), оптимізувати промпти та виконувати додаткову валідацію згенерованого матеріалу.

У цьому контексті особливої уваги потребує розроблення інформаційної технології, яка здатна інтегрувати сучасні моделі генерації тексту в освітнє

середовище, забезпечити точність створених питань, їх відповідність змісту навчальних матеріалів, а також можливість масштабування й адаптації до різних видів документів. Актуальність теми зумовлена зростаючим попитом на автоматизовані засоби оцінювання та необхідністю оптимізації діяльності викладачів у сучасних цифрових університетах.

Результати дослідження можуть бути використані в освітніх платформах, електронних системах тестування, корпоративних навчальних середовищах та сервісах дистанційного навчання. Розроблена технологія дає можливість автоматизувати створення тестового контенту, підвищити ефективність контролю знань, скоротити витрати часу на підготовку матеріалів та забезпечити стандартизовану якість тестових запитань.

Метою кваліфікаційної роботи є розроблення інформаційної технології генерації тестових запитань для освітньої платформи, що застосовує великі мовні моделі для формування структурованих, логічно коректних та змістовно правильних тестів.

Для досягнення поставленої мети необхідно виконати такі завдання:

– проаналізувати сучасні підходи, методи й інструменти автоматичної генерації тестів;

– дослідити особливості використання великих мовних моделей у задачах генерації навчального контенту;

– визначити вимоги до підсистеми автоматичної генерації тестів;

– розробити інформаційну технологію та архітектуру підсистеми генерацыъ тестів;

– реалізувати програмний модуль генерації тестів;

– виконати експериментальну перевірку ефективності інформаційної технології та якості сформованих тестових запитань.

1 АНАЛІЗ ОБ'ЄКТУ ДОСЛІДЖЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Аналіз процесу генерації тестів

Процес створення тестових запитань традиційно є трудомістким та вимагає значних зусиль з боку викладачів. Залежно від обсягу навчального матеріалу, викладач має уважно опрацювати конспект, виділити ключові положення, сформулювати запитання різних рівнів складності та підібрати коректні варіанти правильних та неправильних відповідей. Такий підхід забезпечує якість оцінювання, проте забирає значну кількість часу, особливо в умовах великих навчальних груп або необхідності регулярного оновлення тестів. Процес автоматизованої генерації тестових запитань із використанням великої мовної моделі представлено на рисунку 1.1.

Рисунок 1.1 – Схема процесу генерації тестових запитань з використанням LLM

Поєднання великих мовних моделей із предметно-орієнтованими пайплайнами (деталізованими та стандартизованими послідовністями етапів (стадій)) обробки даних сьогодні виходить далеко за межі освітньої сфери.

Можливість аналізувати великі масиви текстової інформації, виділяти ключові смислові елементи й формувати структурований результат робить такі системи ефективним інструментом у різних галузях [4].

Наприклад, у юридичній сфері подібні технології застосовують для автоматизованого аналізу договорів і нормативних документів, система може виділяти важливі пункти, знаходити потенційні ризики та формувати короткі підсумки для юристів. Це суттєво скорочує час підготовки документів і зменшує ймовірність пропуску важливих деталей.

У фінансовому секторі мовні моделі допомагають здійснювати первинний аналіз звітності, підготовку аналітичних довідок, оцінку кредитних ризиків та виявлення нетипових операцій. Поєднання класичних алгоритмів фінансової аналітики з мовними моделями дозволяє гнучко аналізувати контекст та пояснювати висновки в більш зрозумілій формі.

У технічній підтримці такі системи використовуються для створення чат-ботів, здатних опрацьовувати звернення користувачів, шукати інформацію у базах знань та пропонувати інструкції. Тут LLM працює у поєднанні з базою технічної документації та модулями пошуку, що дозволяє автоматизувати значну частину рутинних звернень.

Важливою сферою застосування LLM, є і маркетинг, де моделі допомагають формувати опис товарів, резюме для презентацій, короткі інформаційні повідомлення або персоналізовані листи. У цьому випадку LLM поєднується з даними про аудиторію, історією взаємодій користувачів та аналітичними інструментами.

Усе це свідчить про те, що інтеграція мовних моделей у прикладні пайплайни стає стандартною практикою. Завдяки здатності швидко аналізувати текст великого обсягу та генерувати структуровані результати, такі системи допомагають скоротити ручну працю, зменшити навантаження на фахівців та підвищити точність обробки інформації. Тому використання подібного підходу в автоматизованій генерації тестів не лише є логічним, а й відображає загальні тенденції розвитку сучасних інтелектуальних технологій.

Останніми роками з’явилися технологічні підходи, що дозволяють автоматизувати окремі етапи генерації тестів. Зокрема, використання великих мовних моделей суттєво змінює спосіб роботи з навчальними матеріалами. Модель, отримавши текст лекції чи конспекту, здатна визначити ключові поняття, знайти смислові зв’язки та сформувати тестові завдання, орієнтуючись на структуру й наповнення документа. Такі можливості відкривають шлях до побудови системи, яка може частково або повністю автоматизувати генерацію тестів, забезпечуючи при цьому прийнятну якість та охоплення теми.

Щоб зрозуміти логіку цього процесу, важливо розглядати генерацію тестів з використанням мовної моделі не як одномоментний акт, а як послідовність взаємопов’язаних етапів. В основі лежить робота з навчальним текстом, який система повинна спершу коректно отримати та привести до придатної для аналізу форми. На цьому етапі здійснюється вилучення тексту з документів (PDF або DOCX), видалення сторонніх символів, злиття розірваних абзаців і визначення мови матеріалу. Така підготовка дозволяє уникнути помилок під час подальшої генерації та забезпечує цілісність змісту.

Другий етап пов’язаний із формуванням контексту, який мовна модель може опрацювати. Через обмеження на довжину вхідної послідовності великі документи поділяють на змістові фрагменти – чанки. Як свідчить практика, саме поділ на логічні блоки з невеликим перекриттям дає змогу моделі працювати з матеріалом уважніше та уникати втрати контексту на межах фрагментів. Після такого структурування кожен блок окремо передається до мовної моделі разом із промптом, у якому чітко визначено формат майбутніх тестових запитань.

Ключовим моментом у процесі генерації є якість інструкції – промпта. Саме промпт визначає, якої складності та якого типу будуть запитання, чи потрібні варіанти відповідей, як саме структурований вихідний формат. Від точності цього елементу залежить те, наскільки узгодженою та послідовною буде вся добірка запитань. На практиці моделі здатні не лише відтворювати

зміст тексту у формі тестів, а й коректно розподіляти складність запитань, уникати повторів і пропонувати логічні варіанти відповідей, якщо промпт достатньо чіткий.

Після отримання первинних результатів необхідна постобробка. Модель може генерувати подібні запитання в різних чанках, тому виникає потреба у видаленні дубльованих або надто схожих формулювань. Додатково перевіряється відповідність правильних відповідей змісту документа та узгодженість формату. У деяких випадках доцільно застосовувати окрему перевірку коректності, щоб уникнути помилкових або некоректних формулювань.

Узагальнюючи, процес генерації тестових запитань можна розглядати як структуровану послідовність, від вилучення тексту й підготовки даних до формування тестів, їх перевірки та впорядкування. Такий підхід дозволяє систематизувати роботу й забезпечити стабільний рівень якості навіть при обробці великих обсягів навчальних матеріалів. У межах даної дипломної роботи цей процес доповнюється інженерними рішеннями, що підвищують точність, зменшують витрати та роблять систему практично придатною для використання в освітніх платформах.

1.2 Огляд існуючих підходів до автоматичної генерації тестів

Автоматична генерація тестових запитань в освітніх технологіях давно є предметом дослідження, однак інтенсивний розвиток цього напряму розпочався відносно недавно. Перші системи були здебільшого правильні і покладалися на заздалегідь визначені шаблони, за якими формували запитання. Такий підхід дозволяв отримувати передбачуваний результат, проте його гнучкість була обмеженою, система могла працювати лише з чітко структурованим текстом і не враховувала складних смислових зв’язків.

Подальший розвиток пов’язаний із використанням методів обробки природної мови. Статистичні алгоритми та моделі машинного навчання дали змогу визначати ключові слова, важливі фрагменти тексту та будувати запитання на основі виявлених шаблонів. Наприклад, поширеним підходом стало автоматичне генерування запитань на основі частотних характеристик термінів або виявлення іменованих сутностей. Такі методи працювали краще за повністю правилові системи, але їхні можливості все ще залежали від якості вихідних даних та чітких структурних ознак тексту.

З появою глибокого навчання почалася нова хвиля розвитку. Рекурентні нейронні мережі та перші трансформерні моделі дали змогу моделювати текст на рівні складніших залежностей, що дозволило формувати більш змістовні запитання. Проте навіть ці системи вимагали великих обсягів навчальних даних і часто були спрямовані на окремі, досить вузькі завдання – наприклад, створення запитань з пропусками, класифікація типів запитань або автоматичний добір відповідей.

Подальший розвиток трансформерної архітектури привів до створення великих мовних моделей (LLM), здатних працювати із загальними мовними задачами. Їхнє використання в автоматичному генеруванні тестів дало змогу отримувати запитання, що наближені до тих, які створює людина, і робити це без необхідності тривалої підготовки або спеціального навчання моделі. LLM опрацьовують значні обсяги тексту, узагальнюють зміст і формують запитання різних типів – від простих тестів з вибором відповіді до аналітичних відкритих запитань. Важливою перевагою цих моделей є можливість керування результатом через промпти, чіткі інструкції дозволяють визначити стиль, формат та складність майбутніх тестових запитань.

Окрім універсальних великих мовних моделей, поширеним є поєднання LLM з предметно-орієнтованими інструментами. Такий підхід дозволяє підсилити систему за рахунок попередньої обробки тексту – визначення ключових термінів, структурування документа, поділу його на змістові фрагменти та вилучення найважливіших концептів. Поєднання класичних

NLP-методів, алгоритмів пошуку та великої мовної моделі робить процес генерації більш керованим і точним, особливо якщо навчальний матеріал має складну структуру або містить спеціальну термінологію.

Попри значні досягнення, кожен із підходів має свої обмеження. Правилові системи недостатньо гнучкі, статистичні моделі часто не враховують глибокий контекст, а нейронні мережі потребують значних навчальних ресурсів. Великі мовні моделі, у свою чергу, залежать від чіткості інструкції та можуть створювати неоднорідні результати без додаткової перевірки. Тому сучасні рішення тяжіють до комбінованих підходів, де генерація тестів включає декілька етапів – попередню обробку тексту, використання LLM для формування запитань і подальшу валідацію результатів.

1.2.1 Правила і шаблони генерації запитань (rule-based підходи)

Одним із найперших напрямів автоматичної генерації тестових запитань стали підходи, засновані на правилах та шаблонах. Такі системи працюють за заздалегідь визначеними логічними інструкціями, де кожен тип запитання формується за фіксованим набором умов. Наприклад, якщо в тексті виявлено певну структуру – термін, визначення чи дата – система застосовує відповідний шаблон і формує запитання у встановленому форматі.

У межах правилового підходу алгоритм зазвичай включає три ключові етапи: розпізнавання важливих фрагментів тексту, співставлення їх із відповідними шаблонами та побудову запитання згідно з цими правилами. Наприклад, речення, що містять конструкції «є…», «визначається як…», часто обробляються як кандидати на створення запитання типу «Що таке…?». Подібна логіка дозволяє системі працювати швидко та передбачувано.

Перевагою таких методів є їхня простота та контрольованість. Оскільки

всі правила задаються вручну, результати генерації є стабільними і не потребують складних обчислень. Rule-based системи можна легко адаптувати під конкретний предмет чи структуру навчальних матеріалів, що робить їх корисними в умовах вузькоспеціалізованого навчання.

Однак цей підхід має суттєві обмеження. По-перше, він залежить від точності формулювань і структури тексту, якщо фрази складні або містять кілька смислових рівнів, система може неправильно визначити потрібний шаблон. По-друге, правила не здатні охопити всі можливі мовні конструкції, тому якість результатів знижується при роботі з неформатованими або неоднорідними матеріалами. По-третє, створення великої кількості правил є трудомістким, а підтримка їх актуальності потребує постійного вдосконалення.

Попри ці недоліки, системи, що ґрунтуються на правилах й сьогодні застосовуються у навчальних платформах, де важлива передбачуваність та контроль за змістом тестових запитань. У поєднанні з іншими методами – наприклад, попереднім вилученням ключових термінів або семантичним аналізом – rule-based підходи можуть доповнювати сучасні моделі та підвищувати якість генерації тестів у спеціалізованих випадках.

1.2.2 Статистичні та NLP-підходи

Після правилових систем наступним етапом розвитку автоматичної генерації тестових запитань стали статистичні та класичні методи обробки природної мови. На відміну від rule-based підходів, вони менш залежні від жорстко заданих шаблонів і здатні працювати з текстами більш довільної структури.

Одним із найпоширеніших методів став аналіз частотності термінів. Основна ідея полягає в тому, що слова та словосполучення, які часто

зустрічаються в документі, зазвичай є ключовими для розуміння теми. На основі таких термінів система може автоматично формувати тестові запитання на визначення понять або їх застосування. Подібні алгоритми прості у реалізації та дають змогу швидко виокремити важливі елементи тексту, однак вони не враховують контекст, у якому використано ті чи інші слова.

Іншим класичним методом є виявлення іменованих сутностей – людей, дат, місць, термінів чи назв об’єктів. Такі сутності часто стають основою для запитань на фактологічні знання. Наприклад, фрази, що містять важливі дати або назви процесів, можуть бути автоматично перетворені на запитання типу «Коли…?» або «Що характеризує…?». У цьому підході важливо правильно налаштувати моделі NER (Named Entity Recognition), щоб уникнути пропусків або помилкових спрацювань.

Окремим напрямом стали алгоритми класифікації та виділення ключових речень. Такі методи використовують показники TF-IDF, векторизацію тексту або інші статистичні особливості для визначення речень, що найкраще передають зміст розділу. На основі таких речень можна формувати як запитання з вибором відповіді, так і завдання на встановлення відповідності або доповнення тверджень. Попри простоту, ці методи дають доволі стабільні результати для структурованих текстів.

Важливо зазначити, що традиційні NLP-підходи добре працюють у випадках, коли навчальний матеріал має чітку логіку викладу та містить характерні мовні маркери. Однак вони обмежені у роботі зі складними або багатошаровими текстами, де важливо враховувати смислові зв’язки між абзацами. Крім того, статистичні підходи не здатні самостійно будувати запитання складних типів, які потребують розуміння причинно-наслідкових зв’язків або глибокої інтерпретації змісту.

Попри ці недоліки, класичні NLP-методи і сьогодні застосовуються як проміжний етап у комплексних системах генерації тестів. Вони дозволяють підготувати текст, структурувати його, виділити ключові поняття та передати більш релевантний матеріал моделям глибокого навчання або сучасним

мовним моделям, підвищуючи їх точність і зменшуючи навантаження на інші елементи системи.

1.2.3 Моделі на основі нейронних мереж

Подальший розвиток автоматичної генерації тестових запитань став можливим завдяки появі нейронних мереж, здатних моделювати складні мовні залежності. Одними з перших архітектур, що отримали широке поширення для задач генерації тексту, були seq2seq-моделі. Вони складаються з двох основних компонентів – енкодера та декодера. Енкодер послідовно обробляє вхідний текст і перетворює його у внутрішнє представлення, тоді як декодер генерує новий текст на основі цього представлення. Такий підхід дав змогу створювати запитання, які враховували не лише окремі ключові слова, а й загальний зміст вхідного фрагмента.

Важливою особливістю seq2seq-архітектур є механізм attention, що дозволяє моделі фокусуватися на найбільш значущих частинах тексту під час генерації запитання. Завдяки цьому система краще розуміє внутрішню логіку документа та може формувати більш точні й змістовні запитання, особливо для структурованих навчальних матеріалів.

Революційним кроком у розвитку генеративних моделей стало появлення трансформерів [5, 6]. На відміну від рекурентних мереж, трансформери не обробляють текст посимвольно чи поступово, а працюють із усією послідовністю одночасно. Це дає змогу глибше аналізувати контекст та виявляти смислові зв’язки, що знаходяться на значній відстані один від одного. Такі властивості особливо важливі під час формування тестів, коли необхідно врахувати одразу кілька концептів чи взаємопов’язаних тверджень. Трансформерні моделі навчилися створювати різні типи запитань, від простих запитань на визначення до запитань, що потребують аналізу чи логічного

узагальнення. Їхня гнучкість дала можливість переходу від жорстко фіксованих шаблонів до більш природної генерації, яка враховує контекст та змістовну глибину матеріалу. Це стало фундаментом для появи сучасних великих мовних моделей, що застосовуються сьогодні у багатьох освітніх сервісах.

Попри значні переваги, нейронні мережі потребують ретельного налаштування та якісних даних для навчання. Якщо навчальний матеріал містить неоднорідну термінологію або складні формулювання, модель може формувати неточні запитання чи пропускати важливі деталі. Крім того, традиційні трансформери працюють у межах фіксованої довжини контексту, що обмежує їх можливості при обробці великих документів без попереднього поділу тексту.

Однак саме ці моделі стали основою для появи більш потужних систем, здатних працювати з великими текстовими масивами та формувати якісні тестові завдання без додаткового навчання. Тому seq2seq-архітектури та трансформери можна розглядати як важливий етап еволюції, що підготував ґрунт для сучасних мовних моделей та їх застосування у генерації тестів, переглянути порівняння між (RNN) та Transformer можна за таблицею 1.1.

Таблиця 1.1 – Порівняння Seq2Seq (RNN) та Transformer

Критерій	Seq2Seq (RNN, LSTM/GRU)	Transformer
Архітектура	Послідовна обробка тексту: кожне слово читається по одному, залежності передаються через приховані стани	Паралельна обробка всієї послідовності, механізм self-attention розглядає всі слова одночасно
Контекст	Обмежений «пам’яттю» RNN; складно враховувати далекі залежності	Добре враховує довготривалі залежності завдяки attention
Швидкість роботи	Повільніші, бо працюють послідовно	Значно швидші завдяки паралельності обчислень

Продовження таблиці 1.1

Критерій	Seq2Seq (RNN, LSTM/GRU)	Transformer
Навчання	Важче піддаються навчанню, більше часу на епоху	Ефективне навчання на великих корпусах, висока масштабованість
Якість генерації тексту	Часто менш природна, іноді втрачається контекст	Висока природність тексту, логічність і зв'язність
Вимоги до даних	Можуть працювати з відносно малими наборами даних	Найкраще працюють з великими обсягами даних
Проблеми	«Забування» контексту, вибух або затухання градієнтів	Високі обчислювальні вимоги, складніша архітектура
Типові завдання	Машинний переклад початкових поколінь, прості генеративні задачі	Сучасні системи генерації тексту, QA, чат-боти, контекстний аналіз
Придатність для генерації тестів	Підходить для простих запитань на основі коротких фрагментів	Найкраще підходить, особливо для складних текстів і різних типів запитань

1.2.4 Порівняння сучасних програмних рішень для автоматичної генерації тестових запитань

Сьогодні існує значна кількість програмних рішень та вебсервісів, що пропонують можливості автоматичного створення тестових запитань. Ці інструменти по-різному реалізують процес генерації запитань – від простих правилових механізмів до використання сучасних мовних моделей. Аналіз таких сервісів дозволяє окреслити їх сильні та слабкі сторони, а також визначити, яких функціональних можливостей бракує та що може бути вдосконалено у власній системі.

Більшість наявних рішень орієнтовані на роботу з короткими навчальними матеріалами, де інформація подана структуровано та

передбачувано. Частина сервісів забезпечує лише базовий функціонал, створюючи запитання на основі ключових термінів або використовуючи заздалегідь підготовлені шаблони. У той же час, платформи нового покоління застосовують великі мовні моделі й здатні обробляти значно складніші тексти, формуючи запитання різних рівнів складності. Проте попри наявність таких можливостей, більшість рішень залишаються обмеженими у гнучкості або не дозволяють повноцінно працювати з великими документами, звідки виникає потреба у спеціалізованих системах.

Порівняння сучасних сервісів генерації тестів наведено у таблиці 1.2. Якщо узагальнити результати порівняння, можна виділити декілька сервісів, підходи яких заслуговують на особливу увагу. Наприклад, рішення на основі LLM, такі як Smodin або Mindsmith, демонструють гнучкість у формуванні запитань і здатні адаптуватися до різних стилів та форматів навчальних матеріалів. Їхня сила полягає у швидкості та можливості генерувати доволі природні й змістовні формулювання без попереднього структурування тексту.

Інший підхід, який варто відзначити, представлений системами типу Moodle. Хоча вони здебільшого спираються на шаблонні механізми та правила, їх перевагою є чіткість, передбачуваність і можливість гнучкого управління процесом тестування у межах навчальних курсів. Такі системи не прагнуть до повної автоматизації генерації, проте надають стабільні результати, важливі для формальної оцінки знань.

Найбільш перспективним у контексті створення тестів із великих документів виглядає підхід, що використовує мовні моделі через API. Він дозволяє застосовувати сучасні механізми обробки тексту, поєднуючи їх із власними пайплайнами, що забезпечують структурність та контроль над логікою генерації. Це відкриває можливість формувати запитання різної складності, працювати з великим обсягом матеріалу та уникати шаблонності, властивої простішим сервісам.

Таблиця 1.2 – Порівняльна таблиця сучасних сервісів генерації тестів

Сервіс / тип рішення	Тип генерації	Переваги	Недоліки	Якість запитань	Придатність для навчальних курсів
Quizlet (створення карток + AI-generate)	LLM-орієнтована генерація	Простий інтерфейс, швидке формування карток	Обмежений контроль формату, мало типів тестів	Середня – підходить для базових понять	Добре підходить для простих курсів
Smodin Test Generator	LLM + шаблони	Підтримка різних типів запитань, автоматичний аналіз тексту	Може створювати неточні запитання при складному матеріалі	Висока для простих текстів, середня для складних	Підходить для гуманітарних курсів
Moodle (плагіни автогенерації)	Rule-based + NLP	Гнучка інтеграція в LMS, широкий вибір типів запитань	Шаблонність результатів, потреба вручну готувати структуру	Стабільна, але без глибокого аналізу	Оптимально для формальних тестів

Продовження таблиці 1.2

Сервіс / тип рішення	Тип генерації	Переваги	Недоліки	Якість запитань	Придатність для навчальних курсів
Google Forms + аддони (Flubaroo тощо)	Частково автоматизова-на обробка	Простота використання, інтеграція з Google Drive	Відсутність повноцінної генерації запитань	Низька – користувач створює запитання вручну	Для базової тестації
AI-орієнтовані генератори (Mindsmith, ClassPoint AI)	LLM-базована генерація	Автоматичне створення тестів із тексту чи презентації, швидкість	Може дублювати запитання або пропускати важливий контекст	Висока, але залежить від промпта	Підходять для інтерактивних лекцій
GPT-орієнтовані API-системи	Повноцінна LLM-генерація з можливістю налаштування пайплайнів	Гнучкість, глибокий аналіз тексту, можливість оновлення логіки	Потребують інтеграції та опрацювання результатів	Дуже висока за умови якісного промпта	Ідеально для складних або великих курсів

Таким чином, огляд існуючих рішень показує, що жоден із доступних сервісів не забезпечує одночасно високу гнучкість, можливість роботи з великими документами та повний контроль над генерацією. Це підкреслює доцільність створення спеціалізованої системи [7], яка об’єднає сильні сторони сучасних інструментів та усуне їх ключові обмеження.

1.3 Вимоги до підсистеми генерації тестів для освітньої платформи

Підсистема автоматичної генерації тестових запитань є ключовим елементом сучасних освітніх платформ, які прагнуть забезпечити гнучкість, адаптивність та високу якість навчального процесу. Для успішного функціонування такої підсистеми необхідно визначити набір вимог, що гарантують коректність обробки навчальних матеріалів, стабільність роботи та відповідність створених тестів змісту навчального контенту. У цьому підрозділі сформульовано основні вимоги, які повинна задовольняти система генерації тестів, розроблена в межах даної дипломної роботи.

1.3.1 Функціональні вимоги

1.3.1.1 Прийом і обробка текстових матеріалів:

а) підсистема повинна приймати навчальні матеріали у форматах PDF, DOCX, TXT;

б) підсистема має здійснювати попередню обробку документа: вилучення тексту з конспекту чи лекції, очищення від артефактів, нормалізацію структури;

в) користувач має мати можливість завантажувати матеріал, обирати

типи запитань та обирати складність згенерованих тестів. До кожної складності є свої правила генерації тестів для користувача.

г) підсистема повинна надавати інформативні повідомлення про хід обробки документа.

1.3.1.2 Аналіз змісту навчального матеріалу:

а) підсистема має підтримувати поділ великих документів на змістові блоки (чанки) для подальшої передачі у мовну модель;

б) має бути забезпечена можливість адаптивного вибору типів запитань користувачем, залежно від структури тексту.

1.3.1.3 Генерація тестових запитань:

а) необхідно забезпечити генерування тестових запитань таких типів;

1) запитання з вибором однієї правильної відповіді;

2) запитання з кількома правильними відповідями;

3) запитання на відповідність;

4) запитання на встановлення послідовності;

5) відкриті запитання короткої відповіді.

б) генерація повинна здійснюватися на основі мовної моделі з урахуванням заданих параметрів (кількість запитань, складність);

в) потрібно уникати дублювання запитань, навіть якщо вони були згенеровані з різних частин документа.

1.3.1.4 Перевірка коректності результатів:

а) підсистема повинна виконувати базову валідацію сформованих запитань для перевірки такого;

1) відповідності запитання змісту документа;

2) логічності формулювань;

3) коректності варіантів відповіді;

б) у разі виявлення невідповідностей підсистема має повторно генерувати запитання або позначати їх як такі, що потребують ручного перегляду.

1.3.1.5 Зрозумілість результатів:

а) згенеровані тести повинні відображатися у структурованому вигляді: запитання → варіанти відповідей → правильні відповіді;

б) бажано, щоб система забезпечувала попередній перегляд результату перед збереженням.

1.3.2 Нефункціональні вимоги

1.3.2.1 Вимоги до продуктивності:

а) підсистема повинна забезпечувати обробку документів середнього обсягу (до 30–50 сторінок);

б) час генерації тестів не повинен перевищувати технологічно обґрунтованих меж для освітніх платформ (кілька десятків секунд за наявності LLM API).

1.3.2.2 Надійність і стійкість роботи:

а) система повинна коректно працювати з документами різного формату;

б) у разі помилки обробки має застосовуватися механізм повторної спроби або виведення інформативного повідомлення.

1.3.2.3 Якість згенерованих запитань:

а) запитання повинні бути логічно узгодженими, змістовними і відповідати академічному рівню.

б) необхідно забезпечити відсутність штучності, стилістичних викривлень або випадкових відхилень від теми.

1.3.2.4 Інтеграція з мовною моделлю: має бути реалізований механізм передачі чанків тексту, промптів та параметрів генерації [11].

1.3.2.5 Інтеграція з освітньою платформою:

а) підсистема повинна передавати результати (тести та їхні структури) у форматі, сумісному з внутрішніми модулями освітньої платформи.

б) має бути забезпечена можливість подальшого редагування

згенерованих запитань викладачем.

Таким чином, підсистема генерації тестових запитань повинна поєднувати потужність сучасних мовних моделей із методами структурної обробки тексту, забезпечуючи високу якість тестів, гнучкість роботи з документами та зручність інтеграції в освітню платформу. Чітке визначення вимог на цьому етапі дозволяє створити систему, здатну працювати стабільно, ефективно та відповідно до потреб користувача.

1.4 Постановка задач магістерської кваліфікаційної роботи

На основі проведеного аналізу процесів автоматичної генерації тестових запитань, розгляду існуючих підходів та систем, а також визначення вимог і ризиків використання мовних моделей, можна сформулювати цілі та завдання магістерської роботи. Постановка задач дозволяє окреслити логіку подальших етапів дослідження та визначити основні напрями розробки підсистеми генерації тестів.

Об’єктом дослідження в рамках магістерської кваліфікаційної роботи є процес автоматичного створення тестових запитань на основі навчальних матеріалів.

Предметом дослідження є методи та моделі, що використовуються для побудови підсистеми генерації тестів з використанням мовних моделей.

Метою дослідження є розроблення підсистеми автоматичної генерації тестових запитань для освітньої платформи, що використовує сучасні мовні моделі та інструменти обробки тексту для формування якісних, змістовних і логічно коректних тестів на основі навчальних матеріалів у різних форматах.

Для досягнення поставленої мети передбачено виконання таких запитань дослідження:

1) проаналізувати існуючі підходи до автоматичної генерації тестових

запитань;

2) розробити вимоги до підсистеми генерації тестів;

3) провести аналіз та вибір реалізації великої мовної моделі (LLM), найбільш придатної для генерації навчальних тестів;

4) дослідити особливості використання LLM для генерації навчальних тестів, виявити основні ризики та проблеми, пов’язані з використанням LLM, запропонувати способи їх мінімізації;

5) розробити інформаційну технологію генерування тестів в освітній платформі з використанням реалізації генеративної LLM;

6) виконати програмну реалізацію підсистеми генерації тестів у складі освітньої платформи, яка впроваджує розроблену ІТ;

7) виконати експериментальну перевірку отриманих наукових результатів.

Очікуваним результатом має бути інформаційна технологія, яка забезпечує якісну автоматичну генерацію тестових запитань на основі текстових документів, здатну інтегруватися в освітню платформу та підтримувати різні формати навчальних матеріалі.

2 ТЕОРЕТИЧНІ ОСНОВИ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ ТА ОСОБЛИВОСТІ ІХ ЗАСТОСУВАННЯ ДЛЯ ГЕНЕРАЦІЇ ТЕСТІВ

2.1 Основи обробки природної мови

Сучасні великі мовні моделі (LLM), такі як GPT, BERT, Claude, PaLM тощо, є результатом багаторічного розвитку обробки природної мови (NLP). Хоча цей проект та дослідження зосереджено на використанні LLM як інструменту для генерації тестових завдань, варто розглядати ці моделі в ширшому контексті – як частину еволюції методів NLP.

NLP – це міждисциплінарна галузь на перетині лінгвістики, інформатики та штучного інтелекту, яка досліджує алгоритми аналізу, розуміння та генерації тексту. Саме в межах цієї області формувалися теоретичні та практичні основи, на яких згодом побудовано LLM. Таким чином, обговорення базових принципів NLP є необхідною передумовою для подальшого розгляду сучасних мовних моделей.

У контексті освітніх систем методи NLP дають змогу автоматично виділяти змістовні одиниці з навчальних матеріалів, ідентифікувати ключові поняття, аналізувати структуру тексту та готувати його до подальшої обробки мовною моделлю. Це особливо актуально при побудові систем автоматизованої генерації тестів, де якість первинної текстової обробки безпосередньо впливає на змістовність і коректність сформульованих запитань.

NLP базувався на статистичних методах, що використовували частотний аналіз, словникові моделі та прості евристики – токенізацію, лематизацію, видалення стоп-слів. Ці методи давали змогу структурувати текст, однак не забезпечували глибокого розуміння значення слів у контексті. Незважаючи на це, класичні NLP-алгоритми залишаються актуальними як проміжна ланка в підготовці даних для LLM, особливо коли йдеться про чистку та структуризацію вхідного тексту.

Справжній прорив у NLP стався з появою моделей глибокого навчання, насамперед трансформерної архітектури, яка стала основою сучасних LLM. Механізм self-attention, характерний для трансформерів, дозволяє моделі враховувати контекст на значних текстових відстанях, що покращує якість аналізу й генерації.

У межах цього дослідження, методи NLP застосовуються для:

– семантичного розбиття навчального матеріалу на логічні блоки;

– виокремлення ключових понять і термінів;

– формування вхідних даних для LLM із урахуванням структури й змісту;

– попереднього очищення тексту від шуму та стилістичних невідповідностей.

Ці завдання не можуть бути якісно вирішені без застосування базових принципів NLP. Таким чином, навіть попри центральну роль LLM у побудові системи, обробка природної мови виступає як необхідна підготовча стадія. Вона забезпечує узгодженість, структурованість і тематичну релевантність даних, з якими працює мовна модель [8].

Місце NLP у системі сучасних технологій ілюструє рисунок 2.1. Він свідчить, що NLP є прикладною галуззю, яка поєднує досягнення машинного навчання та штучного інтелекту. LLM у цій схемі виступають як одна з найсучасніших реалізацій NLP, яка відкриває нові можливості, зокрема для освітніх застосувань.

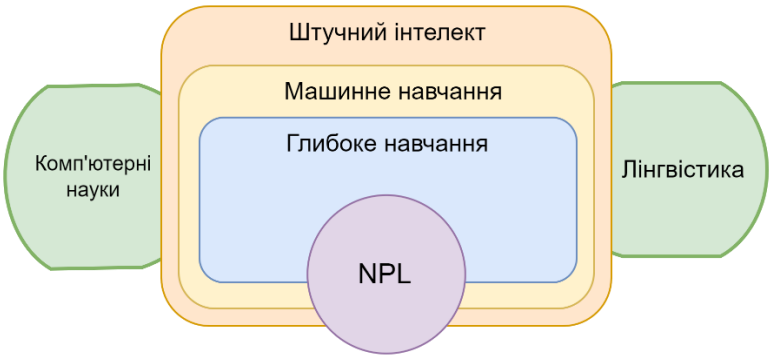


Рисунок 2.1 – Структурно-ієрархічна модель обробки природної мови (NLP) та її взаємозв'язок з суміжними дисциплінами

2.2 Архітектура нейронних мереж типу «Трансформер»

Архітектура трансформера (Transformer) стала ключовим етапом розвитку сучасних моделей обробки природної мови. На відміну від попередніх підходів, таких як рекурентні або seq2seq-мережі, трансформери працюють без послідовної обробки тексту. Вони аналізують усю вхідну послідовність одночасно, що дозволяє краще враховувати як локальні, так і віддалені смислові зв’язки між словами. Саме ця особливість зробила трансформери основою сучасних мовних моделей, включаючи ті, що розглядаються в цій роботі.

Розглянемо ключові компоненти архітектури трансформера, кожен із яких виконує окрему функцію в процесі аналізу та генерації тексту.

Self-Attention (механізм самоуваги) визначає, наскільки важливими є окремі слова в контексті одне одного. Під час обробки речення модель формує для кожного слова три вектори: query (запит), key (ключ) та value (значення). На основі їх взаємодії обчислюються ваги уваги, що дозволяє моделі «зосереджуватися» на тих частинах тексту, які найбільше впливають на значення поточного слова. На практиці це означає, що трансформер здатний правильно врахувати залежності між словами, навіть якщо вони розташовані далеко одне від одного у реченні.

Multi-Head Attention (багатоголова увага) розширює можливості механізму уваги. Замість одного набору ваг (query, key, value) використовується декілька паралельних «голів» уваги. Кожна з них незалежно обчислює власні ваги, аналізуючи послідовність під різними кутами: одна може фокусуватися на граматичних зв’язках, інша – на контекстних або тематичних патернах. Результати всіх голів об’єднуються, що дозволяє отримати багатовимірне та всебічне представлення тексту, значно підвищуючи якість аналізу та роблячи модель стійкою до різних стилів і структур документів.

Positional Encoding (позиційне кодування) – це компонент, який виконує позиційне кодування слів тексту. Оскільки трансформер обробляє всі слова вхідної послідовності одночасно, він не має вбудованого розуміння їх порядку. Щоб передати моделі цю критично важливу інформацію, до векторних представлень слів додаються спеціальні математичні сигнали – позиційні кодування. Найчастіше для цього використовуються синусоїдальні функції, які унікально кодують позицію кожного токена в послідовності. Це дозволяє моделі розуміти як відносне, так і абсолютне розташування слів, що необхідно для коректного аналізу синтаксису та структури речення.

Feed-Forward Network (FFN) – це повнозв’язна нейронна мережа, яка потрібна для нелінійного перетворення та «поглиблення» представлень. Після обробки механізмом уваги отримане для кожного слова контекстуальне представлення далі трансформується через невелику повнозв’язну нейронну мережу (FFN). Цей шар застосовується однаково до всіх позицій у послідовності. Його основним завданням є нелінійне перетворення та подальше «поглиблення» представлень, що дозволяє моделі краще узагальнювати інформацію та формувати більш гнучкі та абстрактні внутрішні представлення тексту.

Взаємодія цих компонентів у багатошаровій архітектурі трансформера наочно представлена на рисунку 2.2.

На схемі можна спостерігати ключові принципи роботи трансформера, а саме:

1) ітеративна обробка: дані послідовно проходять через шари, що складаються з механізмів уваги (Self-Attention, Encoder-Decoder Attention) та повнозв'язних мереж (Feed Forward);

2) залишкові зв'язки: кожен новий шар отримує інформацію не лише від попереднього шару, але й від вихідних даних через блоки «Add & Normalize», що запобігає зникненню градієнтів і дозволяє будувати глибокі мережі;

3) різновиди уваги: схема демонструє наявність як самоуваги (Self-Attention) для аналізу контексту вхідного тексту, так і уваги енкодер-декодер

(Encoder-Decoder Attention), яка є ключовою для завдань генерації (наприклад, створення тестових завдань на основі тексту).

У межах розробки підсистеми генерації тестових запитань трансформерна архітектура використовується як основа для мовної моделі, яка:

– аналізує зміст навчальних матеріалів;

– виявляє смислові фрагменти, які можуть бути основою для запитань;

– формує структуровані та логічно коректні тестові завдання;

– адаптується до різних форматів тексту;

– підтримує роботу з великими документами за рахунок контекстної обробки.

Підсумовуючи, можна виділити ключові переваги трансформерної архітектури:

1) контекстуальна обізнаність: завдяки механізму Self-Attention модель «бачить» весь документ цілісно, що є критичним для формулювання питань, що стосуються взаємозв'язків різних його частин;

2) масштабованість та паралелізм: одночасна обробка всієї послідовності значно пришвидшує навчання та роботу моделі порівняно з послідовними архітектурами, що важливо при роботі з великими обсягами навчальних матеріалів;

3) гнучкість та багатогранність: Multi-Head Attention дозволяє моделі паралельно аналізувати різні аспекти тексту (синтаксис, семантику, логіку), що веде до створення різнотипних запитань (на визначення, на порівняння, на аналіз);

4) збереження структури**:** Positional Encoding гарантує, що модель чутлива до порядку слів та структури речення, а отже, може генерувати граматично правильні та логічно впорядковані формулювання;

5) глибина аналізу: каскад шарів FFN дозволяє послідовно видобувати та перетворювати складні абстракції, необхідні для розуміння суті матеріалу та формування до нього питань;

Саме поєднання механізмів multi-head attention, positional encoding та FFN дозволяє мовній моделі враховувати широкі контекстні залежності й генерувати запитання, які відображають зміст документа, а не лише окремі слова чи фрази [9].

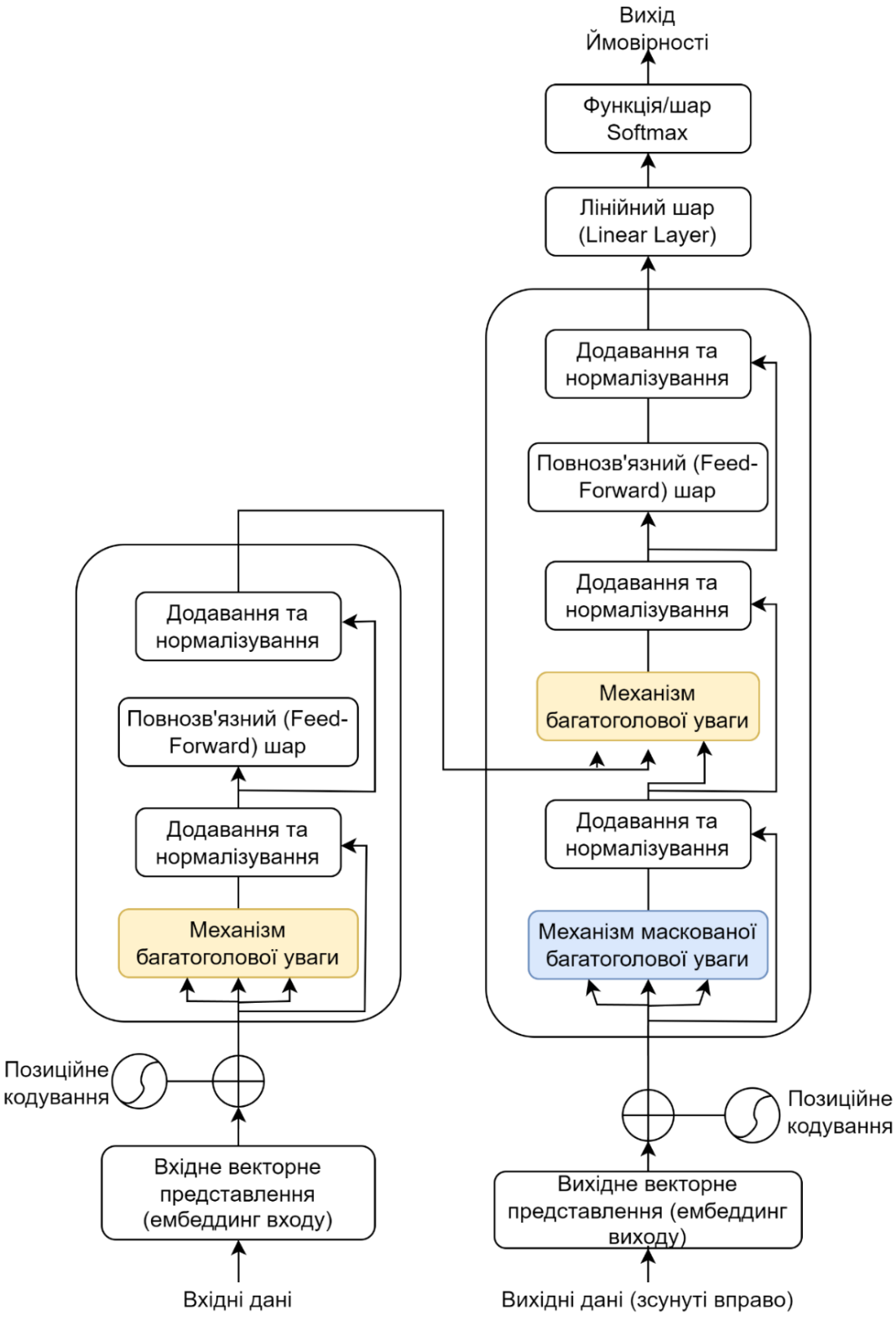


Рисунок 2.2 – Архітектура трансформера

2.3 Типи мовних моделей і принципи генерації

Сучасні мовні моделі базуються на різних архітектурних підходах, кожен з яких має свої особливості у роботі з текстом. Від вибору архітектури залежить, наскільки добре модель «розуміє» контекст, наскільки точно формулює відповіді та чи може взагалі генерувати осмислені речення.

У цьому розділі розглядаються три найпоширеніші типи архітектур трансформерів (transformer architectures): encoder-only, decoder-only та encoder–decoder, а також пояснюється, чому для генерації тестів обрано саме decoder-only.

Encoder-only моделі зосереджені на глибокому опрацюванні й розумінні тексту, без прямої орієнтації на генерацію речень [10]. Під час роботи кожне слово проходить через кілька шарів трансформера, які поступово уточнюють його контекстне представлення. Це дозволяє моделі точно виявляти логіку, структуру, ключові фрагменти, словом, усе, що потрібно для розуміння. Такі моделі широко використовуються для класифікації, пошуку сутностей, аналізу структури документів. Але, проте, самостійно «вигадувати» нові речення вони не можуть – така архітектура просто не передбачає генерації. Архітектуру encoder-only моделі наведено на рисунку 2.3.

Decoder-only архітектури побудовані спеціально для генерації тексту, і саме це визначає їхню ключову роль у сучасних мовних системах. Принцип їхньої роботи базується на autoregressive-підході, коли кожне нове слово створюється з урахуванням усіх попередніх. На кожному кроці модель аналізує вже сформовану частину речення, оцінює найімовірніші варіанти продовження й обирає конкретний токен, що найкраще відповідає контексту [11].

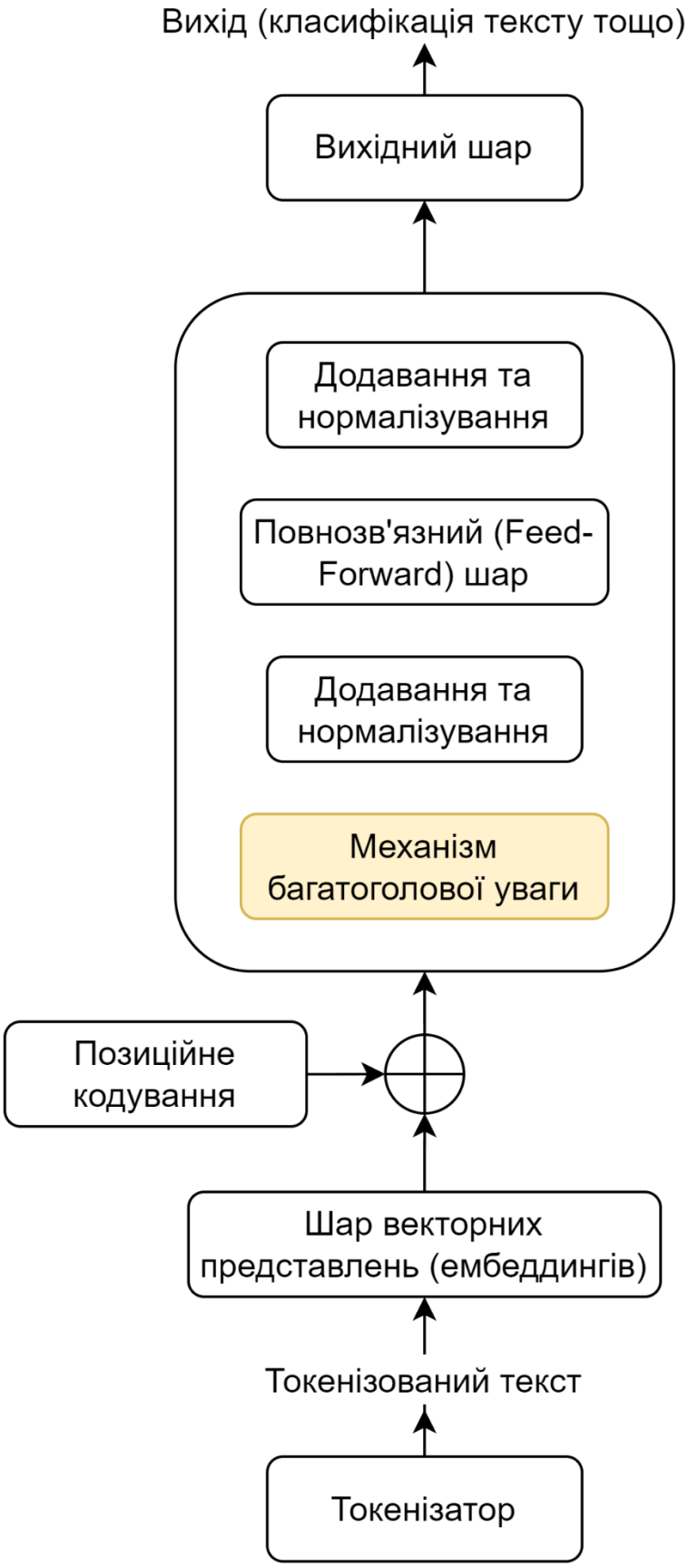


Рисунок 2.3 – Архітектура encoder-only моделі

Архітектуру моделі типу Decoder-only наведено на рисунку 2.4.

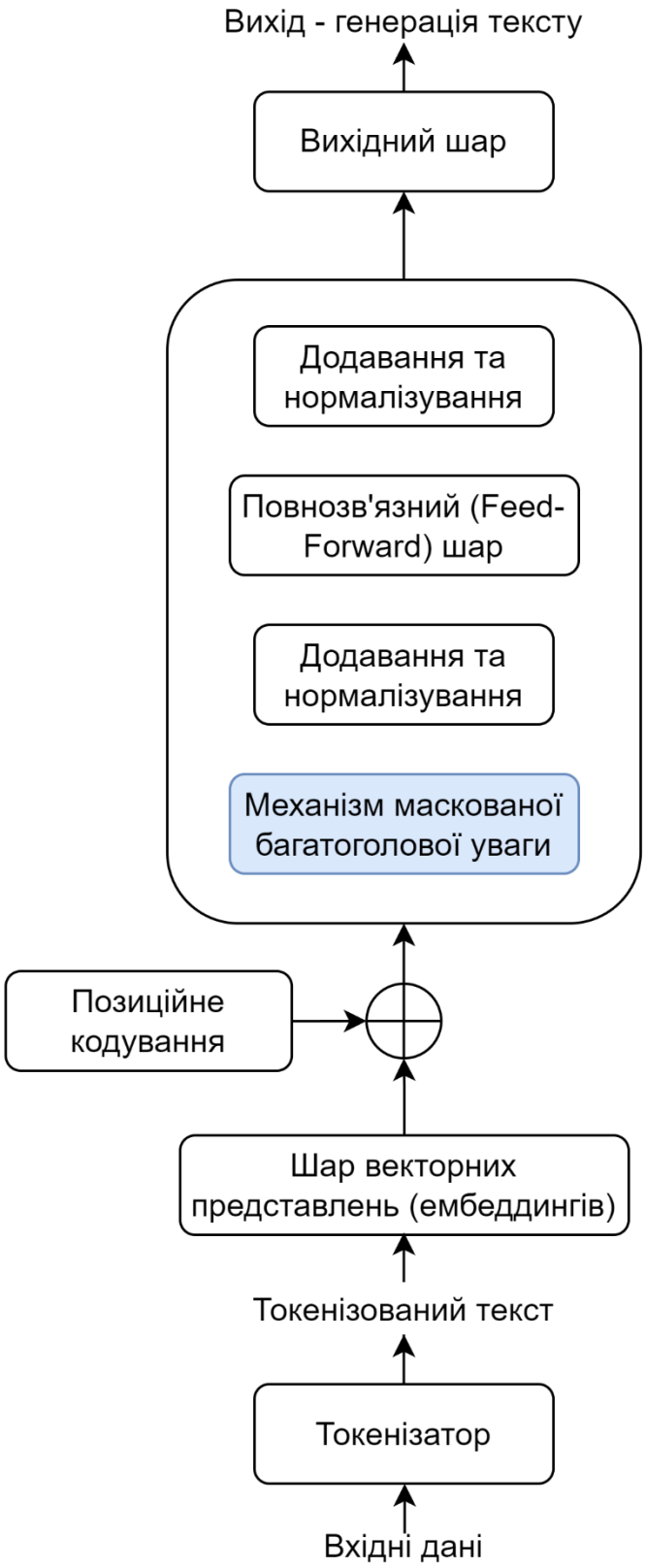


Рисунок 2.4 – Архітектура decoder-only моделі

Encoder-decoder моделі використовують розділення процесу аналізу й генерації на два окремі блоки. Спочатку encoder перетворює вхідний текст на компактне внутрішнє представлення, де зібрана основна інформація й ключові залежності. Далі decoder на основі цього представлення створює новий текст - переклад, узагальнення чи структуровану відповідь [12]. Це дозволяє досягати високої точності перетворення, але така архітектура чутлива до втрати контексту, особливо в довгих документах, де важлива цілісність змісту. Особливо це відчутно під час роботи з великими документами, де важлива цілісність смислових зв’язків. Архітектуру моделі типу encoder-decoder наведено на рисунку 2.5.

Для наочності основні характеристики, сильні сторони та обмеження розглянутих архітектур зведені у порівняльній таблиці 2.1.

Таблиця 2.1 – Порівняльна таблиця архітектур моделі

Архітектура	Характеристика	Сильні сторони	Обмеження
Encoder-only	Створює глибокі контекстні подання, але не генерує текст	Точність аналізу, розуміння структури	Не підходить для генерації запитань
Decoder-only	Генерує текст послідовно, опираючись на весь попередній контекст	Природність формулювань, універсальність, гнучкість	Велике навантаження на пам’ять та GPU
Encoder–decoder	Поєднання аналізу й генерації через два блоки	Висока якість перетворення текстів	Можлива втрата деталей при довгих документах

У задачі генерації тестів критично важливо не лише проаналізувати текст, а й сформулювати запитання, причому логічне, доречне і стилістично «живе». Архітектура типу decoder-only ідеально підходить для цього: вона дозволяє генерувати природні запитання, які відповідають змісту тексту, навіть коли йдеться про великі масиви інформації. Крім того, такий тип моделей добре працює з так званим chunking-поділом довгого тексту на частини, що важливо для обробки лекцій чи конспектів.

Саме тому в цій роботі для вирішення завдання генерування тестів з навчальних дисциплін вибрана трансформерна нейронна мережа з архітектурою decoder-only, як найбільш універсальний і функціональний інструмент для автоматичного створення тестових завдань.

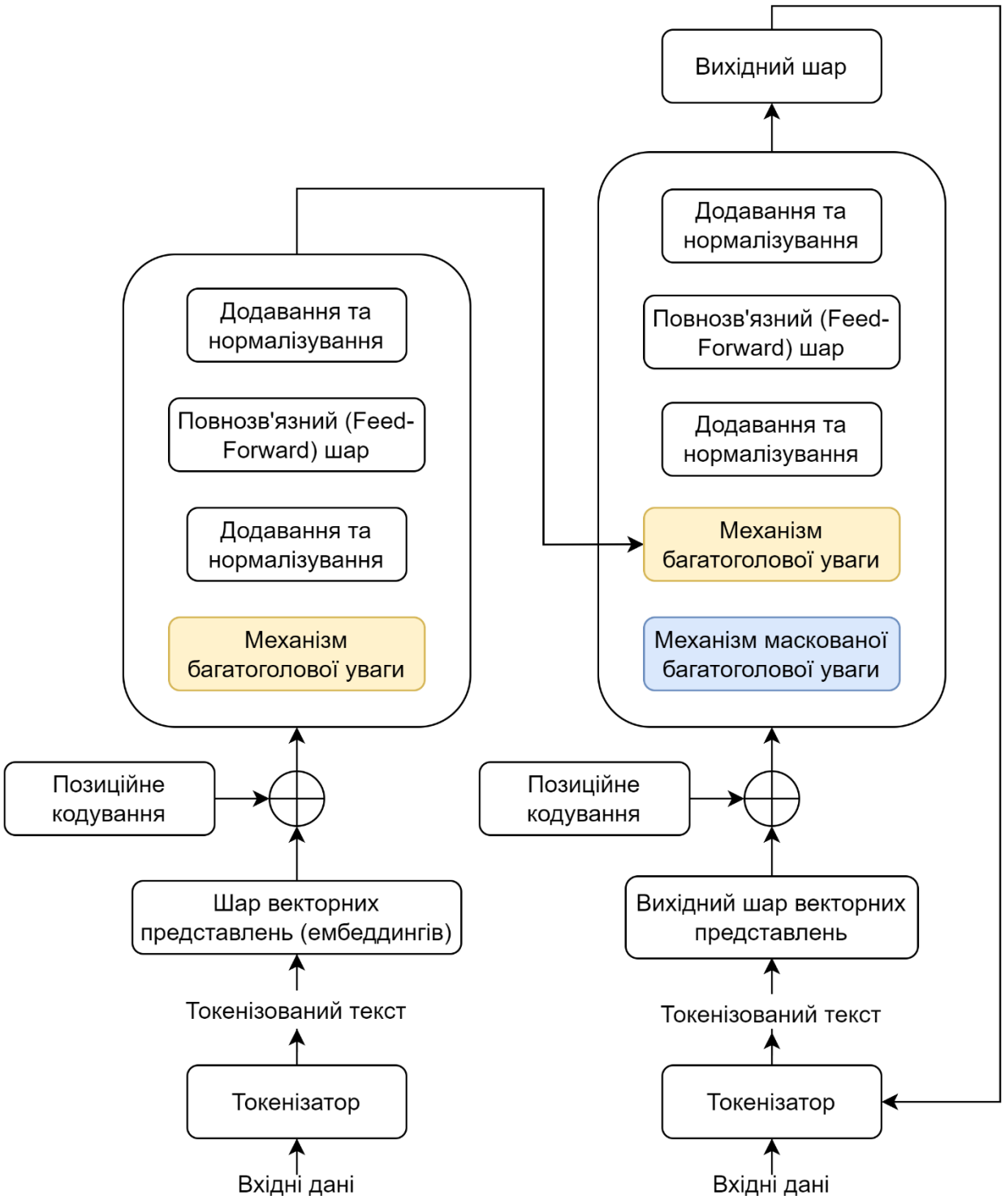


Рисунок 2.5 – Архітектура encoder-decoder моделі

2.4 Механізм побудови запитань

Сучасні підходи до автоматичної генерації тестових запитань (Question Generation, QG) на основі великих мовних моделей часто базуються не на прямій побудові питання з тексту, а на двоетапній процедурі «answer-first» (спочатку відповідь, потім запитання). Цей підхід дозволяє підвищити точність, контекстну релевантність та логічну узгодженість між запитанням і правильною відповіддю. Механізм побудови одного тестового завдання може бути представлений як послідовність взаємопов'язаних етапів [13].

Перший етап «Фрагмент навчального тексту» отримує на вхід семантично завершений фрагмент тексту «чанк», що містить визначену інформаційну одиницю: опис поняття, факту, процесу, причинно-наслідкового зв'язку. Цей контекст обмежує область пошуку для моделі та гарантує, що всі наступні елементи будуть виключно на нього орієнтовані, мінімізуючи ризик вигадування «галюцинацій».

На другому етапі «Генерація відповіді» модель аналізує наданий фрагмент тексту та виокремлює в ньому ключову інформацію, яка може виступити як потенційна відповідь на майбутнє запитання. Це може бути конкретний термін, ім'я, дата, назва процесу, основна характеристика або причинно-наслідковий зв'язок. Ідентифікація відповіді перш ніж питання дозволяє системі чітко сфокусуватися на найважливішій частині контенту. Формально цю передбачену моделлю відповідь можна позначити як â.

На третьому етапі «Генерація запитання» модель, отримавши визначену відповідь â та вихідний контекст, формулює природномовне запитання прямої або непрямої відповіді. Запитання генерується таким чином, щоб воно:

- безпосередньо випливало з контексту;
- однозначно спрямовувало на відповідь â;
- відповідало заданому типу (наприклад, «що?», «коли?», «чому?», «яка основна функція?»).

Четвертий етап «Перевірка узгодженості» – це критичний етап циклічного самоконтролю якості. Його мета – це верифікувати, що згенероване запитання дійсно веде до тієї відповіді, яка була визначена на етапі 2. На практиці це може реалізовуватися шляхом повторного аналізу запитання (іншою моделлю або тим самим підходом повторного аналізу результату) для отримання відповіді ã та її порівняння з вихідною â. Математично ідеалом є виконання умови â = ã, що символізує внутрішню узгодженість і логічну цілісність побудованого завдання. Якщо умова не виконується, запитання може бути відправлене на доопрацювання або відкинуте.

П’ятий етап «Перевірка якості запитання» – це інальний етап перед включенням запитання до тесту. Оцінюється не лише зв’язок з відповіддю, а й загальна якість формулювання: граматична та синтаксична коректність, стилістична адекватність (відповідність академічному стилю), однозначність, а також відповідність заданому рівню складності. Цей етап забезпечує педагогічну придатність згенерованого матеріалу для користувача. Схематично описаний механізм представлений на рисунку 2.6.

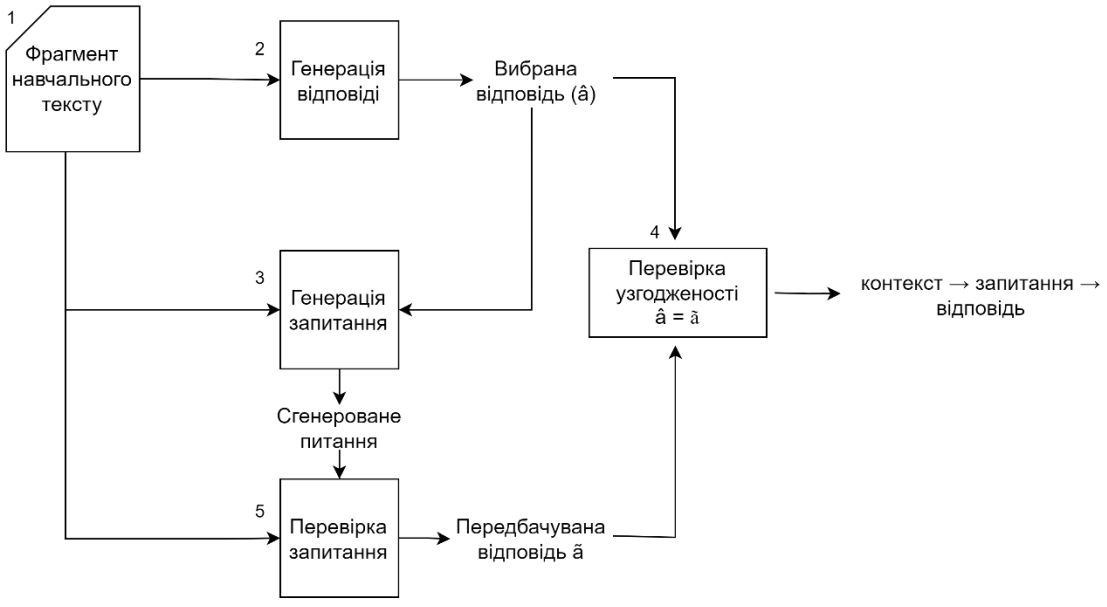


Рисунок 2.6 – Схема механізму побудови тестового запитання за підходом «answer-first»

Таким чином, механізм побудови запитань представляє собою контрольований ітеративний процес, де акцент зміщений з простої генерації тексту на виділення ключового знання та побудову навколо нього перевірочного запитання. Такий підхід суттєво знижує ймовірність появи беззмістовних або нефактологічних запитань.

2.5 Правила конструювання промптів і стратегії формування запитів

Те, як саме сформульовано запит до мовної моделі (промпт), безпосередньо впливає на якість згенерованих тестових запитань. Оскільки генерація відбувається на основі ймовірнісних припущень, чіткість і конкретність інструкції – це не просто рекомендація, а практична необхідність. Вдалий промпт дає змогу звузити простір можливих відповідей, уникнути зайвої креативності і забезпечити відповідність завдання вихідному матеріалу.

Починається все з формулювання чіткої інструкції. Модель має розуміти, що саме від неї очікується: тип запитання, формат, кількість варіантів відповіді, стиль подачі. Розпливчасті або загальні запити, як правило, дають непередбачуваний результат з логічними зсувами, зміненою стилістикою або зайвими варіантами відповіді. Тому структурування промпту – це основа стабільної генерації [14].

Не менш важливим є контекст, тобто файл навчального матеріалу, на основі якого генерується запитання. Якщо модель отримує лише інструкцію, вона опирається на загальні знання, а не на конкретний текст. Це відкриває двері до вигаданих фактів. Навпаки, промпт із контекстом дозволяє точніше відтворити зміст і зменшує ризик «галюцинацій».

Залежно від складності завдання застосовують такі підходи до формування промпту:

− zero-shot підхід, коли модель отримує лише інструкцію без прикладів. Такий підхід дозволяє уникнути впливу сторонніх шаблонів і підтримує гнучкість генерації. Проте під час створення технічних або комплексних тестових запитань zero-shot може бути недостатнім, оскільки модель має занадто широкий вибір можливих формулювань;

− few-shot підхід, який передбачає включення до промпта кількох прикладів правильно сформованих тестових запитань. Це дозволяє моделі зрозуміти стилістику та структуру очікуваного результату. У контексті цієї роботи такий підхід допомагає стабілізувати форму тестових завдань і забезпечити їхню одноманітність.

Окреме місце при формуванні промпту займає явище, відоме як instruction tuning (налаштування інструкції). Це здатність моделі сприймати людську мову як інструкцію, реагувати на фрази типу «створи лише одну правильну відповідь», «оформи у JSON» або «не повторюй запитання». Така поведінка стала можливою завдяки додатковому навчанню сучасних моделей на наборах інструкцій, і її варто використовувати повністю.

У підсумку, добре спроєктований промпт – це не технічна дрібниця, а один із ключових факторів якості. Саме завдяки чітким інструкціям, наданому контексту та правильно обраній стратегії (zero/few-shot), система може стабільно створювати логічні, точні та придатні для навчального використання тести.

2.6 Відмінності генеративних моделей

У галузі великих мовних моделей (LLM), побудованих на архітектурі трансформера, можна виділити генеративні моделі, основним завданням яких є створення нового, когерентного тексту. У контексті задачі автоматичної генерації тестових запитань це розділення має вирішальне значення, оскільки

від обраного типу моделі залежить не лише якість результату, а й сама принципова можливість його досягнення. Таким чином, для генерації тестових запитань придатні саме генеративні моделі.

Серед генеративних LLM існує низка популярних архітектур та конкретних реалізацій, які суттєво відрізняються за продуктивністю, вимогами до ресурсів, якістю генерації та архітектурними особливостями [15]. До таких моделей належать GPT-4 (розробка OpenAI), LLaMA 3 (відкрита модель Meta), Mistral/Mixtral (відкриті ефективні моделі), а також T5/FLAN-T5, які представляють архітектуру encoder-decoder. Для обґрунтованого вибору конкретного інструменту необхідно порівняти їх між собою за ключовими для задачі генерації запитань параметрами. Порівняльний аналіз основних генеративних моделей наведено в таблиці 2.3.

Таблиця 2.3 – Порівняння генеративних моделей між собою

Модель	Архітектура	Сильні сторони	Слабкі сторони	Придатність QG
GPT-4/4оя	Decoder-only	Висока точність, стабільна генерація, мінімум помилок	Високе навантаження на API	Дуже висока
LLaMA 3	Decoder-only	Відкрита модель, хороша якість генерації	Гірше працює з довгими текстами	Висока
Mistral / Mixtral	Decoder-only	Швидкість, компактність	Деколи нестабільні відповіді	Середня–висока
T5 / FLAN-T5	Encoder-decoder	Висока точність у задачах перефразування та резюмування	Менш природна генерація, обмежений контекст	Середня

Після аналізу таблиці стає зрозумілим, що для генерації тестових запитань найкраще підходять моделі, побудовані на архітектурі decoder-only (сімейства моделей, як GPT (OpenAI). Це зумовлено кількома ключовими факторами, які випливають з їхньої архітектури та принципів роботи:

1) здатність до природної генерації тексту: GPT-моделі генерують текст поступово, крок за кроком. Це дозволяє будувати плавні, логічно зв’язані запитання, які не виглядають штучно;

2) підтримка великих контекстів: такі моделі здатні працювати з десятками тисяч токенів, тому легко «оперують» великими фрагментами лекцій, конспектів або методичних текстів;

3) формування повної структури тесту: це означає що моделі GPT можуть одразу сформулювати все, що потрібно: запитання, правдоподібні варіанти відповідей, правильну відповідь і навіть вивести результат у заданому форматі (наприклад, JSON);

4) баланс між універсальністю та точністю: ці моделі поєднують в собі аналітичні й генеративні функції, вони розуміють зміст і водночас здатні створювати нові формулювання, адаптувати стиль, зберігаючи точність.

З огляду на вищезазначене, саме генеративні моделі, побудовані на архітектурі decoder-only, є оптимальним вибором для реалізації підсистеми генерації тестових запитань. Однак для остаточного вибору конкретної моделі серед представників цієї архітектури необхідно застосувати детальну систему критеріїв, враховуючи специфічні вимоги до якості, ефективності та інтеграції в освітню систему, що й буде зроблено в наступному підрозділі.

2.7 Система критеріїв вибору моделі генерації тестових запитань

Розробка підсистеми автоматичної генерації тестових запитань вимагає ретельного обґрунтування вибору базової LLM. Від її характеристик безпосередньо залежать якість, коректність та ефективність всієї системи. Генерація навчальних запитань (Question Generation, QG) є комплексною задачею обробки природної мови, яка поєднує глибинне розуміння контексту, виділення ключових понять, формулювання запитання та побудову

дистракторів. Тому вибір моделі має ґрунтуватися на системі спеціалізованих критеріїв, що відображають вимоги саме до освітніх систем.

У цьому підрозділі наведено та обґрунтовано систему з семи ключових критеріїв для оцінювання мовних моделей. Критерії сформовано з урахуванням специфіки освітнього контексту, де надійність, точність і якість контенту є першочерговими [16].

Сформована система критеріїв включає наступне:

1) критерій якості генерації тексту: згенеровані тексти (питання та варіанти відповідей) повинні бути змістовними, логічно побудованими та стилістично коректними, щоб звучати природно, не містити синтаксичних помилок та однозначно відображати зміст навчального матеріалу. Низька якість генерації безпосередньо призводить до створення неоднозначних або граматично невірних завдань, що суттєво знижує педагогічну цінність тесту та може спричинити непорозуміння у користувача;

2) критерій точності відтворення змісту (фактичної узгодженості): модель повинна мінімізувати явище «галюцинацій» – генерації інформації, відсутньої в наданому контексті, оскільки у тестовому питанні будь-який вигаданий факт є критичним дефектом. Цей критерій безпосередньо визначає надійність та достовірність всієї системи, вимагаючи формування запитань виключно на основі поданого навчального матеріалу;

3) критерій підтримки великого контексту: оскільки навчальні матеріали можуть становити об’ємні тексти, модель має ефективно обробляти довгі документи або інтегруватися з механізмом поділу на фрагменти (чанкінгу) зі збереженням семантичних зв’язків. Недостатня довжина контексту, що обробляється, призводить до втрати важливих деталей та зменшення релевантності згенерованих запитань;

4) критерій стійкості до zero-shot / few-shot інструкцій: модель має чітко інтерпретувати та виконувати інструкції, закладені в промпт, дотримуючись заданого формату, типу запитання, кількості варіантів відповідей і рівня складності, оскільки погане розуміння інструкцій веде до неконсистентних

результатів, що ускладнює їх автоматичну обробку та порушує стабільність функціонування підсистеми;

5) критерій варіативності формулювань: при генерації набору запитань модель має уникати лексичних та структурних повторів, забезпечуючи різноманітність формулювань для подібних концепцій. Низька варіативність знижує якість тесту, створюючи враження шаблонності та сприяючи навчанню користувача конкретній формі запитання, а не його змісту;

6) критерій обчислювальної ефективності: швидкість генерації та ресурсомісткість моделі повинні бути прийнятними для інтеграції в інтерактивну освітню платформу без суттєвих затримок для користувача, оскільки надмірно «важкі» моделі можуть бути непрактичними через високі операційні витрати або неприйнятно тривалий час відгуку;

7) критерій можливості інтеграції: для промислового використання модель має надавати стабільний програмний інтерфейс (API) або мати можливість локального розгортання з чітко визначеними вимогами, що безпосередньо впливає на технічну здійсненність проекту, складність розробки та довгострокові перспективи підтримки системи.

Для вибору оптимальної моделі проведено порівняльний аналіз провідних комерційних та відкритих LLM щодо відповідності вищезазначеним критеріям. Результати представлено у таблиці 2.4.

Таблиця 2.4 – Порівняльна оцінка мовних моделей за ключовими критеріями

Критерій	GPT-3.5 Turbo	GPT-4 / GPT-4o	GPT-4o-mini
Якість генерації	Середня	Відмінна	Висока
Точність змісту	Середня	Висока	Висока
Робота з контекстом	Обмежена	Висока	Задовільна
Варіативність	Середня	Висока	Висока
Ефективність	Висока	Низька	Висока
Інтеграція (API)	Відмінна	Відмінна	Відмінна

На підставі проведеного порівняльного аналізу встановлено, що модель GPT-4o-mini є оптимальним вибором для реалізації підсистеми генерації тестових запитань у межах даної роботи. Вона демонструє збалансоване співвідношення всіх ключових характеристик, а саме високу якість і точність генерації, чудове виконання інструкцій, достатню варіативність, прийнятну роботу з контекстом, а також задовільну обчислювальну ефективність та зручний механізм інтеграції через API. Ця модель позбавлена основних недоліків GPT-3.5 Turbo (низька точність, слабке виконання інструкцій) і є економічно та технічно більш доцільним рішенням порівняно з повноцінними GPT-4/4o, що критично важливо для впровадження в освітню платформу.

2.8 Основні ризики та проблеми використання великих мовних моделей при генерації тестових запитань

Попри високу ефективність LLM у задачах генерації навчального контенту, їх застосування пов’язане з низкою суттєвих ризиків. Ці ризики можуть негативно впливати як на якість тестових завдань, так і на загальну надійність освітнього процесу, що базується на автоматизованій генерації [17]. Розуміння цих обмежень є критично важливим для коректного проєктування підсистеми генерації, що базується на LLM.

2.8.1 Семантичні та фактологічні ризики

Хоч сучасні мовні моделі здатні генерувати граматично правильні й, на перший погляд, логічні запитання, це ще не означає, що вони завжди змістовно точні. Особливо в освітньому середовищі, де фактична достовірність є

ключовою, така «псевдологіка» може стати серйозною проблемою.

Галюцинації під час генерування запитів є найпоширенішою помилкою, у цьому випадку мовна модель може формувати запитання, які виглядають переконливо з точки зору структури та синтаксису, але не мають під собою фактичного підтвердження в навчальному матеріалі. Це може бути результатом, коли модель отримує нечіткий промпт або не має чітко заданого контексту. У тестах такі запитання виглядають як логічні, але стосуються неіснуючих концептів або подій, яких немає у вихідному тексті, матеріалі. Наприклад, модель може «вигадати» поняття або послатися на уявну класифікацію, що не є достовірною інформацією.

Ризики вигаданих фактів, ще один різновид проблеми. цей ризик є окремим різновидом галюцинацій генерації конкретних фактів, що не мають жодного зв’язку з навчальним джерелом. До таких випадків належать, зокрема, вигадані дати, прізвища авторів, назви процесів, хибні цитати тощо. Модель може, наприклад, створити запитання з варіантами відповідей, серед яких «правильна» відповідь не має нічого спільного з оригінальним матеріалом, але звучить переконливо. У технічних або спеціалізованих дисциплінах це може мати серйозні наслідки, навіть дрібна неточність здатна зруйнувати адекватність перевірки знань.

Ризики неправильного фокусу запитання, або іншими словами зміщення фокусу також мають місце. Часом модель помиляється не в самих фактах, а в акцентах, а саме під час генерації LLM іноді обирають як основу для запитання не ключові поняття чи важливі твердження, а другорядні або периферійні елементи тексту. Це проявляється в тому, що запитання фокусуються на малозначимих деталях, які не відображають навчальну мету розділу або модуля. Наприклад, замість формулювання запитання за суттю, вона фокусується на дрібниці, згаданій побіжно.

Окремої уваги потребують ризики, пов'язані з генерацією якісних відволікаючих факторів (distractors) – неправильних варіантів відповідей. Серед типових помилок можна виділити наступне:

– варіанти настільки очевидні, що не вимагають навіть розуміння теми;
– двозначні чи нечіткі формулювання, які лише плутають;
– distractors, які не пов’язані логічно з темою запитання;
– варіанти, які звучать занадто подібно одне до одного.

Такі недоліки суттєво знижують диференційну здатність тесту, роблячи перевірку знань неефективною.

2.8.2 Ризики, які пов’язані з довгими текстами

Мовні моделі, попри свою потужність, мають природне обмеження довжини контексту, з яким вони можуть працювати одночасно. Залежно від архітектури, ця межа зазвичай становить від 4 до 16 тисяч токенів. І хоча цього достатньо для багатьох задач, у випадку з навчальними матеріалами це може бути критично мало. Тексти часто значно довші, у випадку коли багато тексту, спрацьовує механізм поділу, так званий чанкінг. Саме на цьому етапі й виникають особливі ризики.

Обрізання контексту (context truncation) трапляється, коли довжина тексту перевищує допустимий ліміт, модель просто обрізає його, зазвичай з початку або кінця. Якщо у цих обрізаних частинах були важливі визначення чи ключові ідеї, вони просто не потрапляють в обробку. У результаті модель працює з неповною картиною й може сформувати запитання, що базується на фрагменті без достатнього контексту, а це вже пряма дорога до помилки з тестуванням.

Розриви абзаців, алгоритмічний поділ тексту іноді не враховує логіку, речення чи абзац розривається навпіл, і його частини потрапляють у різні чанки. У таких випадках модель отримує лише половину ідеї, а отже, або просто її ігнорує, або робить хибні висновки. Це схоже на спробу зрозуміти речення, прочитавши тільки його кінець.

Зміщення акцентів або іншими словами зсув семантичного ядра трапляється, коли є зміщення фокусу моделі з основної теми абзацу на другорядні згадки, що опинилися ближче до початка чанку. При розриві тексту в довільному місці модель може не «побачити» контекст, який формує значення терміну або пояснення, і натомість побудувати завдання на базі фрагменту, який потрапив до неї випадково.

Втрата важливої інформації при неправильному chunking, на мій взгляд, – це один із найбільш критичних сценаріїв коли логічно пов’язані частини, наприклад, визначення → приклад → виняток, опиняються в різних чанках. Без цієї зв’язності модель не може зрозуміти структуру знання і формує поверхневі або навіть хибні запитання.

Це особливо проблематично для тем, де багато термінології або абстракцій. У таких випадках навіть невелика втрата контексту здатна суттєво вплинути на якість і самого запитання, і загального процесу навчання.

2.8.3 Технічні ризики API-генерації

Інтеграція мовної моделі через API – це зручно і гнучко, але водночас пов’язано з низкою технічних ризиків. На відміну від змістових чи логічних помилок, ці проблеми виникають на рівні самої взаємодії з інтерфейсом або через специфіку конфігураційних параметрів. У випадку автоматичної генерації тестів подібні збої можуть призвести до некоректного форматування, втрати структури або навіть повної непридатності згенерованого результату.

Пошкодження JSON-структури. При цій помильці систем очікують не просто текст, а структурований результат з чітким поділом на запитання, варіанти відповідей, правильну відповідь тощо. Якщо відповідь порушує синтаксис JSON, це унеможливлює автоматичний парсинг. Типові помилки:

– незакриті дужки;

– зайві лапки або коми;

– неправильна вкладеність об’єктів;

– змішування форматованого й звичайного тексту;

– відсутність усіх необхідних елементів (наприклад, кількість запитань менша за очікувану).

Такі помилки потребують додаткової обробки або навіть повної повторної генерації.

Також існує ризик нестабільності відповідей при підвищеному значенні параметра temperature. Параметр temperature у мовних моделях визначає рівень варіативності (креативності) моделі. Вищі значення (0.7–1.0) сприяють більш варіативним відповідям, але водночас збільшують такі ризики:

– розбіжностей у структурі при тих самих запитах;

– втрати логіки у варіантах відповідей;

– «вільної» інтерпретації інструкцій, яка не відповідає джерелу.

Ризик нестабільності робить результат непередбачуваним, навіть якщо вхідні дані не змінюються, модель може повертати зовсім різні формати.

Неоднорідний стиль – це коли запити до моделі надсилаються по частинах, у різних чанках, або з певними паузами, у такому випадку виникає ризик стилістичної розбалансованості. Наприклад:

– одні запитання мають форму твердження, інші, класичних відкритих запитань;

– структура відповідей відрізняється: список проти вбудованого варіанту;

– варіюється рівень формальності, довжина речень, лексика.

Це знижує цілісність тесту, створює враження, ніби запитання формулювали різні люди, без загального стилістичного узгодження.

У контексті цієї роботи стає проблема з кількістю варіантів, іноді навіть попри чіткі інструкції у промптах, модель не надає бажану кількість тестових запитань, а саме:

– формує не ту кількість варіантів (наприклад, два або п’ять замість

чотирьох);

– дублює варіанти;

– не включає правильної відповіді взагалі.

Як було сказано вище, у тестах з автоматизованою перевіркою це критично, система або не зможе знайти правильну відповідь, або працюватиме з помилками.

2.8.4 Педагогічні ризики

Окрім технічних і змістових викликів, автоматизована генерація тестових завдань за допомогою великих мовних моделей породжує низку педагогічних ризиків. Вони здебільшого пов’язані з потенційною невідповідністю згенерованих завдань освітнім цілям, рівню підготовки студентів і навчальній програмі. Така невідповідність може знецінити дидактичну функцію тестування, зробити результати оцінювання хибними, а саму систему, педагогічно неефективною.

Накручування простих запитань, одна з типових проблем, надмірна схильність моделей до формулювання запитань на рівні репродуктивного знання: «що таке», «яка з наведених», «оберіть правильне визначення». Такі запитання не виходять за межі базового запам’ятовування і не стимулюють розвиток критичного або аналітичного мислення. В результаті, навіть якщо студент, або любий інший користувач вірно відповідає на більшість завдань, це ще не означає, що він глибоко засвоїв матеріал.

На мій взгляд особливо гостро це проявляється, коли при генерації не задається бажаний когнітивний рівень (наприклад, за таксономією Блума). У таких випадках модель орієнтується на найбільш очевидні шаблони, що лише поверхнево торкаються теми.

Втрата зв’язку з навчальними матеріалами – це вже інша важлива

педагогічна проблема, невідповідність згенерованих запитань змісту лекцій чи підручників. Без чіткої інструкції або контексту LLM може будувати запитання на загальних уявленнях, отриманих під час тренування, а не на конкретному навчальному матеріалі. В наслідок цього, студент стикається з тестом, який перевіряє знання, що взагалі не викладались у курсі, або перевіряє їх інакше, ніж було представлено на заняттях.

Недостатній контроль за складністю, навіть якщо запитання формально коректне, воно може бути надто легким або, навпаки, непропорційно складним для поточного етапу навчання. Без додаткового налаштування модель не здатна враховувати педагогічну послідовність або адаптувати складність до цільової аудиторії. Це призводить до нерівномірного когнітивного навантаження та у деяких випадках до зниженняя мотивації до навчання.

2.8.5 Етичні ризики

У процесі впровадження автоматизованих систем генерації тестових завдань на основі великих мовних моделей виникає також спектр етичних ризиків, що стосуються відповідальності за якість перевірки знань, коректність поданих запитань та дотримання принципів академічної доброчесності. За своєю природою ці ризики торкаються ключових засад освітнього процесу: справедливості, об’єктивності та надійності оцінювання [18].

Недовіра до результатів перевірки знань від великих мовних моделей не мають глибокого розуміння змісту, їхні відповіді формуються на основі ймовірнісних зв’язків між словами, а не осмислення предмету. Це призводить до кількох критичних моментів:

- запитання можуть не перевіряти ті знання, на які спрямовано курс;
- тестові завдання здатні вводити користувачів в оману, акцентуючи не

на ключовому, а на поверхневому;

– у фокус потрапляє не зміст, а формулювання, що знижує глибину перевірки.

Особливо небезпечними є критичні помилки в ризикових дисциплінах – тих, що стосуються безпеки, здоров'я та права. Помилка в тестовому завданні з медицини чи інженерії – це не просто формальна неточність. Наприклад, хибна інформація щодо протипоказань до препарату або технічних вимог безпеки може мати негативні наслідки. У таких випадках відповідальність лежить не лише на розробнику, а й на закладі, який використовує систему без належної експертизи.

Останнім етичним ризиком, на мій погляд, є мовна інтерференція (language interference) у багатомовному середовищі, де навчальні матеріали можуть бути частково українською, англійською або любою іншую мовою, модель іноді формує запитання з лінгвістичною гібридизацією: україномовне формулювання, англійський термін, наприклад, польські варіанти відповідей. Таке поєднання створює низку проблем:

– порушується стилістична цілісність;

– ускладнюється розуміння для студентів із нерівномірною мовною підготовкою;

– виникає враження недбалої підготовки матеріалів.

Ці фактори впливають на сприйняття тесту як неякісного, помилкового та у деяких випадках непрохідного тестування.

2.9 Стратегічні підходи до мінімізації ризиків та забезпечення якості генерації

Використання великих мовних моделей для створення тестових завдань вимагає не лише генерації, а й ретельного контролю якості. У межах цієї

роботи було реалізовано комплексну систему валідації, яка охоплює технічні, змістовні, лінгвістичні та педагогічні аспекти. Основу складає поетапна стратегія, що включає три ключові кроки:

– generate (генерація**)** – формування запитання та варіантів відповідей на основі структурованого промпта, який явно задає формат вихідних даних (JSON)**,** мову генерації тексту**,** кількість варіантів відповідей **у** питанні, рівень складності тощо;

– verify (перевірка) – перевірка коректності синтаксису JSON, відповідності запитання темі вихідного тексту, змістовної повноти, педагогічної логіки та якості варіантів відповідей;

– refine (доопрацювання) – це доопрацювання згенерованих запитань: заміна слабких варіантів відповідей, редагування стилю, балансування позицій правильної відповіді або повторна генерація.

Такий підхід дає змогу зберігати сталість якості на кожному етапі та адаптувати систему до конкретних освітніх задач.

Щоб уникнути змішування мов (укр./англ. чи будь-якої іншої), необхідно дотримуватися контролю мови та стилістичної однорідності. Для цього система передбачає:

– автоматичне виявлення мови коментарів до навчального матеріалу;

– примусове завдання мови генерації у промптах;

– автоматичне виявлення мови вихідного тексту;

– постобробку з очищенням або редагуванням фрагментів, що містять сторонню лексику.

Це забезпечує стилістичну цілісність тестів і підвищує комфорт сприйняття.

Оскільки вихідний формат даних генерації використовується у вигляді структурованого JSON, система валідації має реалізовувати такі кроки:

– синтаксичну перевірку (всі об’єкти закриті, немає зайвих ком);

– логічну валідацію (наявність усіх ключів – запитання, варіанти, правильна відповідь);

– автоматичне «лікування» JSON, якщо структура пошкоджена частково, наприклад, при нестачі лапок або втраті частини даних.

Для фільтрації низькоякісних запитань застосовується низка евристичних і логічних критеріїв:

– мінімальна довжина запитання;

– відсутність повторів формулювань;

– перевірка на семантичну віддаленість від теми;

– виявлення банальних шаблонів «Що таке...?» без уточнення контексту);

– наявність distractors, які дійсно відволікають, а не очевидно хибні.

Фільтрація відсіює слабкі або формальні запитання, які не несуть навчального навантаження.

Для досягнення стабільності результатів і уникнення небажаної варіативності система використовує такі обмеження температури та інших параметрів генерації:

– низьке значення temperature (0.2–0.4), що зменшує креативність на користь передбачуваності;

– контроль за max_tokens для уникнення перевантаження;

– обмеження top_p, щоб уникнути генерації за рідкісними шаблонами.

Це зменшує непередбачуваність і полегшує стандартизацію результатів.

Перед самою генерацією великі тексти попередньо розбиваються на семантично цілісні фрагменти у процесі інтелектуального поділу тексту (chunking). При цьому враховується параметри:

– логічної структури документа (заголовки, абзаци, тематичні блоки);

– обсягу тексту (щоб уникати обрізання контексту через ліміт токенів);

– важливості інформації (на основі ключових слів і частоти термінів).

Це дозволяє моделі зосередитись на актуальному змісті, не втрачаючи контекст і не створюючи «відірвані» від теми запитання.

Стратегія надлишкової генерації (overgeneration) передбачає, що для компенсації втрат через фільтрацію система генерує на 30–50% більше

запитань, ніж потрібно. Ця надлишкова генерація використовується для:

– відбору найбільш якісних варіантів;

– заміни слабких або пошкоджених запитань;

– формування тестів з різним рівнем складності.

Це дозволяє сформувати якісний фінальний набір навіть при неповній успішності окремих промптів.

Щоб уникнути несвідомої упередженості у виборі позиції правильної відповіді, застосовується балансування позицій правильних відповідей. Для цього підсистема генерування запитів має виконувати таке:

– аналізувати частоту позицій у попередніх запитаннях;

– за потреби виконувати перестановку варіантів;

– повінна намагатися рівномірно розподіляти правильні відповіді по всіх позиціях (A, B, C, D) в переліку запитань.

Такий підхід покращує сприйняття тесту як об’єктивного і знижує ризик випадкового вгадування за шаблоном.

Використання шаблонів генерації відповідно до рівня складності полягає в тому, що запитання формуються на основі темплейтів, узгоджених із когнітивними рівнями:

– рівень 1 (знання): пряме визначення терміну;

– рівень 2 (розуміння): застосування знання до прикладу;

– рівень 3 (аналіз): порівняння або встановлення взаємозв’язків.

Це забезпечує цілеспрямоване формування тестового інструментарію, що дозволяє не лише механічно перевіряти знання, а й диференційовано оцінювати різні рівні їх засвоєння – від відтворення фактів до аналізу та синтезу (відповідно до конкретних навчальних цілей).

У підрозділі 1.1 на рисунку 1.1 представлено узагальнену схему процесу генерації тестових завдань на основі навчального документа.

Після аналізу та розробки рішень щодо мінімізації та усунення проблем генерації запитань, було розроблено удосканалену схему процесу генерації, яка враховує розроблені рішення (див. рисунок 2.7).

У порівнянні з узагальненою схемою з першого розділу, нова схема процесу генерації запитань включає додаткові етапи обробки даних та модулі, що будуть розроблені в рамках покращення системи. Ці модулі забезпечують: стабілізацію JSON-відповідей, фільтрацію дублікатів, компенсацію нестачі запитань, контроль мови, розширену валідацію та балансування варіантів відповідей.

Оновлена схема відображає не лише загальний процес генерації тестів, але й внутрішні механізми інтелектуальної обробки, які забезпечують високу якість кінцевого результату.

Рисунок 2.7 –Удосконалена схема процесу генерації тестових запитань з використанням LLM

3 ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ГЕНЕРАЦІЇ ТЕСТІВ В ОСВІТНІЙ ПЛАТФОРМІ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Опис інформаційної технології генерації тестових запитань

Для реалізації процесу генерації тестових завдань розроблено інформаційна технологія (ІТ) генерації тестів в освітній платформі із використанням мовної моделі великого розміру на основі навчальних матеріалів у форматах PDF та DOCX. Її опис виконано з використанням методології функціонального моделювання IDEF0.

На рисунку 3.1 наведена діаграма IDEF0, яка відображає ІТ та демонструє повний цикл обробки даних – від моменту завантаження користувачем навчального документа до формування фінального тесту, придатного для використання в освітньому процесі.

У межах діаграми показано, як навчальний матеріал проходить етапи попередньої обробки, поділу на логічні фрагменти, формування промптів, генерації тестових запитань за допомогою LLM, а також подальші етапи стабілізації, валідації, компенсації нестачі запитань, фільтрації дублікатів і балансування правильних відповідей. Функціонування кожного етапу підтримується відповідними механізмами, зокрема програмним забезпеченням підсистеми генерації запитань, мовною моделлю, обчислювальними ресурсами та керуючими правилами.

IDEF0 включає дев’ять основних функціональних етапів, які забезпечують керований, ітеративний та надійний процес формування якісних тестових завдань.

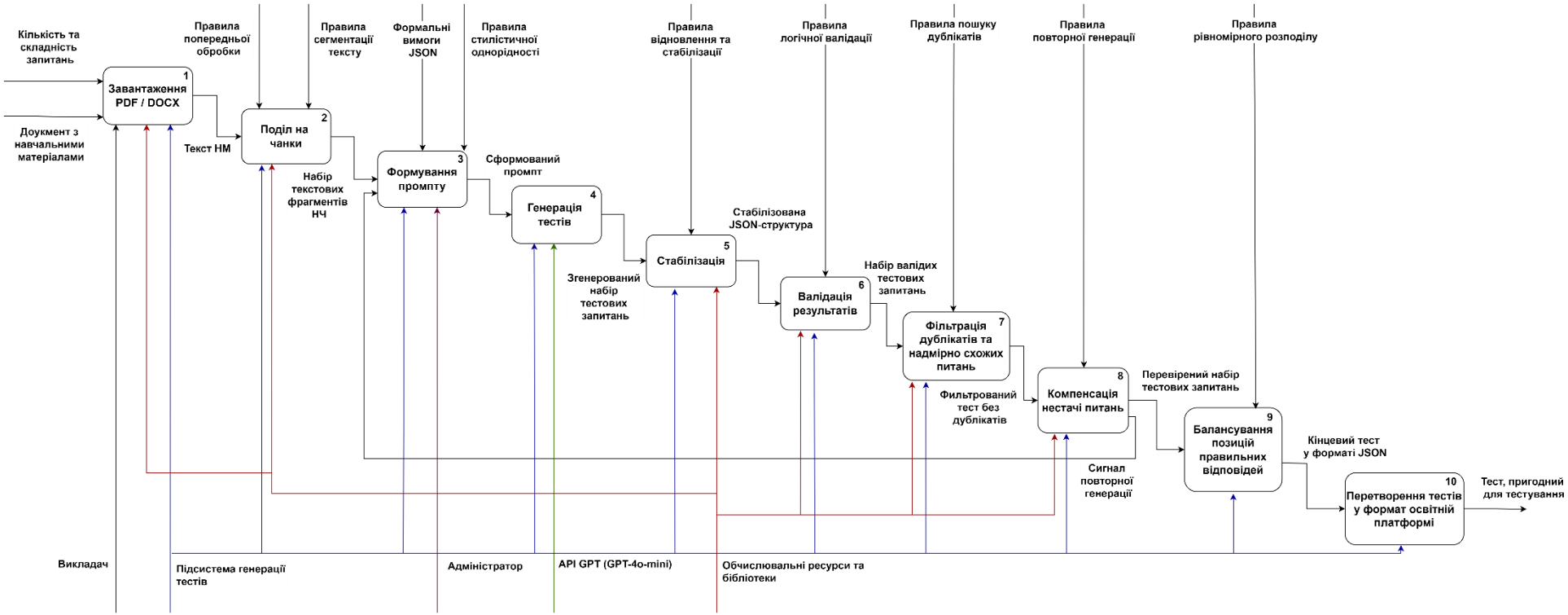


Рисунок 3.1 – IDEF0-діаграма ІТ генерації тестових запитань

IDEF0-діаграма інформаційної технології наочно визначає її етапи, механізми керування, та контроллери.

Етап 1 «Завантаження PDF / DOCX» відповідає за введення навчального матеріалу до системи. Вхідними даними цього етапу є документ із навчальним матеріалом, який надається користувачем або викладачем у форматах PDF або DOCX. Механізмами виконання етапу виступають викладач та підсистема генерування тестів. Вихідними даними є документ, доступний для подальшої обробки в межах підсистеми.

Етап 2 «Поділ на чанки» виконує сегментацію обробленого тексту навчального матеріалу на логічно завершені фрагменти – «чанки». Вхідними даними є текст навчального матеріалу, отриманий після попередньої обробки. Керуючими елементами виступають правила сегментації тексту та правила попередної обробки. Механізмами етапу є програмне забезпечення (ПЗ) підсистеми, обчислювальні ресурси та зовнішні бібліотеки. Вихідними даними є набір текстових фрагментів, підготовлених для налаштування промптів.

Етап 3 «Налаштування промпту» відповідає за формування структурованого запиту до мовної моделі. Вхідними даними етапу є набір текстових фрагментів навчального матеріалу. Керуючими елементами є формальні вимоги до JSON-структури, правила стилістичної та мовної однорідності. Механізмами виступають ПЗ підсистеми та адміністратор, який визначає загальні параметри генерації, та у випадку пошкоджень, відновлює промпт. Вихідним результатом є згенерований промпт, підготовлений для передачі до мовної моделі.

Етап 4 «Генерація тестів» виконує безпосередню генерацію тестових запитань за допомогою LLM. Вхідними даними є сформований промпт. Механізмом виконання є API мовної моделі GPT та ПЗ підсистеми. Вихідними даними є згенерований набір тестових запитань у форматі JSON.

Етап 5 «Стабілізація JSON-відповідей» відповідає за перевірку та відновлення структури JSON-відповіді, отриманої від LLM. Вхідними даними

для нього є згенерований набір тестових запитань. Керуючими елементами виступають правила відновлення та стабілізації JSON-структури. Механізмами є обчислювальні ресурси та бібліотеки. Вихідними даними є стабілізований текст із валідною структурою.

Етап 6 «Валідація результатів» виконує логічну та структурну перевірку тестових запитань. Вхідними даними є стабілізований JSON із тестовими запитаннями. Керуючими елементами є правила логічної валідації. Механізмами є ПЗ підсистеми генерації тестів, обчислювальні ресурси та бібліотеки. Вихідними даними є набір валідних тестових запитань.

Етап 7 «Фільтрація дублікатів та надмірно схожих запитань» забезпечує усунення повторюваних або семантично подібних запитань. Вхідними даними є набір валідних тестових запитань. Керуючими елементами є правила пошуку дублікатів. Механізмом виконання є ПЗ підсистеми генерації тестів, обчислювальні ресурси та бібліотеки. Вихідним результатом є профільтрований тест без дублікатів.

На етапі 8 «Компенсація нестачі запитань» виконується контроль кількості згенерованих тестових завдань. У разі виявлення нестачі запитань формується керуючий сигнал повторної генерації, який передається до етапу налаштування промпту. Керуючими елементами є правила повторної генерації. Механізмами тут виступають ПЗ підсистеми генерації тестів, обчислювальні ресурси та бібліотеки. Вихідними даними є перевірений набір тестових запитань та сигнал повторної генерації.

Етап 9 «Балансування позицій правильних відповідей» є завершальним етапом, який забезпечує рівномірний розподіл правильних відповідей у варіантах тестових запитань. Вхідними даними етапу є перевірений набір тестових запитань. Керуючими елементами є правила рівномірного розподілу балансування тестових выдповыдей. Механізмом виконання є ПЗ підсистеми генерації тестів, обчислювальні ресурси та бібліотеки. Вихідними даними є текст, пригодний для тествування.

3.2 Механізми покращення генерації тестових запитань

З огляду на підрозділ 2.8 та на рисунок 2.5, у межах дипломної роботи було реалізовано низку додаткових механізмів (покращень), які суттєво розширюють базову схему процесу генерації тестових запитань з використанням LLM та роблять процес генерації більш ефективним, ніж при базової схемі процесу генерації тестових запитань (див. рисунок 1.1).

До ключових покращень, відповідно до удосконаленої схеми процесу генерації, належать:

– стабілізація та відновлення JSON-відповідей мовної моделі;

– алгоритм компенсації нестачі запитань (over-generation) через надгенерацію та повторні запити;

– фільтрація дублікатів та надмірно схожих запитань;

– механізм контролю мови та стилістичної однорідності;

– розширена валідація структури, логіки та змісту запитань;

– балансування позицій правильної відповіді для уникнення статистичних перекосів.

У цьому розділі детально описано кожне з запропонованих та реалізованих і наведено відповідні схеми алгорітмів, котрі описують принцип роботи коду та розроблені у вигляді UML-діаграм діяльності. Безпосередньо програмний код розроблених алгоритмів наведено в додатку А.

Почнемо з покращення стабілізації та відновлення JSON-відповідей. У цілому однією з найбільш критичних проблем під час взаємодії з великими мовними моделями є нестабільність структури відповідей. Незважаючи на вимогу повернення даних у форматі JSON, модель інколи генерує частково пошкоджений або синтаксично некоректний результат. Це може призвести до неможливості розпарсити відповідь та до повної зупинки процесу генерації тестових запитань.

Для вирішення цієї проблеми у системі було реалізовано багаторівневий

механізм стабілізації JSON-відповідей. На першому етапі здійснюється спроба стандартного парсингу. У разі виникнення помилки активується модуль реконструкції, який вилучає сторонні символи та відновлює структуру об’єкта. Якщо це не дає результату, система повторно звертається до мовної моделі з тим самим запитом. Такий підхід гарантує безперервність процесу та значно підвищує стійкість до нерегулярної поведінки LLM.

Принцип роботи механізму стабілізації наведено на рисунку 3.2, де показано етапи перевірки JSON, його відновлення та повторного отримання відповіді.

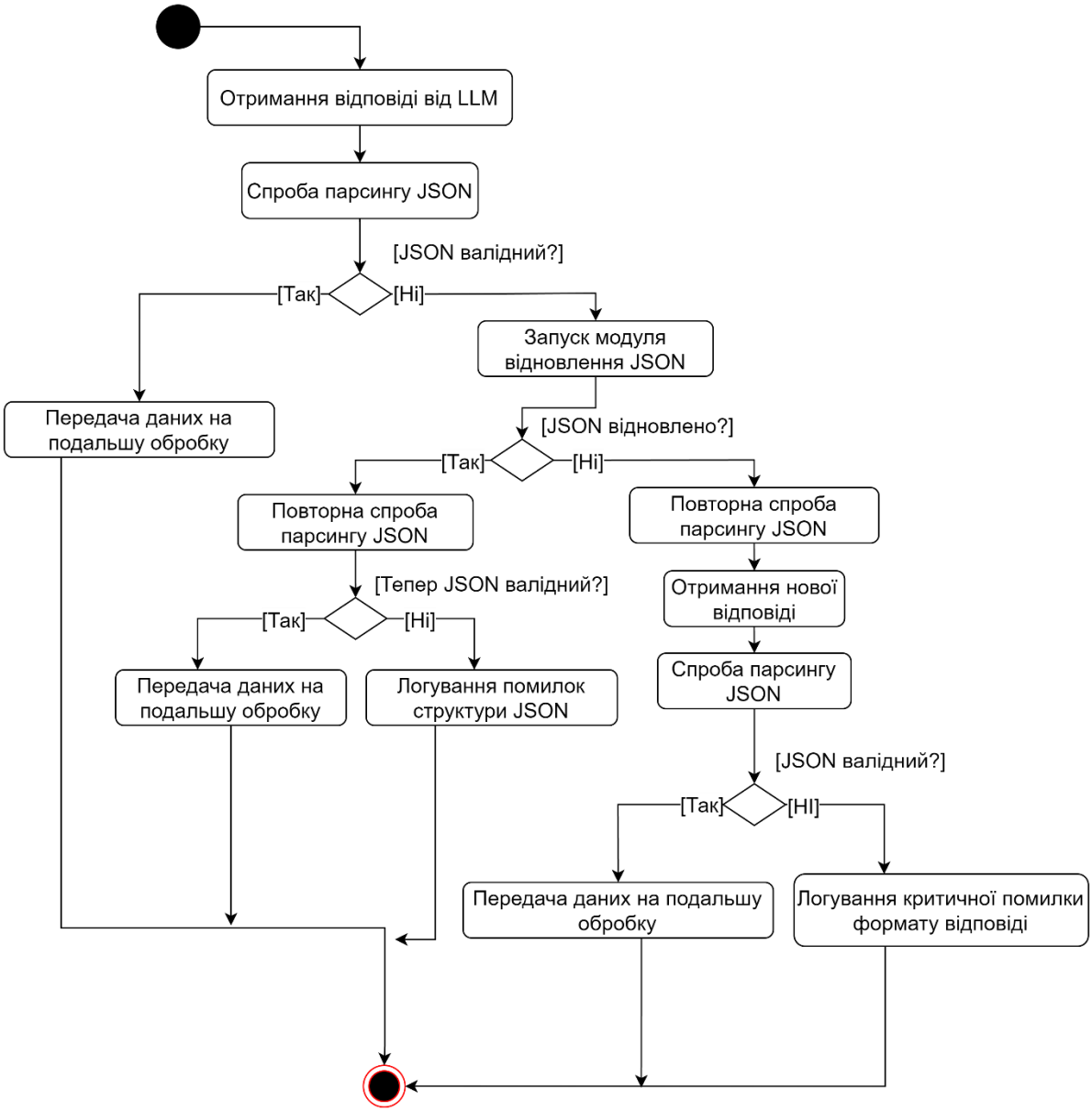


Рисунок 3.2 – Алгоритм покращення стабілізації та відновлення JSON-відповідей

Це покращення забезпечує надійність у всіх сценаріях і усуває одну з найпоширеніших проблем взаємодії з мовними моделями. Реалізація у вигляді коду, знаходиться у додатку А.

Тепер розглянемо покращення компенсації нестачі запитань та стратегію надгенерації запитань. Під час генерації тестових завдань, модель часто повертає меншу кількість запитань, ніж очікується, або формує запитання з порушенням структури. Це характерна вада LLM, що виникає як наслідок ненадійної відповідності вказаній кількості елементів у промпті.

Щоб уникнути цього, реалізовано механізм надгенерації (over-generation): модель свідомо генерує більшу кількість запитань, ніж потрібно. Після цього виконується їхня валідація та фільтрація, а з отриманого набору формується необхідний обсяг коректних завдань. Якщо після фільтрації запитань все ще недостатньо, ініціюється повторний запит для даного текстового фрагмента.

Такий підхід гарантує, що навіть за умов нестабільних відповідей LLM система сформує повноцінний набір запитань.

Принцип роботи механізму компенсації нестачі запитань наведено на рисунку 3.3, де показано взаємодію між надгенерацією, валідацією та повторними викликами моделі.

Це покращення забезпечує стабільність під час роботи з великими документами та дозволяє отримувати достатній обсяг якісних запитань незалежно від поведінки моделі. Реалізація у вигляді коду, знаходиться у додатку А.

Реалізація покращення усунення дублювання та надмірної схожості запитань у LLM виконана у зв’язку з тім, що часто формуються запитання, які відрізняються лише незначними синтаксичними змінами, але залишаються однаковими за змістом. Це негативно впливає на якість тесту, роблячи його менш інформативним і менш корисним у навчальному процесі.

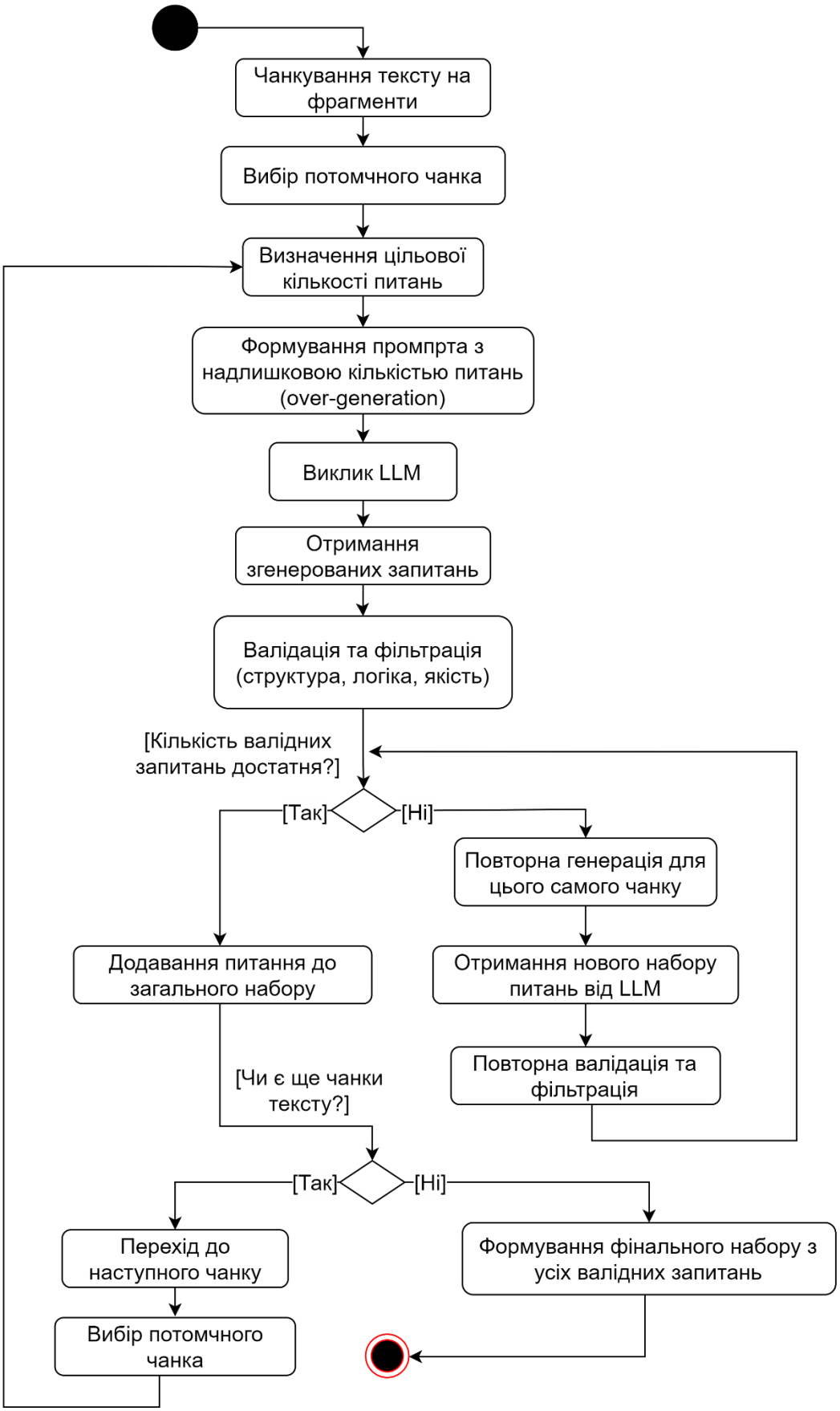


Рисунок 3.3 – Схема алгоритму покращення компенсації нестачі запитань та стратегії надгенерації запитань

Для розв’язання цієї проблеми у системі реалізовано механізм фільтрації дублікатів, який працює на декількох рівнях. Спочатку здійснюється точне порівняння – запитання з ідентичним текстом не додаються до підсумкового набору. Далі виконується часткове порівняння: аналізується початок формулювання та можливість вкладеності одного запитання в інше. Якщо запитання надмірно схоже на вже додане, воно відхиляється. Завдяки цьому механізму система формує набір дійсно унікальних запитань, що суттєво підвищує змістовну різноманітність тесту.

Схема роботи механізму фільтрації дублікатів наведена на рисунку 3.4, де відображено послідовність перевірок текстових збігів. Це покращення дозволяє уникнути повторення однакової інформації та забезпечує максимально рівномірне покриття навчального матеріалу.

Код програмної реалізації механізму фільтрації дублікатів наведено в додатку А.

Механізм покращення мовної стабілізації та контроль стилістичної однорідності було запропоновано тому, що через специфіку роботи LLM можлива поява змішаних мов у згенерованих запитаннях, наприклад, українського тексту з англійськими. Такі змішані конструкції є неприйнятними у тестуванні та суттєво знижують якість матеріалу.

У підсистемі генерації тестів передбачено механізм мовної стабілізації. На першому етапі визначається мова вихідного документа. Далі у системному промпті жорстко фіксується мова, якою має бути сформовано відповідь. Після генерації виконується додаткова постобробка, під час якої вилучаються конструкції, що містять фрагменти іншими мовами. Також перевіряється стилістична узгодженість запитань, щоб усі вони відповідали одному формату.

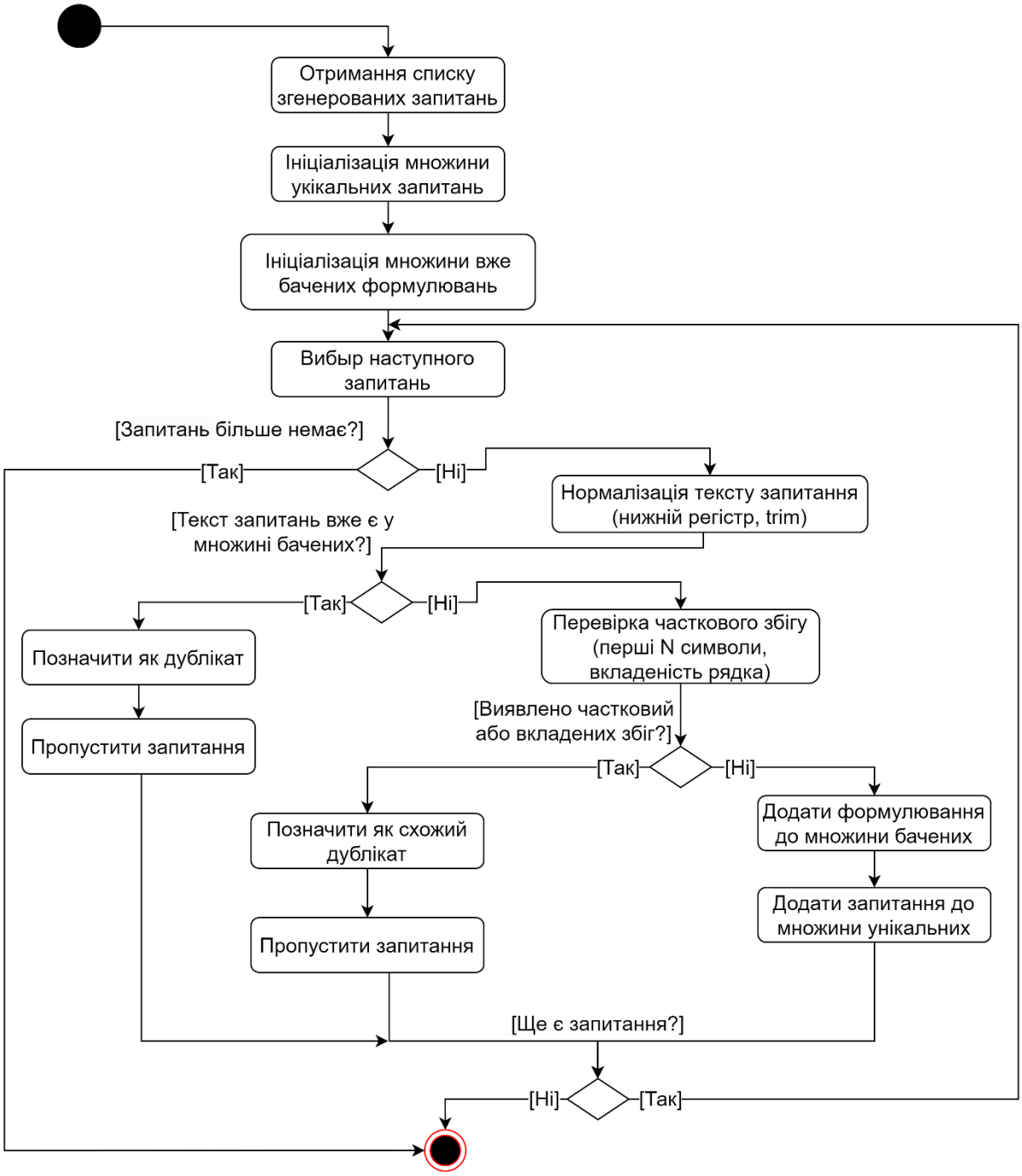


Рисунок 3.4 – Схема алгоритму усунення дублювання та надмірної схожості запитань

На рисунку 3.5 наведено схему алгоритма покращення мовної стабілізації. Вона пояснює, як система визначає мову тексту, передає її у промпт, а також здійснює очищення змішаних формулювань у постобробці.

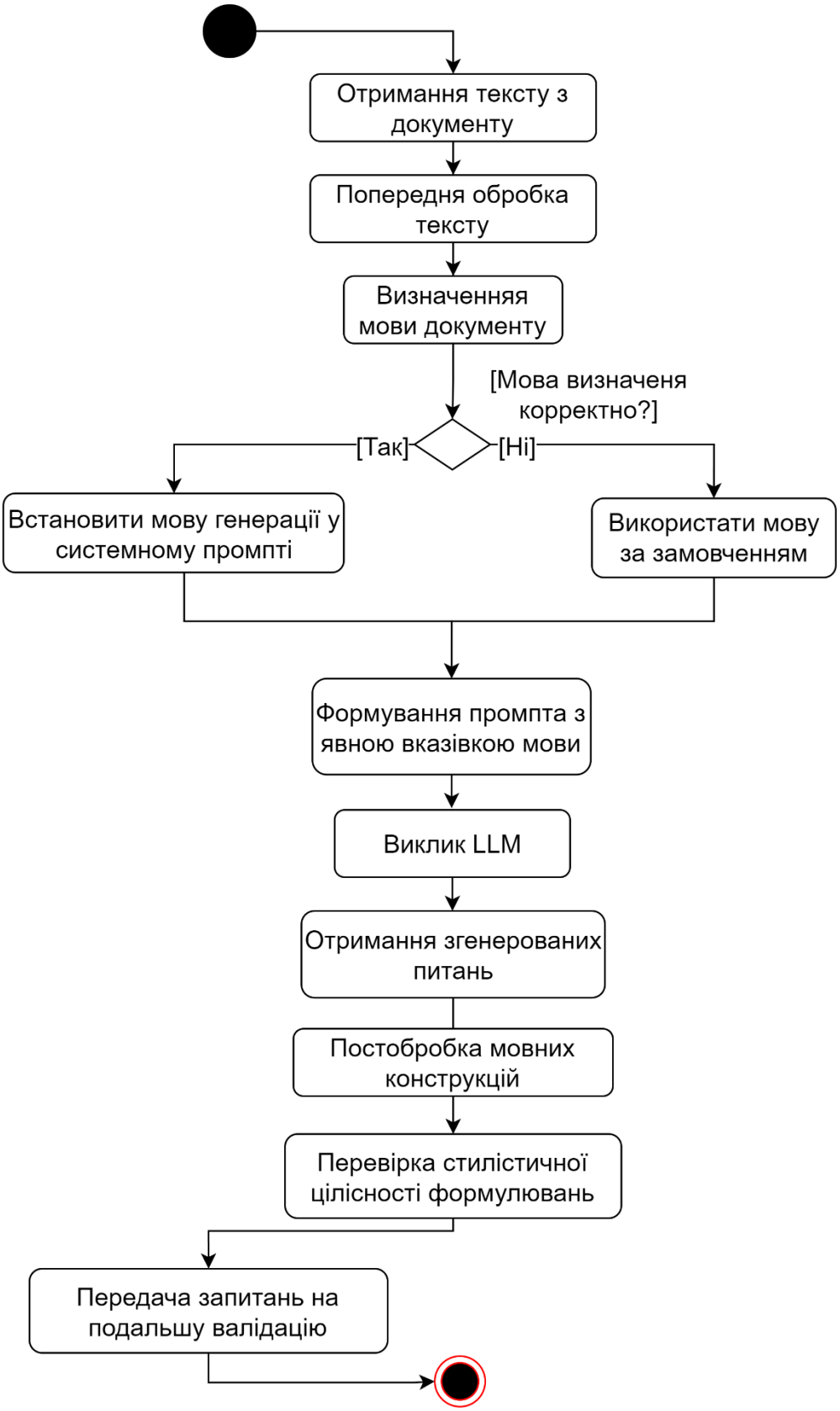


Рисунок 3.5 – Схема алгоритму мовної стабілізації та контролю стилістичної однорідності

Завдяки механізму мовної стабілізації тестові завдання стають стилістично єдиними та придатними для використання у навчальному процесі. Реалізація цього механізму у вигляді коду, наведена у додатку А.

Розширена валідація структури та логіки запитань, згенерованих мовною моделлю необхідна у зв’язку з тім, що часто трапляються структурні помилки, такі як відсутність одного з ключових полів, некоректна кількість варіантів відповіді або занадто короткі, беззмістовні формулювання. Такі запитання не можуть бути використані у тестуванні.

Підсистеми генерації тестів містить спеціальний модуль розширеної валідації, який проводить поетапну перевірку кожного запитання. На першому рівні перевіряється структура: наявність полів question, options, correct_answer. Далі перевіряється логічний зміст запитання: довжина, граматична коректність, адекватність формулювання. Обов’язковою умовою є наявність рівно чотирьох варіантів відповіді. Також система видаляє запитання, що містять мета-інформацію про саму модель чи систему.

На рисунку 3.6 наведено схему алгоритму роботи модуля валідації, яка включає структурні та логічні перевірки. Це покращення дозволяє отримувати лише якісні та коректно сформовані тестові запитання. Реалізація у вигляді коду, знаходиться у додатку А.

У підсистеми генерації тестів реалізовано запропонований механізм балансування позицій правильної відповіді. Після формування всіх запитань підраховується частота появи правильної відповіді на кожній із чотирьох позицій. Якщо фіксується перекос у бік певної позиції, частина варіантів відповідей переставляється так, щоб розподіл був максимально рівномірним. Наприкінці формується остаточно збалансований набір запитань.

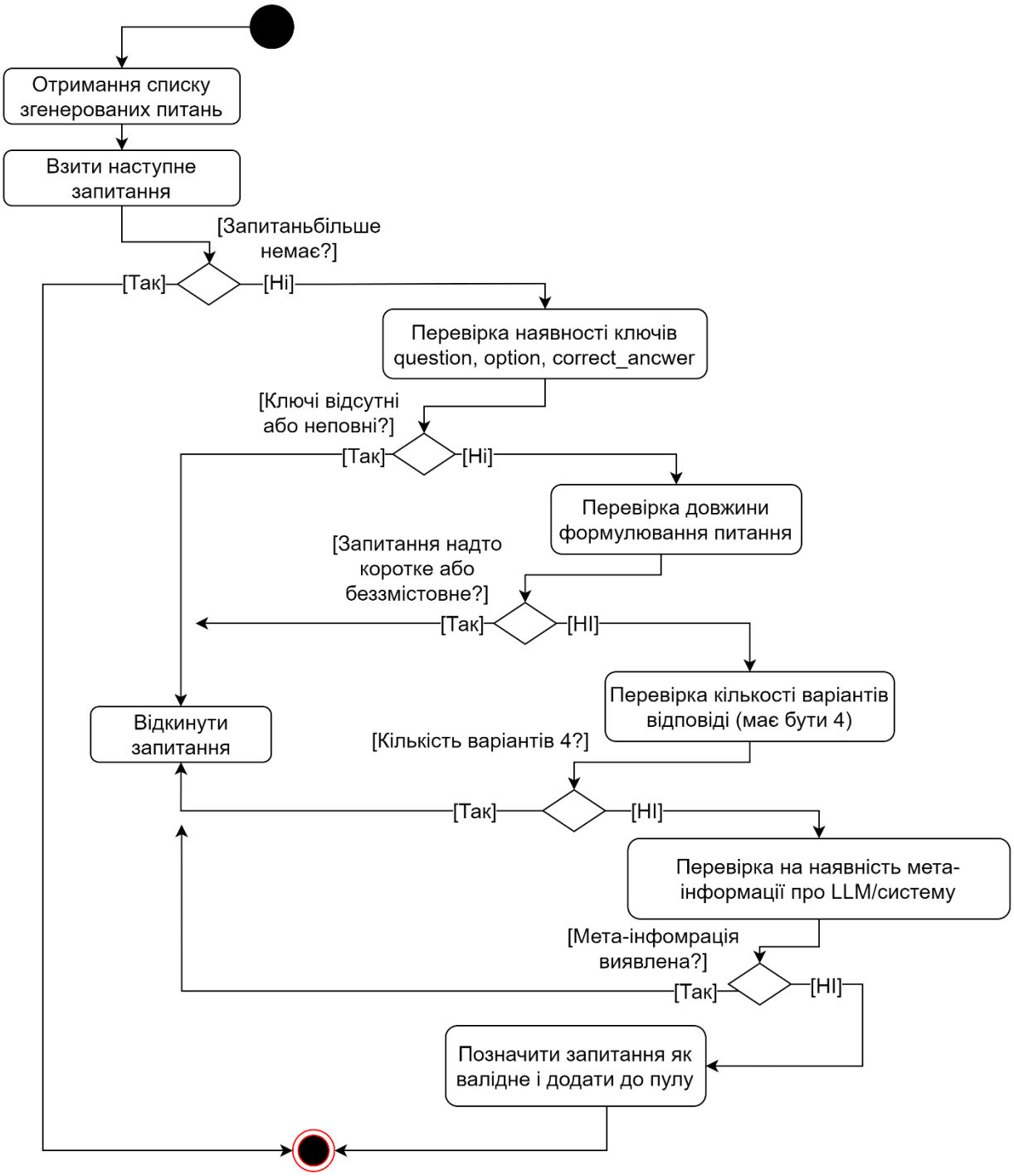


Рисунок 3.6 – Схема алгоритму модуль розширеної валідації

На рисунку 3.7 подано схему логіки цього процесу балансування позицій правильної відповіді, включно з аналізом позицій і процедурою їх вирівнювання.

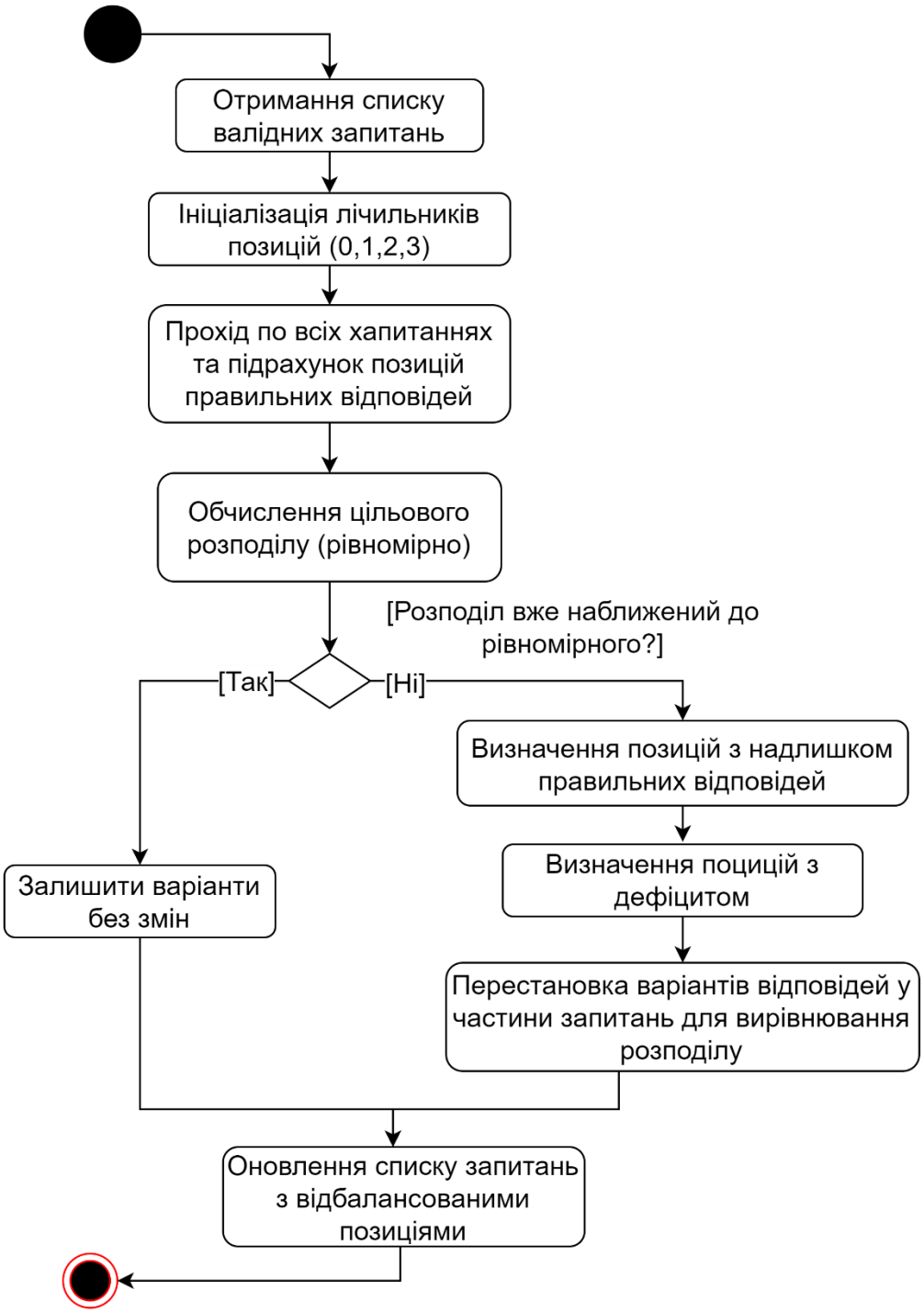


Рисунок 3.7 – Схема алгоритму балансування позицій правильної відповіді

Таке покращення підвищує педагогічну справедливість тесту та унеможливлює вгадування відповідей за допомогою статистичних патернів. Реалізація алгоритму балансування позицій правильної відповіді у вигляді коду наведено у додатку А.

3.3 Інтеграція з API мовної моделі

Ключовим механізмом інтеграції мовних моделей є API, який забезпечує передачу підготовлених текстових фрагментів «чанків , формування промптів, отримання відповідей у структурованому форматі та подальшу їх обробку.

Взаємодія підсистеми генерації тестів из LLM відбувається через стандартний HTTP-запит до API мовної моделі (у проєкті використовується клієнтська бібліотека, яка інкапсулює мережеву частину). Архітектурно цей процес складається з чотирех етапів.

На першому єтапі виконується процес формування структури запиту до моделі. В цей момент підсистема готує логічно цілісний об’єкт запиту, що містить усі необхідні дані для роботи мовної моделі. Процес формування структури запиту поділяється на рольові повідомлення, параметри генерації, формування інструкцій та передачу текстового фрагменту.

Рольові повідомлення (system + user) – це набір інструкцій, що визначає контекст генерації. Системне повідомлення (system message), задає правила поведінки моделі, опис обмежень і загальну стратегію відповіді. Повідомлення користувача (user message), містить конкретний фрагмент навчального тексту, на основі якого мають формуватися тестові запитання.

Параметри генерації (temperature, max_tokens, top_p) керують поведінкою моделі, а саме:

– temperature обмежує рівень випадковості, забезпечуючи стабільність відповідей;

– max_tokens визначає максимальний обсяг згенерованого тексту;

– top_p дозволяє контролювати фокус моделі на найбільш ймовірних варіантах. Таким чином, правильне налаштування цих параметрів забезпечує кращу структурованість та передбачуваність відповідей.

Формування інструкції у вигляді промпта – це процес об’єднання текстових правил, шаблону JSON-структури, вимог до формату та кількості

тестових завдань у єдиний інструкційний блок «промпт», який чітко визначає очікуваний результат.

Останнім етапом у формуванні запиту є передача текстового фрагменту (чанку). До запиту додається виділений фрагмент документа «чанк». Це дозволяє моделі працювати строго в межах наданого контенту та уникати вигадування інформації, якої немає у вихідному матеріалі.

Таким чином формується запит, який містить як контекст, так і формальні правила, за якими має формуватися відповідь моделі.

На другому етапі сформований запит безпосередньо передається до API мовної моделі. Виклик здійснюється через відповідний метод бібліотеки, який:

– забезпечує доставку всіх рольових повідомлень і параметрів генерації;

– надсилає інструкцію у потрібному форматі;

– запускає механізм генерації відповіді на сервері моделі.

Особливо важливим є параметр response_format={"type": "json_object"}, який зобов’язує модель повернути відповідь у вигляді коректного JSON-об’єкта. Це мінімізує ризики синтаксичних помилок та полегшує подальшу обробку результату.

Третій етап взаємодії представляє з себе отримання відповіді від моделі, яку користувач отримує після обробки запиту, модель повертає відповідь, що містить:

– службові дані API;

– основний текст відповіді моделі у структурованому вигляді.

Система з цього результату виокремлює саме JSON-фрагмент, у якому містяться сформовані тестові запитання. Зазвичай цей фрагмент розміщується у полі згенерованої відповіді. На цьому етапі важливо відкинути допоміжні елементи й залишити тільки ту частину, що повністю відповідає вимогам промпта.

Останній четвертий етап взаємодії з API моделі – це подальша обробка після отримання відповіді, вона складається з таких дій:

1) парсинг JSON – це коли система намагається інтерпретувати

отриманий текст як JSON-об’єкт. Якщо структура коректна, дані одразу переходять до наступного етапу;

2) стабілізація структури при помилках. Вона відбувається у разі пошкодженої або неповної JSON-відповіді. При цьому запускаються спеціальні механізми реконструкції. Вони виправляють помилки форматування, видаляють зайві символи або відновлюють порушені частини структури;

3) валідація та фільтрація – система перевіряє отримані дані за низкою таких критеріїв;

– наявність усіх необхідних полів (питання, варіанти, правильна відповідь);

– коректність структури масивів;

– релевантність змісту;

– відсутність дублікатів, надлишкових чи некоректних формулювань;

– дані, що не відповідають вимогам, відсіюються.

У результаті формується набір якісних тестових запитань, готових до відображення у формі для тестування. Таким чином API-інтеграція виступає ключовим каналом взаємодії між системою та мовною моделлю, забезпечуючи контрольовану та стабільну генерацію тестових завдань.

3.4 Розширений розбір розробленої структури промпту

Промпт – це не просто інструкція для моделі, а складний логічний шаблон, який керує всією поведінкою LLM. У розробленій системі він складається з декількох функціональних компонентів, системна інструкція, користувацький інструктивний блок, блок контролю якості та формат відповіді.

Системна інструкція (system message) – це частина промпта, в якої визначені глобальні правила роботи моделі. Вона містить:

– чітку заборону вигадувати інформацію, що особливо важливо у навчальних матеріалах, де точність змісту критична;

– вимогу формувати запитання виключно на основі наданого фрагмента, модель не має права додавати сюжети, яких у тексті не було;

– вказівку створювати відповідь у форматі JSON, що забезпечує структурованість даних;

– встановлення мови (української) та стилю формулювань, щоб запитання відповідали академічному формату;

– вказівку генерувати рівно чотири варіанти відповідей з одним правильним.

Користувацький інструктивний блок (user message) – це центральна частина промпта. Цей блок є точкою входу для контексту, саме він визначає, що модель буде генерувати. Користувацький інструктивний блок включає:

– фрагмент навчального тексту «чанк», реальний зміст, на основі якого треба генерувати запитання;

– бажану кількість запитань, наприклад «згенеруй 6 запитань на основі цього фрагмента»;

– опис структури JSON, що визначає, які ключі має повернути модель;

– приклад JSON-відповіді, який слугує шаблоном, зазвичай модель охоче наслідує наведений формат, наприклад опис тестових запитань, та відповідей до них;

– попередження про недопустимість мета-пояснень чи відступів від формату.

Блок контролю якості окремо у промпті задає обмеження, що покращують якість результатів. Цей блок значно зменшує кількість помилкових або непридатних запитань. Він включає у себе такі інструкції:

– унеможливлювати дублікати: модель намагається створювати унікальні запитання;

– використовувати чітку структуру «питання → варіанти → правильна відповідь»;

– дотримуватись вказаної складності. У промті прописано 3 рівні складності, котрі вліяють на генерацію тексту;

– підтримувати стиль і термінологію тексту, з якого формуються запитання.

Формат відповіді, якого LLM має дотримуватись, також зазначено у промті. Він дозволяє встановити такі правила:

– модель має повернути тільки чистий JSON;

– відповіді не повинні містити супровідного тексту;

– JSON повинен містити строго визначені ключі;

– всі варіанти відповідей повинні бути у масиві.

Це суттєво покращує обробку та дозволяє автоматизувати весь процес без ручного втручання.

Для передачі цих інструкцій моделі використовується стандартна структура запиту до API, яка наведена в лістнгу 1.

Лістинг 1 – Загальна структура запиту (request body) для отримання JSON-відповіді від API мовної моделі

```
{
  "model": "gpt-4o-mini",
  "messages": [
    { "role": "system", "content": "<інструкція>" },
    { "role": "user", "content": "<фрагмент тексту +
вимоги>" }
  ],
  "temperature": 0.3,
  "max_tokens": 1500,
  "response_format": { "type": "json_object" }

СТРУКТУРА JSON:
{{
    "questions": [
        {{
            "id": "q1",
            "question": "Текст запитання?",
            "options": ["Варіант A", " Варіант B", "
Варіант C", " Варіант D"],
```

```
            "correct_answer": " Варіант А"
        }}
    ]
}}
```

Основні властивості даних:

– messages – це набір ролей, що керують поведінкою моделі;

– temperature – це контроль творчості (у системі низький для стабільності);

– max_tokens – це обмеження на довжину відповіді;

– response_format – це вимога повернути валідний JSON.

Очікуваний формат відповіді від моделі – це JSON об’єкт з масивом запитань. Приклад відповіді можна переглянути на лістингу 2.

Лістинг 2 – Приклад JSON-об'єкта запитання у очікуваному форматі відповіді моделі

```
{
      "id": "q7",
      "question": "Яку функцію виконує позиційне кодування в
трансформерах?",
      "options": [
        "Забезпечує порядок слів",
        "Прискорює процес навчання",
        "Стискає дані",
        "Пророкує наступне слово"
      ],
      "correct_answer": "Забезпечує порядок слів"
    }
```

Важливо, що модель повертає не заготовлений текст, а строго структуровані дані, що дозволяють провести подальшу автоматичну обробку:

– фільтрацію;

– перевірку якості;

– балансування відповідей;

– збереження у тестовому форматі.

Узагальнуючи усе вищеперераховане, промпт відіграє роль інтелектуального шаблону, який компенсує недоліки типових відповідей

мовної моделі таким чином:

– системна інструкція визначає поведінку та правила;

– користувацький блок фокусує модель на конкретному фрагменті тексту;

– структурний опис JSON змушує модель формувати дані у формалізованому вигляді;

– обмеження якості зменшують кількість помилок, дублювань і нерелевантних формулювань;

– чітка вимога щодо мови гарантує стилістичну однорідність.

Сукупно це формує стійкий процес генерації, у якому результат моделі стає детальним, контрольованим та придатним для обробки.

4 ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА НАУКОВИХ РЕЗУЛЬТАТІВ

4.1 Опис корпусу тестових документів та тестової генерації запитань

У першому документі розглядається тематика Другої світової війни як глобального військово-політичного конфлікту XX століття. У тексті послідовно висвітлюються причини та передумови війни, основні етапи бойових дій, а також ключові події, що визначили перебіг і завершення конфлікту. Окрему увагу приділено гуманітарним наслідкам війни, зокрема масовим втратам серед цивільного населення, Голокосту та політичним і міжнародно-правовим підсумкам післявоєнного періоду. На основі навчального тексту історичної тематики було виконано автоматичну генерацію тестових запитань із використанням розробленої підсистеми. Форму для завантаження файлу, вибору кількості та складності, було наведена на рисунку 4.1.

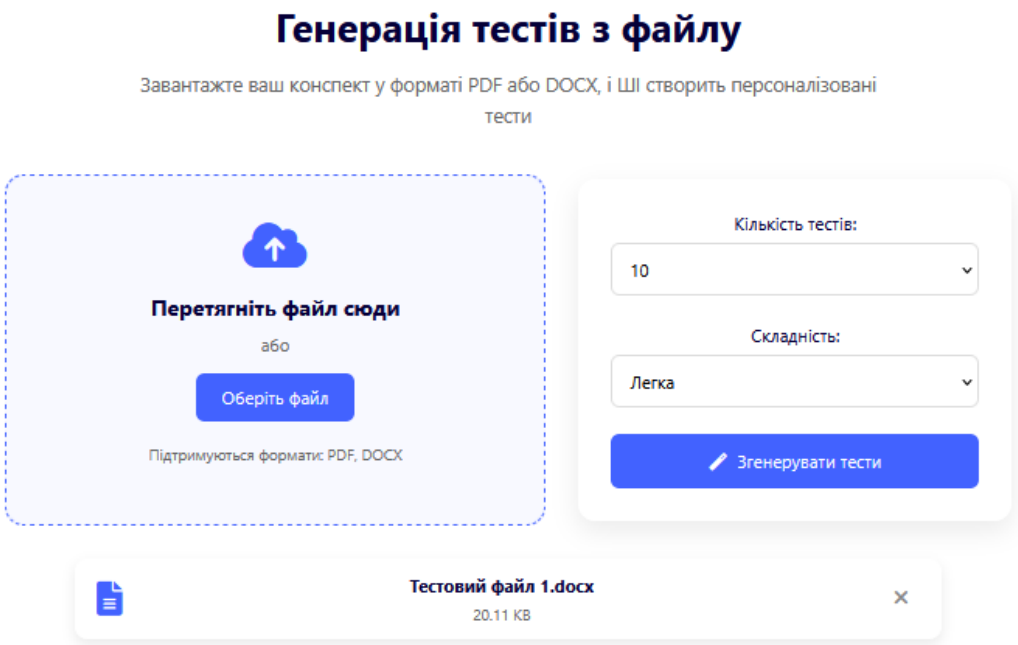


Рисунок 4.1 – Форма завантаження першого документу та вибір складності тестів

Приклад результатів генерації тестів за файлом навчального матеріалу наведено на рисунку 4.2.

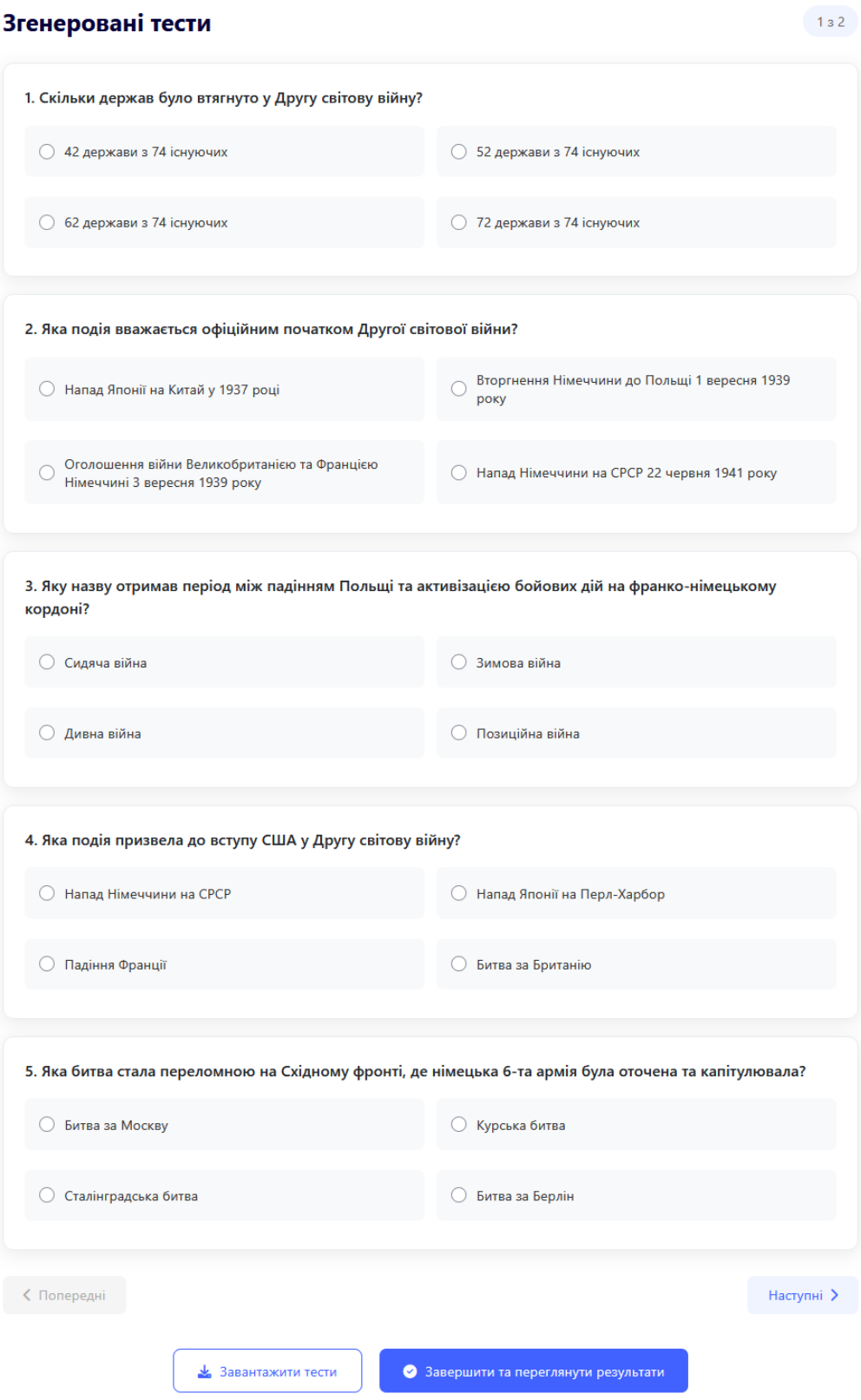


Рисунок 4.2 – Фрагмент тесту, згенерованого на основі першого навчального матеріалу

У другому документі розглядається галузь філософії: логіка (наука про форми та закони правильного мислення), етика (вчення про мораль та моральну поведінку), естетика (філософія прекрасного та мистецтва) та філософська антропологія (дослідження природи людини). У тексті порівнюються ключові концепції та підходи в цих тематиках, такі як дедукція та індукція в логіці, а також проблеми становлення особистості та міжкультурної взаємодії в антропології. На основі філософського тексту за допомогою розробленої підсистеми було автоматично сгенеровано тестові завдання, спрямовані на перевірку розуміння та порівняння основних філософських концепцій. Форму для завантаження файлу, вибору кількості та складності, було наведена на рисунку 4.3.

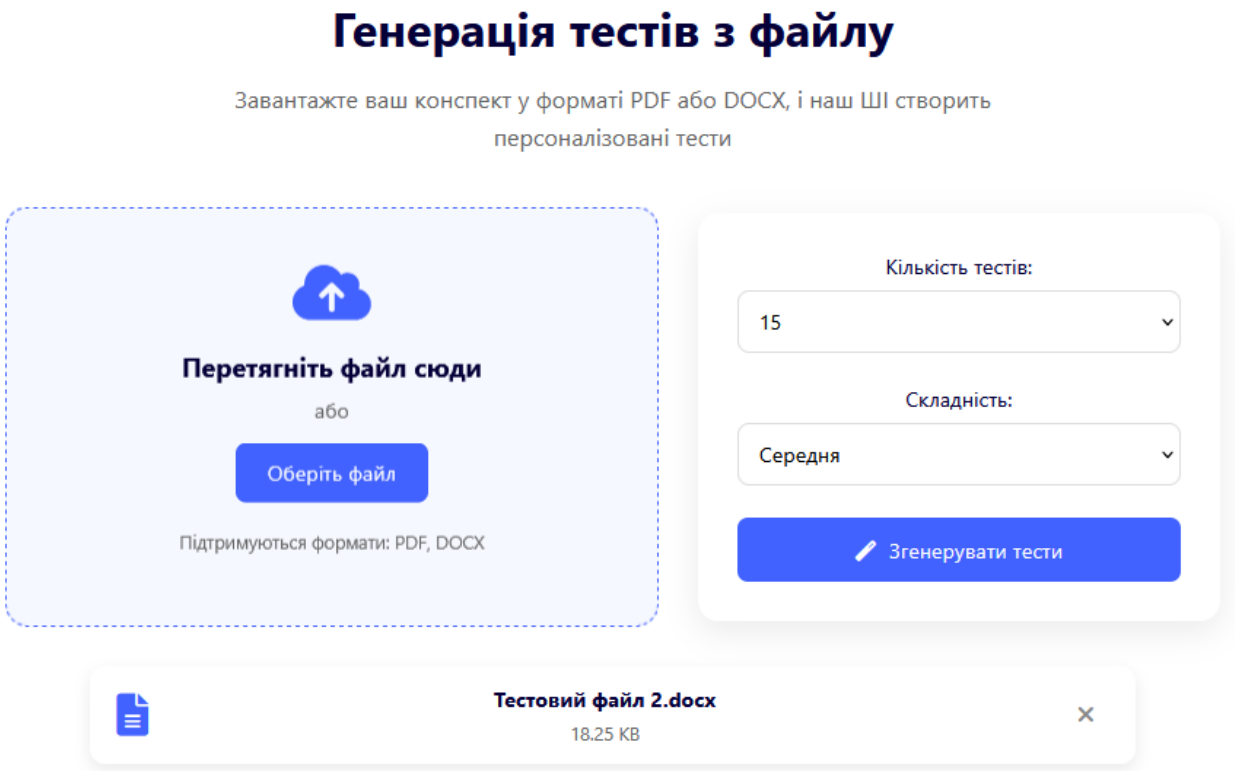


Рисунок 4.3 – Форма завантаження другого документу та вибір складності тестів

Приклад результатів генерації тестів за файлом навчального матеріалу наведено на рисунку 4.4.

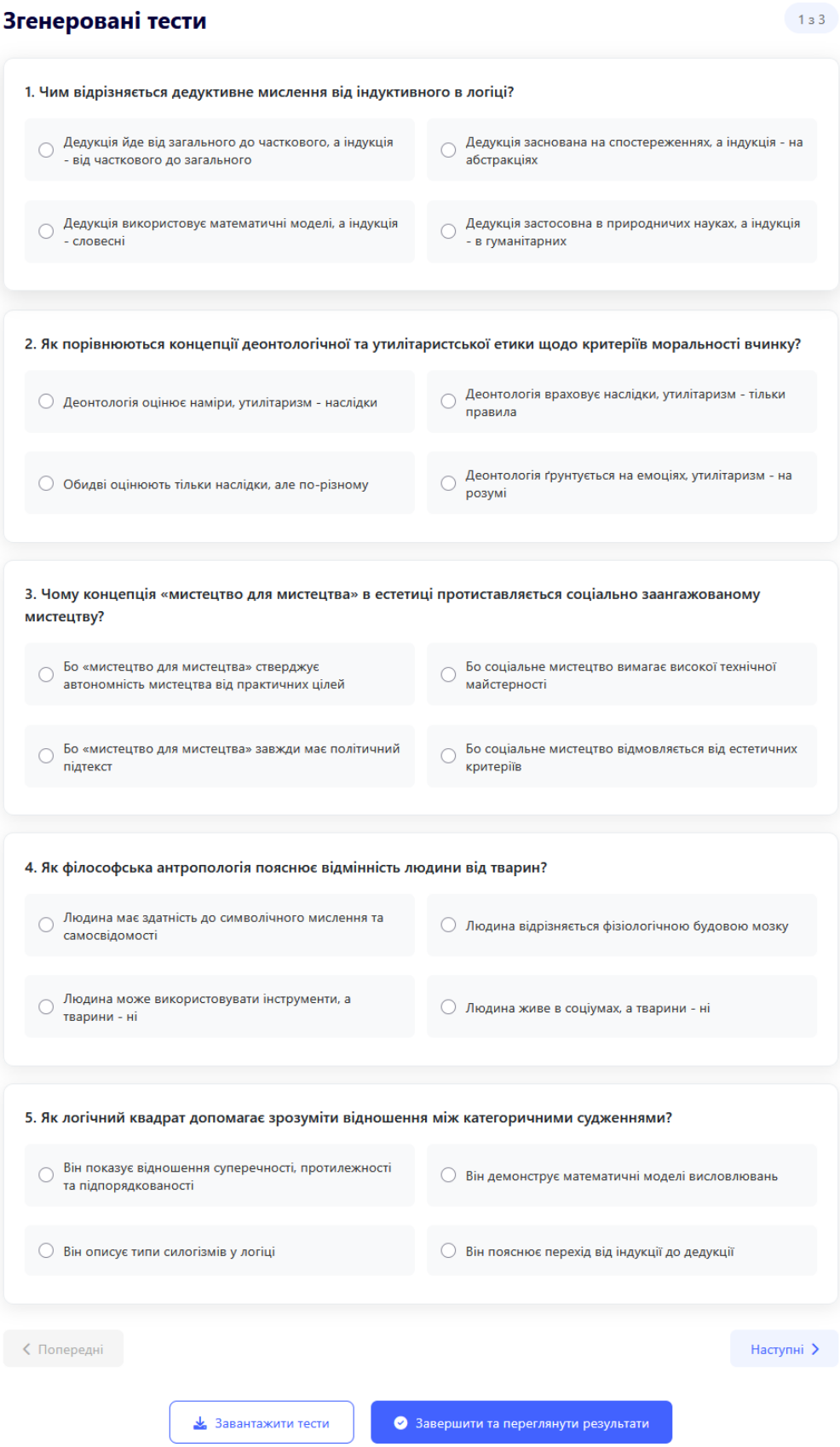


Рисунок 4.4 – Фрагмент тесту, згенерованого на основі другого навчального матеріалу

У третьому документі розглядаються ключові аспекти біології: клітинна будова (організація та функції органел, мембранні системи, метаболічні процеси) та еволюційна теорія (механізми природного відбору, генетичні зміни, видоутворення, адаптації). У тексті аналізуються складні взаємозв'язки між структурою та функцією на клітинному рівні, а також причинно-наслідкові зв'язки в еволюційних процесах, такі як вплив мутацій та пристосованість, і ряд інших тем. На основі біологічного тексту у цьому документі за допомогою розробленої підсистеми було автоматично сгенеровано тестові завдання високого рівня складності, спрямовані на перевірку аналізу причинно-наслідкових зв'язків та оцінки наукових тверджень. Форму для завантаження файлу, вибору кількості та складності, було наведена на рисунку 4.5.

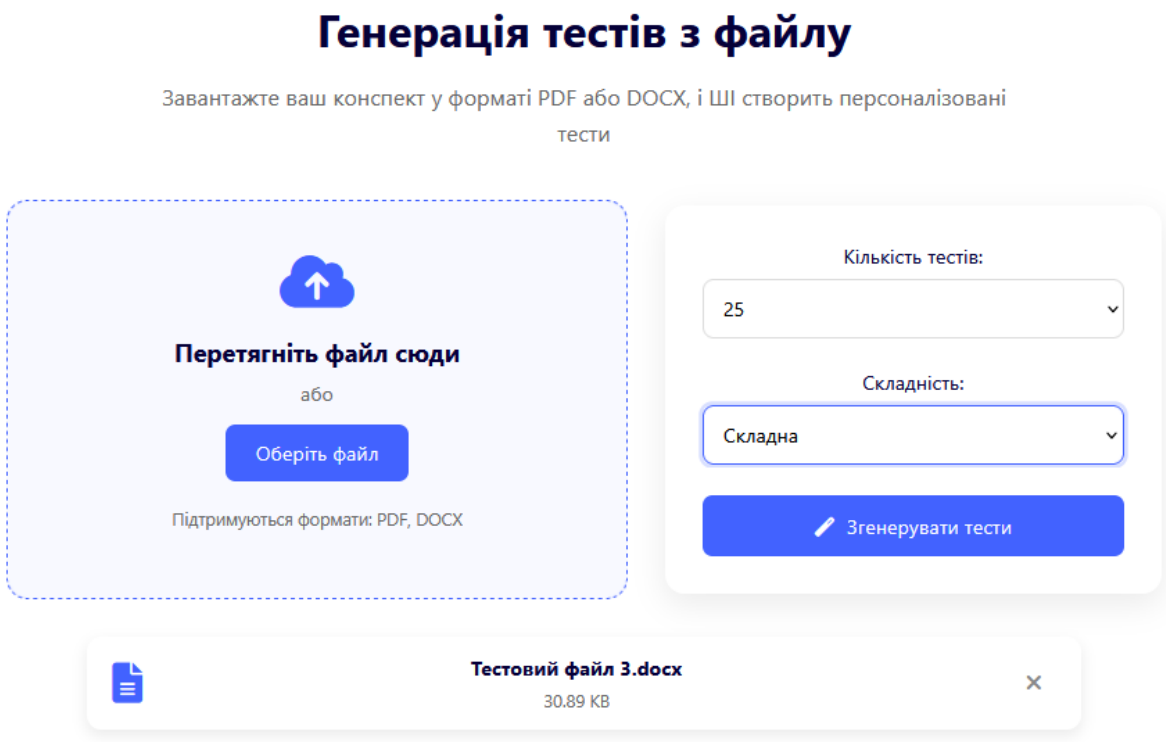


Рисунок 4.5 – Форма завантаження третього документу та вибір складності тестів

Приклад результатів генерації тестів за файлом навчального матеріалу наведено на рисунку 4.6

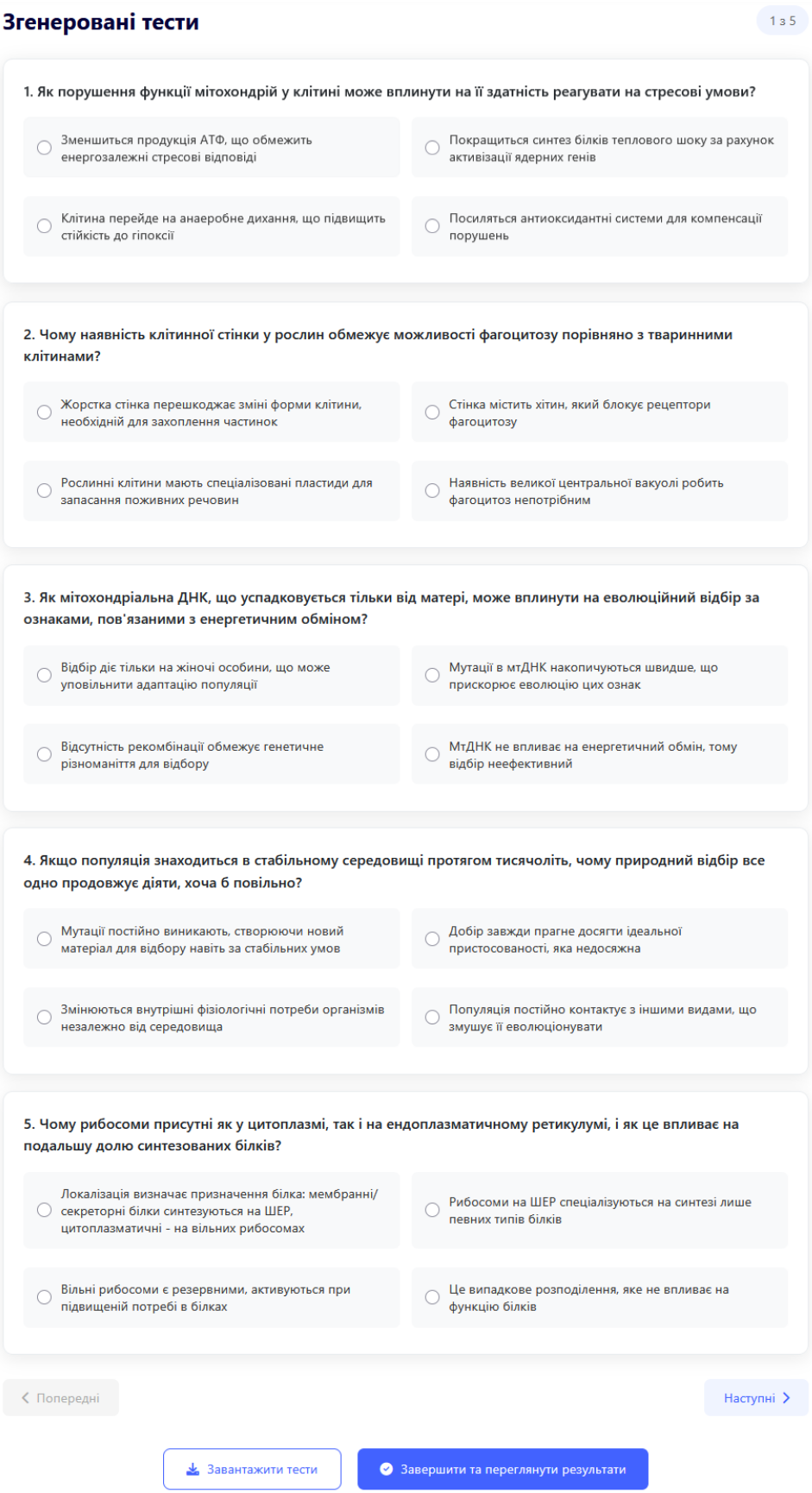


Рисунок 4.6 – Фрагмент тесту, згенерованого на основі третього навчального матеріалу

Для перевірки роботи розробленої системи було використано навчальні матеріали різного обсягу з різною кількістю символів. Для генерації було встановлено середній рівень складності та цільову кількість 15 запитань.

Процес генерації супроводжувався детальним логуванням кожного етапу, що дозволило проаналізувати роботу ключових механізмів системи у різних умовах. При викононанні цього експеремену не було виявлено відхилень та помилок, під час геерування. У цьому випадку журнал містить тільки перелік дій, які виконувались.

Але взагалі в іншних випадках, можуть бути зафіксовані три характерні сценарії, які демонструють ефективність реалізованих алгоритмів відновлення, валідації та балансування. Кожен сценарій представляє різний рівень пошкодження відповіді від мовної моделі та показує, як система адаптується до цих умов.

У першому, ідеальному сценарії система успішно обробила текст, розбила його на семантичні частини та сформувала промпт. Мовна модель повернула відповідь у коректному JSON-форматі, який система успішно розпарсила, отримавши 15 запитань. Усі запитання пройшли повний цикл валідації, перевірку на відповідність структурі, логічну цілісність та відсутність дублікатів. Система також автоматично вирівняла розподіл правильних відповідей між варіантами. Результат обробки та генерації з різними сценаріями можна побачити на рисунках 4.7 – 4.9.

Аналіз першого сценарію з рисунку 4.7, показує, що система виконала всі етапи без помилок, що демонструє її ефективність при ідеальних умовах. Модель повернула валідний JSON, що дозволило одразу отримати цільову кількість запитань. Механізми валідації підтвердили коректність кожного запитання, перевіривши наявність всіх необхідних полів, логічну цілісність та відповідність змісту документа. Балансування позицій правильних відповідей показало невеликий перекос (4-4-4-3), що відповідає прийнятному розподілу. Цей сценарій демонструє ідеальну роботу системи при коректній відповіді від LLM та ефективність реалізованих механізмів постобробки.

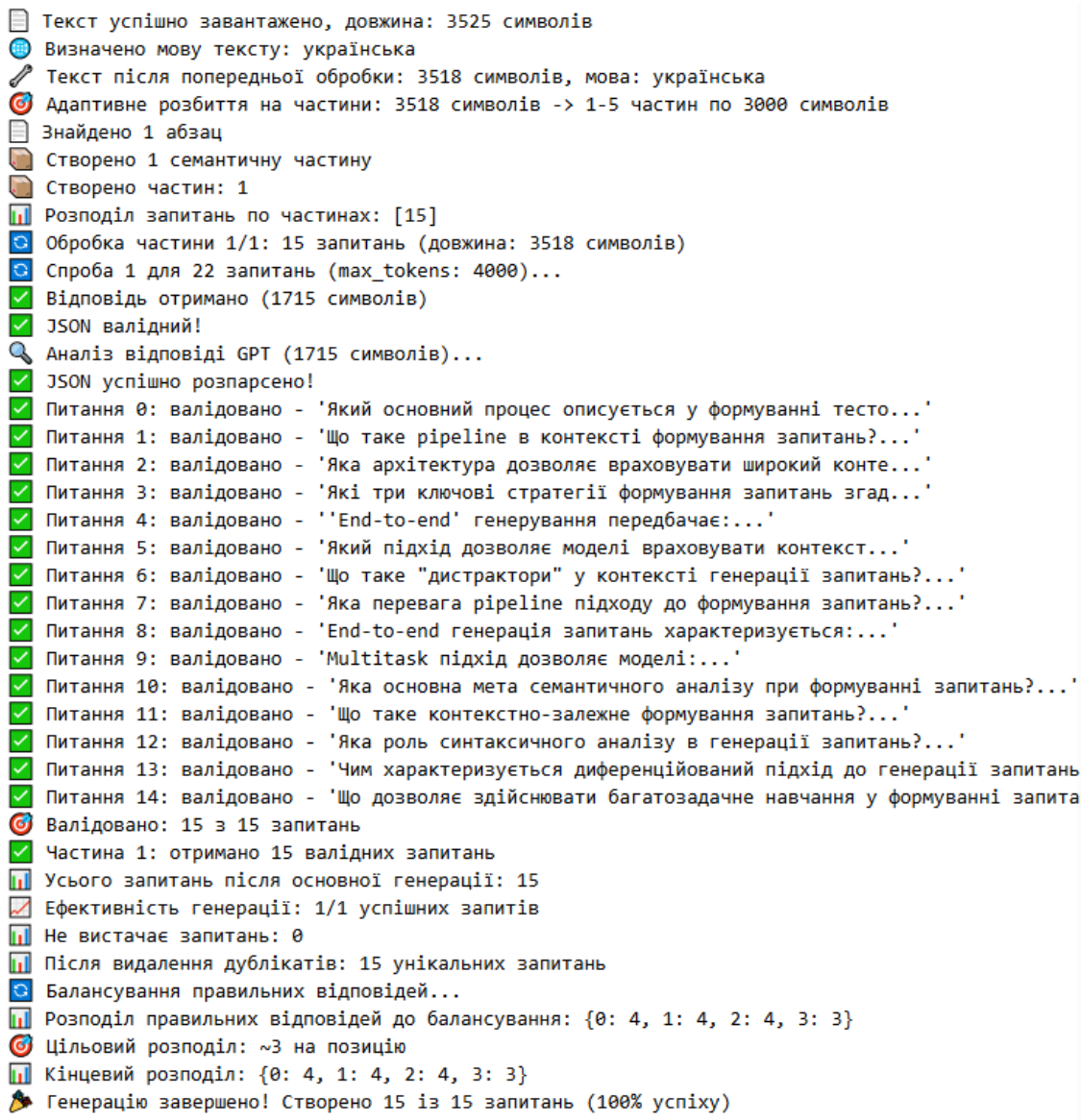


Рисунок 4.7 – Журнал успішної генерації тестових запитань з валідним JSON

У другому сценарії (рис. 4.8) відповідь від мовної моделі містила синтаксичні помилки (відсутність подвійних лапок у назвах полів), що зробило неможливим стандартний парсинг JSON. Система автоматично активувала механізм відновлення структури, а коли це не вдалося через критичний характер пошкоджень, перейшла до резервного парсингу за допомогою регулярних виразів. Цей механізм дозволив витягти 12 коректних запитань. Для досягнення цільової кількості система ініціювала додаткову генерацію 3 запитань, після чого виконала всі етапи постобробки, включаючи видалення дублікатів та балансування відповідей.

```
Текст успішно завантажено, довжина: 3525 символів
Визначено мову тексту: українська
Текст після попередньої обробки: 3518 символів, мова: українська
Адаптивне розбиття на частини: 3518 символів -> 1-5 частин по 3000 символів
Знайдено 1 абзац
Створено 1 семантичну частину
Створено частин: 1
Розподіл запитань по частинах: [15]
Обробка частини 1/1: 15 запитань (довжина: 3518 символів)
Спроба 1 для 22 запитань (max_tokens: 4000)...
Відповідь отримано (1715 символів)
Невалідний JSON: Expecting property name enclosed in double quotes: line 1 column 2 (char 1)
Спроба відновити JSON...
Не вдалося відновити JSON
Резервний парсинг...
Резервним методом витягнуто 12 питань
Питання 0: валідовано - 'Які основні проблеми виникають при автоматичній генерації тестів?'
Питання 1: валідовано - 'Чим відрізняються encoder-only та decoder-only архітектури?'
Питання 2: валідовано - 'Яка мета використання positional encoding у трансформерах?'
Питання 3: валідовано - 'Що таке few-shot промптинг та які його переваги?'
Питання 4: валідовано - 'Які педагогічні ризики пов'язані з використанням LLM для генерації
Питання 5: валідовано - 'Чому важливо контролювати температуру генерації для тестових запита
Питання 6: валідовано - 'Що таке контекстне обмеження LLM та як воно впливає на генерацію?'
Питання 7: валідовано - 'Які механізми захисту від дублювання запитань реалізовано в системі
Питання 8: валідовано - 'Як працює механізм балансування позицій правильних відповідей?'
Питання 9: валідовано - 'Що таке семантичний чанкінг та чому він важливий?'
Питання 10: валідовано - 'Які етапи включає процес валідації згенерованих запитань?'
Питання 11: валідовано - 'Чому JSON є кращим форматом для обміну даними з LLM?'
Валідовано: 12 з 12 запитань
Частина 1: отримано 12 валідних запитань
Усього запитань після основної генерації: 12
Ефективність генерації: 1/1 успішних запитів
Не вистачає запитань: 3
Запуск додаткової генерації...
Додаткова спроба 1: генеруємо 3 запитання...
Відповідь отримано (1523 символи)
JSON валідний!
Аналіз відповіді GPT (1523 символи)...
JSON успішно розпарсено!
Питання 0: валідовано - 'Які типи тестових запитань підтримує система генерації?'
Питання 1: валідовано - 'Що таке мультимодальна генерація та чи підтримується вона в системі
Питання 2: валідовано - 'Які мови підтримує система для генерації тестів?'
Валідовано: 3 з 3 запитань
Додаткова спроба 1: додано 3 запитання
Додатково додано 3 запитання
Кінцева кількість запитань: 15
Після видалення дублікатів: 15 унікальних запитань
Балансування правильних відповідей...
Розподіл правильних відповідей до балансування: {0: 4, 1: 4, 2: 4, 3: 3}
Цільовий розподіл: ~3 на позицію
Кінцевий розподіл: {0: 4, 1: 4, 2: 4, 3: 3}
Генерацію завершено! Створено 15 із 15 запитань (100% успіху)
```

Рисунок 4.8 – Журнал генерації тестових запитань з пошкодженою відповіддю від LLM

Аналіз другого сценарію з рисунку 4.8, показує, що система продемонструвала гнучкість та відмовостійкість при обробці пошкоджених даних. Механізм відновлення JSON не спрацював через критичні синтаксичні помилки, проте резервний парсинг дозволив витягти 12 коректних запитань.

Компенсаційний механізм (over-generation) автоматично ініціював додаткову генерацію недостаючих 3 запитань, що дозволило досягти цільового числа 15.

У третьому, найскладнішому сценарії (рис. 4.9) відповідь від моделі була настільки пошкодженою, що жоден механізм відновлення не дозволив отримати достатню кількість запитань. Резервний парсинг витягнув лише 4 запитання, з яких лише 3 пройшли валідацію. Система автоматично виявила критичну нестачу (3 запитань з 15) та ініціювала повторний запит до мовної моделі з тими самими параметрами. При другій спробі модель повернула коректний JSON, з якого було отримано 12 запитань. Загалом, після двох спроб система досягла цільової кількості 15 запитань та успішно виконала всі етапи постобробки.

Аналіз третього сценарію з рисунку 4.9, показує, що він демонструє максимальну стійкість системи у критичних умовах. При повній неспроможності обробки першої відповіді система автоматично виконує повторний запит, гарантуючи отримання цільової кількості запитань навіть у разі критичних збоїв. Важливо відзначити, що система не просто повторює запит, а аналізує кількість вже отриманих валідних запитань та генерує саме стільки, скільки не вистачає для досягнення цільового числа. Після двох спроб було досягнуто 15 запитань, і всі наступні етапи (фільтрація дублікатів, балансування) виконані успішно.

Аналіз трьох сценаріїв роботи підтверджує такі особливості реалізованих механізмів обробки відповідей від LLM, які підтверджують їх ефективність:

1) багаторівнева обробка JSON-відповідей: система застосовує послідовний підхід від стандартного парсингу через механізми відновлення до резервного парсингу, що забезпечує обробку даних різного рівня пошкодженості;

2) стратегія надлишкової генерації (over-generation): система завжди генерує на 30-50% більше запитань, ніж потрібно, що дозволяє компенсувати втрати під час валідації та фільтрації;

3) автоматичні повторні запити при критичних збоях: система самостійно визначає недостатню кількість валідних запитань та ініціює повторну генерацію для досягнення цільового результату.

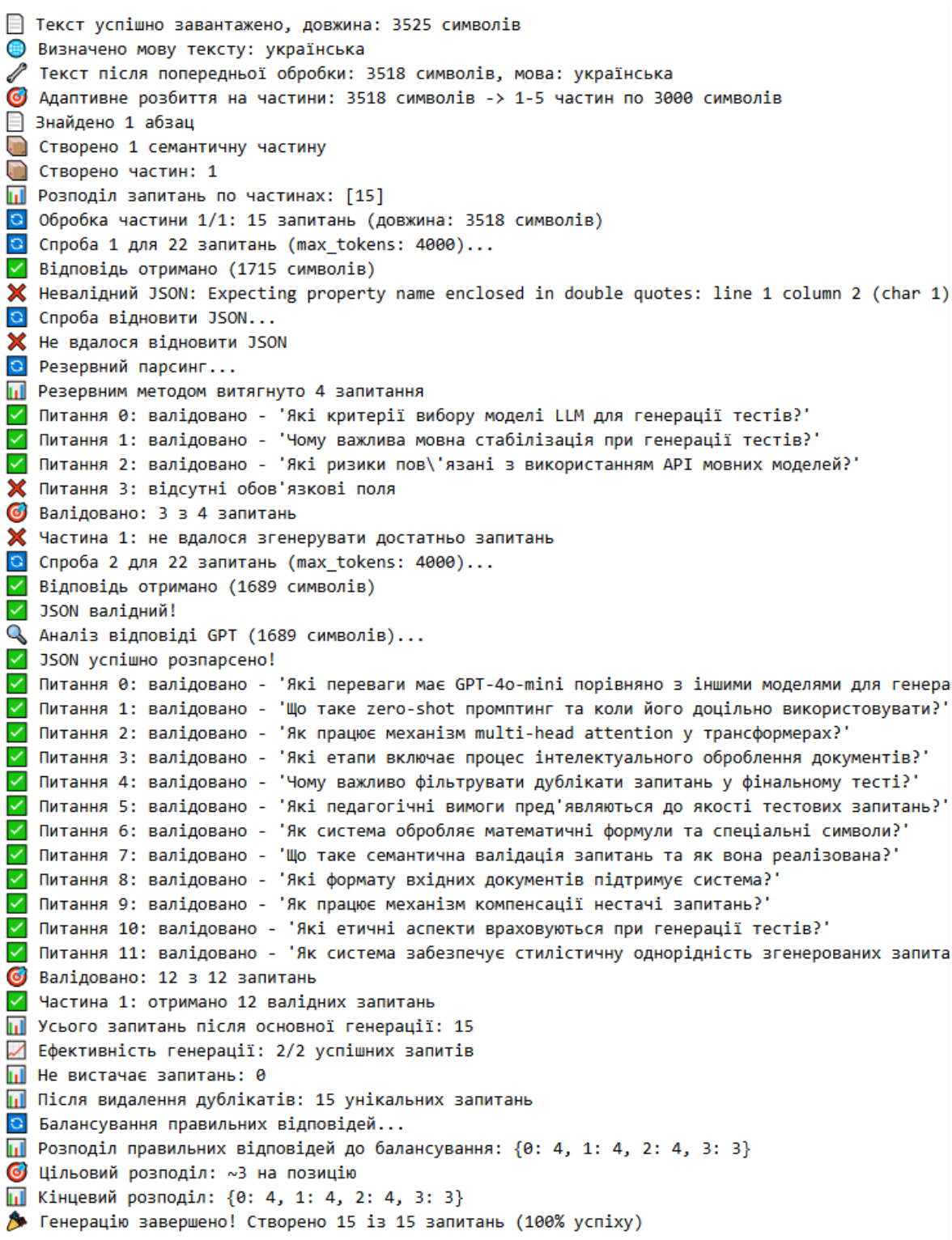


Рисунок 4.9 – Журнал генерації тестових запитань з дуже пошкодженою відповіддю від LLM

Підсистема доволяє редагувати згенеровані запитання тесту за допомогою форми, яка наведенноеї на рисунку 4.10

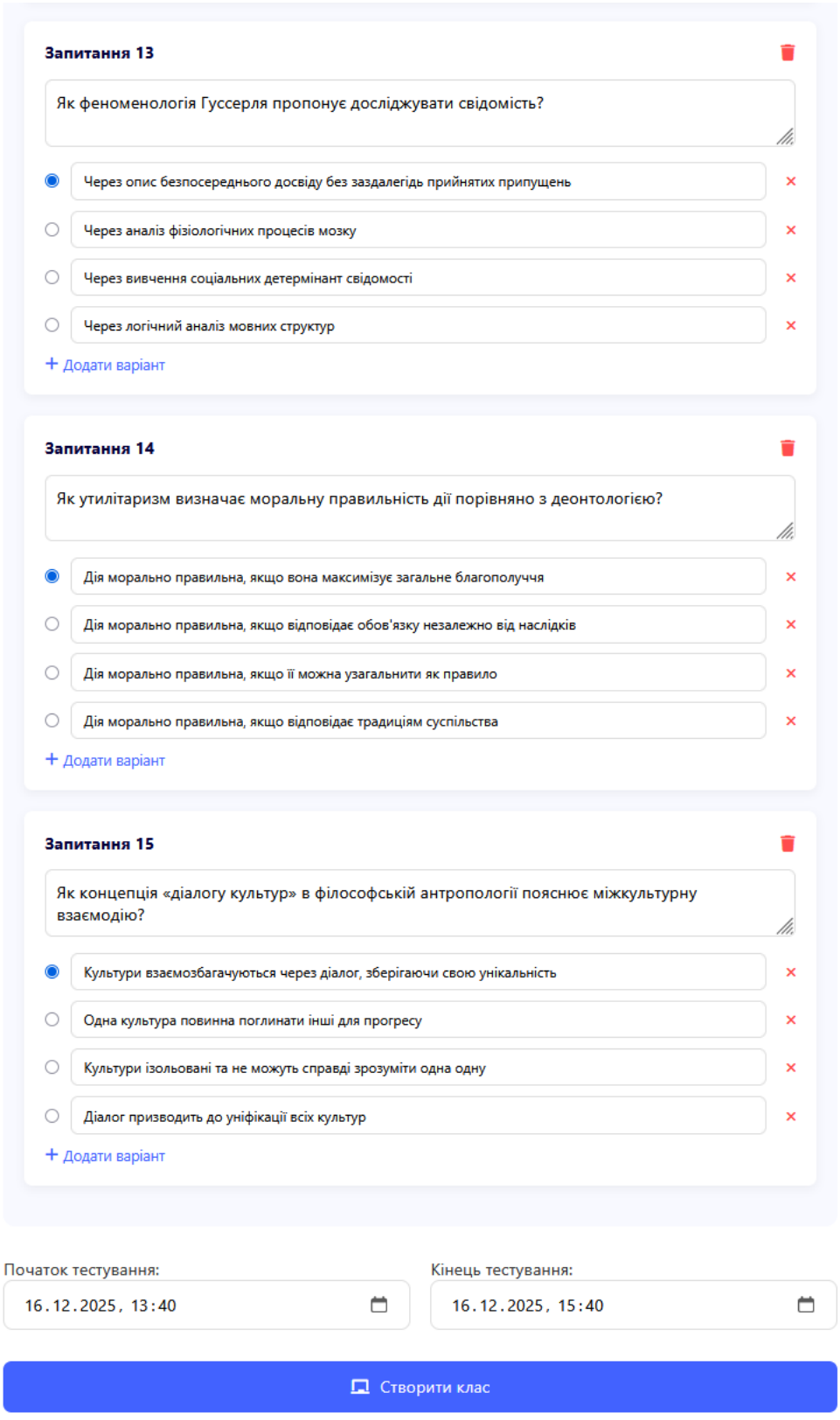


Рисунок 4.7 – Форма редагування запитань, отриманих від LLM

4.2 Оцінка якості та ефективності генерації запитань

Оцінювання якості автоматично згенерованих тестових запитань є нетривіальною задачею, оскільки коректність таких завдань визначається не лише граматичною правильністю формулювань, а й їхньою змістовною відповідністю навчальному матеріалу та педагогічною цінністю. Якісне тестове запитання повинно бути однозначним щодо правильної відповіді, не допускати альтернативних трактувань і відповідати заявленому рівню складності. У випадку використання мовних моделей додатковою складністю є те, що модель оперує статистичними закономірностями тексту та не володіє явним розумінням навчальної мети, що ускладнює автоматичне визначення якості результату.

Застосування виключно автоматичних метрик оцінювання не дозволяє отримати повну картину якості згенерованих тестів. Такі показники, як precision і recall, дають змогу кількісно оцінити частку коректних запитань та ступінь охоплення основних тем навчального документа. Метрики BLEU та ROUGE можуть використовуватися для аналізу лексичної схожості формулювань, проте вони не враховують логічну структуру запитання, якість дистракторів та педагогічну коректність. У зв’язку з цим автоматичні метрики розглядаються лише як допоміжний інструмент аналізу та не можуть бути єдиною основою для оцінювання якості генерації.

З метою отримання об’єктивних результатів у роботі застосовано комбінований підхід до оцінювання, який поєднує автоматичний аналіз і оцінку згенерованих тестових запитань відповідно людиною. Такий підхід дозволяє врахувати як кількісні характеристики результатів генерації, так і якісні аспекти, що не піддаються формалізації. Оцінка проводилася для тестів, згенерованих на основі навчальних матеріалів різних предметних областей та для різних рівнів складності (легкий, середній і складний), було проаналізовано 130 згенерованих тестів.

Оцінювання якості кожного тестового запитання здійснювалося за сукупністю критеріїв, до яких належать такі:

1) відповідність змісту навчального документа;

2) коректність формулювання;

3) однозначність правильної відповіді;

4) відсутність двозначності інтерпретації та відповідність заявленому рівню складності.

Для кількістного показу результатів використовувалась п’ятибальна шкала, де значення «1» відповідало некоректним і непридатним до використання запитанням, а значення «5» – повністю коректним і структурово правильним.

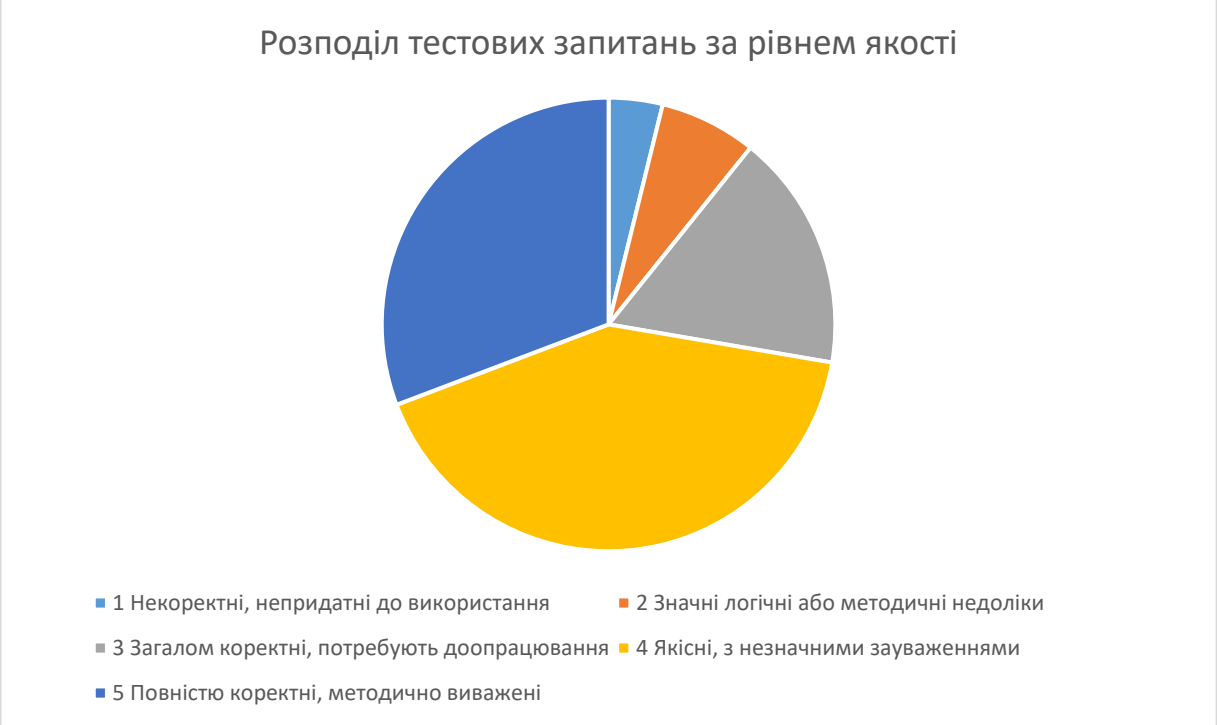
Результати оцінювання узагальнено у таблиці 4.1, де наведено розподіл згенерованих тестових запитань за рівнями якості відповідно до п’ятибальної шкали.

Таблиця 4.1 — Розподіл згенерованих тестових запитань за рівнем якості

Оцінка	Характеристика якості	Кількість запитань	Частка, %
1	Некоректні, непридатні до використання	5	3,8 %
2	Значні логічні або методичні недоліки	9	6,9 %
3	Загалом коректні, потребують доопрацювання	22	16,9 %
4	Якісні, з незначними зауваженнями	54	41,5 %
5	Повністю коректні, методично виважені	40	30,8 %
Разом		130	100 %

Як видно з таблиці 4.1, найбільшу частку становлять запитання з оцінками 4 та 5, що свідчить про переважно високу якість результатів автоматичної генерації. Частка позитивних оцінених тестів (оцінки 4 і 5) становить 72,3 % від загальної кількості проаналізованих завдань.

Для наочного подання результатів оцінювання було побудовано діаграму розподілу тестових запитань за рівнем якості. На рисунку 4.11 представлено співвідношення кількості запитань, що отримали оцінки від 1 до 5, що у свою чергу дозволяє візуально оцінити ефективність роботи розробленої підсистеми генерації тестів.



Рисуно 4.11 – Кругова діаграма розподілу тестових запитань за рівнем якості

Переважання секторів, що відповідають оцінкам 4 та 5, свідчить про стабільну якість генерації для різних предметних областей і рівнів складності.

Таким чином, результати оцінки підтверджують доцільність використання розробленої підсистеми для автоматичної генерації тестових запитань на основі навчальних матеріалів. Поєднання автоматичних метрик та експертного аналізу забезпечує всебічну оцінку якості результатів і дозволяє визначити напрями подальшого вдосконалення системи.

Слід зазначити, що якість згенерованих тестових запитань значною мірою залежить від типу та структури вихідного навчального документа. У випадку матеріалів, які містять велику кількість математичних формул,

символічних позначень або спеціалізованих нотацій, автоматична генерація тестів може призводити до появи некоректних або методично непридатних запитань. Це зумовлено обмеженнями текстового представлення формул у форматі JSON, а також відсутністю повноцінної підтримки математичної розмітки під час подальшого відображення згенерованого контенту у веб-інтерфейсі системи. У таких випадках мовна модель не завжди коректно інтерпретує семантику формульних виразів, що негативно впливає на якість формулювання запитань і варіантів відповідей. Разом із тим зазначене обмеження не є принциповим і може бути усунуте в подальших версіях системи.

ВИСНОВКИ

У межах магістерської кваліфікаційної роботи було проведено дослідження методів і моделей автоматичної генерації тестових запитань, спрямоване на створення підсистеми для освітньої платформи, здатної формувати тестові завдання різних типів на основі навчальних матеріалів.

У роботі проаналізовано процес генерації навчальних тестів та існуючі підходи до їх автоматичної генерації, розроблено вимоги до підсистеми генерації тестових запитань у складі освітньої платформи.

У рамках роботи проведено аналіз та вибір реалізації великої мовної моделі, найбільш придатної для генерації навчальних тестів, розроблено систему критеріїв вибора реалізації генеративних LLM сімейства GPT.

Також було досліджено особливості використання LLM для генерації навчальних тестів, виявлено ризики і проблеми, пов’язані з використанням LLM, запропоновано підходи та механізми мінімізації ризиков та забезпечення якості генерації, запропоновано удосканалення процесу генерації тестів за рахунок включення до нього нових етапів, які реалізують ці підходи та механізми.

Розроблено інформаційну технологію генерації навчальних тестів з використанням реалізації генеративної LLM та алгоритми виконання додаткових етапів у процесі генерації, які забезпечують якость генерації тестів.

Крім того виконано програмну реалізацію підсистеми генерації тестів у складі освітньої платформи, яка впроваджує розроблену ІТ. За її допомогою виконано експериментальну перевірку ІТ та наукових результатів на корпусі навчальних матеріалів. Результати експерименту підтвердили можливість використання генеративних LLM та додаткових запропонованих підходів і механізмів для якісної автоматичної генерації тестових запитань.

Отримані результати демонструють, що розроблена підсистема

генерації тестів може бути успішно інтегрована в сучасні освітні платформи, що дозволить значно зменшити навантаження на викладачів, забезпечити оперативне формування тестів і підвищити ефективність контролю знань.

Перспективами подальших досліджень є розширення інтелектуальної перевірки якості тестів, адаптивне визначення складності, створення системи самооцінювання та покращення інтеграції з LMS-платформами.

Результати роботи були опубліковані у форматі тез доповіді на конференції: V International Scientific and Theoretical Conference «Current scientific goals, approaches and challenges» (December 12, 2025). Dresden, Germany. [18].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методичні вказівки щодо розробки та оформлення кваліфікаційної роботи для (здобувачів вищої освіти усіх форм навчання другого (магістерського) рівня вищої освіти спеціальності 122 Комп'ютерні науки освітньо-професійної програми «Інформаційні управляючі системи та технології»). / Упоряд.: К.Е. Петров, С.Ф. Чалий, О.Д. Міхнова. – Харків: ХНУРЕ, 2024. – 19 с.

2. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. Чинний від 2017-07-01. – Київ: ДП «УкрНДНЦ», 2016. – 26 с.

3. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Чинний від 2016-07-01. – Вид. офіц. Київ : УкрНДНЦ, 2016. 16 с.

4. Brown T., Mann B., Ryder N., Subbiah M., Kaplan J., Dhariwal P. et al. Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems. 2020. Vol. 33. P. 1877–1901. URL: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html (дата звернення: 12.11.2025).

5. Radford A., Wu J., Child R., Luan D., Amodei D., Sutskever I. Language Models are Unsupervised Multitask Learners. OpenAI Blog. 2019. URL: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf (дата звернення: 12.11. 2025).

6. Wolf T., Debut L., Sanh V., Chaumond J., Delangue C., Moi A. et al. 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. URL: https://aclanthology.org/2020.emnlp-demos.6/ (дата звернення: 12.11. 2025).

7. Raffel C., Shazeer N., Roberts A., Lee K., Narang S., Matena M. et al.

Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Journal of Machine Learning Research. 2020. Vol. 21, no. 140. P. 1–67. URL: https://jmlr.org/papers/v21/20-074.html (дата звернення: 12.11. 2025).

8. Liu P., Yuan W., Fu J., Jiang Z., Hayashi H., Neubig G. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. ACM Computing Surveys. 2023. Vol. 55, no. 9. P. 1–35. URL: https://dl.acm.org/doi/10.1145/3560815 (дата звернення: 12.11. 2025).

9. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł., Polosukhin I. Attention Is All You Need. Advances in Neural Information Processing Systems (NeurIPS 2017), Proceedings of the 31st International Conference on Neural Information Processing Systems**.** 2017. P. 5998–6008. URL: https://papers.neurips.cc/paper/7181-attention-is-all-you-need.pdf (дата звернення: 21.11. 2025).

10. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT). 2019. Vol. 1. P. 4171–4186. URL: https://aclanthology.org/N19-1423/ (дата звернення: 21.11. 2025).

11. Radford A., Narasimhan K., Salimans T., Sutskever I. Improving Language Understanding by Generative Pre-Training. OpenAI Blog. 2018. URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf (дата звернення: 21.11. 2025).

12. Raffel C., Shazeer N., Roberts A., Lee K., Narang S., Matena M., Zhou Y., Li W., Liu P. J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Journal of Machine Learning Research. 2020. Vol. 21, no. 140. P. 1–67. URL: https://jmlr.org/papers/v21/20-074.html (дата звернення: 21.11. 2025).

13. Wei J., Tay Y., Bommasani R., Raffel C., Zoph B., Borgeaud S. et al. Emergent Abilities of Large Language Models. Transactions on Machine Learning

Research. 2022. URL: https://openreview.net/forum?id=yzkSU5zdwD (дата звернення: 21.11. 2025).

14. White J., Fu Q., Hays S., Sandborn M., Olea C., Gilbert H. et al. A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. Proceedings of the 2023 Conference on Human Factors in Computing Systems (CHI). 2023. URL: https://dl.acm.org/doi/10.1145/3544548.3581318 (дата звернення: 21.11. 2025).

15. OpenAI. GPT-4 Technical Report. OpenAI. 2023. URL: https://cdn.openai.com/papers/gpt-4.pdf (дата звернення: 21.11. 2025).

16. Kasneci E., Seßler K., Küchemann S., Bannert M., Dementieva D., Fischer F. et al. ChatGPT for good? On opportunities and challenges of large language models for education. Learning and Individual Differences. 2023. Vol. 103. P. 102274. URL: https://www.sciencedirect.com/science/article/pii/S1041608023000195 (дата звернення: 21.11. 2025).

17. Bender E. M., Gebru T., McMillan-Major A., Shmitchell S. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency. 2021. P. 610–623. URL: https://dl.acm.org/doi/10.1145/3442188.3445922 (дата звернення: 21.11. 2025).

18. Kamsiuk D., Borysenko T. Проблеми та стратегії підвищення якості генерації тестів в освітній платформі з використанням великих мовних моделей // V International Scientific and Theoretical Conference «Current scientific goals, approaches and challenges» (December 12, 2025). Dresden, Germany. Pp. 151-215. URL: DOI https://doi.org/10.36074/scientia-12.12.2025.

19. Hugging Face. Large Language Models Explained. Hugging Face Blog. 2023. URL: https://huggingface.co/blog/large-language-models (дата звернення: 12.12.2025).

20. ACM. Prompt Engineering for Large Language Models. ACM Digital Library. 2023. URL: https://dl.acm.org/doi/10.1145/3544548.3581318 (дата звернення: 12.12.2025).

21. Hugging Face. Text Generation Strategies. Hugging Face Documentation.

2024. URL: https://huggingface.co/docs/transformers/main/en/generation_strategies (дата звернення: 16.12.2025).

22. OpenAI. Structured Outputs and JSON Mode. OpenAI. 2024. URL: https://platform.openai.com/docs/guides/structured-outputs (дата звернення: 12.12.2025).