Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Розробка генеративної моделі створення навчального контенту
на основі великих мовних моделей
(тема)

Виконав:
здобувач другого року навчання,
групи ДСм-24-1
Жеребний В.В.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Науки про дані (Data Science)
(повна назва спеціалізації)
Керівник доц. Магдаліна І.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Лариса ЧАЛА
(підпис) (прізвище, ініціали)

2025 р

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

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 Науки про дані (Data Science)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Жеребному Віктору Валерійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка генеративної моделі створення навчального контенту на основі великих мовних моделей

затверджена наказом університету від 24 листопада 20 25 р. № 1057Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18 грудня 20 25 р.

3. Вихідні дані до роботи Науково-технічні публікації, дані Інтернет-джерел та відомих наукових проєктів, Python documentation, набір даних для тренування та тестування системи

4. Перелік питань, що потрібно опрацювати в роботі

1) Аналіз предметної галузі та постановка задачі

2) Аналіз наявних підходів побудови систем штучного інтелекту

3) Побудова додатку для створення навчального контенту

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів передатестаційної практики	Термін виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	24.11.2025	виконано
2	Аналіз предметної області великих	27.10.2025	виконано
	генеративних мовних моделей		
3	Планування архітектури власної генеративної	29.11.2025	виконано
	системи створення навчального контенту		
4	Аналіз підходів до автоматизованої перевірки	01.12.2025	виконано
	достовірності згенерованих матеріалів		
5	Реалізація програмної системи	06.12.2025	виконано
6	Тестування системи створення навчального	07.12.2025	виконано
	контенту		
7	Підготовка висновків	08.12.2025	виконано
8	Оформлення пояснювальної записки	10.12.2025	виконано
9	Нормоконтроль	12.12.2025	виконано
10	Захист кваліфікаційної роботи	18.12.2025	

Дата видачі завдання 24 листопада 20 25 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Магдаліна І.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 74 с., 20 рис., 1 табл., 3 дод., 24 джерела.

ВЕЛИКА МОВНА МОДЕЛЬ, ГЛИБОКЕ НАВЧАННЯ, НАВЧАЛЬНИЙ КОНТЕНТ, ПОПЕРЕДНЄ ТРЕНУВАННЯ, СЕМАНТИЧНИЙ ГРАФ ПЕРЕВІРКИ ЗНАНЬ, ТАКСОНОМІЯ ШТУЧНОГО ІНТЕЛЕКТУ, ШТУЧНИЙ ІНТЕЛЕКТ.

Об’єкт дослідження – задача покращення якості автоматично створеного навчального контенту

Предмет дослідження – розробка генеративної моделі для автоматизованого створення навчального контенту на основі великих мовних моделей (LLM), а також її інтеграція з механізмами семантичної перевірки знань.

Мета роботи – створити архітектуру системи, яка поєднує генеративні можливості LLM з зовнішніми структурами знань. Де пропонується використання семантичного графу з використанням методу Retrieval-Augmented Generation (RAG) для підвищення достовірності й педагогічної цінності згенерованих матеріалів.

Методологія включає аналіз сучасних архітектур LLM (Transformer, Mixture-of-Experts), багаторівневе навчання, а також інтеграцію зовнішніх джерел знань (RAG, графи знань) для автоматизованої верифікації. Розроблено концептуальну архітектуру системи: генератор контенту, модуль структурування дидактики, модуль семантичної перевірки на основі графа знань, модуль фільтрації.

Наукова новизна полягає в поєднанні генеративної моделі із семантичним графом перевірки знань як обов’язковим етапом верифікації перед публікацією контенту, що знижує ризики «галюцинацій» LLM та підвищує якість матеріалів.

ABSTRACT

Master’s thesis contains: 74 pp., 20 fig., 1 tabl., 3 ann., 24 references.

ARTIFICIAL INTELLIGENCE, ARTIFICIAL INTELLIGENCE TAXONOMY, DEEP LEARNING, EDUCATIONAL CONTENT, LARGE LANGUAGE MODEL, PRE-TRAINING, SEMANTIC KNOWLEDGE VERIFICATION GRAPH.

Research Object – the task of improving the quality of automatically generated educational content.

Research Subject – the development of a generative model for automated creation of educational content based on large language models (LLMs), as well as its integration with mechanisms for semantic knowledge verification.

Purpose of the Work – to create a system architecture that combines the generative capabilities of LLMs with external knowledge structures. It is proposed to use a semantic graph with the Retrieval-Augmented Generation (RAG) method to enhance the reliability and pedagogical value of the generated materials.

Methodology includes the analysis of modern LLM architectures (Transformer, Mixture-of-Experts), multi-level training, as well as the integration of external knowledge sources (RAG, knowledge graphs) for automated verification. A conceptual system architecture has been developed: content generator, didactic structuring module, semantic verification module based on a knowledge graph, and a module for filtering.

Scientific Novelty lies in combining a generative model with a semantic knowledge verification graph as a mandatory verification stage before content publication, which reduces the risk of LLM “hallucinations” and improves the quality of materials.

ЗМІСТ

ВСТУП

Активне впровадження технологій штучного інтелекту в освітні процеси зумовлює перегляд традиційних підходів до створення та поширення навчальних матеріалів. Зростання обсягів інформації, швидке оновлення навчальних програм і підвищені вимоги до якості освітнього контенту формують потребу в інструментах, здатних забезпечити оперативну підготовку актуальних і структурованих навчальних ресурсів. У цьому контексті генеративні мовні моделі розглядаються як перспективний інструмент автоматизації створення навчального контенту, що поєднує гнучкість, масштабованість і можливість адаптації до різних освітніх сценаріїв.

Сучасні великі мовні моделі демонструють високий рівень володіння природною мовою та здатність формувати зв’язний текст, орієнтований на задану тематику й стиль викладу. Завдяки цьому вони можуть використовуватися для генерації пояснювальних матеріалів, прикладів, тестових завдань і допоміжних дидактичних елементів. Проте застосування таких моделей у навчальному середовищі супроводжується низкою принципових обмежень, серед яких ключовим є питання достовірності та коректності згенерованої інформації. Статистична природа мовних моделей не гарантує відповідності отриманих тверджень фактичним знанням, що особливо критично в освітньому контексті.

Проблема «галюцинацій» та внутрішніх суперечностей у відповідях мовних моделей зумовлює необхідність пошуку методів контролю якості генерації. Просте збільшення обсягу навчальних даних або ускладнення архітектури моделі не забезпечує повного розв’язання цієї проблеми, оскільки джерело помилок закладене в самому принципі імовірнісної генерації тексту. У зв’язку з цим актуальним є перехід від автономного використання мовних моделей до побудови комплексних систем, у яких

генерація поєднується з механізмами перевірки, узгодження та обґрунтування знань.

Одним із напрямів такого розвитку є інтеграція великих мовних моделей із зовнішніми джерелами інформації та структурованими репрезентаціями знань. Поєднання генеративних можливостей LLM із Retrieval-Augmented Generation дозволяє спиратися на надані навчальні матеріали та зменшувати залежність від внутрішньої параметричної пам’яті моделі. Додаткове використання семантичних графів знань створює умови для формалізації навчального матеріалу, перевірки логічної узгодженості тверджень і підвищення прозорості отриманих результатів.

Таким чином, розробка генеративних систем створення навчального контенту поступово зміщується від задачі простого текстового генерування до задачі побудови керованих, верифікованих і відтворюваних рішень. У цьому контексті особливої уваги потребує дослідження архітектурних підходів, методів навчання та механізмів інтеграції знань, що дозволяють підвищити педагогічну цінність і надійність згенерованих матеріалів. Саме ці аспекти визначають логіку подальшого аналізу предметної галузі та формують підґрунтя для розгляду відповідних технічних рішень у наступних розділах роботи.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз наявних досліджень

Генеративний штучний інтелект для освіти є важливою та критично важливою галуззю для дослідників, викладачів та практиків: постійний прогрес у генеративному штучному інтелекті надає та продовжуватиме надавати безпрецедентні можливості для покращення освіти, а унікальні виклики в освіті вимагатимуть подальших технічних інновацій у ньому [1].

Але існує ряд досліджень на тему мовних генеративних моделей для освітніх цілей, де спільною та однією з головних проблем виступає питання достовірності кінцевого матеріалу та його ефективність у навчальному процесі [2], [3]. Проблематика некомпетентності штучного інтелекту у повсякденних задачах досі вирішується дослідниками передових компаній, а генерація навчального контенту – ставить виклик не тільки технічним реалізаціям, а й етичним нормам та академічної доброчесності [4].

Серед наявних питань, у подальшому дослідженні основну увагу буде приділено методам, що підвищують достовірність генеративного матеріалу. Одним із гарних прикладів практичної реалізації LLM-моделі є Khanmigo від Khan Academy. Готовий продукт, що включає у собі – AI-репетитора-асистента, який використовує GPT-серії та додаткові модифікації для навчання й підтримки учнів та вчителів[5]. Даний продукт використовує вихідний код ChatGPT від компанії OpenAI. Та надає зручний інтерфейс користувача, де запропоновані вже готові шаблони запитів, де потрібно вставити тільки потрібну інформацію.

Також існуює спеціалізований тип інтелектуальних навчальних систем (Intelligent Tutoring Systems), де найвідомішим прикладом є AutoTutor, що використовує комбінації класичних ITS-підходів з сучасними LLM. Програмне забезпечення дає змогу створювати діалоги, адаптивні підказки та індивідуальний контент.

1.2 Постановка задачі

На основі проведеного аналізу архітектур та методів проектування великих мовних моделей, підходів навчання, а також шляхів до інтеграції зовнішніх джерел знань, було сформовано мету дослідження, що визначає зміст цієї роботи. Загальна мета полягає у створенні генеративної системи, здатної формувати текстовий контент на основі наявних знань та забезпечувати його автоматизовану перевірку за допомогою семантичного графа. Для досягнення зазначеної мети необхідно розв’язати такі під задачі:

– обґрунтувати вибір архітектури генеративної моделі, враховуючи особливості сучасних LLM-підходів та їхню придатність до задач генерації структурованого контенту;

– визначити та проаналізувати методи інтеграції зовнішніх джерел знань, серед яких Retrieval-Augmented Generation (RAG), семантичні та онтологічні графи, а також інструментальні підходи виклику зовнішніх функцій;

– налаштувати структуру семантичного графа знань, визначити типи вузлів, відношення між ними та правила узгодження, необхідні для подальшої перевірки фактологічної коректності згенерованих тверджень;

– розробити архітектуру прототипу системи, яка включатиме модуль генерації тексту, модуль пошуку достовірних джерел, модуль семантичної перевірки та механізм узгодження отриманого результату з графом знань;

– реалізувати програмну модель системи із використанням сучасних бібліотек для мовних моделей та інструментів побудови графів знань;

– розробити механізм автоматизованої верифікації згенерованого контенту, що забезпечуватиме: ідентифікацію сутностей, порівняння тверджень із графом знань, виявлення та усунення суперечностей або некоректних фактів.

У подальшій роботі провести експериментальні дослідження функціональних можливостей системи, що включатимуть:

– порівняння генерації без зовнішніх даних та з використанням RAG;

– оцінювання впливу семантичного графа на рівень фактологічної коректності за допомогою ряду систем навчання та тестування, таких як NL4OPT, MAMO-EasyLP, MAMO-ComplexLP, IndustryOR;

– аналіз можливостей різних налаштувань моделі.

Сформулювати висновки та рекомендації щодо можливості використання розробленої системи та перспектив її розвитку, зокрема масштабування графа знань і застосування покращених методів перевірки.

У результаті очікується створення цілісної генеративної системи зі зручним користувацьким інтерфейсом, що поєднує LLM та семантичний граф знань і забезпечує підвищену точність, послідовність та контрольованість згенерованого текстового контенту.

2 АНАЛІЗ НАЯВНИХ ПІДХОДІВ ПОБУДОВИ СИСТЕМ ШТУЧНОГО ІНТЕЛЕКТУ

Протягом останніх десятиліть просліджується тенденція розвитку штучного інтелекту в розгалужену міждисциплінарну сферу, у межах якої співіснують десятки парадигм, методів та архітектур. Зі зростанням різноманіття моделей та їхніх функціональних можливостей виникла потреба у впорядкованій системі класифікації, або таксономії, штучного інтелекту. Однак саме побудова такої таксономії є нетривіальним завданням: різні джерела пропонують несумісні класифікаційні ознаки, змішують рівні абстракції або використовують різні терміни для позначення подібних явищ. Це дає ускладнення для аналізу та порівняння підходів, вибір методів для практичних задач та формування єдиної наукової мови.

У наукових джерелах простежується тенденція класифікувати ШІ одночасно за кількома незалежними вимірами. Наприклад, у дослідженні [6] пропонуються шість ключових осей: можливості (від вузького до суперінтелекту), функціональність (від реактивних до гіпотетичних самосвідомих систем), використані техніки (символьні, статистичні, еволюційні, нейронні, гібридні), домен застосування, середовище виконання, цільове призначення. Подібний багатовимірний опис дозволяє краще охопити різні покоління та підходи до ШІ, однак одночасно оголює головну проблему: штучний інтелект як технологія не має однієї «істинної» класифікації, оскільки:

– різні підходи оперують різними рівнями абстракції (функції, архітектури, алгоритми, прикладні задачі);

– категорії часто перетинаються (наприклад, нейросимвольні системи одночасно належать до кількох парадигм);

– розвиток моделей, особливо мовних, швидко робить попередні рамки застарілими;

– сучасні системи комбінують методи з різних гілок, – наприклад, поєднання мовних моделей, зовнішніх знань та інструментів навчання.

Таким чином, вивчення підходів до побудови ШІ потребує аналізу не лише існуючих моделей, але й структурної організації всієї дисципліни, її концептуальних обмежень та невирішених структурно-класифікаційних питань.

Далі буде розглянуто та проаналізовано саме сучасні великі мовні моделі. Дані ШІ системи додатково загострюють розглянуту проблему, адже поєднують різні властивості з кількох таксономічних вимірів, що можна описати як:

– системи обмеженої пам’яті;

– побудовані на глибокому навчанні (Deep Learning);

– часто рішення потребує інтеграції у гібридні архітектури для вирішення специфічних задач;

– працюють як генеративні моделі, що створюють складний семантичний контент.

Перший розділ даної магістерської роботи буде присвячено дослідженню таксономії великих мовних моделей та специфікації майбутнього технічного рішення, що відображено у темі роботи.

2.1 Архітектурні підходи до побудови великих мовних моделей

На момент написання даної роботи провідним архітектурним підходом для побудови великих мовних моделей є Transformer. Перевагами даного архітектурного підходу є поєднання високої паралельності обчислень та ефективного моделювання довгострокових залежностей. В основі лежить уявлення про LLM як «статистичний апроксиматор розподілу токенів у мовному просторі» [9]. Завдяки цьому Transformer здатен моделювати ймовірність появи наступного елемента, спираючись на складну систему латентних залежностей.

Конструктивним рішенням Transformer є архітектура типу encoder–decoder. Вона складається з двох симетричних стеків шарів, що взаємодіють між собою через механізми уваги (рисунок 2.1).

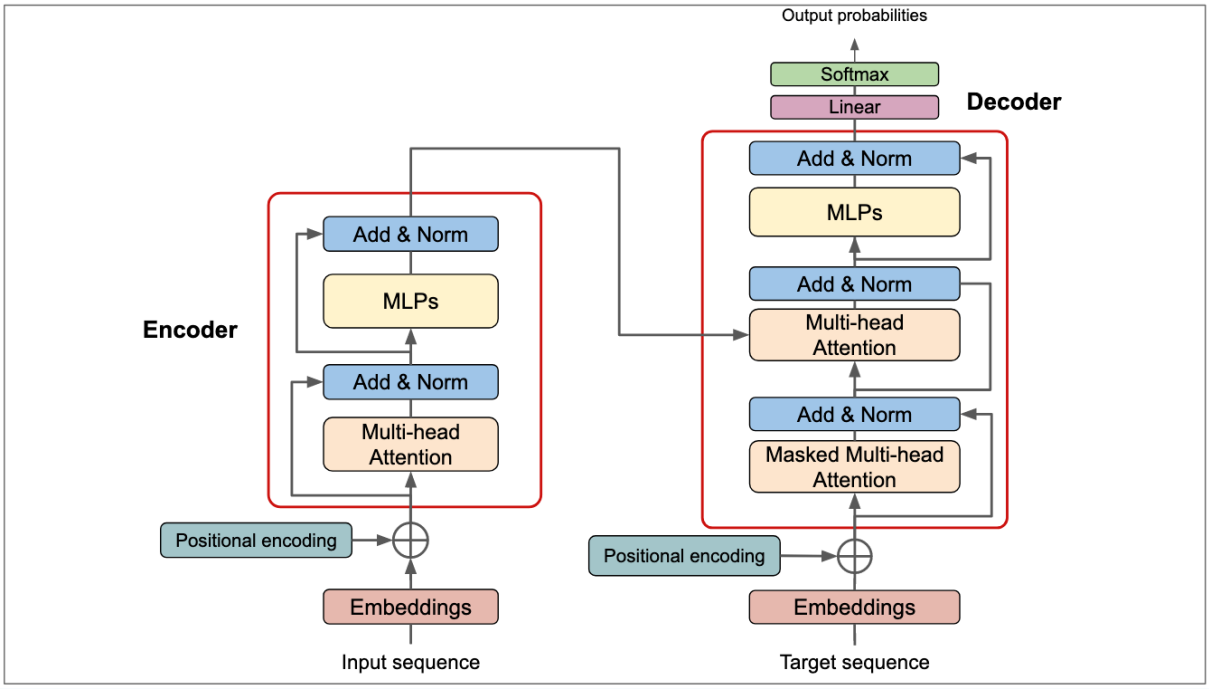

Рисунок 2.1 – Схематичне представлення архітектурного підходу Transformer – джерело [3]

Encoder відповідає за перетворення вхідної послідовності у набір контекстуально збагачених представлень. Кожен шар енкодеру містить:

а) multi-Head Self-Attention – механізм, який дозволяє кожному токену «бачити» всі інші токени в межах послідовності, незалежно від їхньої відстані;

– position-Wise Feed-Forward Network – двошаровий нелінійний блок, який поглиблює семантичне представлення;

– резидентні з’єднання та нормалізацію, що стабілізують і прискорюють навчання;

б) decoder формує вихідну послідовність авто-регресивно. Він складається з трьох типів уваги;

в) masked Self-Attention – обмежує увагу лише попередніми токенами, зберігаючи напрямок генерації;

г) encoder–Decoder Attention – дає можливість кожному позиційному представлення декодера «звертатися» до всіх енкодерних представлень, що дозволяє узгоджувати вхідний і вихідний текст;

д) feed-Forward Network для нелінійних перетворень.

Позиційні кодування (синусоїдальні, RoPE, ALiBi) компенсують відсутність рекурентності або згорток, цим самим забезпечують знання про порядок токенів. Завдяки поєднанню механізму «self-attention» або само-увага та паралельної обробки Transformer досягає мінімального шляху поширення інформації між токенами, що покращує здатність моделі вивчати довгі залежності [3].

Важливою перевагою Transformer є його масштабованість: збільшення кількості шарів, голів уваги або розмірності простору подань підсилює когнітивну складність моделі, але тільки за умови відповідних обчислювальних ресурсів.

Таким чином, архітектура Transformer – це не лише набір механізмів уваги, а цілісна структурна модель типу encoder–decoder, яка стала фундаментом для сучасних LLM і забезпечує можливість ефективного масштабування та високої продуктивності. Тому, дана система має практичну цінність для поточного дослідження створення навчального контенту.

Паралельно ведуться дослідження напрямів, що сфокусовані на покращення ефективності великих моделей. Амбітною є архітектура Mixture-of-Experts (MoE) – це підхід до масштабування нейронних мереж, що дозволяє збільшувати модельну ємність на порядки без пропорційного зростання обчислювальних витрат. Ключовий принцип MoE полягає у способі розподіленні обчислень між великою кількістю експертів (нейронних підмереж), з яких лише маленька підмножина активується для кожного вхідного прикладу (sparse routing). Даний підхід

дає змогу будувати моделі з мільярдами й навіть трильйонами параметрів, зберігаючи прийнятну вартість кроку розподілу навантаження. Основними складовими архітектури є:

а) набір експертів (Experts) – це колекція паралельних feed-forward нейронних мереж, кожна з власними параметрами. У класичній реалізації експерти однотипні, але виконуються незалежно;

б) гейтинг-мережа (Gating Network) – невелика нейромережа, що визначає, які експерти будуть активовані для конкретного вхідного вектора. Застосовується механізм Noisy Top-K, який вибирає k найбільш достовірних експертів та обчислює їх ваги за softmax. Це забезпечує sparse активацію та контрольований розподіл навантаження;

в) sparse маршрутинг (Sparsely-Gated Routing) – результат роботи гейтингу: модель обчислює вихід як зважену суму лише k експертів, обраних із загальної множини n. Це різко зменшує кількість обчислень і дозволяє масштабувати n до тисяч;

г) балансувальні регулятори;

– loss «importance» – вирівнює середні ваги гейтинг-мережі між експертами;

– loss «load» – вирівнює фактичну кількість прикладів, що надходять до кожного експерта. Обидва механізми запобігають «колапсу» до одного чи кількох експертів і зберігають стабільність тренування;

– розподілена реалізація – експерти можуть виконуватися на різних GPU/TPU, поєднуючи data parallel для основної моделі та model parallel для експертів. Це дозволяє масштабувати MoE до масштабних систем.

Перевага MoE полягає в тому, що модель отримує практично необмежену параметричну потужність, але в кожному кроці обробки виконується лише невелика її частина. Саме це робить MoE одним із ключових напрямів у розвитку ефективних LLM, а також часто

використовується як доповнення до архітектурного підходу Transformer. Та у фокусі даної роботи відкриває підвищення достовірності та надійності генерації начального контенту вже на потужних обчислювальних машинах, – чого вимагає даний підхід.

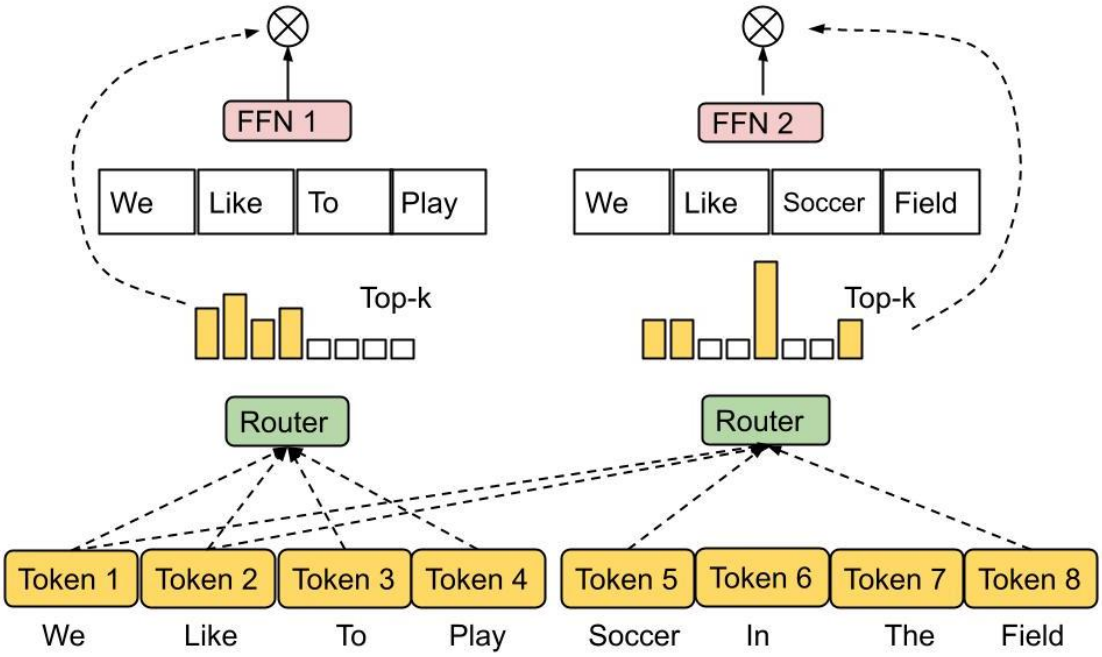

Рисунок 2.2 – Схематичне представлення архітектурного підходу Mixture-of-Experts – джерело [7]

2.2 Методологія навчання LLM

Розвиток великих мовних моделей ґрунтується на багатоступеневому процесі навчання, який забезпечує формування глибинних лінгвістичних репрезентацій, здатності до узагальнення та адаптації до прикладних задач. Відповідні етапи охоплюють попереднє тренування, додаткове навчання, узгодження моделі з людськими преференціями та механізми підтримки актуальності знань. Кожен із цих рівнів виконує окрему функцію у створенні генеративної системи, орієнтованої на побудову навчального контенту.

2.2.1 Pretraining (попереднє тренування)

Базовий етап формування LLM відбувається у вигляді самонавчання на великих корпусах неструктурованих текстів. Загальноприйнята постановка задачі полягає у прогнозуванні наступного токена на основі контексту:

$$P(x_t \mid x_{<t}), \tag{2.1}$$

де x_t– наступний елемент послідовності, а $x_{<t}$– попередні токени. Така авторегресивна модель дозволяє апроксимувати статистичний розподіл природної мови й формує базові лінгвістичні, енциклопедичні та структурні знання моделі.

У статті Large Language Models: A Survey [8] зазначено, що саме цей етап забезпечує фундаментальні можливості LLM, включно зі здатністю до in-context learning та узагальнення шаблонів навіть за широкої варіативності текстів.

Важливою умовою є якість даних: значний вплив на результативність мають очищення корпусу від шуму, дублювання й непослідовностей, що мінімізує накопичення спотворених мовних залежностей. У контексті систем генерації навчального контенту цей етап формує універсальні мовні компетенції, але не забезпечує спеціалізованої педагогічної поведінки – тому потребує наступних фаз адаптації.

2.2.2 Supervised Fine-Tuning (SFT) та Instruction Tuning

Supervised Fine-Tuning є надбудовою над навченою моделлю та передбачає додаткове навчання на парних прикладах «інструкція – відповідь». SFT є ключовим для формування керованої поведінки, що

дозволяє моделі слідувати структурованим завданням та створювати вихід, достовірного контексту запиту.

Instruction tuning – спеціалізована форма SFT, де модель навчається виконувати інструктивні запити різних типів. Цей етап значно покращує узгодженість моделей у діалогових сценаріях, знижує потребу у складному prompt-інжинірингу та забезпечує контрольований стиль відповіді [8].

У рамках розробки генеративної системи навчального контенту SFT дозволяє:

– задати бажаний стиль (академічний, педагогічний, пояснювальний);

– формувати структуру відповідей (визначення, приклади, пояснення, задачі);

– навчити модель формувати коректні дидактичні матеріали.

Якість SFT визначається різноманітністю й точністю інструкційних датасетів. Застосування методів Self-Instruct для масштабування навчальних прикладів є ефективним, але вимагає ручного або автоматизованого фільтрування для усунення помилкових або надлишкових зразків [8].

SFT модель має великий потенціал для реалізації поточної наукової роботи, бо надає можливість налаштовувати не тільки стиль відповідей, але і використовувати попередньо налаштовані датасети дидактичних структур, – що будуть надані на вибір для зручності налаштування кінцевих матеріалів у користувача.

2.2.3 RLHF/RLAIF та альтернативні методи узгодження

Фінальним етапом побудови сучасних LLM є вирівнювання моделі зі вподобаннями користувача, безпечними нормами та бажаною стилістикою відповідей. Найпоширенішим підходом є RLHF (Reinforcement Learning from Human Feedback), де модель оптимізується на основі людських оцінок. У цьому підході формують:

– набір відповідей-кандидатів;

– reward-модель (модель винагороди);

– оптимізацію політики моделі, зазвичай через PPO (Proximal Policy Optimization) – алгоритм підсилювального навчання (Reinforcement Learning), який використовують у RLHF для «фінального калібрування» мовних моделей.

RLHF значно покращує поведінкову відповідність, зменшує токсичність і робить відповіді більш корисними й структурованими. Однак цей підхід вимагає значних ресурсів і масштабних датасетів людських оцінок. Саме тому формується тенденція до заміни RLHF або його часткової автоматизації.

RLAIF (Reinforcement Learning from AI Feedback) є масштабованою альтернативою, де роль людини виконує високоякісна «вчительська» модель. Підхід демонструє порівнянні результати зі значним скороченням людських ресурсів. Альтернативи RLHF – сучасні методи, описані у статті [8], включають:

– DPO (Direct Preference Optimization) – напряму оптимізує політику на основі пар преференцій, без складної RL-парадигми; більш стабільний та ресурсоефективний;

– KTO (Kahneman-Tversky Optimization) – потребує тільки маркування типу «бажано/небажано», що робить його придатним в умовах обмеженої кількості даних.

У підсумку, проаналізувавши сильні та слабкі сторони даного підходу, для дипломної моделі, орієнтованої на навчальний контент, саме поєднання SFT та DPO є оптимальним компромісом між якістю та ресурсами, забезпечуючи одночасно коректність, безпечність та навчальну орієнтацію відповідей.

2.2.4 Continual Learning (додаткове навчання та довгострокова актуальність знань)

Оскільки навчальні матеріали оновлюються, а навчальні програми змінюються, LLM потребує механізмів підтримки актуальності без повного повторного тренування. Проблема катастрофічного забування робить пряме послідовне донавчання мало ефективним, тому сучасні підходи використовують параметр-ефективні методи та зовнішні структури даних. Основні механізми:

– PEFT, LoRA, адаптери – дозволяють оновлювати лише невелику частину параметрів, зберігаючи базові знання. Це мінімізує обчислювальні витрати та стабілізує навчання;

– RAG (Retrieval-Augmented Generation) – поєднання моделі з базою знань, що дає можливість використовувати оновлені джерела без перепідготовки. Це один із найефективніших способів зменшення «галюцинацій» у навчальному контенті;

– rehearsal-стратегія – періодичне повторення невеликої частини оригінального корпуса для запобігання втраті навичок [8].

Тому, у системах створення навчальних матеріалів найбільш доцільним є комбінування RAG для актуалізації фактів та PEFT-підходів для адаптації стилю і структури відповідей під конкретні навчальні програми.

Методологія навчання великих мовних моделей у напрямку генерації начального контенту є багаторівневим процесом, де:

– попереднє тренування забезпечує мовні компетенції;

– SFT/instruction tuning формує керованість і педагогічну стилістику;

– RLHF/DPO/RLAIF узгоджують поведінку моделі з потребами користувача;

– continual learning і RAG підтримують актуальність та зменшують галюцинації;

– процесу навчання з підкріпленням на основі відгуку спеціаліста (рисунок 2.3).

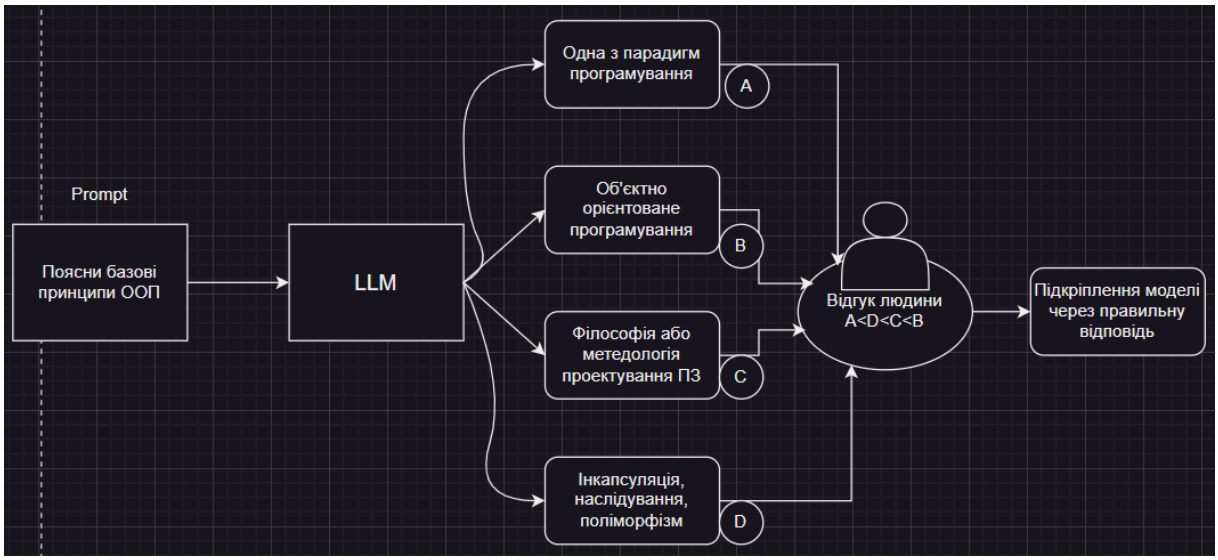

Рисунок 2.3 – Приклад процесу навчання з підкріпленням на основі відгуку спеціаліста (виконано самостійно)

2.3 Інтеграція зовнішніх джерел знань

Однією з ключових проблем великих мовних моделей є схильність до генерації недостовірних або вигаданих фактів. Це явище зумовлене статистичною природою моделей, які опираються на внутрішні параметри як на імпліцитне сховище знань. Відсутність прямого доступу до актуальної, перевіюваної та структурованої інформації ускладнює застосування LLM у знання-інтенсивних задачах, зокрема в освіті, де точність даних є критично важливою. У відповідь на це сучасні дослідження пропонують підходи, що дозволяють комбінувати параметричну пам’ять моделі із зовнішніми джерелами знань.

2.3.1 Retrieval-Augmented Generation (RAG)

Підхід Retrieval-Augmented Generation передбачає поєднання мовної моделі з нефіксованою базою знань, що зберігається у вигляді індексу документів або векторного сховища. У класичній архітектурі RAG модель використовує модуль пошуку для отримання достовірних фрагментів тексту та генератор для побудови відповіді на основі цих фрагментів[9]. У такому випадку LLM перестає покладатися виключно на власні параметри та отримує можливість:

– оновлювати «знання» шляхом модифікації зовнішнього індексу, не потребуючи повторного тренування моделі;

– підвищувати фактичну точність відповідей, використовуючи достовірні текстові джерела;

– зменшувати кількість «галюцинацій» під час генерації відповідей.

Завдяки цьому підходу RAG-моделі демонструють суттєве покращення як у задачах відкритого запитання, так і в генеративних завданнях, оскільки вбудований механізм пошуку забезпечує коректну зовнішню опору на факти, що дає значне покращення генерації контенту, базуючись на вже наданих користувачем методичних матеріалах.

2.3.2 Інтеграція семантичних та онтологічних графів знань

Сучасні моделі великої мови (LLM) демонструють високу здатність до генерації зв’язного тексту, водночас зберігаючи помітну уразливість щодо генерації недостовірних або неправдивих тверджень (так званих «галюцинацій»). Інтеграція структурованих репрезентацій знань – семантичних графів (knowledge graphs, KG) та формальних онтологій – пропонує системний підхід до зниження ризику таких явищ. Семантичні графи забезпечують машинозчитувану структуру фактів і відношень, а

онтології вводять типову таксономію, аксіоми та логічні обмеження, що разом створює основу для перевірки консистентності та простежуваності наукових тверджень.

Семантичний граф знань формально розглядається як орієнтований граф:

$$G=(V,E), \tag{2.2}$$

де V – множина сутностей (entities);

E – множина ребер або триплетів у форматі «суб'єкт–предикат–об'єкт».

Онтологія задає схему S, що містить класи сутностей, властивості, обмеження (типи, кардинальності, ролі) та логічні аксіоми.

Інтеграція структурованих репрезентацій знань – семантичних графів та доменних онтологій – з великими мовними моделями розглядається як один із ключових підходів до підвищення достовірності та відтворюваності генерованого наукового тексту. На відміну від методів, що покладаються виключно на статистичні закономірності текстового корпусу, залучення формальних знань дає змогу запровадити механізми верифікації фактів, логічної перевірки тверджень і простежуваності джерел.

У класичному вигляді семантичний граф складається з триплетів «суб'єкт–предикат–об'єкт». Якщо до цього додається онтологічна схема, яка задає класи сутностей, їхні властивості, типи, обмеження та аксіоми, тоді така комбінована структура стає придатною для формального reasoning, валідації структурного й семантичного контексту. Відповідно, генерація тексту може бути «прив'язана» до фактів, які вже формалізовані в KG, і твердження – перевірені на відповідність онтологічним правилам.

Перевага цього підходу проявляється у кількох аспектах. По-перше, він підвищує фактологічну точність: мовна модель з доступом до KG може звертатися до надійних фактів, а не вигадувати, спираючись лише на текстові патерни. У роботі Give Us the Facts: Enhancing Large Language

Models with Knowledge Graphs for Fact-aware Language Modeling автори показують, що KG-підтримувані моделі суттєво перевищують за якістю моделі, які покладаються лише на статистичні зв'язки, коли йдеться про твердження, що потребують фактологічного обґрунтування [10].

По-друге, інтеграція KG дає змогу впровадити механізм provenance – тобто прив'язку кожного твердження до конкретних вузлів або триплетів у графі. Це підвищує прозорість, дає змогу перевірити джерела, повторно перевірити висновки або змінити граф знань, якщо виявлено помилку або неточність.

По-третє, якщо онтологія визначає формальні аксіоми (наприклад, типи сутностей, діапазони значень, кардинальності відношень), система отримує засоби для виявлення семантичних або логічних невідповідностей, які могли би виникнути у процесі генерації. Це знижує ризик непослідовностей, протиріч чи неправильно сформульованих тверджень.

Четвертий аспект – підтримка багатокрокового логічного виведення (multi-hop reasoning). Семантичні графи дозволяють поєднувати віддалені факти у вигляді ланцюжків відношень, що дає змогу будувати складну наукову аргументацію. У цьому плані важливою є робота KG-GPT: A General Framework for Reasoning on Knowledge Graphs Using Large Language Models – тут показано, як LLM може бути задіяна в pipeline, що включає сегментацію тексту, пошук достовірного підграфа та логічне виведення, з перевіркою отриманих висновків на основі KG [11].

З іншого боку, цей підхід має суттєві обмеження. Перше – неповнота графів знань: навіть потужні та добре опрацьовані KG не покривають усіх фактів домену, особливо якщо мова про спеціалізовані наукові галузі. Якщо в графі відсутні достовірні факти, модель може знову звернутися до внутрішніх ймовірнісних механізмів, що зводить нанівець переваги KG-підходу або частково повертає проблему «галюцинацій».

Друге – складність інтеграції різнорідних джерел. Об'єднання відкритих графів (наприклад загальних) із доменними онтологіями

передбачає процедури вирівнювання сутностей, узгодження таксономій, нормалізації предикатів і, можливо, узгодження одиниць вимірювання чи семантичних інтерпретацій. Такий процес knowledge fusion є нетривіальним і часто потребує значних експертних і обчислювальних ресурсів.

Третє обмеження пов’язане з продуктивністю: великі графи знань із багатьма сутностями і зв’язками вимагають ефективних механізмів індексації, швидкого retrieval підграфів і оптимізації reasoning. Багатокрокове логічне виведення може бути дорогим зі сторони обчислень або навіть експоненційно масштабуватися з ростом глибини reasoning, що потребує запровадження евристик або обмежень на глибину/громіздкість висновків.

Оцінювання ефективності інтеграції KG з LLM має передбачати як кількісні, так і якісні метрики. До кількісних відносяться: точність фактологічної перевірки (наприклад, метрики типу FEVER), показник grounding rate (відсоток тверджень, які можна прив’язати до вузлів графа), частка порушень онтологічних аксіом (ontology violation rate). До якісних – експертний аудит (human evaluation), призначений для перевірки логічної послідовності, коректності трактувань, відповідності термінології й доменній логіці. Комбінація автоматичних метрик і професійної рецензії дає змогу всебічно оцінити вплив інтеграції графів знань на наукову коректність тексту.

Для практичної імплементації слід передбачити: розробку або адаптацію доменної онтології; надійний механізм entity linking для коректного зіставлення фрагментів тексту з вузлами графа; ефективний subgraph-retrieval із пріоритизацією достовірних вузлів; компонент логічного reasoning або валідації, який перевіряє відповідність згенерованих тверджень аксіомам онтології; за можливості – human-in-the-loop для доповнення та корекції графа в разі неповноти або суперечностей.

Загалом, інтеграція семантичних графів і онтологій із генерацією текстів LLM надає основу для створення верифікованих, прозорих, логічно

узгоджених наукових документів. Водночас її застосування вимагає ретельного проєктування, ресурсів і постійної підтримки графів. За належної реалізації цей підхід є одним із найперспективніших напрямів для формування достовірних і реплікуваних наукових текстів.

2.3.3 Toolformer-підходи та виклик зовнішніх інструментів

До класу методів побудови LLM-систем також відноситься Toolformer. Такий підхід дозволяє моделі автономно ініціювати взаємодію із зовнішніми інструментами, такими як: API-сервіси, бази даних, системи обчислень, пошукові механізми, тощо. Підхід Toolformer був представлений у роботі «Language Models Can Teach Themselves to Use Tools» [12].

У праці було продемонстровано, що мовна модель може самостійно навчитися визначати:

- який інструмент потрібно викликати;
- які аргументи передати;
- коли доцільно використовувати зовнішні дані.

Ключовою ознакою підходу Toolformer є створення анотацій про те, коли та як слід використовувати зовнішні інструменти. Замість потреби у великих розмічених наборах даних з API-викликами, Toolformer базується на self-supervised data labeling, де модель генерує потенційні виклики API у контексті, оцінює їх корисність і лише після цього включає до фінального набору даних для додаткового навчання. Такий підхід мінімізує потребу у людській участі та забезпечує масштабованість.

Порівняно з класичними мовними моделями, цей підхід обіцяє універсальність використання, компенсує системні слабкості LLM-систем такі як «галюцинації», робота із даними у реальному часі, автономність інтеграції нових інструментів. На практиці використання цей метод не здобув широкої популярності серед розробників мовних моделей, адже концептуальна ідея не здобула стійкого практичного рішення та потребує

значних внесків для попереднього налаштування. Саме впровадження та підтримка Toolformer-підходів дійсно лише на масштабних проектах з достатньою кількістю людських ресурсів. Або навпаки – для виконання вузькоспеціалізованих тестів для штучного інтелекту [13], про які далі буде йти мова у розділі 2.6.

Так як мета кваліфікаційної роботи має більш практичне значення у генерації навчального матеріалу для перевірки знань, Toolformer-підходи не зможуть забезпечити бажану гнучкість та повну відповідність даним, що надає користувач.

2.3.4 Evidentiality-guided Generation як напрям підвищення достовірності

Окремим підходом до покращення фактологічної якості генерації є методи, що враховують евіденційність – показник цитованості та правдоподібності джерела доказів. Такі моделі не просто використовують надані документи, а й вчаться оцінювати, чи справді фрагмент містить інформацію, необхідну для коректної відповіді. Запропонований [14] механізм додає до генеративного процесу передбачення евіденційності, що дозволяє суттєво зменшити вплив недостовірних або шумових документів та знизити ймовірність генерації неправильних тверджень [14].

Таким чином, сучасні LLM-системи дедалі частіше поєднують у собі параметричні знання з різними формами зовнішніх джерел:

- текстові колекції та векторні бази (RAG);
- семантичні графи та онтології (структуровані знання);
- зовнішні інструменти та API (Toolformer-підходи);
- модулі оцінки доказовості (фільтрація достовірних доказів).

Отже, для задачі створення навчального контенту такі підходи мають особливе значення: вони забезпечують точність, відтворюваність і відповідність сучасного матеріалу науковим джерелам. Інтеграція

зовнішніх джерел знань у генеративну модель є необхідною умовою підвищення її надійності та застосовності у сфері освіти.

2.4 Триплети знань (суб’єкт–предикат–об'єкт) в LLM

Триплет визначається як атомарне твердження про ресурс (суб’єкт), його властивість або зв’язок (предикат) та значення або інший ресурс (об’єкт). У моделі семантичного графа множина таких триплетів утворює семантичну мережу, де вузли – це сутності, а ребра – відношення між ними. Використання формалізованих схем (онтологій) і стандартних серіалізацій (наприклад, Turtle, RDF/XML) дозволяє ефективно зберігати, запитувати (SPARQL) і верифікувати такі дані [15].

Нейромережеві підходи й великі мовні моделі розширили можливості автоматичного витягання, доповнення та використання триплетів у кількох напрямах:

– екстракція та нормалізація: LLM демонструють високу ефективність у видобуванні сутностей і відношень з нерозміченого тексту, перетворюючи природньомовні висловлювання у триплети (zero-/few-shot витягання). Це дає змогу швидко створювати початкові графи знань зі статей, підручників та інших навчальних матеріалів [16];

– поповнення та завершення графа: методи вбудовань знаннєвих графів (knowledge graph embeddings), такі як TransE, моделюють відношення між сутностями в просторі векторних представлень, що сприяє автоматичному передбаченню відсутніх триплетів (completion). Поєднання таких вбудованих методів із LLM дозволяє поєднувати символічну точність і статистичну гнучкість;

– покращення генерації через структурований контекст (Graph-augmented generation): інтегровані підходи, де семантичний граф або набір триплетів використовується як контекст або «м'який промпт» для LLM,

підвищують точність фактів, консистентність і обґрунтованість генерованого тексту (наприклад, у рамці RAG/GraphRAG, GraphCheck) [16].

У контексті автоматизованого створення навчального матеріалу триплети виконують кілька прикладних ролей:

– формалізація навчальної інформації: навчальні факти та політики можуть зберігатися як триплети, що полегшує побудову модулів знань для різних предметних областей [17];

– персоналізація і адаптація: семантичний граф дозволяє пов’язувати навчальні концепти зі складністю, передумовами і цілями учня; LLM може на їхній основі генерувати адаптовані пояснення або питання [17];

– генерація структури курсу і оцінювань: триплети дозволяють алгоритмічно формувати навчальні траєкторії (предумови → концепти → вправи), що потім перетворюються LLM у тексти лекцій, приклади та тести.

У підсумку, можна виділити такі переваги у викроистанні даного підходу:

– підвищена прозорість і відстежуваність джерел знання (кожне твердження має явне семантичне представлення) [15];

– краща інтеграція різнорідних джерел завдяки уніфікованій моделі триплетів і стандартам (RDF/SPARQL) [15];

– сумісність із методами машинного навчання (вбудування, GNN) для індуктивного виведення та раціоналізації [18].

У той же час необхідно пам’ятати про виклики, що виникають разом з використанням триплетів:

– шум і невизначеність витягнутих триплетів: помилки в екстракції призводять до некоректних фактів у навчальному контенті; потрібні механізми валідації (онтологічні правила, людська експертиза, автоматичні перевірки);

– канонізація і узгодження: різні вирази одного і того ж факту потребують нормалізації сутностей і відношень (entity linking, relation canonicalization);

– масштаб і продуктивність: великі графи вимагають оптимізованих сховищ (triple stores) і стратегій індексації для інтерактивного використання з LLM [16].

При побудові генеративної моделі для створення навчального контенту на основі великих мовних моделей семантичний граф знань і, зокрема, представлення знань у вигляді триплетів (суб'єкт–предикат–об'єкт) виконують критично важливу роль. Триплети забезпечують формальну, інтероперабельну структуру для збереження фактів, полегшують механізми пошуку та верифікації, служать мостом між символічними знаннями та статистичними моделями і дозволяють створювати більш достовірний, персоналізований та пояснюваний навчальний контент. Ефективний конвеєр – від витягання триплетів через LLM, їх канонізації і вбудовування до кінцевої генерації навчальних текстів – є перспективним напрямом подальших досліджень і практичної реалізації.

2.5 Тестування LLM-систем

Застосування великих мовних моделей для автоматичної побудови математичних моделей оптимізації з природномовних описів швидко перетворюється на активну дослідницьку ділянку. Основні виклики – точна інтерпретація умов задачі, коректна формалізація змінних та обмежень, а також гарантування сумісності з наявними розв'язувачами. Для систематичної оцінки були запропоновані кілька бенчмарків: NL4Opt (NeurIPS competition), MAMO (EasyLP, ComplexLP) і новіші індустріальні датасети (IndustryOR). Ці набори даних дозволяють порівнювати архітектури LLM і підходи верифікації моделей [19].

Формулювання задач з природної мови. NL4Opt (Natural Language for Optimization) виник як конкурс на NeurIPS і зосереджений на перетворенні текстового опису задач лінійного програмування в формальні моделі. Набір містить близько 289 задач і слугує «конкретним» тестом здатності моделі правильно ідентифікувати змінні, цільову функцію та обмеження у різних стилях опису. Основний критерій оцінки – коректність відтвореної моделі (формулювання), інколи доповнений виконуваністю розв'язку через розв'язувач. NL4Opt стимулював розвиток шаблонних підходів, синтаксичного парсингу та fine-tuning для задач OP [20].

MAMO: процесна оцінка моделювання (EasyLP / ComplexLP). MAMO (Mathematical Modeling) запропоновано як більш широке й глибоке середовище для дослідження «процесних» аспектів моделювання: не лише правильно чи ні, а як саме модель будує рішення (розпізнавання компонентів, обґрунтування, проміжні кроки). MAMO поділено на EasyLP (~652 прості LP-задачі) та ComplexLP (~211 складніших задач) – це дозволяє вимірювати як 0-shot/1-shot продуктивність, так і поведінку при ітеративній взаємодії з моделлю.

Дослідження показують, що сучасні LLM (наприклад, GPT-4 варіанти) досягають значно вищої точності на EasyLP, тоді як на ComplexLP їхня продуктивність падає через семантичну неоднозначність і складніші обмеження. MAMO також пропонує методи компонент-рівневої перевірки (перевірка змінних, обмежень окремо), що підвищує надійність валідації [21].

IndustryOR: індустріальна перевірка. IndustryOR – це спроба наблизити оцінку до реальних промислових сценаріїв: задачі тут часто більші за розміром, містять специфічну бізнес-термінологію, додаткові практичні обмеження (логістика, резерви, політика) і шукають не лише синтаксичну правильність, а й практичну корисність формулювання за участю розв'язувачів та експертів-верифікаторів. IndustryOR включено до порівнянь у кількох недавніх роботах, і показано, що спеціалізовані

підходи (синтез даних, інструктування, інтеграція з зовнішніми перевірками) значно підвищують достовірністьрезультатів у промислових умовах [22].

Стандартні метрики й підходи валідації включають:

– формальну коректність (логічна відповідність моделі опису);

– компонентну точність (правильність змінних, обмежень, коефіцієнтів);

– функціональну перевірку (чи дає модель розв’язок при запуску розв’язувача);

– стійкість до стилістичних змін тексту;

– інтерпретованість/обґрунтування (наскільки модель пояснює свої кроки).

MAMO підкреслює важливість процесних перевірок – часткова валідація компонентів дешевша в обчислювальному сенсі і більш інформативна для навчання. NL4Opt надає сильний корпус для тестування кінцевої коректності, тоді як IndustryOR додає тест «практичності» [21].

Серед найуспішніших підходів покращення продуктивності LLM для OP-задач: спеціалізований fine-tuning на розмічених прикладах (дані з NL4Opt/MAMO), синтез даних і інверсне генерування (OR-Instruct/ORLM підходи для масштабування даних), модульна перевірка, мульти-етапні агенти (декомпозиція задачі: парсинг, моделінг, код/вивід, відлагодження), інтеграція з символьними інструментами для фактичної верифікації.

Роботи 2024–2025 демонструють, що поєднання кількох стратегій (наприклад, ітеративний синтез даних + component checkers) дає найкращий приріст [22].

Незважаючи на успіхи, залишаються відкриті питання: загальна надійність у складних/шумних описах, виявлення і виправлення помилок у згенерованих моделях, рішення-пояснення для інженерів-користувачів, масштабування на великі індустріальні задачі без втрати точності, стандартизовані метрики для порівняння (особливо між академічними та

індустріальними наборами). Синтез нових, багатовимірних бенчмарків та розробка ефективних автономних верифікаторів – ключові напрями роботи 2025 року [23].

Тому, NL4Opt, MAMO і IndustryOR формують комплементарний набір інструментів для всебічної оцінки здатності LLM до формалізації задач оптимізації. Поєднання компонентної валідації, синтезованих даних та інтеграції з розв’язувачами – найперспективніша практика для підвищення надійності моделей у цій домені.

3 ПОБУДОВА ДОДАТКУ ДЛЯ СТВОРЕННЯ НАВЧАЛЬНОГО КОНТЕНТУ

3.1 Завантаження знань про предметну область до системи

Процес завантаження доменних знань є початковим етапом роботи системи та забезпечує її інформаційне наповнення. Система повинна правильно інтерпретувати навчальні матеріали, надані користувачем, а також забезпечувати їх подальше використання як під час генерації контенту, так і під час семантичної верифікації. Для цього реалізовано механізм оброблення вхідних документів у різних форматах та подальшої побудови двох взаємопов’язаних структур: векторного індексу RAG та семантичного графа знань.

3.1.1 Обробка завантажених даних

Система підтримує автоматизоване завантаження навчальних матеріалів користувача та їх подальшу інтеграцію у внутрішню базу знань. На даному етапі забезпечується приймання документів у форматах TXT, PDF та DOCX, що охоплюють основні типи навчальних ресурсів. Після завантаження файл проходить процедуру уніфікованого оброблення: визначається його тип, виконується повна текстова екстракція та перетворення в однорідне текстове представлення. Кожен з підтримуваних форматів оброблюється за унікальною для кожного логікою:

– txt: файли зчитуються у вихідному вигляді, без додаткового структурування або попередньої обробки, з подальшим очищенням від неінформативних символів:

– pdf: здійснюється аналіз сторінкової структури документа з використанням вбудованого механізму екстракції тексту; у випадку

наявності растрових елементів додатково виконується процедура автоматичного розпізнавання тексту (OCR);

– docx: документ обробляється шляхом парсингу внутрішнього XML-представлення з урахуванням логічних блоків, стилів оформлення, таблиць, списків та виносок, що забезпечує коректне відтворення текстового змісту.

На завершальному етапі виконується очищення отриманого текстового масиву від службових елементів форматування, артефактів конвертації та неінформативних символів. Результатом цього процесу є стандартизований raw text, який передається у підсистеми побудови RAG-індексу та графа знань. Завдяки такій процедурі забезпечується коректність подальших операцій із семантичним аналізом та інтеграцією змісту в структури знань системи.

3.1.2 Побудова Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) у запропонованій системі реалізовано шляхом побудови векторного індексу навчальних матеріалів користувача, що дозволяє швидко знаходити релевантні фрагменти тексту для генеративної моделі.

Перед формуванням векторного індексу всі надані документи конвертуються у текстовий формат та очищуються від неінформативних символів. Для забезпечення ефективного пошуку текст поділяється на сегменти (chunks) фіксованого розміру, у реалізації використовуються фрагменти приблизно по 500 символів. Така сегментація дозволяє зберегти локальний контекст кожного шматка тексту та забезпечує оптимальний баланс між точністю пошуку та розміром індексу.

Кожен сегмент тексту перетворюється у векторне представлення за допомогою SentenceTransformer («all-mpnet-base-v2»). Ембеддинги зберігаються у форматі PyTorch tensor, що дозволяє виконувати обчислення

на GPU та підвищує швидкодію генерації векторних репрезентацій. У разі повторного використання індексу система може завантажити вже збережені ембеддинги, що економить обчислювальні ресурси.

Для забезпечення швидкого пошуку релевантних фрагментів тексту ембеддинги конвертуються у формат NumPy та додаються до індексу FAISS. Індекс FAISS реалізує пошук найближчих векторів за евклідовою відстанню (L2), що забезпечує ефективне відновлення найбільш релевантних сегментів для будь-якого запиту користувача. Пошук виконується шляхом обчислення відстаней між вектором запиту та усіма векторними представленнями сегментів тексту, після чого повертаються найближчі фрагменти (рисунок 3.1).

```python
def build_rag_index(file_path, chunk_size=500, batch_size=64):
    device = "cuda" if torch.cuda.is_available() else "cpu"
    embedder = SentenceTransformer("all-mpnet-base-v2", device=device)

    with open(file_path, "r", encoding="utf-8") as f:
        knowledge_text = f.read()

    chunks = textwrap.wrap(knowledge_text, width=chunk_size)

    embeddings = embedder.encode(
        chunks,
        batch_size=batch_size,
        convert_to_tensor=True,
        device=device
    )

    embeddings_cpu = embeddings.cpu().numpy()
    dimension = embeddings_cpu.shape[1]

    index = faiss.IndexFlatL2(dimension)
    index.add(embeddings_cpu)

    return index, chunks, embedder
```

Рисунок 3.1 – Функція побудови RAG на основі текстових даних (виконано самостійно)

Під час обробки запиту користувача текст запиту кодується у вектор за допомогою тієї ж моделі SentenceTransformer. Далі здійснюється пошук у FAISS-індексі для визначення топ-K найбільш релевантних сегментів (рисунок 3.2).

```python
def search_knowledge(query, index, chunks, embedder, top_k=3):
    q_emb = embedder.encode([query])
    _, indices = index.search(q_emb, top_k)
    return [chunks[i] for i in indices[0]]
```

Рисунок 3.2 – Функція пошуку релевантних сегментів за запитом (виконано самостійно)

Результатом є набір текстових фрагментів, які зберігають семантичну близькість до запиту та можуть використовуватися для подальшої генерації.

3.1.3 Побудова семантичного графа знань

Побудова семантичного графа знань передбачає трансформацію неструктурованого тексту у формалізоване представлення у вигляді вузлів (сутностей) та ребер (відношень). Цей процес реалізовано як послідовний пайплайн, що включає попередню обробку текстів, їх сегментацію, вилучення триплетів, канонізацію та формування графової структури.

На першому етапі система зчитує колекцію текстових документів у форматі .txt. Кожен документ очищується від небажаних символів, уніфікуються пробіли, видаляються некоректні та непечатні символи (рисунок 3.3). Це забезпечує узгоджену структуру даних перед обробкою моделями NLP.

```python
def load_and_clean_texts(path="./texts"):
    texts = []
    for f in Path(path).glob("*.txt"):
        t = f.read_text(encoding="utf-8", errors="ignore")
        t = re.sub(r"\s+", " ", t).strip()
        texts.append(t)
    return texts
```

Рисунок 3.3 – Функція завантаження та очищення текстових даних (виконано самостійно)

Після очищення корпусу текстів здійснюється обробка кожного документа засобами природної мовної обробки (NLP). На першому етапі виконується coreference resolution – процедура заміни займенників на відповідні референти. Для цього використовується бібліотека coreferee, інтегрована у конвеєр моделі spaCy (рисунок 3.4). Coreference resolution дозволяє забезпечити цілісність семантичних зв'язків та уникнути ситуацій, коли сутність у тексті позначена займенником, що ускладнює її подальше використання під час побудови триплетів.

```
def resolve_coreferences(text):
    doc = nlp(text)
    tokens = [t.text for t in doc]
    chains = doc._.coref_chains
    if chains:
        for chain in chains:
            main = chain.main
            for m in chain:
                if m != main:
                    for i in range(m.start, m.end):
                        tokens[i] = main.text
    return " ".join(tokens)
```

Рисунок 3.4 – Функція заміни займенників на відповідні референти (виконано самостійно)

Після заміни кореференційних згадок текст розбивається на речення засобами spaCy. Речення виступають базовою одиницею аналізу, оскільки моделі залежнісного синтаксичного розбору працюють на рівні окремих пропозицій. Для кожного речення здійснюється автоматичне визначення двох компонентів: суб'єкта та об'єкта (рисунок 3.5). Їх виділення виконується через аналіз syntactic dependencies, зокрема правил, що охоплюють залежності типу nsubj, dobj, iobj тощо. Це дозволяє коректно витягувати кандидатні сутності навіть у випадках складних синтаксичних конструкцій.

```python
def extract_entities(sentence):
    ent1 = ""
    ent2 = ""
    prv_dep = ""
    prv_text = ""
    prefix = ""
    modifier = ""
    for tok in nlp(sentence):
        if tok.dep_ != "punct":
            if tok.dep_ == "compound":
                prefix = tok.text if prv_dep != "compound" else prv_text + " " + tok.text
            if tok.dep_.endswith("mod"):
                modifier = tok.text
            if "subj" in tok.dep_:
                ent1 = (modifier + " " + prefix + " " + tok.text).strip()
                prefix = ""
                modifier = ""
            if "obj" in tok.dep_:
                ent2 = (modifier + " " + prefix + " " + tok.text).strip()
        prv_dep = tok.dep_
        prv_text = tok.text
    return ent1, ent2
```

Рисунок 3.5 – Функція виділення компонентів із речення (виконано самостійно)

Визначення предиката здійснюється окремо, за допомогою механізму spaCy Matcher. На цьому етапі формується проміжний набір триплетів виду:

(суб’єкт – предикат – об’єкт),

кожен з яких відповідає окремому реченню вихідного документа.

У моделі формується шаблон, який визначає головний дієслівний вузол (ROOT), а також пов’язані з ним можливі препозиції, агенти або прикметникові модифікатори. Виділений фрагмент представляє собою семантичне відношення між суб’єктом та об’єктом.

Реалізація механізму екстракції триплетів представлена на рисунку 3.6.

На основі сформованого набору триплетів формується таблиця знань. Вона включає три колонки: source, edge, target, що відповідають суб’єкту, предикату та об’єкту відповідно. Таблиця є проміжним представленням перед побудовою графа (рисунок 3.7).

```python
def extract_relation(sentence):
    doc = nlp(sentence)
    matcher = Matcher(nlp.vocab)
    pattern = [
        {"DEP": "ROOT"},
        {"DEP": "prep", "OP": "?"},
        {"DEP": "agent", "OP": "?"},
        {"POS": "ADJ", "OP": "?"}
    ]
    matcher.add("rel", [pattern])
    matches = matcher(doc)
    if not matches:
        return None
    span = doc[matches[-1][1]:matches[-1][2]]
    return span.text
```

Рисунок 3.6 – Функція виділення семантичного відношення у реченні (виконано самостійно)

	source	target	edge
0	gravity	ground	keeps
1	gravity	rivers	affects
2	gravity	even galaxies	is responsible
3	who	17th century	was
4	object	other object	proposed
5	why objects	ground	allowed
6	work	two centuries	formed
7	20th albert einstein	general relativity	introduced
8	gravity	mass space	showed
9	other objects	curves	warp
10	movement	gravitational attraction	observed as gravitational
11	theory	black holes	confirmed through
12	gravity	fundamental nature	is
13	scientists	fundamental quantum mechanics	continue
14	research	other string nature	seeks
15	scientists	intricate matter	helps
16	many questions	better gravity	gained
17	understanding gravity	deeper reality	continues

Рисунок 3.7 – Таблиця знань отримана шляхом виділення триплетів із тексту (виконано самостійно)

Перед етапом побудови графа проводиться додатковий етап фільтрації займенників та службових слів, який реалізовано у вигляді окремої перевірки валідності сутності. На цьому етапі кожен кандидантний суб'єкт і об'єкт порівнюється зі списком заборонених лексем, що включає особові та вказівні займенники. Сутності, які містять такі слова, відкидаються й не потрапляють до фінального набору триплетів. Це забезпечує підвищену чистоту семантичної структури, гарантує відсутність узлових вершин на кшталт «he», «she», «it», «this», «those» та усуває помилки, що не вдалося повністю компенсувати під час coreference resolution.

Фінальним етапом є побудова семантичного графа знань засобами бібліотеки NetworkX. Для цього використовується орієнтований мультиграф (MultiDiGraph), який підтримує декілька ребер між одними й тими самими вузлами, що дозволяє зберігати різні типи відношень між однаковими сутностями (рисунок 3.8). Ребра зберігають відповідний предикат, а вузли відповідають канонізованим сутностям, отриманим під час обробки текстів. Граф може бути візуалізований за допомогою вбудованих функцій NetworkX – зокрема, алгоритму розташування spring layout, який розподіляє вузли у просторі за принципом моделі сил.

```
def build_graph(df):
    G = nx.from_pandas_edgelist(
        df,
        "source",
        "target",
        edge_attr=True,
        create_using=nx.MultiDiGraph()
    )
    return G
```

Рисунок 3.8 – Функція побудови графа знань (виконано самостійно)

Для кожного ребра зберігається додаткова метаінформація:

– джерело тексту;

– ідентифікатор фрагмента;

– рівень впевненості моделі;

– контекст (фрагмент речення).

Ця метаінформація дозволяє не лише точно відстежувати походження та достовірність кожного факту, але й оцінювати рівень надійності зв’язків у графі, що є критично важливим для подальшої генерації навчального контенту. Завдяки збереженню контексту та рівня впевненості моделі можна проводити глибший аналіз якості даних, оперативно виявляти потенційно некоректні факти та забезпечувати коректну інтеграцію з механізмами семантичної верифікації.

Приклад візуалізації побудованого розробленою системою графа на основі тексту що містить загальну інформацію про силу гравітації можна побачити на рисунку 3.9.

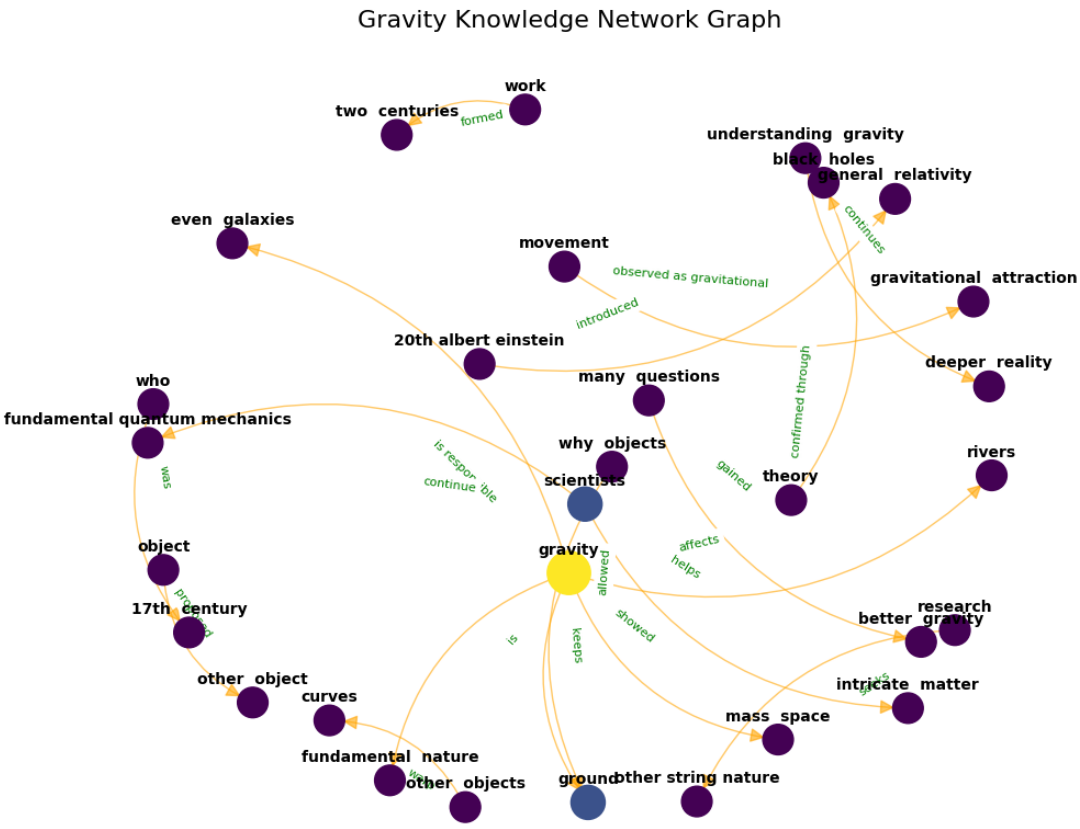

Рисунок 3.9 – Візуалізація побудованого графа знань (виконано самостійно)

Таким чином, побудований конвеєр забезпечує повний цикл автоматичного вилучення та структурування знань: від очищення та корекції тексту до побудови орієнтованого графа, який відображає семантичні зв’язки між сутностями на основі реальних текстових даних.

В кінці роботи пайплайну згенерований граф зберігається у форматі GraphML для подальшого використання під час верифікації згенерованого моделлю контенту.

3.2 Процес генерації навчального контенту

Процес автоматизованої генерації навчального контенту у розробленій системі реалізовано як послідовність взаємопов’язаних етапів, що забезпечують отримання структурованих та семантично узгоджених тестових завдань. Основною метою цього процесу є створення множини питань типу multiple choice, які відповідають визначеній темі, а також забезпечують коректність і релевантність навчальних матеріалів. Нижче наведено детальний опис роботи кожного етапу. Цей крок виступає вхідною точкою всієї системи та визначає межі предметної області, характер знань, що необхідні для генерації та подальші критерії релевантності.

3.2.1 Пошук релевантних матеріалів за допомогою RAG

Після отримання теми система ініціює механізм Retrieval-Augmented Generation (RAG), який виконує добір релевантних фрагментів знань. Даний етап складається з кількох підпроцесів:

– формування векторного представлення запиту – тема користувача перетворюється у вектор (ембединг), що дозволяє вимірювати її семантичну подібність до фрагментів з бази знань;

– пошук у векторній базі – система здійснює пошук у попередньо проіндексованій колекції навчальних матеріалів, які були завантажені користувачем або входять до основної бази знань;

– формування контексту для моделі – найрелевантніші фрагменти агрегуються та готуються як частина контексту для запиту до мовної моделі, що дозволяє зменшити ризик появи галюцинацій та забезпечує опору генерації на фактичні матеріали предметної області.

3.2.2 Формування промпту для мовної моделі

На етапі генерації навчального контенту формування промпту відіграє критичну роль, оскільки саме від якості цього інструктивного блоку залежить коректність, структурованість та семантична обґрунтованість результатів, які повертає мовна модель. У розробленій системі промпт має чітко визначену архітектуру й поєднує статичну інструктивну частину з динамічними компонентами, що включають тему, задану користувачем, та релевантний контекст, отриманий за допомогою механізмів RAG. Така структура забезпечує контрольованість генерації та узгодженість з фактичними даними предметної області. структуру промпту, реалізованого у системі представлено у додатку А.

Конструювання промпту здійснювалося відповідно до принципів сучасного промпт-інжинірингу, що дає змогу забезпечити детермінованість структури відповіді, мінімізувати появу помилок формату та знизити ймовірність генерування некоректних або вигаданих тверджень. Одним із ключових підходів стало застосування role prompting, у межах якого модель заздалегідь позиціонується як спеціалізована система генерації навчальних тестів. Такий метод сприяє стабілізації поведінки моделі та покращує точність виконання інструкцій. Додатково реалізовано жорстке обмеження контекстом (context restriction), у якому чітко вказано, що модель повинна використовувати виключно надану інформацію. Це є необхідним для

запобігання галюцинаціям і забезпечення відповідності результатів даним із семантичного графа знань та RAG-компонента.

Значну увагу приділено формалізації вихідних даних. У промпті застосовано прийом explicit output formatting, що передбачає визначення строгої JSON-структури разом із шаблоном для формування кожного елемента масиву. Для підсилення надійності додано інструкції-заборони, які запобігають включенню коментарів, пояснень або довільного тексту, а також повторне наголошення на вимозі повернення лише JSON-масиву. Така інструкційна ієрархізація (instruction hierarchy) забезпечує узгодженість між усіма вимогами та зменшує ризик їх порушення. Крім того, продемонстрований у промпті приклад JSON-структури відіграє роль «мікро-прикладу» у межах методики few-shot guided formatting, дозволяючи моделі більш стабільно дотримуватись необхідного формату навіть без надання повномасштабних демонстрацій.

Загалом використаний набір технік промпт-інжинірингу формує цілісну, високодетерміновану інструкцію, яка забезпечує надійну, несуперечливу та формально коректну генерацію навчальних тестових завдань, придатних до подальшої автоматизованої семантичної верифікації.

3.3 Верифікація достовірності згенерованого контенту та виправлення помилок

Після завершення процесу генерації тестових завдань та отримання вихідних даних у форматі JSON виконується етап їхньої семантичної верифікації на основі графа знань предметної області. Використання попередньо побудованого графа у форматі GraphML забезпечує можливість формального порівняння тверджень мовної моделі зі структурованою системою доменної інформації, дозволяючи виявляти як фактичні помилки, так і логічні суперечності.

Граф імпортується та нормалізується за допомогою бібліотеки networkx. На етапі завантаження виконується стандартизація лексичних форм концептів і створення допоміжних індексів для пришвидшення семантичного пошуку (рисунок 3.10).

```python
def load_knowledge_graph(path: str) -> nx.Graph:
    graph = nx.read_graphml(path)
    for node in graph.nodes:
        norm = graph.nodes[node].get("label", node).strip().lower()
        graph.nodes[node]["norm_label"] = norm
    return graph
```

Рисунок 3.10 – Функція завантаження графу знань (виконано самостійно)

Верифікація тверджень здійснюється шляхом порівняння концептів, згаданих у згенерованих питаннях і відповідях, з існуючими у графі знаннями (рисунок 3.11). На цьому етапі відбувається пошук релевантних вузлів, визначення характеру зв'язків між ними та перевірка відповідності тверджень структурі графа. Якщо концепти не пов'язано або вони суперечать визначеним у графі семантичним відношенням, твердження маркується як помилкове.

```python
def validate_fact(graph: nx.Graph, concept: str, related: str) -> bool:
    c = concept.lower().strip()
    r = related.lower().strip()
    nodes_c = [n for n, d in graph.nodes(data=True) if d["norm_label"] == c]
    nodes_r = [n for n, d in graph.nodes(data=True) if d["norm_label"] == r]
    if not nodes_c or not nodes_r:
        return False
    for nc in nodes_c:
        for nr in nodes_r:
            if graph.has_edge(nc, nr):
                return True
    return False
```

Рисунок 3.11 – Функція перевірки правильності твердження (виконано самостійно)

У разі виявлення помилки система застосовує багаторівневий підхід до корекції, який включає:

а) автоматичні семантичні корекції;

1) якщо correct_answer не верифікований (status != VERIFIED), але інший варіант відповіді в масиві має статус VERIFIED, автоматично змінити correct_answer на ту, що верифікована, записати provenance і confidence;

2) якщо декілька варіантів отримали статус VERIFIED, відправити питання на повторну генерацію;

б) перегенерація через модель;

1) скласти локалізований повторний промпт із вимогою до регенерації тестового завдання, включити у prompt;

– виявлені суперечності;

– релевантні вузли графа як доказ;

– контекст у вигляді результату пошуку локалізованої інформації за допомогою раніше розробленої RAG-моделі;

– вимогу згенерувати альтернативне питання на задану тематику;

2) викликати LLM зі складеним промптом;

3) повторно провести процес верифікації згенерованого контенту.

Якщо правильна відповідь (correct_answer) у тестовому завданні не підтверджується графом знань, але серед інших варіантів відповідей є хоча б один, що має статус VERIFIED, система автоматично замінює правильну відповідь на верифіковану. Такий підхід дозволяє усунути значну частину типових помилок мовних моделей, пов'язаних з помилковим вибором «правильної» опції при загалом коректному наборі варіантів. Механізм реалізовано через пошук верифікованих відповідей та відповідне оновлення структури завдання (рисунок 3.12).

```python
def auto_correct_answer(question):
    verified = [opt for opt in question["options"] if opt["status"] == "VERIFIED"]
    if len(verified) == 1:
        question["correct_answer"] = verified[0]["text"]
        question["provenance"] = "auto_graph_fix"
        question["confidence"] = 0.95
    elif len(verified) > 1:
        question["status"] = "AMBIGUOUS"
    return question
```

Рисунок 3.12 – Функція виправлення правильної відповіді (виконано самостійно)

У випадках, коли питання містить суперечності або жоден варіант не підтверджений графом знань, використовується локалізований re-prompt, що направляє модель на генерацію альтернативного питання (рисунок 3.13).

```python
def build_reprompt(topic, contradictions, evidence_nodes, rag_context):
    return f"""
The previously generated multiple-choice question contains semantic inconsistencies:
{contradictions}

The following knowledge graph nodes provide relevant evidence:
{evidence_nodes}

Here is relevant context on the topic '{topic}':
{rag_context}

Generate an alternative multiple-choice question on the topic '{topic}'.
The question must include:
- Four answer options, with one correct answer,
- Strict adherence to the facts provided in the evidence nodes and the context,
- No invented or unsupported information.

Return the result strictly in JSON format with the fields:
"question", "options", "correct_answer".
"""
```

Рисунок 3.13 – Функція побудови повторного промпту (виконано самостійно)

Модель повертає скориговане питання, яке повторно проходить верифікацію.

У підсумку, завдяки поєднанню графа знань та механізму ре-промптингу, система здатна:

– автоматично позначати та виправляти помилки в тестових завданнях;

– маркувати питання як VERIFIED, AMBIGUOUS, SUSPECT або CONFLICT залежно від рівня відповідності графу;

– повторно викликати мовну модель для некоректних або суперечливих питань;

– передавати на перевірку людини лише ті завдання, які не підлягають однозначному автоматичному виправленню.

Такий підхід дозволяє формувати високоякісний навчальний контент із високою семантичною узгодженістю та мінімізацією логічних та фактологічних помилок.

3.4 Користувацький інтерфейс

Користувацький інтерфейс розробленої системи представлений у вигляді веб-додатку з двома основними панелями, що забезпечують роботу з навчальними матеріалами та генерацію тестових завдань. Інтерфейс побудований з акцентом на простоту та зручність користування, що дозволяє швидко виконувати основні дії без додаткового навчання. Дизайн панелей виконаний у сучасному стилі з чіткою візуальною структурою, що допомагає користувачу легко орієнтуватися у функціоналі та послідовності дій.

Для коректної інтеграції додатку з розробленою системою реалізований API на Python (рисунок 3.14). Була використана бібліотека FastAPI, що забезпечує високопродуктивну обробку HTTP-запитів, зручну маршрутизацію та автоматичну генерацію документації OpenAPI. Завдяки асинхронній обробці запитів, API дозволяє ефективно взаємодіяти з компонентами системи, включаючи завантаження файлів, актуалізацію

графа знань та генерацію тестових завдань, забезпечуючи при цьому швидку реакцію сервера та масштабованість при зростанні навантаження.

Кожен ендпоінт реалізовано з перевіркою вхідних даних і обробкою можливих помилок, що підвищує надійність і стабільність роботи системи. Також API інтегровано з логуванням операцій і механізмами обробки статусів виконання, що дозволяє відслідковувати стан процесів і своєчасно інформувати користувача про результати його дій.

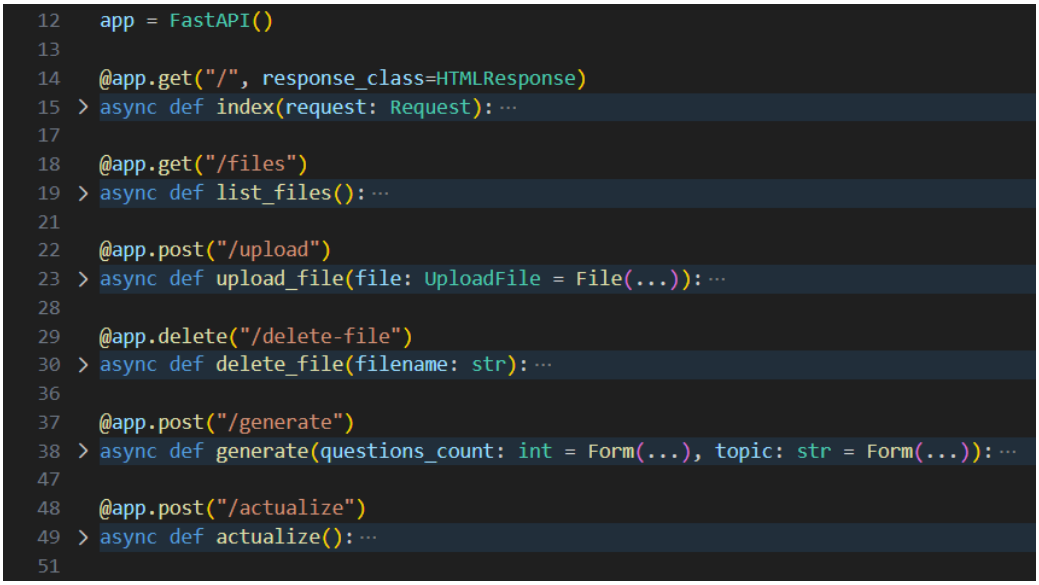

Рисунок 3.14 – Сигнатура ендпоінтів додатку (виконано самостійно)

Інтерфейс додатку реалізовано на HTML із використанням бібліотеки TailwindCSS для швидкого створення адаптивного і сучасного дизайну.

Перша панель призначена для роботи з вихідними даними користувача та інтеграції їх у граф знань. Вона містить наступні елементи:

– список завантажених файлів, що відображає всі документи, завантажені користувачем. Для кожного файлу передбачена кнопка видалення;

– форма завантаження нового файлу після вибору файлу користувач натискає кнопку «Upload File»;

– кнопка «Actualize Knowledge Base», що запускає процедуру обробки та інтеграції завантажених файлів у граф знань. Після завершення процесу користувачу виводиться повідомлення про статус актуалізації.

Нижче на рисунку 3.15 представлений зовнішній вигляд панелі для роботи із завантаженням знань до системи:

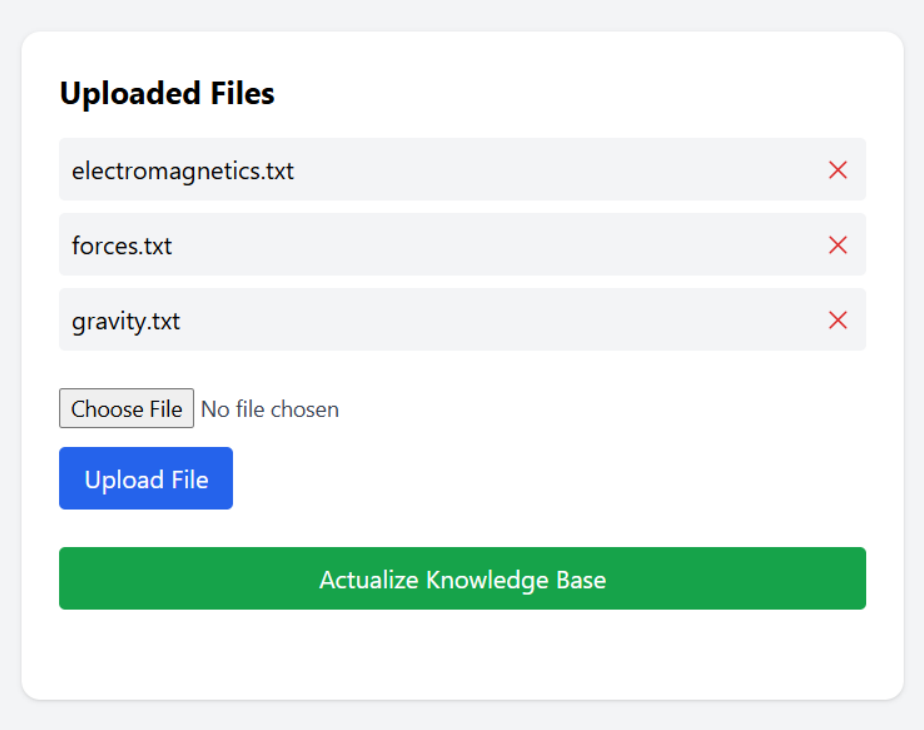

Рисунок 3.15 – Панель завантаження даних до системи (виконано самостійно)

Ендпоінти, що використовуються панеллю:

– /files (GET) – отримання списку завантажених файлів для відображення у списку;

– /upload (POST) – завантаження нового файлу на сервер;

– /delete-file (DELETE) – видалення конкретного файлу за назвою;

– /actualize (POST) – ініціалізація обробки файлів та актуалізації графа знань.

Ця панель дозволяє користувачу гнучко управляти матеріалами та забезпечуючи підготовку коректних даних для наступного етапу генерації тестів.

Друга панель відповідає за створення навчальних завдань на основі заданої тематики та кількості питань. Панель містить форму введення параметрів генерації:

– поле «Questions Count» – кількість запитуваних тестових питань;

– поле «Topic» – тематична область, на яку повинні бути згенеровані питання;

– кнопка «Generate» – запуск процесу генерації, перевірки правильності отриманного матеріалу та виправлення помилок.

Нижче на рисунку 3.16 представлений зовнішній вигляд панелі для генерації тестових завдань.

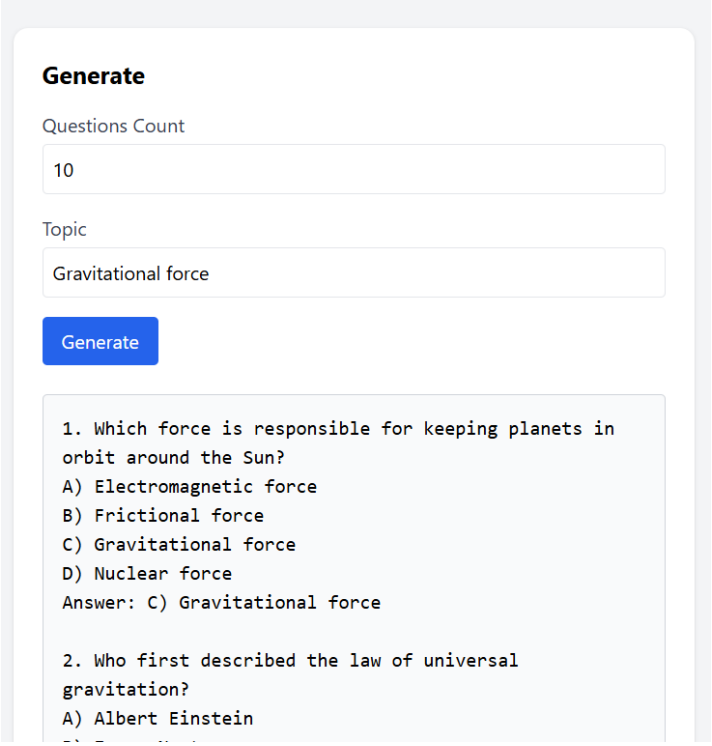

Рисунок 3.16 – Панель генерації тестів (виконано самостійно)

Результат генерації відображається у полі виводу із підтримкою форматування JSON, що дозволяє користувачу переглянути згенеровані питання та варіанти відповідей у структурованому вигляді.

Інтерфейс, завдяки поділу на дві панелі, забезпечує чітку логіку роботи (рисунок 3.17): спочатку підготовка та актуалізація даних, потім генерація перевіреного навчального контенту.

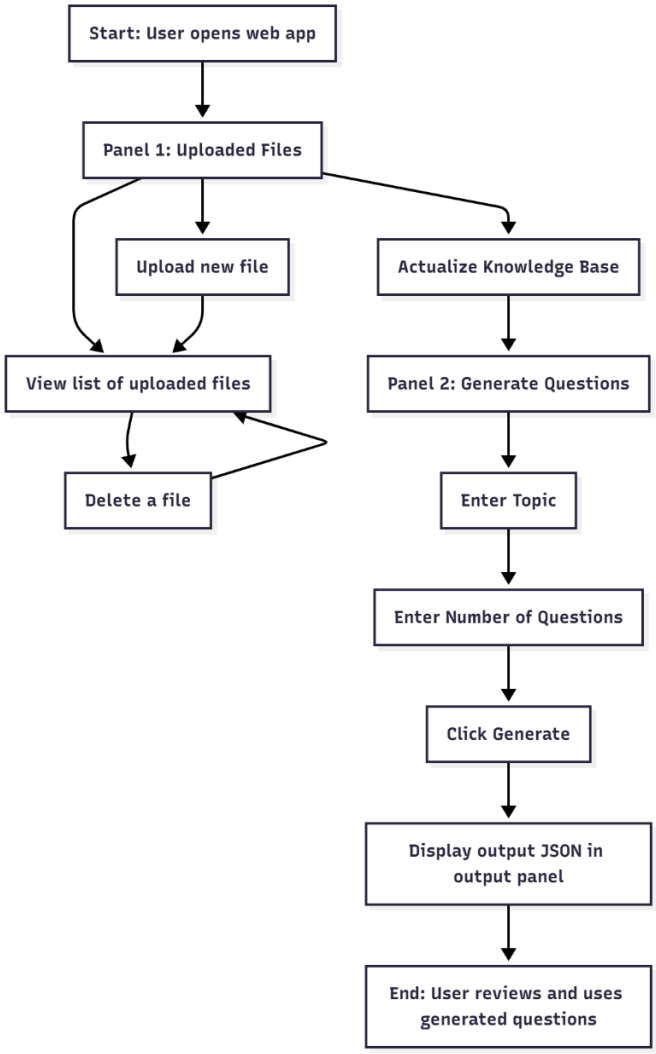

Рисунок 3.17 – User flow додатку (виконано самостійно)

Використання сучасного фреймворку TailwindCSS забезпечує зручність та адаптивність інтерфейсу, а інтеграція з серверними

ендпоінтами гарантує безперервний та контрольований потік даних від завантаження файлів до отримання коректних тестових завдань.

3.5 Результати тестування розробленої системи

Для оцінки ефективності розробленої системи було проведено серію експериментів, спрямованих на вимірювання коректності та якості згенерованих тестових завдань. Основним критерієм оцінки виступала семантична відповідність питань та варіантів відповідей фактам предметної області, представленим у графі знань. Кожне питання перевірялося на наявність логічних суперечностей, правильність правильного варіанту відповіді та відповідність базовим фактам.

Для кількісної оцінки коректності використовувалася метрика Accuracy, яка визначалась як відношення кількості питань із статусом VERIFIED до загальної кількості згенерованих питань:

$$Accuracy = \frac{N_{VERIFIED}}{N_{total}} \times 100\% \quad (3.3)$$

де $N_{VERIFIED}$ – кількість правильно згенерованих питань, підтверджених експертом (людиною);

N_{total} – загальна кількість питань у вибірці.

Для кращої візуалізації зміни Accuracy в залежності від використаних методів покращення якості генерації, були розроблені та проведені наступні сценарії тестування:

а) гола модель (LLM без контексту) – на цьому етапі модель генерувала тестові питання лише на основі внутрішніх знань LLM без використання зовнішнього контексту, результати показали високий рівень помилок та логічних суперечностей через феномен «галюцинацій» і обмежену точність внутрішніх знань моделі;

б) модель + контекст із RAG – додавання релевантного контексту через RAG дозволило суттєво підвищити точність відповідей, модель отримувала додаткову інформацію з документів користувача, що зменшувало кількість фактологічних помилок та логічних суперечностей. Спостерігалося відчутне зростання метрики Accuracy порівняно з «голою» моделлю;

в) модель + контекст із RAG + механізм корекції за допомогою графа знань – на фінальному етапі до системи додано розроблений алгоритм семантичної верифікації та автоматичної корекції тестів із використанням графа знань:

– автоматичну зміну correct_answer, якщо серед варіантів є VERIFIED;

– маркування питань як AMBIGUOUS при наявності декількох верифікованих варіантів;

– повторну генерацію питань через LLM із локалізованим re-prompt, що враховує суперечності, вузли графа та RAG-контекст.

У результаті проведеного тестування було встановлено, що впровадження автоматичної корекції та семантичної верифікації дозволяє покращити коректність згенерованих тестових завдань (таблиця 3.1). Модель було проінструктовано згенерувати по 100 питань у 3 різних предметних областях. Проте, хоча спостерігається помітне, хоч і незначне, підвищення метрики Accuracy порівняно з використанням моделі з контекстом RAG, результати свідчать про наявність потенціалу для подальшого вдосконалення системи.

Подальші дослідження можуть бути спрямовані на розширення графа знань, удосконалення локалізованих промптів та впровадження додаткових механізмів оцінки якості питань, що дозволить досягти стабільно високої точності та узгодженості згенерованого навчального контенту.

Таблиця 3.1 – Результати тестування

Сценарій	Gravitational force (%)	Electromagnetics (%)	Arithmetics (%)	Основні спостереження
Гола модель	75	77	42	Висока частка фактологічних помилок та логічних суперечностей, плутанина термінів та прикладів
Модель + RAG	82	83	39	Використання RAG дозволяє зменшити кількість помилок та уточнити факти, покращується узгодженість (окрім аріфметичних завдань)
Модель + RAG + граф знань	86	87	47	Автоматична корекція і семантична верифікація незначно але помітно покращують результати

Крім того, в розроблену систему можна впровадити методи автоматичної самоаналізу та рейтингової оцінки якості питань, засновані на метриках схожості з фактичними даними графа знань і статистичних показниках моделей. Такі підходи дозволяють не лише коригувати окремі питання, а й формувати адаптивну стратегію генерації навчальних матеріалів, що враховує історичні результати верифікації та рівень довіри до різних вузлів графа.

ВИСНОВКИ

У результаті проведеної роботи було розроблено комплексну систему для генерації навчального контенту з автоматичною семантичною верифікацією та корекцією тестових завдань на основі графа знань. Основною метою дослідження було підвищення достовірності та коректності матеріалів, що генеруються великими мовними моделями (LLM), шляхом інтеграції механізмів перевірки знань і виправлення потенційних помилок. Розроблена система поєднує підхід Retrieval-Augmented Generation (RAG) для надання релевантного контексту мовній моделі та алгоритми семантичної верифікації на основі побудованого графа знань. Такий підхід дозволяє не лише виявляти суперечливі твердження у сгенерованому контенті, а й автоматично коригувати їх із високим ступенем достовірності.

У процесі дослідження було реалізовано алгоритм генерації тестових завдань у форматі multiple-choice, який включає кілька етапів: пошук релевантної інформації через RAG, формування промпту для LLM з вимогою повернення JSON-запису питань із варіантами відповідей та правильною відповіддю, а також подальшу семантичну верифікацію результатів. Алгоритм корекції помилок передбачає автоматичне виправлення некоректних відповідей на основі перевірених вузлів графа знань, маркування суперечливих результатів як AMBIGUOUS та локалізоване повторне формулювання запитів (re-prompting) для моделі. Реалізація цього підходу показала, що навіть невеликі корекції за допомогою графа знань дозволяють суттєво зменшити кількість помилкових тверджень у тестах і підвищити загальну достовірність навчального контенту.

Тестування системи проводилося у кілька етапів із поступовим нарощуванням рівня складності та додаткових механізмів контролю: використання «голої» LLM без контексту, LLM з контекстом RAG та LLM

з контекстом RAG і семантичною корекцією графа знань. Результати показали помітне, хоча і не драматичне, підвищення точності генерації тестових завдань. Найбільш ефективним виявився сценарій із повною інтеграцією RAG і графа знань: у цьому випадку виявлялися та автоматично виправлялися більшість некоректних тверджень, що зменшувало ймовірність логічних суперечностей і феномену «галюцинацій» моделі. Аналіз результатів для різних предметних областей – Gravitational force, Electromagnetics та Arithmetics – показав, що запропонований підхід є універсальним та стабільно підвищує достовірність контенту незалежно від тематики.

Розроблений користувацький інтерфейс веб-додатку забезпечує інтуїтивно зрозумілий флоу користувача, розділяючи роботу з навчальними матеріалами та генерацію тестових завдань на окремі панелі. Перша панель дозволяє завантажувати документи, переглядати список файлів та актуалізувати базу знань, друга – задавати тему, вказувати кількість питань і отримувати структурований вихідний результат у JSON-форматі. Така організація інтерфейсу сприяє швидкій інтеграції системи у навчальний процес та зручності використання без спеціальної технічної підготовки.

Системна архітектура та використання API на основі FastAPI забезпечили гнучкість, масштабованість і можливість подальшого розширення функціоналу. Впровадження RAG як механізму надання релевантного контексту для LLM дозволяє моделі враховувати специфіку предметної області і суттєво зменшувати частоту помилкових тверджень у створюваних завданнях. Водночас інтеграція графа знань відкриває перспективи для подальшого розвитку системи: можна автоматично виявляти суперечності у власних даних графа, контролювати якість бази знань та впроваджувати механізми навчання на помилках, що виявляються під час генерації тестів.

Перспективи подальшого розвитку системи включають кілька напрямів. По-перше, можливе покращення механізму семантичної верифікації через розширення графа знань і інтеграцію багаторівневих джерел інформації. Це дозволить підвищити точність перевірки більш складних та міждисциплінарних завдань. По-друге, впровадження адаптивного re-prompting із урахуванням контексту попередніх некоректних відповідей та статистики успішності може ще більше підвищити якість генерованих тестів. По-третє, інтеграція системи з навчальними платформами та використання аналітики результатів студентів дозволить не лише генерувати якісний контент, а й формувати персоналізовані навчальні траєкторії.

Таким чином, проведена робота підтвердила ефективність поєднання LLM, RAG та графа знань для генерації достовірного навчального контенту. Запропонований підхід показав значне підвищення точності тестових завдань порівняно з використанням лише моделі, а також забезпечує можливості для подальшого вдосконалення системи. Отримані результати створюють основу для розвитку інтерактивних освітніх платформ з високим рівнем автоматизації та контролю якості навчального матеріалу, що відповідає сучасним вимогам цифрової освіти.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Attention-Aware Long-Context Model. URL: https://arxiv.org/pdf/2402.01580 (дата звернення: 20.11.2025).

2. Frontier Language Models: Scaling Behavior and Capabilities. URL: https://arxiv.org/pdf/2507.02180 (дата звернення: 20.11.2025).

3. Large Language Models and Multi-Agent Reasoning. URL: https://arxiv.org/pdf/2402.14601 (дата звернення: 20.11.2025).

4. AI Research Brief 1. College Board. URL: https://research.collegeboard.org/media/pdf/AI%20Research%20Brief%201_vf_0.pdf (дата звернення: 20.11.2025).

5. Khanmigo: AI-powered tutoring platform. URL: https://www.khanmigo.ai (дата звернення: 20.11.2025).

6. Dutta A. Taxonomy of Artificial Intelligence. 2025. URL: https://www.researchgate.net/publication/395299914_Taxonomy_of_Artificial_Intelligence (дата звернення: 20.11.2025).

7. Vaswani A., Shazeer N., Parmar N. Attention Is All You Need. 2017. URL: https://arxiv.org/abs/2005.1140 (дата звернення: 20.11.2025).

8. Mixture-of-Experts (MoE). URL: https://arxiv.org/abs/1701.06538 (дата звернення: 20.11.2025).

9. Minaee S., Mikolov T., Nikzad N. Large Language Models: A Survey. 2025. URL: https://arxiv.org/abs/2402.06196 (дата звернення: 20.11.2025).

10. Lewis P., Perez E., Piktus A. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. 2021. URL: https://arxiv.org/abs/2005.11401 (дата звернення: 20.11.2025).

11. Long-form Reasoning with LLMs. URL: https://arxiv.org/abs/2306.11489 (дата звернення: 20.11.2025).

12. Findings of EMNLP 2023. URL: https://aclanthology.org/2023.findings-emnlp.631 (дата звернення: 20.11.2025).

13. Schick T., Dwivedi-Yu J., Dessì R. Toolformer: Language Models Can Teach Themselves to Use Tools. 2023. URL: https://arxiv.org/abs/2302.04761 (дата звернення: 20.11.2025).

14. ArXiv:2510.01253v1 (HTML version). URL: https://arxiv.org/html/2510.01253v1 (дата звернення: 20.11.2025).

15. Asai A., Gardner M., Hajishirzi H. Evidentiality-guided Generation for Knowledge-Intensive NLP Tasks. 2021. URL: https://arxiv.org/abs/2112.08688 (дата звернення: 20.11.2025).

16. RDF Primer. W3C Recommendation. URL: https://www.w3.org/2001/09/rdfprimer/rdf-primer-20021108.html (дата звернення: 20.11.2025).

17. GenAIK 2025 Proceedings. URL: https://aclanthology.org/2025.genaik-1.10.pdf (дата звернення: 20.11.2025).

18. Knowledge Graph Embeddings Survey. URL: https://arxiv.org/abs/2003.02320 (дата звернення: 20.11.2025).

19. A Fast Learning Algorithm for Deep Belief Networks. NeurIPS 2013. URL: https://proceedings.neurips.cc/paper_files/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf (дата звернення: 20.11.2025).

20. Ramamonjison R. та ін. Proceedings of Machine Learning Research. Vol. 220. URL: https://proceedings.mlr.press/v220/ramamonjison23a.html (дата звернення: 20.11.2025).

21. Ramamonjison R. та ін. Proceedings of Machine Learning Research. Vol. 221. URL: https://proceedings.mlr.press/v220/ramamonjison23a.html (дата звернення: 20.11.2025).

22. ArXiv:2405.13144v2. URL: https://arxiv.org/pdf/2405.13144v2 (дата звернення: 20.11.2025).

23. ArXiv:2405.17743. URL: https://arxiv.org/abs/2405.17743 (дата звернення: 20.11.2025).

24. IJCAI 2025 Proceedings. URL: https://www.ijcai.org/proceedings/2025/1192.pdf (дата звернення: 20.11.2025).