Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів ін'єкції знань
для великих мовних моделей
(тема)

Виконав:
здобувач 2 року навчання
групи ІПЗм-23-4

Владислав ФЛЯГІН
(Власне ім'я, ПРІЗВИЩЕ)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-наукова

Керівник доц. Олексій ТУРУТА
(посада, Власне ім'я, ПРІЗВИЩЕ)

Допускається до захисту
Зав. кафедри _____ Кирило СМЕЛЯКОВ
(підпис) (Власне ім'я, ПРІЗВИЩЕ)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти другий (магістерський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми освітньо-наукова програма
Освітня програма Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Флягіну Владиславу Костянтиновичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Дослідження методів ін'єкції знань для великих мовних моделей»
Затверджена наказом по університету від 15.04. 2025р. № 290 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 16.06.2025
3. Вихідні дані до роботи набір текстів для створення бази знань, згенерований граф знань у Neo4j, великі мовні моделі (GPT, Llama, Gemini) для тестування, фреймворк DeepEval, а також обчислювальні ресурси (GPU або хмарна інфраструктура).
4. Перелік питань, що потрібно опрацювати в роботі
аналіз та порівняння існуючих великих мовних моделей та методів ін'єкції знань, вибір підходящих великих мовних моделей та методів ін'єкції для дослідження, проектування даних для проведення експериментальних досліджень, написання програмних рішень, проведення експериментів та аналіз отриманих результатів

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	16.04	*виконано*
2	Аналіз предметної галузі і постановка задачі	17.04 – 30.04	*виконано*
3	Теоретичне дослідження	01.05-09.05	*виконано*
4	Практичне дослідження	10.05-20.05	*виконано*
5	Підготовка до апробації результатів дослідження. Публікація матеріалів	18.04-20.05	*виконано*
6	Підготовка пояснювальної записки	21.05-29.05	*виконано*
7	Підготовка презентації та доповіді	27.05-02.06	*виконано*
8	Перевірка на плагіат	03.06	*виконано*
9	Нормоконтроль	03.06-04.06	*виконано*
10	Рецензування	05.06-10.06	*виконано*
11	Попередній захист	06.06	*виконано*
12	Занесення диплома в електронний архів	10.06	*виконано*
13	Допуск до захисту у зав. кафедри	11.06	*виконано*

Дата видачі завдання 16 квітня 2025р.

Студент (ка) _____ Владислав ФЛЯГІН
(підпис)

Керівник роботи _____ доц. Олексій ТУРУТА
(підпис) (посада, Власне ім’я, ПРІЗВИЩЕ)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка містить: 98 с., 62 рис., 2 табл., 40 джерел.

ВЕЛИКІ МОВНІ МОДЕЛІ, ДОНАВЧАННЯ, МЕТОДИ ІН'ЄКЦІЇ ЗНАНЬ, DEEPEVAL, FINE-TUNING, NEO4J, RAG.

Об'єктом дослідження є методи ін'єкції знань для великих мовних моделей.

Метою роботи є дослідити вплив різних методів ін'єкції знань на якість великих лінгвістичних моделей у реальних задачах. Також, метою є отримання рекомендацій щодо використання того чи іншого методу ін'єкції знань в залежності від типу задачі та наявних даних.

Методами розробки та проектування є аналіз проблемної області дослідження та вибір великих лінгвістичних моделей для майбутнього проведення дослідження шляхом вирішення багатокритеріальної задачі прийняття рішень. Також у якості методів розробки та проектування було обрано мову програмування Python, фреймворк DeepEval та графову базу даних Neo4j.

У результаті кваліфікаційної роботи було досліджено сучасний стан галузі генеративного штучного інтелекту, його проблеми та тенденції. Також, детально було розглянуто варіації методів ін'єкції знань, їх переваги та недоліки у порівнянні один з одним. Наостанок, було виконано 32 експерименти з різними конфігураціями RAG системи, по результатам яких було зроблено висновки щодо доцільності використання тої чи іншої модифікації.

Як висновок з експериментів, найбільш ефективними модифікаціями є ті, які використовують гібридний пошук. Щодо таких гіперпараметрів як кількість отримуваних чанків, так само як і розмір чанку, то вони є індивідуальними в залежності від ваших даних. Отримані конфігурації можуть бути адаптовані для використання в будь-якій сфері.

Для більш повної картини можна продовжити дослідження з використанням претреновування у якості альтернативного методу ін'єкції знань.

LARGE LANGUAGE MODELS, MODEL-TUNING, KNOWLEDGE INJECTION METHODS, DEEPEVAL, FINE-TUNING, NEO4J, RAG.

The object of research is methods of knowledge injection in large language models.

The aim of the study is to investigate the impact of different knowledge injection methods on the quality of large language models in real-world tasks. Also, the aim is to obtain recommendations on the use of a particular knowledge injection method depending on the type of task and available data.

The methods of development and design are the analysis of the problem area of research and the selection of large linguistic models for future research by solving a multi-criteria decision-making problem. The Python programming language, the DeepEval framework, and the Neo4j graph database were chosen as development methods.

As a result of the qualification work, the current state of the field of generative artificial intelligence, its problems and trends were investigated. Also, the variations of knowledge injection methods, their advantages and disadvantages in comparison with each other were considered in detail. Finally, 32 experiments were performed with different configurations of the RAG system, based on the results of which conclusions were drawn about the feasibility of using a particular modification.

According to the experiments, the most effective modifications are those that use hybrid search. As for such hyperparameters as the number of chunks obtained and the size of the chunk, they are individual depending on your data. The resulting configurations can be adapted for use in any field.

For a more complete picture, you can continue this research using pre-training as an alternative method of knowledge injection.

Завідувачу кафедри
ПІ_____
(скорочена назва кафедри)
проф. Кирилу СМЕЛЯКОВУ
(вчене звання, сласне ім'я, прізвище)

ЗАЯВА
щодо самостійності виконання кваліфікаційної роботи та можливості її публікації (та/або публікації анотації кваліфікаційної роботи) в електронному архіві відкритого доступу ElAr KhNURE

Я, Флягін Владислав Костянтинович
(прізвище, ім'я, по батькові)

здобувач вищої освіти на другому (магістерському) рівні вищої освіти академічної групи ІПЗм-23-4

кафедра програмної інженерії
(повна назва кафедри)

заявляю: моя кваліфікаційна робота на тему
Дослідження методів ін'єкції знань для великих мовних моделей
(назва роботи)

що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в репозиторії "ElArKhNURE". Погоджуюся з авторським договором, відповідно до Положення про репозиторій ХНУРЕ "ElArKhNURE". Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з вимогами академічної доброчесності, згідно з якими виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

Дата 03.06.2025 Підпис

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних

API – application programming interface

CS – Chunk Size

GPU – Graphics Processing Unit

HS – Hybrid Search

KI – Knowledge Injection

LLM – Large Language Model

LoRA – Low-Rank Adaptation

NLP – Natural Language Processing

QLoRA – Quantized Low-Rank Adaptation

RAG – Retrieval-Augmented Generation

SS – Sentence Splitter

TF-IDF – TermFrequency-InverseDocumentFrequency

TTS – TokenTextSplitter

ВСТУП

За останні кілька років наш світ колосально змінився. З появою штучного інтелекту стало зрозуміло що багато речей можна дізнатись значно швидше та написати значно швидше. Дуже важливим аспектом штучного інтелекту є те, що ми не просто читаємо статтю або підручник на конкретну тему, а ми можемо обговорювати нову інформацію. Це можна порівняти як самостійне читання підручника, дослідження та розуміння чому саме така формула наведена, чи немає десь помилки і яка логіка за цим стоїть, та випадком коли вам хтось пояснює цей матеріал. Будь-то ваш друг, колега, або викладач в університеті. Найважливіша різниця в тому, що ви можете задавати зустрічні питання та отримувати на них відповіді. Це значним чином прискорює процес навчання. При правильному використанні, та без логічних помилок, штучний інтелект у сучасному вигляді є корисним для людства.

Щодо логічних помилок, це стається в більшій мірі через те, що модель знає лише про те, що бачила під час тренування. Там може не бути деталей щодо певних доменів, про які ви хочете спитати. Це означає що в модель потрібно зробити ін’єкцію знань конкретного домену, щоб вона краще могла відповідати на поставлені запитання та бути більш корисною. Але у наш час існує багато методів ін’єкції знань у великі лінгвістичні моделі.

Ця кваліфікаційна робота як раз націлена на дослідження методів ін’єкції знань для великих мовних моделей.

Тема дослідження тісно пов’язана з науковим напрямком кафедри ПІ, яка досліджує інноваційні технології та методи їх використання.

Метою роботи є дослідити вплив різних методів ін’єкції знань на якість великих лінгвістичних моделей у реальних задачах. Також, метою є отримання рекомендацій щодо використання того чи іншого методу ін’єкції знань в залежності від типу задачі та наявних даних.

Методами розробки та проектування є аналіз проблемної області дослідження та вибір великих лінгвістичних моделей для майбутнього проведення дослідження шляхом вирішення багатокритеріальної задачі прийняття

рішень. Також у якості методів розробки та проектування було обрано мову програмування Python, фреймворк DeepEval та графову базу даних Neo4j.

У результаті кваліфікаційної роботи було досліджено сучасний стан галузі генеративного штучного інтелекту, його проблеми та тенденції. Також, детально було розглянуто варіації методів ін’єкції знань, їх переваги та недоліки у порівнянні один з одним. Наостанок, було виконано 32 експерименти з різними конфігураціями RAG системи, по результатам яких було зроблено висновки щодо доцільності використання тої чи іншої модифікації, що є певним кроком вперед для цієї ніші.

Як висновок з експериментів, найбільш ефективними модифікаціями є ті, які використовують гібридний пошук. Щодо таких гіперпараметрів як кількість отримуваних чанків, так само як і розмір чанку, то вони є індивідуальними в залежності від ваших даних. Отримані конфігурації можуть бути адаптовані для використання в будь-якій сфері.

Також, додаткове дослідження, яке стосується підходу до узгодження великих мовних моделей з специфічними базами знань з використанням агентів та RAG, було представлено на воркшопі конференції MIT@AIS-2025. Ця робота удосконалює існуючі методи узгодження лінгвістичних моделей шляхом використанням графу знань як динамічної бази даних обмежень для моделі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ І ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної галузі дослідження

В цьому розділі наведено поверхневий огляд сучасного стану, конкретних методів, проблем та перспектив обраної галузі.

1.1.1 Сучасний стан галузі

Великі лінгвістичні моделі (LLM) стали невід'ємною частиною сучасної обробки природної мови. Моделі, які побудовані на основі архітектури трансформерів (наприклад, GPT-4, Gemini, Llama), здатні ефективно вирішувати широкий спектр завдань, таких як генерація тексту, автоматичний переклад (зараз навіть досліджуються можливості створення більш керованих систем для перекладу [1]), резюмування, класифікація тексту та багато іншого (див. рис. 1.1).

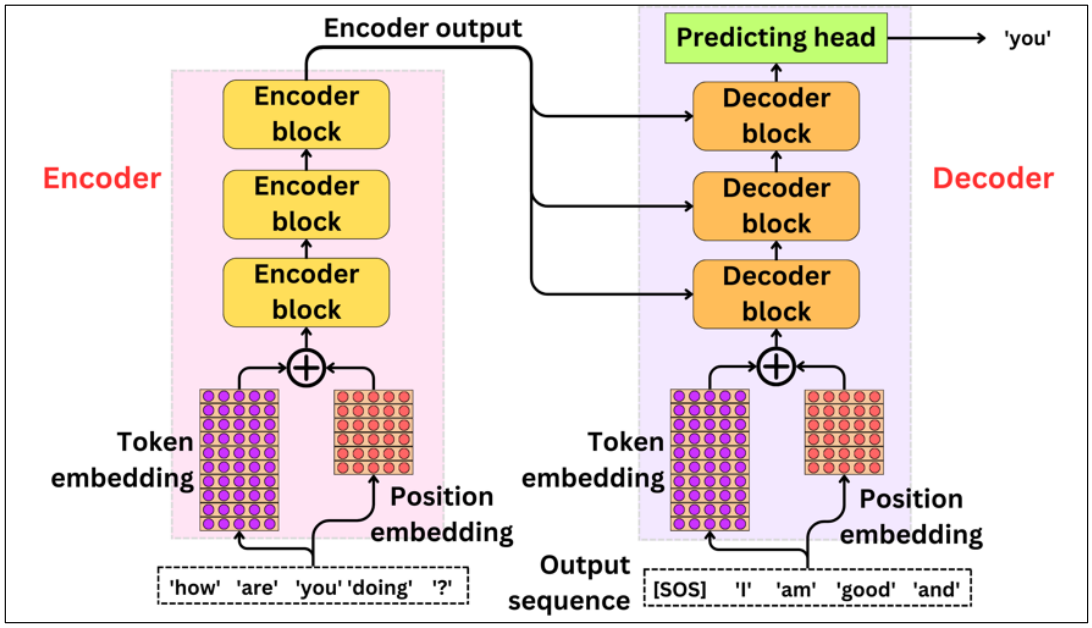


Рисунок 1.1 – Архітектура трансформера (за даними [2])

За архітектурою операційних систем, моделі самі по собі не розуміють символів. Моделі розуміють числову інформацію. Саме тому мовні моделі мають токенізатори. Токенізатор – це інструмент, який розбиває дані на токени (найменші одиниці інформації, що обробляються моделлю). Для тексту,

токенізатор перетворює слова чи їх частини на токени. Для зображень, розділяє їх на маленькі фрагменти, наприклад, 16x16 пікселів. Після попереднього розбиття на токени, кожен токен кодується числом і подається на вхід мережі для обробки. Цей процес є зворотнім, так як модель передбачує на виході теж токени (див. рис. 1.2).

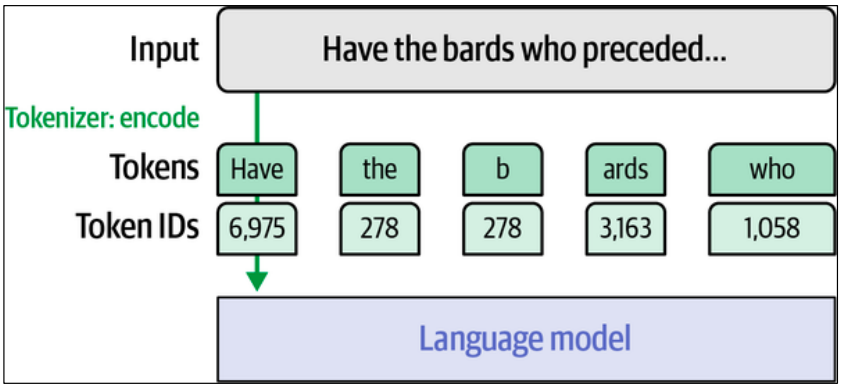


Рисунок 1.2 – Приклад роботи токенізатора (за даними [3])

У 2025 році ми вже маємо великі мультимодальні моделі, які вміють працювати не лише з текстом, а вміють працювати з зображеннями, відео, аудіо та різними форматами файлів. Хоча лише два з половиною роки тому, 30 листопада 2022 року було представлену першу публічно доступну версію ChatGPT, який міг працювати лише з текстом. Такий шалений ріст штучного інтелекту за останні роки суттєво впливає на продукти по всьому світу. Майже у кожному сайті або додатку вже є вбудований чат інтерфейс з ботом, який допоможе відповісти на всі ваші запитання, або хоча б спробує це зробити. У цього є і свої недоліки, основним з яких є те, що бот живе в певного роду бульбашці, і він не знає всього про навколишній світ. Саме через це, наприклад, за 1 хвилину розмови з оператором можна вирішити 99% запитань, хоча за цей самий час з ботом можна вирішити лише 50% запитань, що неодноразово перевірено на власному досвіді. У той же час, використання ботів значно дешевше та простіше для сторони продукту, через що цей підхід і стає всі більш популярним. Але з плином часу настане момент, коли боти зможуть відповідати на значно більший відсоток реальних запитань користувачів та стануть кориснішими для людей.

Основна цінність мовних моделей полягає у здатності аналізувати великі обсяги даних, узагальнювати їх та використовувати за потреби. Однак, попри свої можливості, сучасні LLM мають низку обмежень, що пов’язані з ін’єкцією нових знань. Ін’єкція знань передбачає інтеграцію внутрішньої або зовнішньої інформації, яка дозволяє моделі бути більш точною, інформативною та достовірною. Знання можуть бути представлені у вигляді структурованих (бази знань) чи неструктурованих (набору тексту або зображень) джерел.

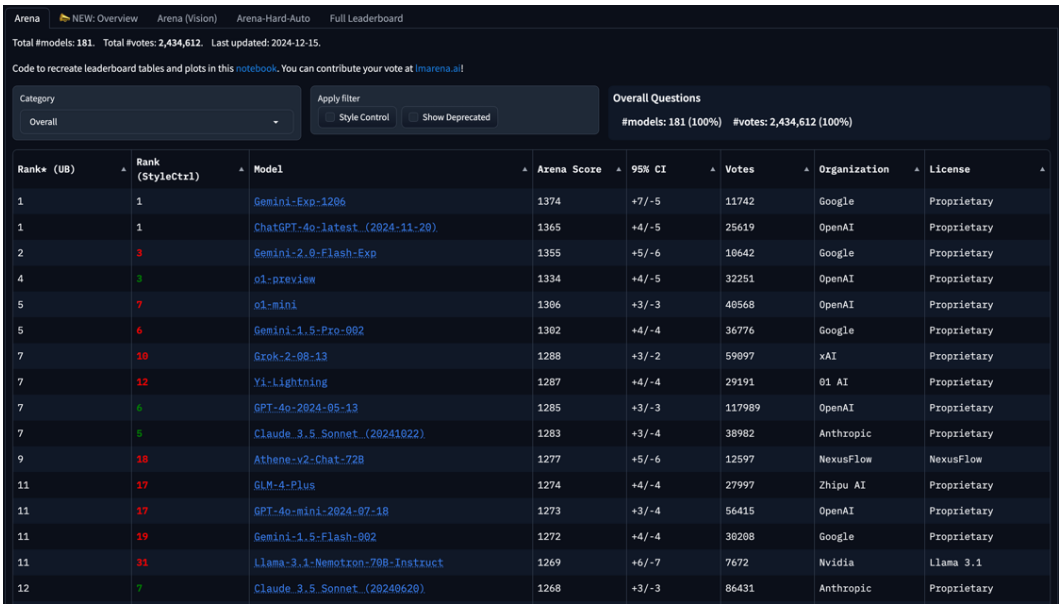
Інтеграція нових знань у лінгвістичні моделі є критично важливим завданням для забезпечення їхньої функціональності в задачах, де потрібна висока точність. Наприклад, у медицині, юриспруденції, бізнес-аналітиці або банкінгу. Під точністю мається на увазі низка аспектів, таких як безпосередня точність відповіді, повнота відповіді та ґрунтовність відповіді на конкретних даних.

Будь-яка модель починається з стадії початкового тренування. Під час подібного тренування, методами оптимізації ми знаходимо оптимальний набір параметрів нейронної мережі для виконання певної задачі (наприклад, передбачення пропущеного слова всередині речення, або передбачення наступного слова в реченні). На стадії попереднього тренування модель вивчає семантику кількох різних мов (у випадку якщо на вхід моделі надаються багатомовні дані). Сучасні моделі, такі як GPT-4o, підтримують більше ніж 50 різних мов [4].

Після цього етапу ми отримуємо мовну модель в чистому вигляді. Цікаве тут те, що насправді ці моделі не використовуються дуже широко. Причина в тому, що базова модель, яка отримана після попереднього навчання не оптимізована для використання у режимі інструкцій, до якого всі вже встигли звикнути. Для цього потрібно донавчити модель на датасеті з інструкцій. І ось вже оптимізовані під інструкції моделі набули великої популярності.

Для спрощення процесу вибору та порівняння моделей було створено кілька арен, або таблиць лідерів, які мають на меті демонстрацію рівня якості моделей на кількох датасетах. Окремо можна виділити LM Arena, де моделям призначається

ELO рейтинг базуючись на фідбеку користувачів стосовно того, яка модель краще відповіла на поставлене запитання (при цьому користувачі бачать лише відповіді моделей, але не бачать де яка саме модель) (див. рис. 1.3).



Rank* (UB)	Rank (StyleCtrl)	Model	Arena Score	95% CI	Votes	Organization	License
1	1	Gemini-Exp-1206	1374	+7/-5	11742	Google	Proprietary
1	1	ChatGPT-4o-latest (2024-11-20)	1365	+4/-5	25619	OpenAI	Proprietary
2	3	Gemini-2.0-Flash-Exp	1355	+5/-6	10642	Google	Proprietary
4	3	o1-preview	1334	+4/-5	32251	OpenAI	Proprietary
5	7	o1-mini	1306	+3/-3	40568	OpenAI	Proprietary
5	6	Gemini-1.5-Pro-002	1302	+4/-4	36776	Google	Proprietary
7	10	Grok-2-08-13	1288	+3/-2	59097	xAI	Proprietary
7	12	Yi-Lightning	1287	+4/-4	29191	01 AI	Proprietary
7	6	GPT-4o-2024-05-13	1285	+3/-3	117989	OpenAI	Proprietary
7	5	Claude 3.5 Sonnet (20241022)	1283	+3/-4	38982	Anthropic	Proprietary
9	18	Athene-v2-Chat-72B	1277	+5/-6	12597	NexusFlow	NexusFlow
11	17	GLM-4-Plus	1274	+4/-4	27997	Zhipu AI	Proprietary
11	17	GPT-4o-mini-2024-07-18	1273	+3/-4	56415	OpenAI	Proprietary
11	19	Gemini-1.5-Flash-002	1272	+4/-4	30208	Google	Proprietary
11	31	Llama-3.1-Nemotron-70B-Instruct	1269	+6/-7	7672	Nvidia	Llama 3.1
12	7	Claude 3.5 Sonnet (20240620)	1268	+3/-3	86431	Anthropic	Proprietary

Рисунок 1.3 – LM Arena leaderboard (за даними [5])

Також, існує окремо LLM Leaderboard від HuggingFace, де акцент робиться саме на відкритих рішеннях та їх оцінці (див. рис. 1.4).

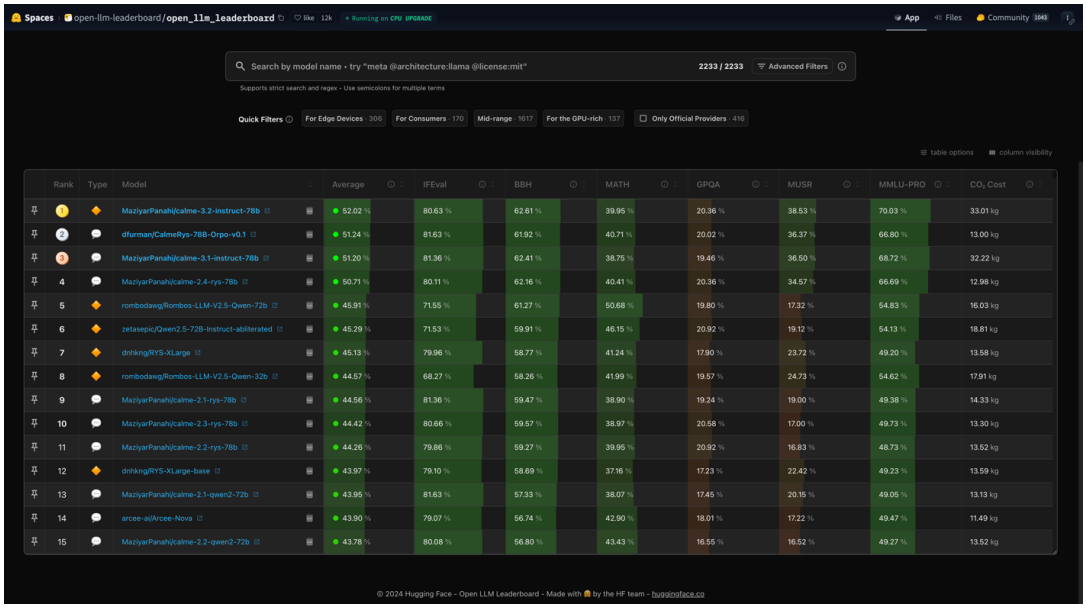


Рисунок 1.4 – HuggingFace LLM leaderboard (за даними [6])

Відомий по багатьом іншим таблицям порівнянь ArtificialAnalysis leaderboard наведено на рисунку 1.5

	FEATURES		QUALITY	PRICE	OUTPUT TOKENS/S	LATENCY	
MODEL	CREATOR	CONTEXT WINDOW	QUALITY INDEX *Normalized avg*	BLENDED *USD/1M Tokens*	MEDIAN *Tokens/s*	MEDIAN *First Chunk (s)*	FURTHER ANALYSIS
o1-preview	OpenAI	128k	86	$26.25	141.2	25.00	Model Providers
o1-mini	OpenAI	128k	84	$5.25	230.1	9.95	Model Providers
GPT-4o (Aug '24)	OpenAI	128k	78	$4.38	80.2	0.67	Model Providers
GPT-4o (May '24)	OpenAI	128k	78	$7.50	98.0	0.64	Model Providers
GPT-4o (Nov '24)	OpenAI	128k	73	$4.38	110.3	0.36	Model Providers
GPT-4o mini	OpenAI	128k	73	$0.26	113.2	0.68	Model Providers
Llama 3.3 70B	Meta	128k	74	$0.67	172.3	0.39	Model Providers
Llama 3.1 405B	Meta	128k	74	$3.50	29.8	0.73	Model Providers
Llama 3.1 70B	Meta	128k	68	$0.75	73.0	0.44	Model Providers
Llama 3.2 90B (Vision)	Meta	128k	68	$0.81	39.1	0.38	Model Providers
Llama 3.2 11B (Vision)	Meta	128k	54	$0.18	129.0	0.31	Model Providers
Llama 3.1 8B	Meta	128k	54	$0.10	182.4	0.35	Model Providers
Llama 3.2 3B	Meta	128k	49	$0.06	203.1	0.33	Model Providers
Llama 3.2 1B	Meta	128k	26	$0.04	550.8	0.34	Model Providers
Gemini 2.0 Flash (exp)	Google	2m	82	$0.00	169.2	0.54	Model Providers
Gemini 1.5 Pro (Sep)	Google	2m	81	$2.19	59.5	0.80	Model Providers
Gemini 1.5 Flash (Sep)	Google	1m	72	$0.13	180.7	0.39	Model Providers
Gemma 2 27B	Google	8k	61	$0.26	47.1	0.69	Model Providers
Gemma 2 9B	Google	8k	55	$0.12	168.0	0.43	Model Providers
Gemini 1.5 Flash (May)	Google	1m		$0.13	313.6	0.31	Model Providers

Рисунок 1.5 – ArtificialAnalysis leaderboard (за даними [7])

1.1.2 Методи ін’єкції знань

Маючи попереднє уявлення про початкове тренування моделі і її донавчання, розглянемо всі сучасні методи інтеграції знань у великі лінгвістичні моделі (включаючи вже вищезазначені методи попереднього навчання та донавчання).

1.1.2.1 Попереднє навчання (pretraining)

Цей підхід передбачає навчання моделі на текстових наборах даних (у наш час датасет скоріше за все містить також зображення, а може і аудіо), які містять значні обсяги знань. Розміри датасетів, які використовуються наразі під час тренування великих мовних моделей – це колосальні об’єми даних.

До переваг такого типу навчання можна віднести те, що модель отримує базові знання, які вона може використовувати під час виконання майбутніх завдань.

Основним недоліком можна назвати те, що знання стають статичними після навчання, тому модель потребує перенавчання для оновлення інформації (модель натренована в 2023 році буквально нічого не знає про те, що сталось після її навчання).

1.1.2.2 Донавчання (fine-tuning)

Цей підхід дозволяє налаштовувати модель для виконання конкретних завдань, використовуючи додаткові доменні дані. Наприклад, GPT-4o можна налаштувати для роботи в програмуванні чи фінансовому аналізі. Без цього етапу зараз не відбувається жодний реліз гарної моделі. Перевагами є підвищення ефективності для конкретного домену та ефективність такого підходу у порівнянні з новим тренуванням моделі з нуля. Основними недоліками є залежність від якості даних та невелика кількість доступних даних для донавчання.

1.1.2.3 Налаштування інструкцій (prompt engineering)

Основна ідея полягає в тому, щоб надати моделі кілька прикладів того, що ми даємо на вхід, та що очікуємо на вихід. Існують різновиди так званого запуску без інструкцій (zero-shot), надання одного прикладу (one-shot), кількох прикладів (few-shot) та системних інструкцій (system prompt), призначенням яких є опис задачі та методів її вирішення. Добре постаравшись та маючи досить невелику область, налаштуванням інструкцій можна повністю замінити процес донавчання моделі. Насправді, жодне реальне використання великих мовних моделей в діючих проектах не відбувається без цього кроку. Ми завжди використовуємо хоча б системні інструкції, а дуже часто і надання кількох прикладів. Це робить модель більш детерміністичною, а саме це і потрібно бізнесу.

У якості переваги можна віднести простоту впровадження, так як це є найлегшим способом надання моделі нових знань. До недоліків можна віднести залежність якості результату від формулювання інструкцій. Нечіткі, розпливчасті або помилкові інструкції можуть суттєво погіршити якість відповідей моделі. Саме тому налаштування інструкцій є одною з важливих задач сучасних проектів, які зав’язані на використанні генеративного штучного інтелекту. Поступово починають з’являтись фреймворки для автоматичної оптимізації інструкцій, але поки що це досить дорогий та не до кінця відшліфований процес.

1.1.2.4 Дистиляція знань (knowledge distillation)

Це метод машинного навчання для передачі знань від великої та складної моделі вчителя до меншої та ефективнішої моделі студента. Основна мета – зберегти продуктивність і точність великої моделі, при цьому значно зменшивши обчислювальні витрати. Модель вчителя, зазвичай, це глибока нейронна мережа, яка навчається на великих наборах даних. Вона забезпечує високу якість передбачень, але вимагає багато обчислювальних ресурсів (GPT-4o, як приклад). Модель студента, у той самий час, вчиться імітувати поведінку вчителя, зберігаючи схожу точність, але працює швидше і використовує менше пам'яті (GPT-4o-mini, як приклад).

До переваг дистиляції можна віднести ефективність, так як модель студента менша і менш ресурсовибаглива, що дозволяє використовувати її на пристроях із обмеженими обчислювальними ресурсами (наприклад, смартфони або IoT). Також, зменшення кількості параметрів веде за собою зменшений час передбачень, що є завжди гарною новиною.

До недоліків можна віднести втрату точності в порівнянні з моделлю вчителя, хоча це може бути і не така велика різниця. Також слід сюди віднести чутливість до параметрів під час передачі знань. Незважаючи на те, що кінцева модель виходить невеликого розміру, модель вчителя визначається як якісна та велика модель, що означає необхідність в обчислювальних ресурсах та наявності даних.

1.1.2.5 Інтеграція з базами знань (knowledge graphs)

Це концепція, коли моделі під час відповідей можуть звертатися до зовнішніх баз знань, таких як Wikipedia, IMBD, Google пошук, прогноз погоди та інші. Окрім доступу до відкритих ресурсів модель також може мати доступ до закритих даних конкретної компанії. Це можуть бути внутрішні документи, каталог продуктів чи база даних користувачів з уподобаннями. Наприклад, існує архітектура RAG (Retrieval-Augmented Generation), яка інтегрує механізми пошуку специфічних даних з лінгвістичною моделлю (див. рис. 1.6).

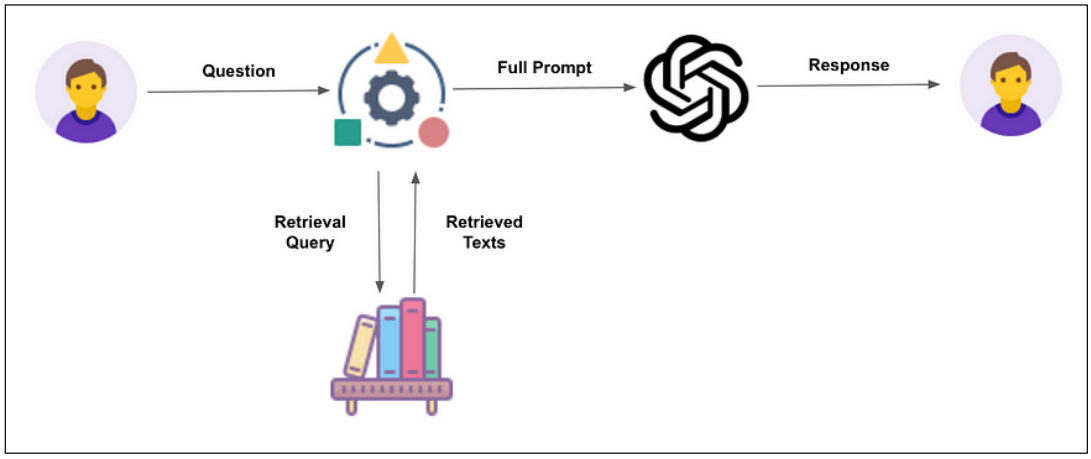


Рисунок 1.6 – Архітектура RAG (за даними [8])

До переваг можна віднести постійну актуальність знань і зменшення необхідності донавчання моделі або видозміни прикладів інструкцій. Це єдиний підхід який дозволяє досить легко масштабуватись, при цьому не витрачаючи значно більше ресурсів на донавчання або повне перенавчання моделі.

До недоліків відносять ускладнення архітектури та певні затримки у роботі моделі. Затримки виникають через час очікування відповідей від сторонніх сервісів або баз даних.

1.1.3 Проблеми та перспективи ринку

Як і будь-який ринок, генеративний штучний інтелект, а конкретно великі мовні моделі, мають наступні проблеми станом на зараз:

- статичність знань: сучасні LLM мають обмежену здатність до оновлення знань, що робить їх менш ефективними у швидкозмінних сферах;
- галюцинації: моделі можуть генерувати переконливі, але неправдиві відповіді, що значно знижує їхню достовірність (також, модель може просто повертати якесь слово багато разів, або просто не слухатись інструкцій – це більше стосується невеликих моделей);
- нестача структурованих даних: більшість знань представлена у неструктурованій формі, що ускладнює їхню інтеграцію та якісне донавчання моделей на них;
- обчислювальні витрати: навчання або донавчання моделей потребує значних ресурсів;
- інтерпретація логіки моделей: наразі можна досить легко отримати відповідь від моделі на те чи інше запитання, але зрозуміти чому саме модель зробила такий вибір, або наскільки він є достовірним може буде зовсім не тривіальною задачею.

Відповідно до проблем є низка перспектив та трендів ринку, на які варто звернути увагу:

- автоматизація інтеграції знань: розробка механізмів для автоматичного збагачення моделей знаннями через бази даних чи API у реальному часі, чим фактично вже зараз є MCP (Model Context Protocol);
- інтерпретованість моделей: розвиток інструментів, які дозволяють пояснювати, на основі яких знань моделі генерують відповіді (та наскільки вони впевнені в своїх висновках);
- зменшення обчислювальних витрат: поступове зменшення обчислювальних витрат, яке ми можемо бачити за останній рік, явно вказує на те, що ціни на АПІ до всіх моделей будуть поступово зменшуватись;
- генерація штучних даних: наразі для багатьох досліджень, як і для навчання нових моделей, використовуються набори даних згенеровані третіми моделями; цей підхід є ефективним з точки зору часу для

отримання структурованою інформації на бажану тему та з заданим об’ємом;

– агентні та мультиагентні системи: світ все далі рухається шляхом використання не лише моделей, а саме агентів, які є фактично розумною обгорткою для моделей, які дозволяють зменшувати довжину контекстного вікна шляхом поділу задач на менші, більш атомарні частини, які в свою чергу обробляються тим чи іншим агентом, який підлаштований саме під це.

Інноваційність даної роботи полягає в дослідженні методів інтеграції знань, що є однією з найважливіших тем сучасних досліджень. Це дозволить зручно масштабувати реальні проекти, у яких застосовується велика кількість специфічних знань, доступ до яких обмежено.

1.2 Огляд й аналіз літературних, наукових джерел

Цей розділ містить детальний огляд робіт пов’язаних з великими мовними моделями та методами ін’єкції знань у великі мовні моделі.

1.2.1 Великі лінгвістичні моделі

Рівень якості моделі суттєво залежить від того, наскільки великою є модель та наскільки гарно підготовлені дані використовувались для навчання конкретної моделі. Для приблизної оцінки кількості параметрів відомих нейронних мереж було досліджено кілька статей ([9-11]). Нижченаведені дані зібрані за 2023-2024 роки, але не для всіх нещодавніх релізів моделей відома кількість параметрів. Узагальнені результати досліджень наведено на рисунку 1.7.

Згідно неточних оцінок, сучасні моделі, такі як Claude Opus, можуть мати більше 2х трильйонів параметрів, але не набагато більше. Поки що невідомо про значно більші моделі, але все ще попереду.

Стаття «Training Compute-Optimal Large Language Models» пропонує новий підхід до оптимального тренування великих лінгвістичних моделей із заданим обчислювальним бюджетом [12]. Автори виявили, що сучасні моделі є переважно

недотренованими, оскільки акцент робився на збільшенні їхніх розмірів, а не на кількості навчальних токенів. Рекомендовано масштабувати модель і обсяг даних рівномірно.

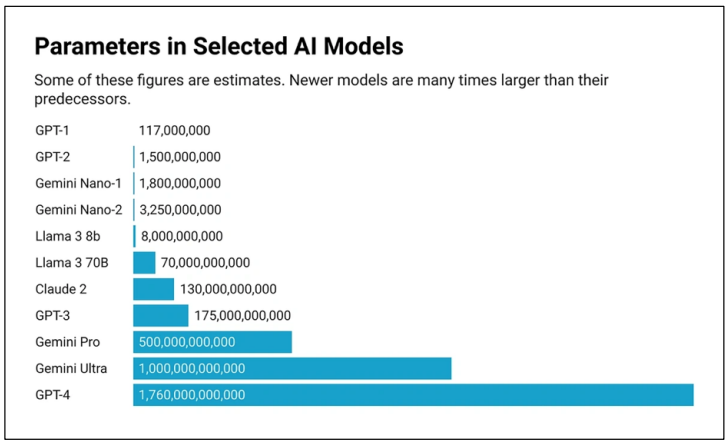


Рисунок 1.7 – Оцінка кількості параметрів нейронних мереж (за даними [10])

Для перевірки було створено модель Chinchilla (70B параметрів), яка перевершила більші моделі, такі як Gopher (280B), знижуючи витрати на обчислення й спрощуючи використання на менших апаратних ресурсах. Також пропонується оцінка, що для оптимального навчання великих мовних моделей співвідношення кількості навчальних токенів до кількості параметрів має бути в районі 15. Це означає, що моделі з меншою кількістю параметрів, але навчені на більших обсягах даних, можуть демонструвати кращі результати, ніж більші моделі з меншими обсягами навчальних даних. Також, по приблизним оцінкам вважаємо один токен рівним 5.5 байтам для прози та десь 4 байтам для задачі генерації коду.

Маючи подібну оцінку, підрахуємо, що для того щоб оптимально навчити модель розміром 1.8T параметрів (GPT-4o) необхідно мати датасет розміром 123T. З урахування того, що це була лише текстова модель, то відповідно і датасет мав бути лише текстовим. 123 терабайти текстової інформації це надзвичайно велика кількість інформації. Для цього великі компанії парсять буквально всю корисну інформацію, яка може бути тим чи іншим чином використана під час тренування. Через це і багато судових позовів, від таких

видань як New York Times, на те, що їх дані були без дозволу використані під час тренування моделей [13].

Хоча розміри таких датасетів викликають довіру, реально корисних з тих даних є не так багато. Людина за все життя не змогла б осягнути ту кількість знань якими володіють зараз мовні моделі, хоча при цьому людина не допускала б купи помилок які допускають сучасні мовні моделі. Це означає лише те, що ми навчаємо моделі неефективно, або самі моделі за своєю архітектурою не підходять до поставленої задачі [14] (див. рис. 1.8).

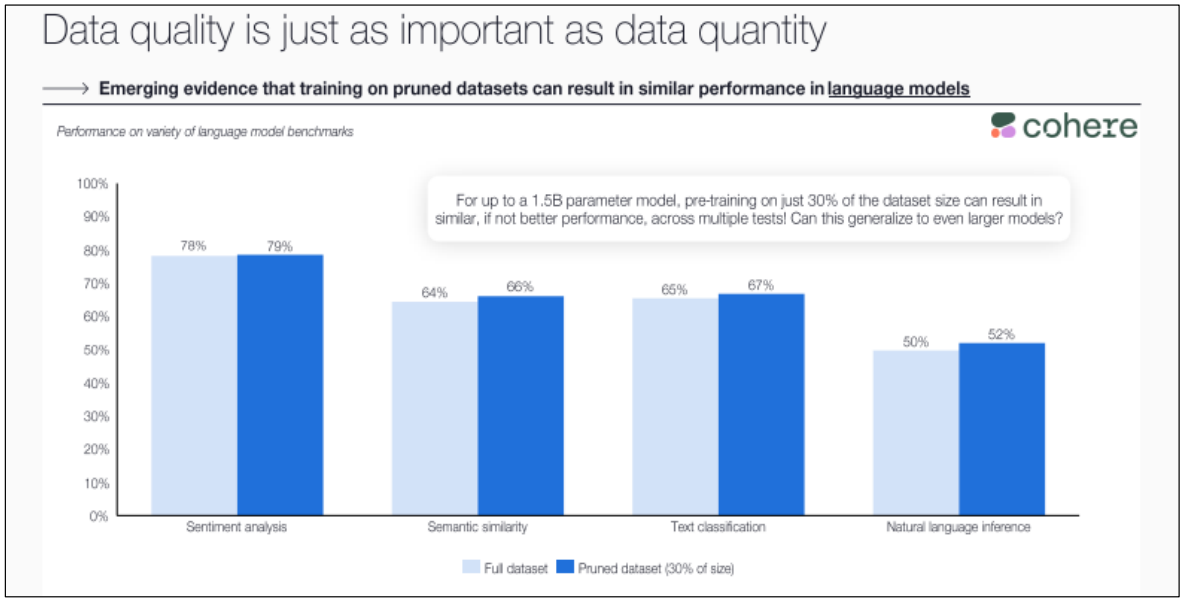


Рисунок 1.8 – Як менший датасет може дати кращу якість (за даними [14])

Стаття у Financial Times «Nvidia and the AI boom face a scaling problem» розглядає виклики масштабування в епоху AI [15]. Останніми роками спостерігається уповільнення проривів у розвитку ШІ, і деякі експерти припускають, що ми можемо стикатися з «плато масштабування». Незважаючи на величезне зростання обчислювальної потужності та розмірів моделей, результати стають дедалі менш вражаючими щодо реальних інновацій. Це може свідчити про обмеження сучасних підходів, що залежать від збільшення даних та ресурсів, і наголошує на необхідності переосмислення фундаментальних методів або нових парадигм у ШІ.

1.2.2 Методи ін’єкції знань у великі мовні моделі

У минулому популярні методи ін'єкції знань включали пошук та доповнення за допомогою певної зовнішньої бази знань [16]. У даному випадку, під базою знань розуміється набір прототипів – речень, взятих із зовнішніх текстів, які містять задані концепти або схожі за змістом фрази. У процесі відбору актуальних прототипів використовується позиційний індикатор, а під час безпосередньо процесу генерації використовується так званий модуль масштабування. Ці дві компоненти доповнюють пару кодувальника та декодувальника, створивши один з перших аналогів динамічної інтеграції знань шляхом отримання найбільш актуальних прикладів до поставленої задачі (див. рис. 1.9).

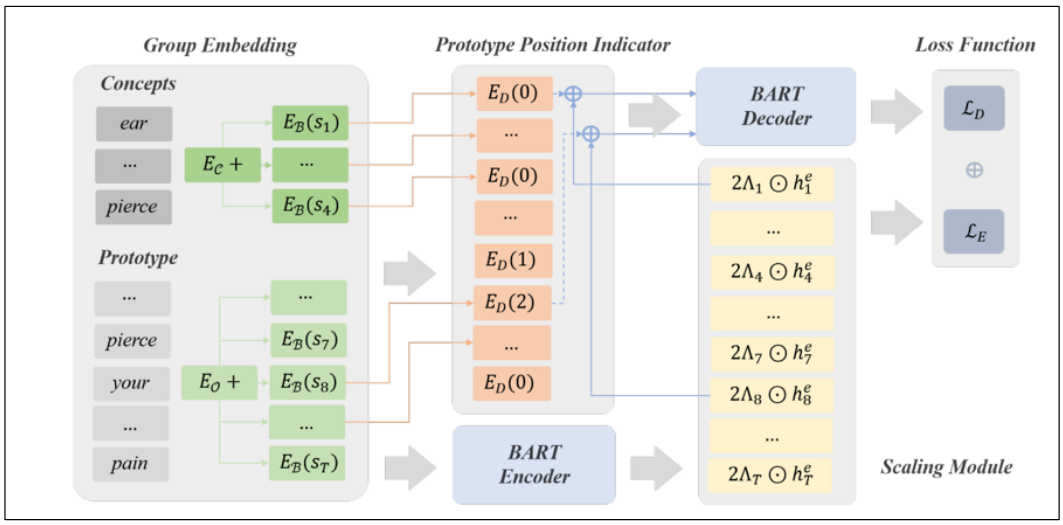


Рисунок 1.9 – Метод з позиційним індикатором та модулем масштабування (за даними [16])

Як можна побачити, база знань не обов'язково повинна бути векторною, як це практикується в сучасних системах RAG. Стаття «Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks» пропонує модель RAG, яка поєднує попередньо навчені моделі з непараметричною пам'яттю, що є індексом текстів із Вікіпедії, доступним через нейронний пошуковий модуль [17]. Це дозволяє моделі динамічно отримувати актуальну інформацію, покращуючи її

здатність генерувати точні та контекстно ґрунтовні відповіді. Автори порівнюють дві модифікації RAG: одну, яка використовує однакові витягнуті шматки протягом усієї генерації, і іншу, яка дозволяє використовувати різні шматки для кожного токена. Експерименти показують, що моделі RAG перевершують традиційні моделі послідовностей та архітектури, орієнтовані на конкретні завдання. Такий підхід дозволяє отримувати більш специфічні, різноманітні та ґрунтовні відповіді, що вирішує обмеження великих попередньо навчених моделей. Наприклад, проблеми з оновленням знань і наданням джерел для прийнятих рішень (див. рис. 1.10).

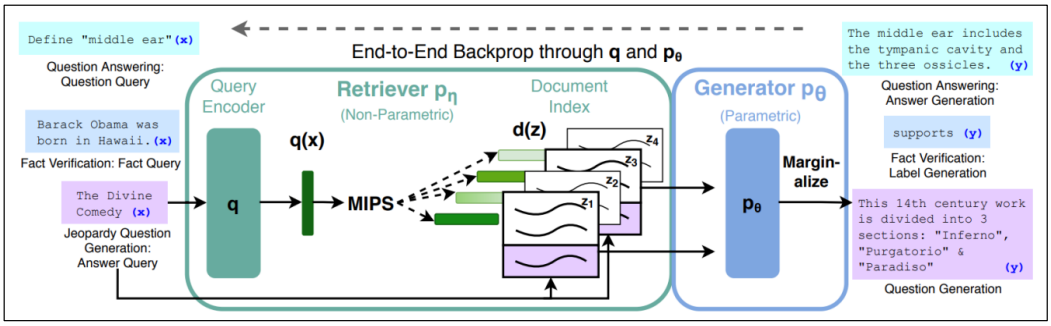


Рисунок 1.10 – Модель RAG з пошуком даних по індексу (за даними [17])

З впровадженням RAG систем, які побудовані на базі векторної бази даних, було запропоновано альтернативні конструкції, такі як графи знань. Стаття «Knowledge Injection to Counter Large Language Model (LLM) Hallucination» досліджує метод ін’єкції знань (KI) для зменшення явища галюцинацій у великих лінгвістичних моделях [18]. Вона показує, як контекстні дані з графу знань, що описують сутності, можна використовувати для формування запитів до LLM. Такий підхід зменшує кількість фактичних помилок у згенерованому тексті. Дослідження на прикладі відповіді на онлайн-відгуки показало, що KI допомагає моделі генерувати більш ґрунтовний та якісний текст, навіть коли використовуються менші моделі, ніж тодішній OpenAI аналог (GPT-3). Це дозволяє економити ресурси та покращує якість генерованого контенту (див. рис. 1.11).

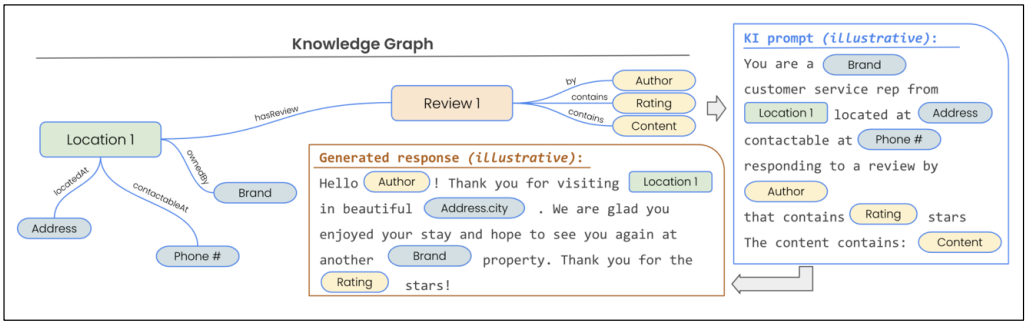


Рисунок 1.11 – Приклад використання графу знань (за даними [18])

У якості покращення методів RAG, нещодавно від Anthropic вийшла стаття про новий стандарт у запровадженні RAG систем [19]. У порівнянні з попередніми версіями, у ній наявний TF-IDF індекс, або іншими словами – пошук по ключовим словам. Основна ідея у використанні двох індексів замість одного полягає у знаходженні більш релевантних частин з бази даних. Після отримання контексту з обох індексів використовуються алгоритми ранжування (див. рис. 1.12).

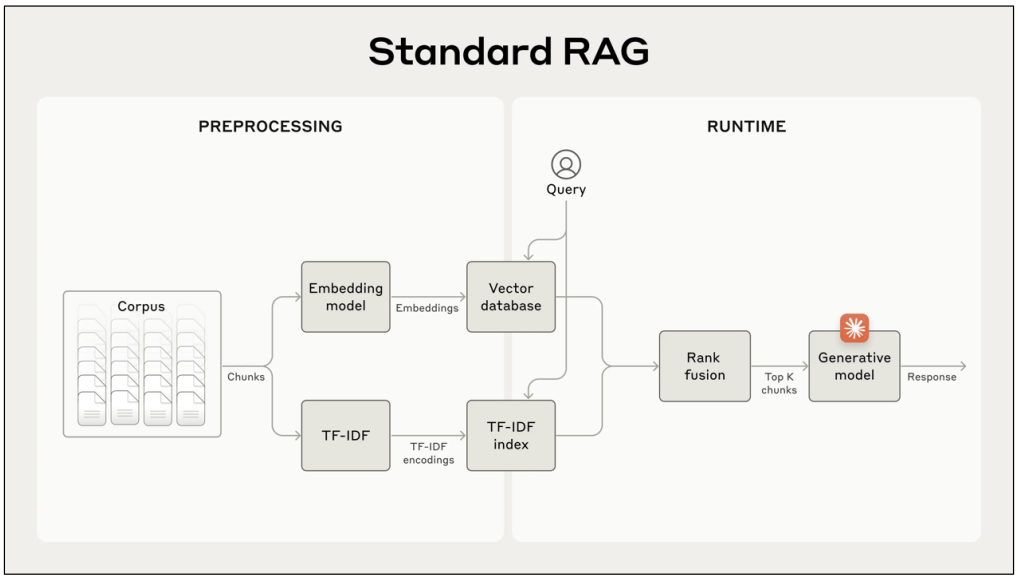


Рисунок 1.12 – Сучасний стандарт RAG (за даними [19])

Також, у цій статті представлено ще одну варіацію підготовки даних до індексації (див. рис. 1.13). Основна ідея полягає в додаванні до кожного шматка документу додаткової інформації, яка описує його значення у початкову текст.

Таким чином ми отримуємо більш насичені інформацією шматки даних, які в майбутньому має бути легше знаходити під час запитів.

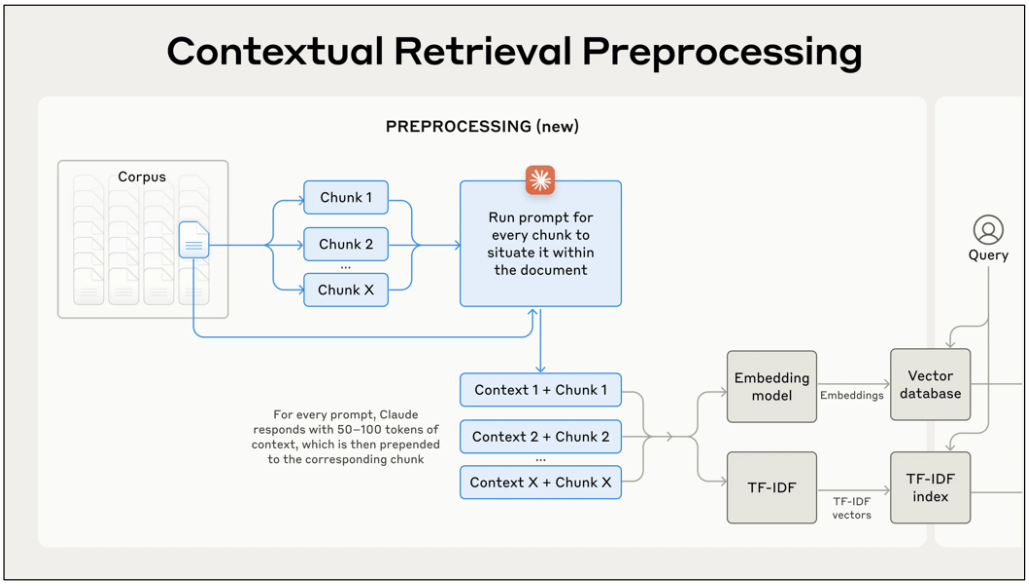


Рисунок 1.13 – Спеціальна підготовка даних до індексації (за даними [19])

Вище описаний більш загальний випадок RAG системи. У випадках, коли від вхідного тексту може значно залежати вибір методу обробки, перед використанням RAG системи використовують семантичні маршрутизатори. У статті «Benchmarking Conversation Routing in Chatbot Systems Based on Large Language Models» автори досліджують різні підходи до семантичної маршрутизації в чат-ботах, що використовують великі мовні моделі (LLM) [20]. Вони пропонують багатомовний тестовий набір даних для оцінки ефективності цих підходів та проводять експерименти для виявлення їхніх переваг і недоліків. У результаті дослідження виявлено, що семантичний маршрутизатор на основі латентних векторів забезпечує результати, близькі до LLM-класифікатора, але водночас надає високу швидкість, інтерпретованість і керованість завдяки налаштуванню набору прикладів і меж відхилення для кожного маршруту.

Ще один з поширених підходів ін'єкції знань описано в статті «LORA: LOW-RANK ADAPTATION OF LARGE LANGUAGE MODELS» [21]. Основним з статті є метод LoRA (Low-Rank Adaptation) для адаптації великих мовних моделей, таких як Llama-3.2 90B, без необхідності повного донавчання. LoRA

заморожує ваги моделі та додає матриці низького рангу для адаптації під нові знання. Це значно знижує кількість параметрів, які потрібно тренувати. LoRA значно покращує якість моделі, одночасно знижуючи апаратні вимоги у порівнянні зі звичайним донавчанням, що робить її ефективним рішенням для дійсно великих моделей. (див. рис. 1.14).

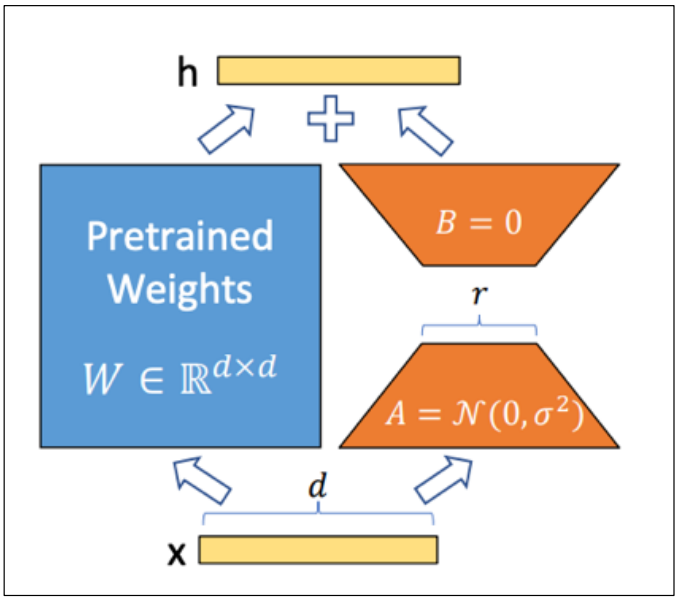


Рисунок 1.14 – Графічне представлення LoRA (за даними [21])

Існують різні конфігурації того, які шари нейронної мережі потрібно заморожувати під час такого донавчання, а які потрібно залишати незамороженими. Вибір конфігурації донавчання з використанням LoRA значною мірою впливає на забування моделлю попередніх знань, тривалість навчання та необхідні обчислювальні ресурси.

Продовженням ідеї LoRA стала стаття «QLORA: Efficient Finetuning of Quantized LLMs», яка пропонує метод QLoRA [22]. Цей метод дозволяє ефективно донавчати великі мовні моделі, такі як Llama 65B параметрів, на споживчому GPU із 48GB відео пам’яті. QLoRA використовує 4-бітну квантизацію з новим форматом NormalFloat, метод подвійної квантизації та оптимізатори, які керують піками пам’яті. Це зберігає якість 16-бітного донавчання, суттєво знижуючи апаратні вимоги (див. рис. 1.15).

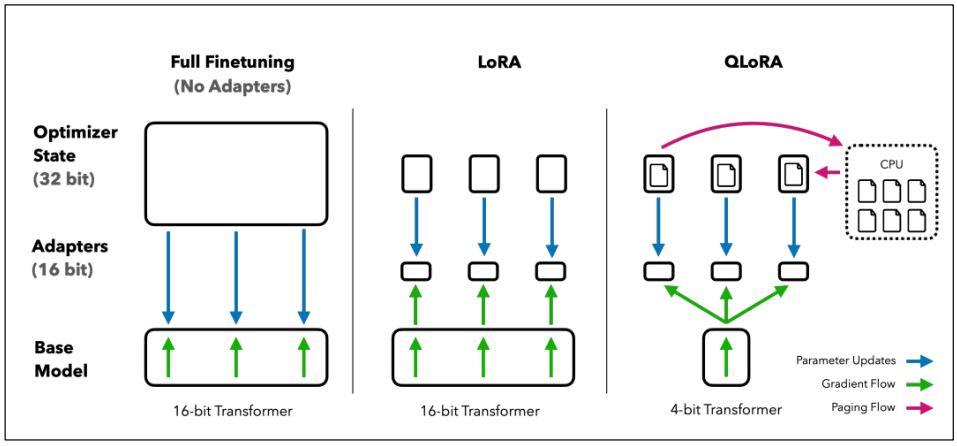


Рисунок 1.15 – Графічне зображення алгоритму QLoRA (за даними [22])

Існує досить цікава публікація «Injecting New Knowledge into Large Language Models via Supervised Fine-Tuning» щодо дослідження методів донавчання моделі [23]. У статті детально досліджуються методи оновлення знань великих мовних моделей за допомогою додавання нової, специфічної для домену інформації, зокрема щодо останніх спортивних подій. Основна ідея полягає у дослідженні впливу структурованості набору даних на узагальнення знань моделі. Розглянуто два підходи до створення навчальних даних:

– масштабування на основі токенів: збільшення обсягів тренувальних даних шляхом додавання більшої кількості тексту (токенів), що покращує точність відповіді, але може призводити до нерівномірного охоплення знань;
– масштабування на основі фактів: зосередженість на окремих фактах для створення більш структурованих даних, що дозволяє ефективніше інтегрувати нові знання в модель.

Експерименти з GPT-4 показали, що використання підходу на основі фактів значно покращує результати моделі у відповідях на запитання, пов'язані з новими даними (див. рис. 1.16).

Також вартує додати, що навчання на датасеті, який згенеровано на основі фактів, показує стабільні покращення в залежності від кількості ітерацій. У той

самий час, набір даних на основі токенів, зі збільшенням кількості ітерацій, має тенденцію до перенавчання та втрати початкових знань.

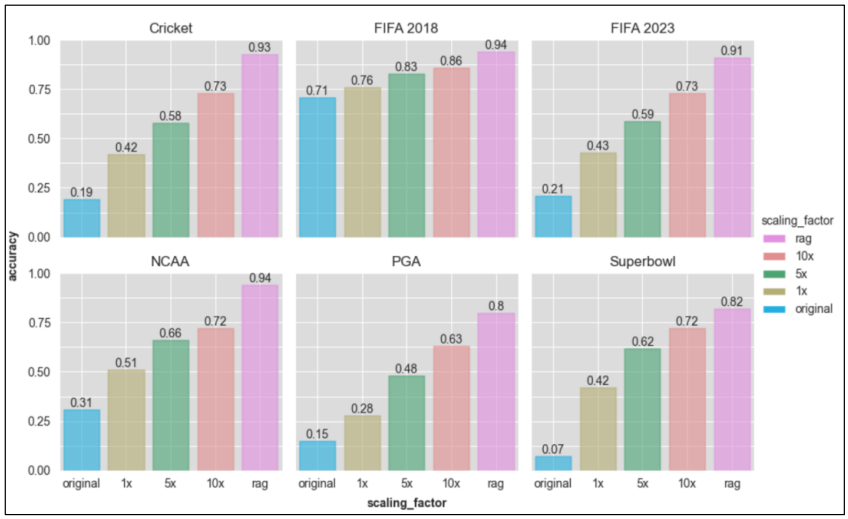


Рисунок 1.16 – Результати підходу на основі фактів (за даними [23])

Стаття «Long-context LLMs Struggle with Long In-context Learning» присвячена аналізу можливостей великих мовних моделей (LLM) обробляти довгі контексти за допомогою нового бенчмарку LongICLBench [24]. Автори підкреслюють, що існуючі LLM, хоча й здатні працювати з довгими текстами, але мають обмежену ефективність у складних задачах, що вимагають повного розуміння контексту. Використовуючи класифікацію з аномально великим числом класів (до 174), дослідження демонструє, що продуктивність моделей значно знижується з ростом довжини вхідних даних. Крім того, важливу роль грає позиція даних у контексті. Автори закликають до вдосконалення моделей для обробки довгих текстів (див. рис. 1.17).

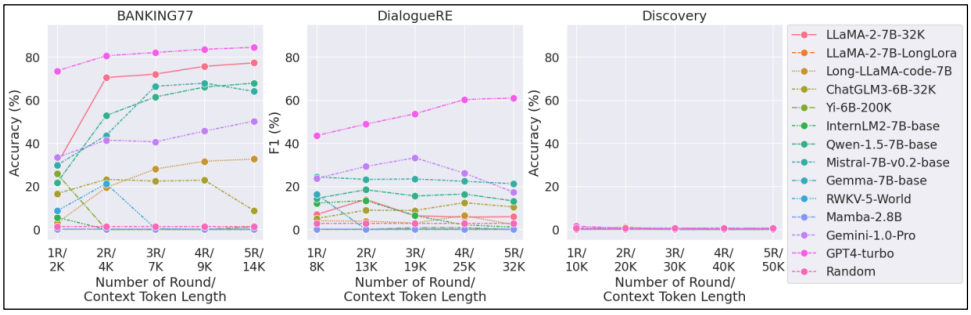


Рисунок 1.17 – Результати моделей на бенчмарку LongICLBench (за даними [24])

Було проведено окремий ряд дослідження у OpenAI, по результатам якого, в залежності від того, під знаннями моделі мається на увазі стилістичні особливості або конкретні знання, існують різні особливості щодо набуття моделлю цих знань [25]. Часто проекти починають з налаштування інструкцій та рухаються у той чи інший бік оптимізації в залежності від конкретної задачі та даних. Для отримання найкращих кращих результатів використовуються кілька методів ін'єкції знань разом, нівелюючи недоліки один одного (див. рис. 1.18).

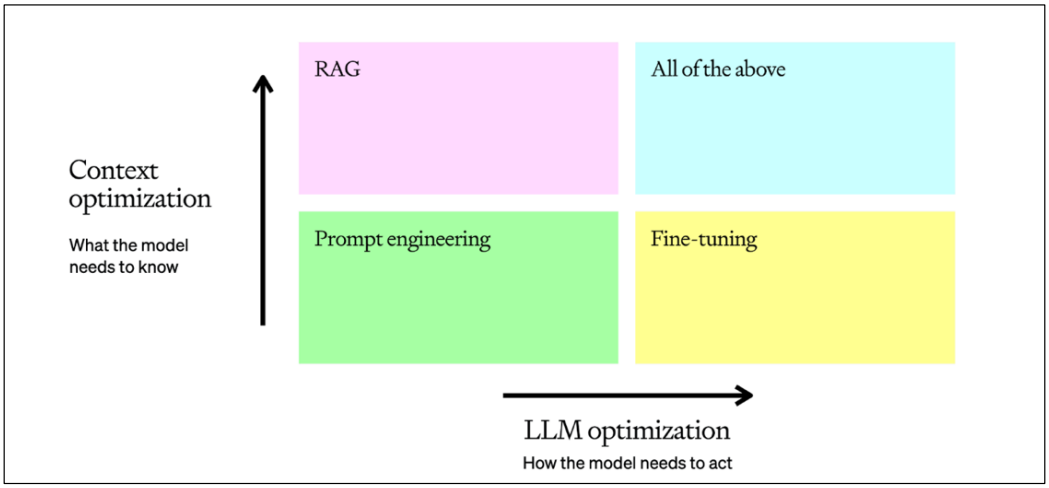


Рисунок 1.18 – Оптимізація роботи LLM для ін'єкції знань (за даними [25])

Проаналізувавши багато публікацій на тему ін'єкції знань, не було знайдено детального порівняння модифікацій при використанні RAG систем на конкретному датасеті.

1.3 Постановка задачі

Цей підрозділ описує деталі щодо задачі дослідження, методів дослідження та ресурсів, необхідних для виконання дослідження.

1.3.1 Опис задач дослідження

Основна ідея зосереджена на дослідженні методів ін'єкції знань у великі лінгвістичні моделі (LLM) з метою оцінки їх ефективності при виконанні реальних завдань. Основна мета – зробити порівняльний аналіз різних типів

ін’єкції знань, оцінити їхню ефективність на спеціально згенерованому датасеті та визначити найкращі підходи для підвищення точності та релевантності відповіді, що генерується моделями.

У якості фреймворку для оцінки якості моделей буде використано DeepEval.

Виконання цих завдань дозволить отримати картину ефективності різних RAG підходів до ін’єкції знань у LLM та сприяти вдосконаленню моделей для практичного використання.

1.3.1.1 Генерація спеціалізованого датасету, який включає тексти, питання та відповіді на основі визначеного набору даних

У якості методу генерації буде обрано потужні великі мовні моделі сучасності, такі як GPT-4o. У першу чергу потрібно виокремити предметну область, або кілька з них, для детального аналізу щодо можливих даних у цих областях. Після цього потрібно підлаштувати інструкції для конкретної великої лінгвістичної моделі та перевірити що отриманий набір даних є належної якості. Датасет повинен містити не менше 20000 токенів. Маючи датасет знань, потрібно згенерувати тестовий датасети питань та відповідей. Тестовий датасет повинен містити не менше 50 прикладів.

1.3.1.2 Реалізація та тестування кількох підходів до ін’єкції знань

Конкретно пропонується розглянути наступні підходи:

- гібридний пошук та просто семантичний пошук (гібридний пошук включає в себе пошук по ключовим словам на основі TF-IDF; під семантичним пошуком мається на увазі векторний пошук);
- кількість актуальних чанків які модель буде отримувати з бази даних (було вирішено зупинитись на 3 та 5 чанках відповідно);
- розмір чанків з даними (було вирішено зупинитись на 128 та 256 токенах);

– токенайзер на чанки (так як наші дані є в більшій мірі простою текстовою інформацією то було вирішено зупинитись на розбитті чанків по символам та по реченням).

1.3.1.3 Побудова порівняльної таблиці продуктивності зазначених підходів

Для об’єктивного оцінювання підходів буде використано наступний перелік метрик:

– актуальність відповідей, у якості основного критерія порівняння;
– ціна обробки конкретної конфігурації;
– ґрунтовність моделі на додаткових знаннях;
– відповідність отриманого контексту до початкових запитань.

1.3.2 Обґрунтування вибору методів дослідження

Для задачі дослідження методів ін’єкції знань потрібно вибрати кілька релевантних лінгвістичних моделей.

У порівнянні беруть участь як закриті, так і відкриті моделі. Від OpenAI було обрано 3 моделі (GPT-4o, GPT-4o-mini, o1-mini) які коштують досяжно для досліджень. Від прямого конкурента, Google, було обрано моделі Gemini 1.5 Pro та Gemini 1.5 Flash. Також, у трійці закритих провайдерів є Anthropic, звідки було взято модель Claude 3.5 Sonnet.

З відкритих моделей було обрано Llama 3.2 11B, як одну з найновіших моделей цього сімейства. Також було вирішено розглянути Llama 3.1 405B (найбільшу відкриту модель), та Gemma 2 27B (як найефективнішу модель свого розміру). Llama 3.2 90B було обрано як щось середнє між Llama 3.1 405B та Gemma 2 27B.

У якості критеріїв для оцінки було обрано наступні:

– якість знань (нормалізована зважена якість моделі виходячи з багатьох бенчмарків [7]); базова якість моделі значно впливає на майбутні можливості з додатковими знаннями;

- масштабованість (можливість динамічно та швидко масштабувати систему на базі конкретної моделі); цей показник є важливим, через те що людство цікавлять корисні рішення які масштабуються, а не ті, які обмежуються лише експериментами;
- довжина контекстного вікна (максимальна кількість токенів на вхід до моделі); цей критерій є важливим, через те що довжина контекстного вікна напряму впливає на кількість знань, які можна додатково передати моделі; чим більше контекстне вікно – тим краще для інтеграції нових знань;
- мультимодальність (підтримка моделлю різних типів інформації, таких як текст, зображення, відео, аудіо); це є важливим критерієм, так як більш мультимодальну модель можна використати у більшій кількості задач, що безпосередньо впливає на варіативність даних для методів ін’єкції знань;
- закритість (ступінь доступу до моделей); цей показник є дуже важливим, бо чим менш закритою є модель, тим більше можна зробити маніпуляцій над моделлю для інтеграції додаткових знань.

У якості методу вирішення оптимізаційної задачі було обрано лінійну адитивну згортку з ваговими коефіцієнтами, так як у нашому випадку важливий внесок кожного з факторів. Фактори було нормалізовано за принципом максимальності, після чого було використано принцип Парето для виключення недомінуючих альтернатив. У якості методу визначення вагових коефіцієнтів було обрано метод ранжування. Масштабованість – 5, Відкритість – 4, Довжина контекстного вікна – 3, Якість знань – 2, Мультимодальність – 1. У результаті отримуємо наступні коефіцієнти: Масштабованість – 0.33, Відкритість – 0.27, Довжина контекстного вікна – 0.2, Якість знань – 0.13, Мультимодальність – 0.07.

Було розраховано корисність кожної з альтернатив з використанням лінійної адитивної згортки (див. табл. 1.1). Значення в таблиці нижче наведенні з заокругленнями.

Таблиця 1.1 – Розрахунок корисності альтернатив

	Якість знань	Масштабованість	Довжина контекстного вікна	Мультимодальність	Відкритість	Корисність
o1-mini	0,82	1,00	0,06	0,25	0,33	0,556
Gemini 1.5 Pro	0,80	1,00	1,00	1,00	0,67	**0,885**
Llama 3.2 90B	0,67	0,40	0,06	0,50	1,00	0,537
Llama 3.2 11B	0,53	0,80	0,06	0,50	1,00	**0,651**
Llama 3.1 405B	0,72	0,20	0,06	0,25	1,00	0,459
Gemma 2 27B	0,61	0,80	0,00	0,25	1,00	0,632

У результаті можна явно побачити, що у нас є лідер, і це Gemini 1.5 Pro. Це сталось за рахунок усіх високих показників та неймовірно великого контекстного вікна, яке зробило його однозначним лідером по цьому показнику. Це перша модель, яку буде використано для подальшого дослідження.

Наступною за корисністю є модель Llama 3.2 11B. Її також буде використано у якості моделі для подальшого дослідження, через те що вона є абсолютно відкритою, а тобто ми можемо проводити будь-які маніпуляції з вагами моделі.

Використання таких інструментів, як DeepEval, забезпечує інтегральну оцінку продуктивності, тоді як стандартні метрики дозволяють виміряти якість згенерованого тексту з лінгвістичної точки зору.

Neo4j, завдяки своїй структурі, ідеально підходить для бази знань, яка природно моделює складні зв'язки між даними. Neo4j дозволяє зберігати й обробляти графи знань, забезпечуючи швидкий доступ до релевантної інформації через мову запитів Cypher.

1.3.3 Необхідні ресурси

Основними ресурсами, необхідними для проведення дослідження є наступні:

- доступ до великих мовних моделей (Gemini, GPT-4o, Llama 3.2);
- обчислювальні ресурси (GPU із достатньою кількістю пам’яті для інференсу; наприклад, NVIDIA RTX 4090);
- інструменти для створення та управління базами знань (Neo4j);
- програмне забезпечення для оцінки моделей, зокрема бібліотека DeepEval;
- згенеровані датасети.

2 ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ

2.1 Дані

У нашому дослідженні було вирішено згенерувати датасет з використанням сучасних великих лінгвістичних моделей, таких як GPT-4o. Це було зроблено з урахуванням того, наскільки часто саме GPT-4o використовується для генерації датасетів у різного роду дослідженнях та донавчанні моделей. У якості інтерфейсу до GPT-4o буде використано OpenAI Playground, як часткову альтернативу платній підписці на ChatGPT [26] (див. рис. 2.1).

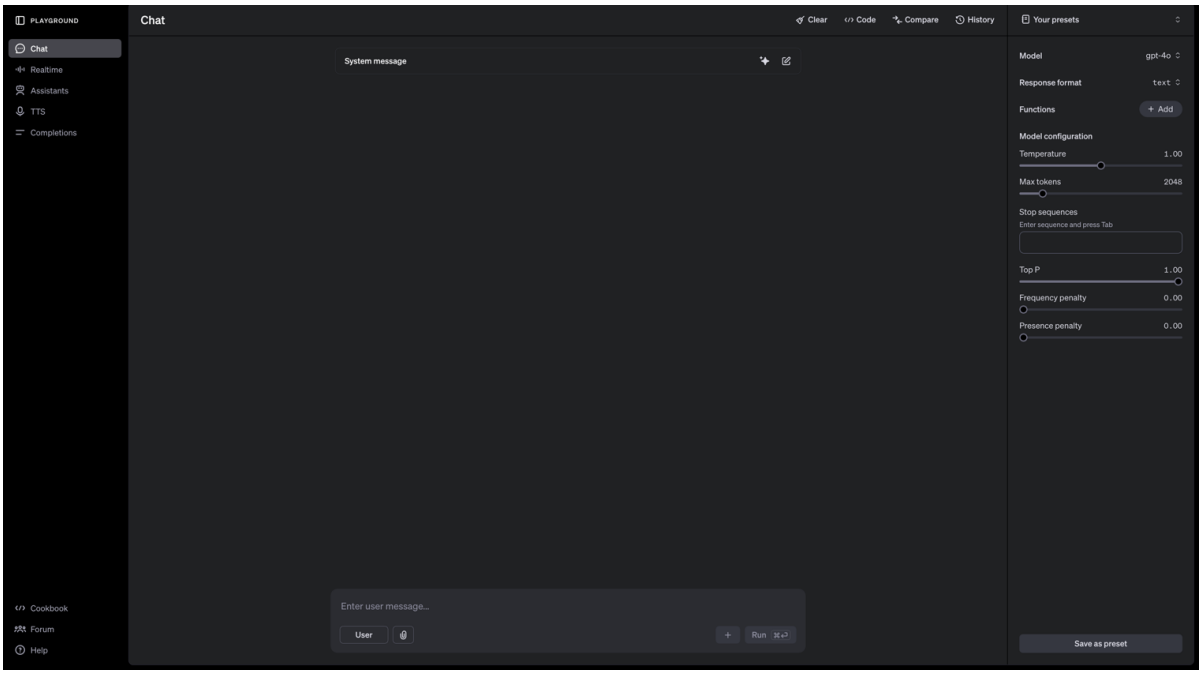


Рисунок 2.1 – Інтерфейс OpenAI Playground (за даними [26])

Вибір саме OpenAI Playground зумовлений тим, що тут можна вручну обрати модель для роботи, задати параметри генерації та швидко експериментувати з отриманим набором даних. У якості попередньої теми для генерацій було обрано перелік уявних компаній, які мають певну історію, перелік продуктів, штат людей з детальним описом хто чим займається, за що відповідає, та багато додаткової інформації. Акцент саме на необхідності штучних даних зроблено по результатам теоретичних досліджень. Ідея цього полягає в тому, щоб

зробити експеримент більш достовірним та зменшити шанси того, що моделі вже колись бачили ці дані до нуля.

У якості сховища даних було обрано графову базу даних Neo4j. З корисних особливостей Neo4j можна виділити те, що при реєстрації надається безкоштовний екземпляр бази даних, який є досить зручним у користуванні та експериментах [27] (див. рис. 2.2).

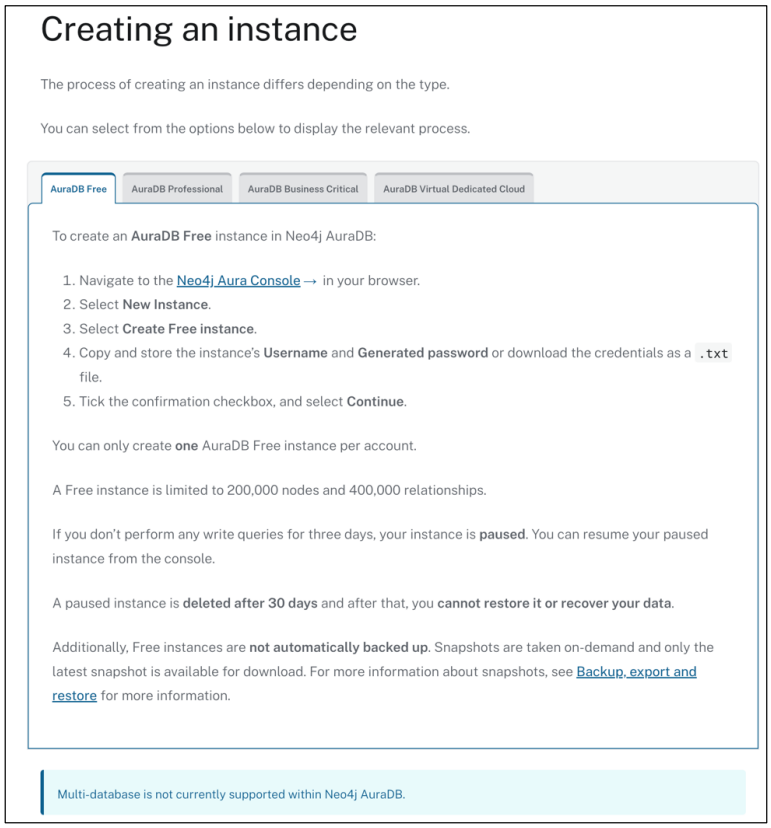


Рисунок 2.2 – Реєстрація безкоштовного екземпляру Neo4j (за даними [27])

Безкоштовна база даних має свої обмеження, такі як: кількість вершин не більше 200 000 та кількість ребер не більше 400 000. Також, певні незручності приносить той факт, що екземпляр бази даних сам вимикається якщо у нього нічого не записувати протягом 3х днів та повністю видаляється якщо нічого не робити протягом 30 днів. Але, у якості бази даних для нашого дослідження це не є суттєвими обмеженнями які будуть заважати експериментам. Особливістю Neo4j є те, що у якості мови запитів використовується Cypher. Основна його ідея в тому, щоб спростити всі маніпуляції з даними до інтуїтивно зрозумілих [28]. На

рисунку 2.3 наведено графічне зображення побудови базового запиту мовою Cypher.

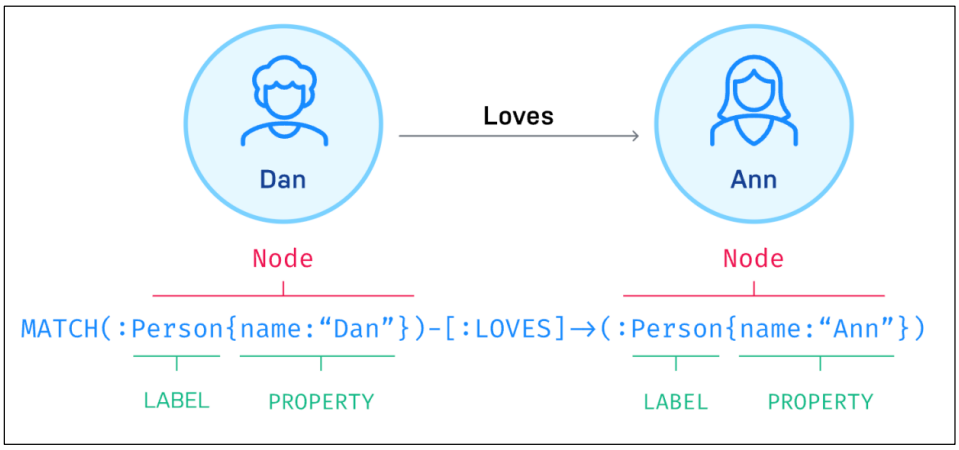


Рисунок 2.3 – Графічне зображення побудови базового запиту мовою Cypher (за даними [28])

Ще одною особливістю екосистеми Neo4j є наявність неймовірно зручного середовища візуалізації бази даних – Aura. Aura є системою управляння бази даних, з вбудованими можливостями редактору запитів до бази даних Neo4j та безпосередньою візуалізацією результатів. Візуалізації можна масштабувати, видозмінювати, міняти кольори, фільтрувати та ще багато чого іншого [29] (див. рис. 2.4).

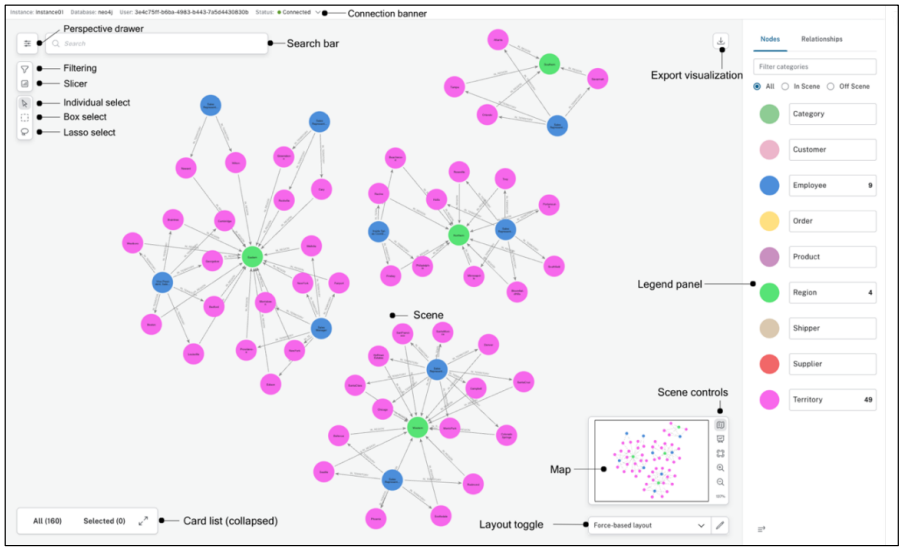


Рисунок 2.4 – Neo4j Aura UI (за даними [29])

2.2 Лінгвістичні моделі

Для дослідження методів ін’єкції знань було обрано 2 моделі, а саме Gemini Pro 1.5 та Llama 3.2 11B. Для доступу до Gemini 1.5 Pro буде використовуватись Google Cloud Platform [30] (див. рис. 2.5).

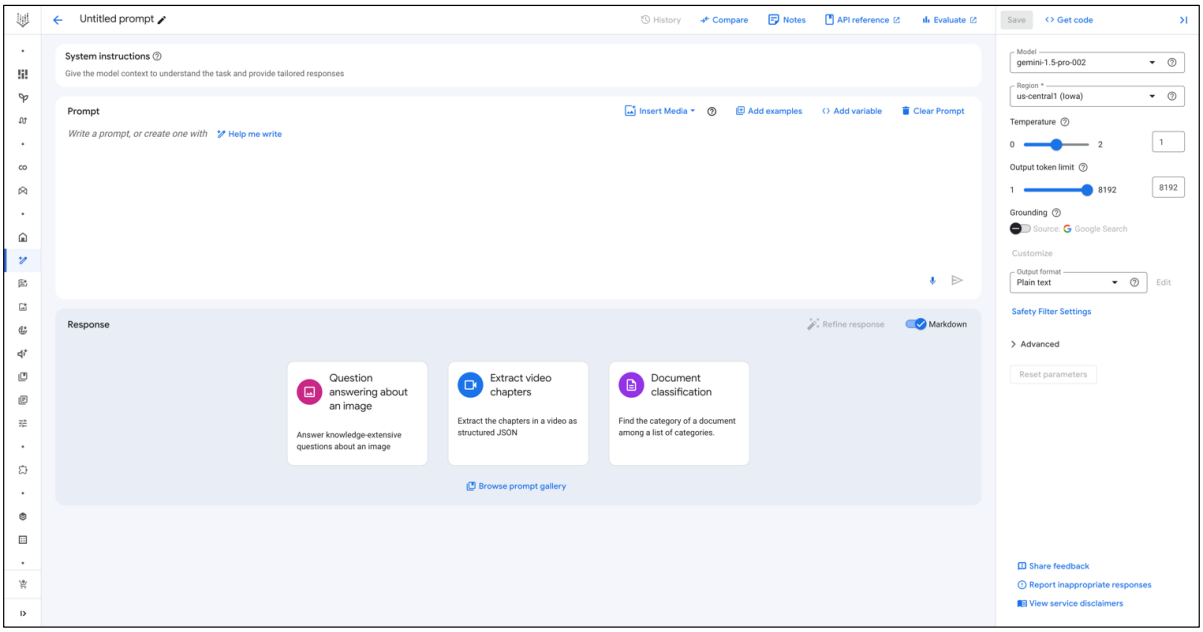


Рисунок 2.5 – Gemini UI через Google Cloud Platform (за даними [30])

Під час дослідження планується використовувати як UI так і API до Gemini Pro. Для використання Gemini API потрібно підключити та активувати модулі Vertex AI та Cloud Storage через Google Cloud Platform.

У той же час, для доступу до Llama 3.2 11B буде використовуватись репозиторій на платформі HuggingFace [31]. За останній місяць модель скачало майже 3 мільйони людей, що є дуже великим значенням для даної платформи. Для майбутнього використання у дослідженні було подано заявку на отримання доступу до моделі (див. рис. 2.6). Також, було отримано позитивну відповідь щодо використання ваг Llama 3.2 11B Vision Instruct для дослідження. Ваги моделі можна також отримати безпосередньо через репозиторій на GitHub, але це менш зручно через необхідність додатково підключати локальну модель через інтерфейси HuggingFace для зручнішої роботи.

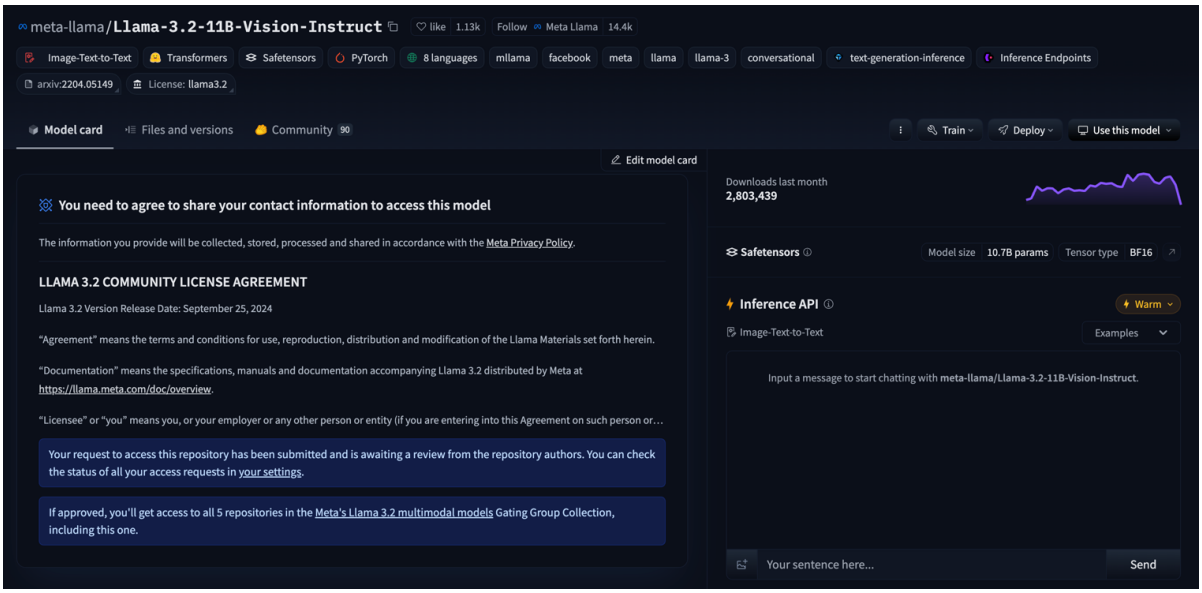


Рисунок 2.6 – Картка моделі Llama-3.2-11B-Vision-Instruct (за даними [31])

Для ін'єкції знань методами налаштування інструкцій, вищенаведені моделі мають досить різні інтерфейси. Для побудови інструкцій при використання Llama-3.2-11B-Vision-Instruct потрібно використовувати спеціальні теги, такі як: «<|start_header_id|>», «<|end_header_id|>», «<|eot_id|>», «<|begin_of_text|>» та «<|image|>» [32] (див. рис. 2.7). Основна ідея полягає в тому, щоб явно виділяти де і яку інформацію ми маємо на увазі, будь-то системні інструкції, додаткова інформація, зображення або запитання до моделі. Подібні теги є сучасною практикою для великих лінгвістичних моделей. Дуже важливим є пильнування за тим, яку саме модель ви використовуєте, бо моделі від різних провайдерів можуть мати зовсім різні теги, або їх значення. Так, навіть при переході між моделями Llama 3.2 3B та Llama 3.2 11B можна побачити те, що додався тег «<|image|>». Це відбувається через те, що ми працюємо з моделлю на нижньому рівні. Приблизно те саме відбувається на серверних частинах сучасних API, таких як OpenAI API, Anthropic API та Google Vertex AI API коли ми передаємо значення змінних у якості вхідних параметрів. На вхід до API достатньо згрупувати інструкції у коректні частини та все зроблять за вас за лаштунками. У цьому є як свої переваги так і недоліки.

Рисунок 2.7 – Приклад використання тегів для створення інструкції (за даними [32])

2.3 Оцінка результатів

Для оцінки чи дві відповіді співпадають не можна просто використовувати порівняння рядків або векторну схожість. Обидві відповіді можуть бути сформульовані абсолютно по різному але мати однаковий зміст. Хоча векторна схожість і є досить непоганим початковим варіантом, пропонується використовувати великі лінгвістичні моделі для порівняння відповіді з дійсно вірною, маючи питання, дві відповіді на нього, та весь набір даних. Таких підхід є значно більш стабільним для оцінки, саме тому його і буде використано у дослідженні.

Для оцінки якості моделей буде використано метрики актуальності відповіді (наскільки модель точно відповіла на поставлене запитання), ґрунтовності відповіді (наскільки ця відповідь ґрунтується на додатковому контексті отриманому з БД) та доречність контексту (наскільки доречні дані ми отримуємо з БД). Для розрахунку буде використано бібліотеку DeepEval. Було зроблено саме такий вибір через те, що вона надає широкий спектр метрик для обрахунку, при цьому маючи дуже зручну реалізацію додаткових викликів у разі проблем з боку моделі [33].

3 ПРАКТИЧНЕ ДОСЛІДЖЕННЯ

3.1 Дані

Головною задачею практичного дослідження є перевірка того, наскільки модель вміє використовувати нові дані. Після проведеного теоретичного дослідження стало зрозуміло, що для впевненості в тому що дані є дійсно новими та модель про них нічого не знає потрібно згенерувати нові дані.

Для генерації було використано OpenAI Playground та моделі gpt-4o та gpt-4.5-preview. Параметри для генерації наведено на рисунку 3.1.

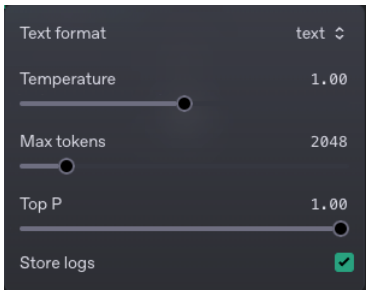


Рисунок 3.1 – Параметри моделі при генерації датасету (створений самостійно)

У якості стратегії для генерації даних було обрано генерацію інформації по конкретним компаніям. На вхід давалась назва компанії, чим вона займається, та по ним генерувався детальний опис цієї компанії, з усіма необхідними деталями (див. рис. 3.2).

```
Generate me a private corpus with some details mentioning the imagined
NeonByte Interactive – Video game development and VR/AR experiences
A list of products, prices, responsible stuff, etc.
I want to use it as my private corpus for the LLM fine-tuning
You can generate really a lot of the text. The more the better.
```

Рисунок 3.2 – Інструкції для генерації датасету (створений самостійно)

Після генерації відповідної інформації про компанію було згенеровано питання та відповіді. Приклад інструкції наведено на рисунку 3.3.

```
generate 20 questions & answers from the dataset above, structure them in
the python list with "question" & "answer" fields
```

Рисунок 3.3 – Інструкції для генерації питань та відповідей (створений самостійно)

Загалом було згенеровано інформацію про 9 різних компаній. Розмір датасету в токенах 21059 (з використанням токенізатора «o200k_base»). Детальна інформація по розподілу токенів наведена на рисунку 3.4.

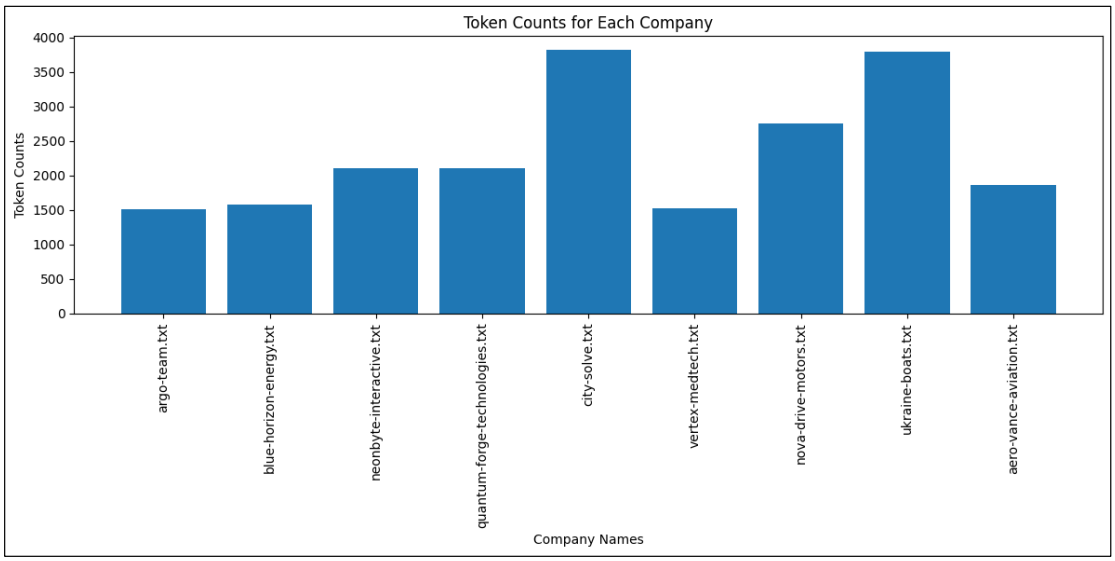


Рисунок 3.4 – Детальна інформація по розподілу токенів (створений самостійно)

Датасет було завантажено у безкоштовний екземпляр бази даних Neo4j (див. рис. 3.5). У експериментах кількість записів в БД варіювалась від 100 до 300.

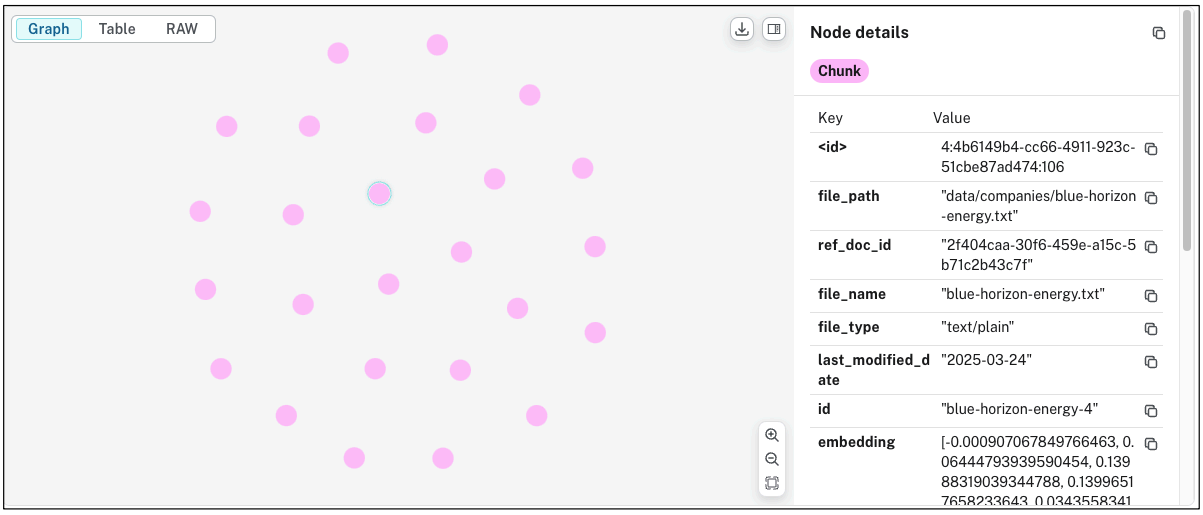


Рисунок 3.5 – Датасет у базі даних Neo4j (створений самостійно)

Що ж стосується датасету запитань, то в загальному було згенеровано 110 запитань. Для експериментів було використано лише 50 запитань, так як

збільшення кількості запитань значно збільшує витрати на експерименти. Приклади запитань наведено в додатку Д. Детальна інформація по розподілу питань по компаніям наведена на рисунку 3.6.

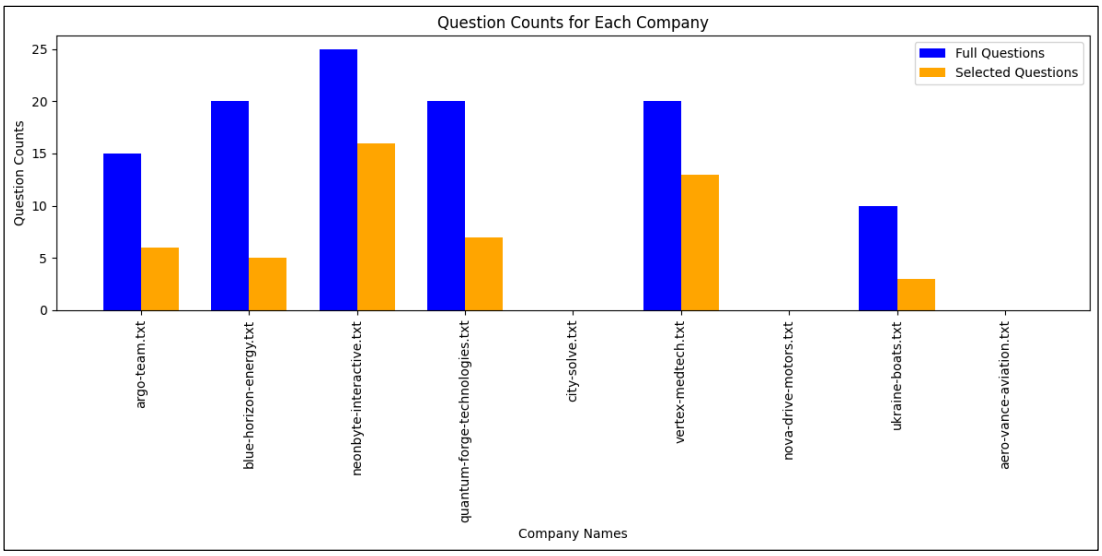


Рисунок 3.6 – Детальна інформація по датасету питань та відповідей (створений самостійно)

3.2 Моделі

По результатам теоретичного дослідження було обрано 2 моделі для експериментів: «Gemini-1.5-pro» та «meta-llama/Llama-3.2-11B-Vision-Instruct». Початково планувалось використання орендованого серверу для використання моделі «Llama», але для економії коштів та спрощення інфраструктури було обрано АПІ від провайдера Together AI [34]. Через цей АПІ було використано модель «meta-llama/Llama-3.2-11B-Vision-Instruct-Turbo» (додатково оптимізована початкова модель).

Ціна за використання моделі «Gemini-1.5-pro»: $1.25 за 1M вхідних токенів та $5.00 за 1M вихідних токенів. Під час експериментів жодні ліміти не заважали роботі з даним провайдером моделей.

У той же час ціна використання моделі «Llama-3.2-11B-Vision-Instruct-Turbo»: $0.18 за 1M токенів (однаково для вхідних та вихідних токенів). Під час експериментів потрібно було додатково враховувати обмеження на кількість

запитів у хвилину – 60. Ці ліміти поступово збільшуються – чим більше витрачаєте коштів тим більші ліміти. Але, навіть максимальні ліміти не вдаються до ніяких порівнянь з лідерами ринку OpenAI та Google.

Для оцінки результатів моделей було використано моделі від OpenAI. Це зроблено для зменшення зсуву у випадках коли ми використовуємо одні й ті самі моделі для передбачень та оцінок. Було використано «gpt-4o-mini» як компроміс розумної та дешевої моделі. Ціна за використання «gpt-4o-mini»: $0.15 за 1M вхідних токенів та $0.60 за 1M вихідних токенів. Що в середньому трохи більше за використання «Llama». Під час оцінки результатів було виявлено ліміти на першому рівні користувача OpenAI, які забороняли більше 10000 запитів на один день. Через це розрахунок метрик було розділено на кілька днів.

Для використання моделей «Gemini» та «Llama» було використано фреймворк LlamaIndex [35] (див. рис. 3.7-3.8).

Рисунок 3.7 – Обгортка для моделі «Gemini-1.5-pro» (створений самостійно)

Рисунок 3.8 – Обгортка для моделі «Llama» (створений самостійно)

Моделі OpenAI використовувались всередині бібліотеки DeepEval.

На основі двох базових моделей було створено перелік компонент для перевірки гіпотези щодо впливу тої чи іншої компоненти RAG підходу (див. рис. 3.9).

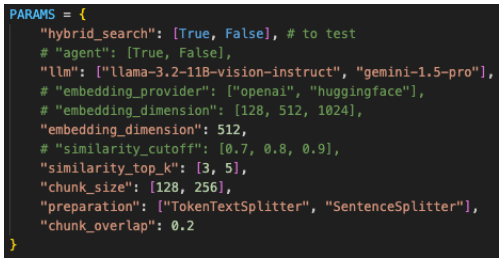


Рисунок 3.9 – Перелік компонент (створений самостійно)

Для кожного експерименту створювались нові екземпляри функцій для запису та вилучення даних з БД (див. рис. 3.10-3.11).

```python
# initialize a file reader
reader = SimpleDirectoryReader(input_files=doc_paths)

# load documents into LlamaIndex Documents
documents = reader.load_data()

vector_store = Neo4jVectorStore(
    username=cfg['config']['env']['NEO4J_USER'],
    password=cfg['config']['env']['NEO4J_PASSWORD'],
    url=cfg['config']['env']['NEO4J_URI'],
    embedding_dimension=cfg['config']['rag']['setup']['embedding_dimension'],
    distance_strategy=cfg['config']['rag']['setup']['distance_strategy'],
    index_name=cfg['config']['rag']['setup']['index_name'],
    text_node_property=cfg['config']['rag']['setup']['text_node_property'],
    hybrid_search=cfg['config']['rag']['setup']['hybrid_search']
)

# setup context
storage_context = StorageContext.from_defaults(
    vector_store=vector_store
)

if preparation == "TokenTextSplitter":
    parser = TokenTextSplitter(
        chunk_size=cfg['config']['rag']['setup']['chunk_size'],
        chunk_overlap=int(np.ceil(cfg['config']['rag']['setup']['chunk_overlap'] * cfg['config']['rag']['setup']['chunk_size'])),
        separator=cfg['config']['rag']['setup']['separator'],
        id_func=id_func
    )
elif preparation == "SentenceSplitter":
    parser = SentenceSplitter(
        chunk_size=cfg['config']['rag']['setup']['chunk_size'],
        chunk_overlap=int(np.ceil(cfg['config']['rag']['setup']['chunk_overlap'] * cfg['config']['rag']['setup']['chunk_size'])),
        separator=cfg['config']['rag']['setup']['separator'],
        id_func=id_func
    )

# parse documents into nodes (chunks)
nodes = parser.get_nodes_from_documents(documents)

db_index = VectorStoreIndex(nodes, storage_context=storage_context, embed_model=embed_model, show_progress=False)
```

Рисунок 3.10 – Запис нових даних в БД (створений самостійно)

```python
###### configure different tools
# custom retriever
retriever = VectorIndexRetriever(
    index=db_index,
    similarity_top_k=cfg['config']['rag']['setup']['similarity_top_k'],
    vector_store_query_mode=cfg['config']['rag']['setup']['vector_store_query_mode']
)

# custom response synthesizer
response_synthesizer = get_response_synthesizer(
    response_mode=cfg['config']['rag']['setup']['response_mode']
)

# combine custom query engine
query_engine = RetrieverQueryEngine(
    retriever=retriever,
    response_synthesizer=response_synthesizer
)

###### configure setup
setup = query_engine
```

Рисунок 3.11 – Вилучення нових даних з БД (створений самостійно)

3.3 Метрики

У якості основних метрик для розрахунку було використано актуальність відповіді, обґрунтованість, та доречність контексту. Для розрахунку кожної з цих метрик було використано велику мовну модель («gpt-4o-mini»). Також, окрім розрахунку метрик, було отримано роздуми чому саме модель вважає що оцінка має бути такою. По порядку важливості – основною метрикою є актуальність відповіді, після йде ґрунтовність та доречність контексту.

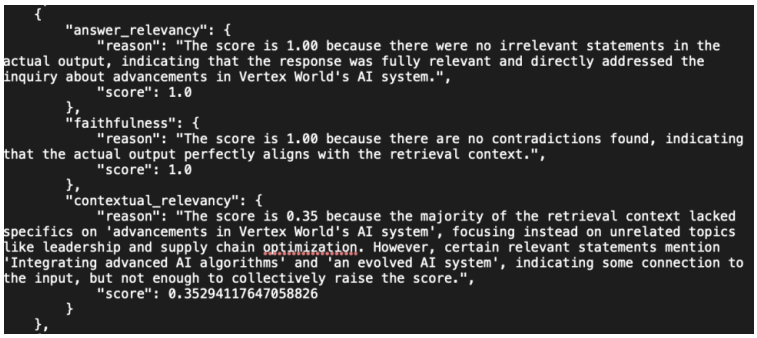
DeepEval фреймворк автоматично пробує отримати оцінку знову у разі невдачі. Це є дуже важливим аспектом для бібліотеки розрахунку метрик, бо час від часу можна отримувати галюцинації від великих лінгвістичних моделей.

За допомогою фреймворку DeepEval було створено LLMTestCase для кожного питання з датасету. Після чого всі тестові кейси було виконано в паралель з використанням 50 потоків (див. рис. 3.12).

```
###### evaluate
if EVALUATE:
    test_cases = []
    for j, item in enumerate(tqdm(dataset[:DATASET_SIZE])):
        # define test case
        test_case = LLMTestCase(
            input=item.get('question'),
            actual_output=predictions[j]['response'],
            expected_output=item.get('answer'),
            retrieval_context=predictions[j]['source_nodes']
        )
        test_cases.append(test_case)

    answer_relevancy_metric = AnswerRelevancyMetric(model=cfg['config']['rag']['models']['llm_openai'])
    faithfulness_metric = FaithfulnessMetric(model=cfg['config']['rag']['models']['llm_openai'])
    contextual_relevancy_metric = ContextualRelevancyMetric(model=cfg['config']['rag']['models']['llm_openai'])

    # time.sleep(60)

    results = evaluate(
        test_cases=test_cases,
        metrics=[answer_relevancy_metric, faithfulness_metric, contextual_relevancy_metric],
        max_concurrent=50,
    )
```

Рисунок 3.12 – Код розрахунку метрик (створений самостійно)

На рисунку 3.13 наведено приклад оцінки роботи моделі на одному з питань датасету.



```
    {
        "answer_relevancy": {
            "reason": "The score is 1.00 because there were no irrelevant statements in the
actual output, indicating that the response was fully relevant and directly addressed the
inquiry about advancements in Vertex World's AI system.",
            "score": 1.0
        },
        "faithfulness": {
            "reason": "The score is 1.00 because there are no contradictions found, indicating
that the actual output perfectly aligns with the retrieval context.",
            "score": 1.0
        },
        "contextual_relevancy": {
            "reason": "The score is 0.35 because the majority of the retrieval context lacked
specifics on 'advancements in Vertex World's AI system', focusing instead on unrelated topics
like leadership and supply chain optimization. However, certain relevant statements mention
'Integrating advanced AI algorithms' and 'an evolved AI system', indicating some connection to
the input, but not enough to collectively raise the score.",
            "score": 0.35294117647058826
        }
    },
```

Рисунок 3.13 – Результат оцінки моделі (створений самостійно)

3.4 Результати

Під час практичного дослідження було проведено 32 експерименти з метою знаходження найважливіших аспектів при побудові RAG системи (див. табл. 3.1). Загальна сума всіх експериментів ~10$.

Таблиця 3.1 – Результати експериментів

Модель	Модифікація	Актуальність відповіді [0-1]	Обґрун-тованість [0-1]	Доречність контексту [0-1]
gemini-1.5-pro	HS/TTS/top5/CS256	**0.944**	0.925	0.130
gemini-1.5-pro	HS/TTS/top3/CS256	0.942	0.873	0.171
gemini-1.5-pro	HS/SS/top5/CS256	0.917	0.860	0.132
llama-3.2-11B	HS/SS/top3/CS256	0.911	0.910	0.198
llama-3.2-11B	HS/TTS/top5/CS256	0.903	0.914	0.140
llama-3.2-11B	HS/SS/top5/CS256	0.897	**0.942**	0.140
llama-3.2-11B	SS/top5/CS256	0.892	0.864	0.132
llama-3.2-11B	HS/TTS/top5/CS256	0.890	0.785	0.165
gemini-1.5-pro	HS/SS/top3/CS128	0.885	0.924	0.215
gemini-1.5-pro	SS/top5/CS256	0.883	0.862	0.134
gemini-1.5-pro	TTS/top5/CS256	0.879	0.840	0.121
gemini-1.5-pro	HS/TTS/top5/CS128	0.879	0.848	0.157
llama-3.2-11B	HS/TTS/top3/CS128	0.872	0.857	0.216
gemini-1.5-pro	HS/SS/top3/CS256	0.861	0.877	0.189
llama-3.2-11B	SS/top3/CS256	0.861	0.840	0.170
gemini-1.5-pro	HS/SS/top5/CS128	0.858	0.880	0.153
llama-3.2-11B	HS/TTS/top3/CS256	0.850	0.857	0.162
llama-3.2-11B	HS/SS/top3/CS128	0.837	0.880	**0.222**
llama-3.2-11B	HS/SS/top5/CS128	0.833	0.895	0.151
llama-3.2-11B	TTS/top5/CS256	0.823	0.866	0.125
gemini-1.5-pro	HS/TTS/top3/CS128	0.823	0.826	0.203
gemini-1.5-pro	TTS/top3/CS256	0.810	0.855	0.158
llama-3.2-11B	TTS/top5/CS128	0.807	0.891	0.164
llama-3.2-11B	SS/top5/CS128	0.806	0.878	0.151
gemini-1.5-pro	SS/top3/CS256	0.795	0.911	0.164
gemini-1.5-pro	SS/top3/CS128	0.768	0.852	0.204

Кінець таблиці 3.1

Модель	Модифікація	Актуальність відповіді [0-1]	Обґрун-тованість [0-1]	Доречність контексту [0-1]
gemini-1.5-pro	TTS/top5/CS128	0.764	0.810	0.160
llama-3.2-11B	TTS/top3/CS256	0.761	0.938	0.154
gemini-1.5-pro	SS/top5/CS128	0.759	0.867	0.145
gemini-1.5-pro	TTS/top3/CS128	0.753	0.859	0.213
llama-3.2-11B	TTS/top3/CS128	0.742	0.867	0.195
llama-3.2-11B	SS/top3/CS128	0.727	0.910	0.191

Відповідно до результатів вище можна зробити наступні висновки:

- ми маємо різницю між найкращою та найгіршою конфігурацією у 22% актуальності відповідей. Це дуже багато з урахуванням того, що експериментів було всього 32. Це підтверджує те, що ми обрали важливі параметри для оптимізації RAG системи;
- лише використання одного гібридного пошуку покращує актуальність відповідей на 8%. Додатковий пошук по ключовим словам надає можливість знаходити речі, які простим семантичним пошуком неможливо знайти;
- використання 3 релевантних чанків у порівнянні з 5 дає приріст у ~5% доречності контексту. Це через те що питання в нашому датасеті досить нескладні та не потребують багато контексту. Збільшення релевантних чанків лише додавало зайвої інформації;
- використання розміру чанків у 256 токенів дає приріст по всім метрикам від 3 до 6%. Це скоріш за все результат того, що більші по розміру чанки повністю захоплювали необхідний шматок інформації;
- від зміни моделі та методу розбиття даних на чанки метрики суттєво не змінюються. Про модель, скоріш за все це результат відносно простого датасету запитань. Розумність моделі не допомагала в цьому випадку, та як і в більшості RAG систем. Щодо методу розбиття на чанки –

результатом є структура нашого датасету. Він є структурованим Markdown форматом. Скоріш за все, якби ми був простий текст на вхід, то SS працював би краще;

– доречність контексту завжди є відносно малою. Якщо проаналізувати результати моделі судді, то виявиться, що всі невеликі оцінки ставились через присутність зайвої інформації в отриманих даних. Це все через те, що в нашому випадку дані є дуже скупченими, тобто ми отримуємо багато зайвої інформації також. В одному чанку (хоч він і досить малий) може бути інформація відразу про кілька різних речей.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було проаналізовано предметну галузь, розглянуто прогрес індустрії штучного інтелекту за останні кілька років, досліджено актуальні проблеми та тенденції галузі. Зокрема, можна виділити наявність все більшої кількості таблиць лідерів великих лінгвістичних моделей по різним критеріям оцінки, які допомагають кожному вибрати найкращу модель в залежності від даних та задачі. Окрему увагу було приділено важливості проведення цього дослідження в рамках сучасної індустрії штучного інтелекту.

Для більш глибокого розуміння тенденцій та розвитку індустрії було детально розглянуто кілька основних публікацій, які стосуються розвитку великих лінгвістичних моделей та технологій штучного інтелекту. З ряду досліджень було виокремлено кілька методів, які було використано під час програмної реалізації дослідження. Також, було приділено увагу розмірам тренувальних датасетів, проблемі з нестачею даних та можливим досягненням «плато масштабування». Що, у свою чергу, має мотивувати дослідників по всьому світу на пошук більш оптимальних та ефективних архітектур нейронних мереж. Окремо було приділено увагу вмінню великих мовних моделей обробляти великі контексти нових знань.

Після зваженого аналізу було детально сформульовано задачі для виконання дослідження, такі як вибір моделі, генерація датасетів, та використання переліку модифікацій RAG систем для перевірки на практиці. Задачу вибору моделі було розглянуто зі сторони багатокритеріальної задачі оптимізації, для розв’язку якої було використано лінійну адитивну згортку по нормалізованим значенням критеріїв. З розглянутих 10 великих мовних моделей, для подальшого дослідження було обрано лише дві: «Gemini Pro 1.5» та «Llama 3.2 11B Vision Instruct». За зваженими показниками якості вони були явними лідерами. Також, було обрано DeepEval у якості фреймворку для оцінки моделей, та Neo4j у якості сховища бази знань.

У розрізі теоретичного дослідження було детально розглянуто конфігурації ін’єкції знань з використанням великих лінгвістичних моделей, приклади

використання мови Cypher для побудови запитів до бази знань Neo4j з використанням зручної надбудови Aura, та деталі отримання доступів до вищезазначених великих лінгвістичних моделей. Наостанок було розглянуто метрики, які буде використано при розрахунку якості рішень, та переваги використання фреймворку DeepEval.

Під час практичного дослідження було згенеровано штучний датасет з детальними описами 9 компаній, загальним розміром 21059 токенів. Для цього датасету було згенеровано 110 пар запитань-відповідей. Після цього було реалізовано 32 експерименти для оцінки ефективності тої чи іншої конфігурації RAG системи на 50 запитаннях-відповідях. Маючи результати передбачень усіх моделей було розраховано метрики з використанням моделі «gpt-4o-mini». Далі було детально проаналізовано отримані результати та візуалізовано статистики по експериментам.

У даній роботі було окреслено та проаналізовано існуючі моделі, методи, обмеження та ресурси необхідні для використання методів ін’єкції знань на практиці. Також було досліджено ефективність налаштування конкретних параметрів у реальних RAG системах.

Як висновок з експериментів, найбільш ефективними модифікаціями є ті, які використовують гібридний пошук. Щодо таких гіперпараметрів як кількість отримуваних чанків, так само як і розмір чанку, то вони є індивідуальними в залежності від ваших даних. Отримані конфігурації можуть бути адаптовані для використання в будь-якій сфері. Повний код експериментів доступний на GitHub [36].

Для більш повної картини можна продовжити дослідження з використанням претреновування у якості альтернативного методу ін’єкції знань.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Erdem, E., Kuyu, M., Yagcioglu, S., Frank, A., Parcalabescu, L., Plank, B., Babii, A., Turuta, O., Erdem, A., Calixto, I., Lloret, E., Apostol, E.-S., Truică, C.-O., Šandrih, B., Martinčić-Ipšić, S., Berend, G., Gatt, A., & Korvel, G. (2022). Neural natural language generation: A survey on multilinguality, multimodality, controllability and learning. Journal of Artificial Intelligence Research, 73, 1131-1207.

2. Error - Substack. URL: https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https://substack-post-media.s3.amazonaws.com/public/images/12cdf506-6cd8-4afa-93a3-b77b82770309_2755x1570.png (дата звернення: 17.04.2025).

3. Agarwal S. Tokenization in Large Language Models (LLMs). Medium. URL: https://medium.com/@shashankag14/tokenization-in-large-language-models-llms-0ba0aea6b1d6 (дата звернення: 19.04.2025).

4. What languages do the models support? | Novelcrafter Help Center. Novelcrafter Help Center. URL: https://docs.novelcrafter.com/en/articles/9324140-what-languages-do-the-models-support (дата звернення: 19.04.2025).

5. Chatbot Arena LLM Leaderboard: Community-driven Evaluation for Best LLM and AI chatbots. URL: https://lmarena.ai/ (дата звернення: 20.04.2025).

6. Open LLM Leaderboard - a Hugging Face Space by open-llm-leaderboard. Hugging Face – The AI community building the future. URL: https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard#/ (дата звернення: 20.04.2025).

7. LLM Leaderboard - Compare GPT-4o, Llama 3, Mistral, Gemini & other models | Artificial Analysis. AI Model & API Providers Analysis | Artificial Analysis. URL: https://artificialanalysis.ai/leaderboards/models (дата звернення: 20.04.2025).

8. Grow Right. What is Retrieval Augmented Generation?. LinkedIn: Log In or Sign Up. URL: https://www.linkedin.com/pulse/what-retrieval-augmented-generation-grow-right/ (дата звернення: 23.04.2025).

9. GPT-4 Parameters Explained. URL: https://hix.ai/hub/chatgpt/gpt-4-parameters (дата звернення: 23.04.2025).

10. Howarth J. Number of Parameters in GPT-4 (Latest Data). Exploding Topics. URL: https://explodingtopics.com/blog/gpt-parameters (дата звернення: 25.04.2025).

11. OpenAI debuts GPT-4o 'omni' model now powering ChatGPT | TechCrunch. TechCrunch. URL: https://techcrunch.com/2024/05/13/openais-newest-model-is-gpt-4o/ (дата звернення: 25.04.2025).

12. Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. 2022. Training compute-optimal large language models. In Proceedings of the 36th International Conference on Neural Information Processing Systems (NIPS '22). Curran Associates Inc., Red Hook, NY, USA, Article 2176, 30016–30030. DOI: 10.5555/3600270.3602446.

13. The Times Sues OpenAI and Microsoft Over A.I. Use of Copyrighted Work. The New York Times - Breaking News, US News, World News and Videos. URL: https://www.nytimes.com/2023/12/27/business/media/new-york-times-open-ai-microsoft-lawsuit.html?smid=url-share (дата звернення: 27.04.2025).

14. Ataccama. The Importance of Data Quality for AI. Ataccama. URL: https://www.ataccama.com/blog/the-importance-of-data-quality-for-AI (дата звернення: 27.04.2025).

15. Bradshaw T. Nvidia and the AI boom face a scaling problem. Financial Times. URL: https://www.ft.com/content/f24ba8d5-4c33-47ef-a91e-8f76340b08c4 (дата звернення: 27.04.2025).

16. Zhihao Fan, Yeyun Gong, Zhongyu Wei, Siyuan Wang, Yameng Huang, Jian Jiao, Xuanjing Huang, Nan Duan, and Ruofei Zhang. 2020. An Enhanced Knowledge Injection Model for Commonsense Generation. In Proceedings of the 28th International Conference on Computational Linguistics, pages 2014–2025, Barcelona, Spain (Online). International Committee on Computational Linguistics.

17. Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim

Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS '20). Curran Associates Inc., Red Hook, NY, USA, Article 793, 9459–9474.

18. Martino, Ariana, Michael Iannelli, and Coleen Truong. "Knowledge injection to counter large language model (LLM) hallucination." In European Semantic Web Conference, pp. 182-185. Cham: Springer Nature Switzerland, 2023.

19. Introducing Contextual Retrieval. Home \ Anthropic. URL: https://www.anthropic.com/news/contextual-retrieval (дата звернення: 28.04.2025).

20. Maksymenko, D., Kryvoshein, D., Turuta, O., Kazakov, D., & Turuta, O. (2024). Benchmarking Conversation Routing in Chatbot Systems Based on Large Language Models. Proceedings http://ceur-ws. org ISSN, 1613, 0073.

21. Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... & Chen, W. (2022). Lora: Low-rank adaptation of large language models. ICLR, 1(2), 3.

22. Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). Qlora: Efficient finetuning of quantized llms. Advances in neural information processing systems, 36, 10088-10115.

23. Mecklenburg, N., Lin, Y., Li, X., Holstein, D., Nunes, L., Malvar, S., ... & Hendry, T. (2024). Injecting new knowledge into large language models via supervised fine-tuning. arXiv preprint arXiv:2404.00213.

24. Li, T., Zhang, G., Do, Q. D., Yue, X., & Chen, W. (2024). Long-context llms struggle with long in-context learning. arXiv preprint arXiv:2404.02060.

25. Optimizing LLM Accuracy. URL: https://platform.openai. com/docs/guides/optimizing-llm-accuracy (дата звернення: 30.04.2025).

26. OpenAI Playground. URL: https://platform.openai.com/playground/ chat?models=gpt-4o (дата звернення: 05.05.2025).

27. Creating an instance. URL: https://neo4j.com/docs/aura/classic/auradb/getting-started/create-database/ (дата звернення: 07.05.2025).

28. Neo4j Cypher Query Language. URL: https://neo4j.com/product/cypher-graph-query-language/ (дата звернення: 07.05.2025).

29. Explore overview. URL: https://neo4j.com/docs/aura/explore/explore-visual-tour/explore-overview/ (дата звернення: 08.05.2025).

30. Google Cloud console. Google Cloud Platform. URL: https://console.cloud.google.com/vertex-ai/studio/freeform?inv=1&invt=AbkjZg&project=speechtotext-352308 (дата звернення: 08.05.2025).

31. meta-llama/Llama-3.2-11B-Vision-Instruct · Hugging Face. Hugging Face – The AI community building the future. URL: https://huggingface.co/meta-llama/Llama-3.2-11B-Vision-Instruct (дата звернення: 08.05.2025).

32. Llama 3.2 | Model Cards and Prompt formats. Llama. URL: https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_2/ (дата звернення: 09.05.2025).

33. DeepEval - The Open-Source LLM Evaluation Framework. DeepEval - The Open-Source LLM Evaluation Framework. URL: https://docs.confident-ai.com/ (дата звернення: 09.05.2025).

34. All Models. Models hosted by Together AI available for serverless or for on demand dedicated endpoints. URL: https://api.together.ai/models (дата звернення: 10.05.2025).

35. Google GenAI - LlamaIndex. LlamaIndex - LlamaIndex. URL: https://docs.llamaindex.ai/en/stable/examples/llm/google_genai/ (дата звернення: 11.05.2025).

36. GitHub repository, Major Diploma. URL: https://github.com/Vlad-Fliahin/major-diploma (дата звернення: 29.05.2025).

37. Maksymenko, Daniil, Danylo Kryvoshein, Olena Turuta, Dmytro Kazakov, and Oleksii Turuta. "Benchmarking Conversation Routing in Chatbot Systems Based on Large Language Models." Proceedings http://ceur-ws. org ISSN 1613 (2024): 0073.

38. Maksymenko, Daniil, and Oleksii Turuta. "Interpretable Conversation Routing via the Latent Embeddings Approach." Computation 12, no. 12 (2024): 237.

39. Saichyshyna, Nataliia, Daniil Maksymenko, Oleksii Turuta, Andriy Yerokhin, Andrii Babii, and Olena Turuta. "Extension Multi30K: Multimodal dataset for integrated vision and language research in Ukrainian." In Proceedings of the Second Ukrainian Natural Language Processing Workshop (UNLP), pp. 54-61. 2023.

40. Erdem, Erkut, Menekse Kuyu, Semih Yagcioglu, Anette Frank, Letitia Parcalabescu, Barbara Plank, Andrii Babii et al. "Neural natural language generation: A survey on multilinguality, multimodality, controllability and learning." Journal of Artificial Intelligence Research 73 (2022): 1131-1207.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ ЗА НАУКОВИМИ НАПРЯМАМИ КЕРІВНИКА ТА НАУКОВЦІВ КАФЕДРИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ

1. Erdem, E., Kuyu, M., Yagcioglu, S., Frank, A., Parcalabescu, L., Plank, B., Babii, A., Turuta, O., Erdem, A., Calixto, I., Lloret, E., Apostol, E.-S., Truică, C.-O., Šandrih, B., Martinčić-Ipšić, S., Berend, G., Gatt, A., & Korvel, G. (2022). Neural natural language generation: A survey on multilinguality, multimodality, controllability and learning. Journal of Artificial Intelligence Research, 73, 1131-1207.

20. Maksymenko, D., Kryvoshein, D., Turuta, O., Kazakov, D., & Turuta, O. (2024). Benchmarking Conversation Routing in Chatbot Systems Based on Large Language Models. Proceedings http://ceur-ws. org ISSN, 1613, 0073.

37. Maksymenko, Daniil, Danylo Kryvoshein, Olena Turuta, Dmytro Kazakov, and Oleksii Turuta. "Benchmarking Conversation Routing in Chatbot Systems Based on Large Language Models." Proceedings http://ceur-ws. org ISSN 1613 (2024): 0073.

38. Maksymenko, Daniil, and Oleksii Turuta. "Interpretable Conversation Routing via the Latent Embeddings Approach." Computation 12, no. 12 (2024): 237.

39. Saichyshyna, Nataliia, Daniil Maksymenko, Oleksii Turuta, Andriy Yerokhin, Andrii Babii, and Olena Turuta. "Extension Multi30K: Multimodal dataset for integrated vision and language research in Ukrainian." In Proceedings of the Second Ukrainian Natural Language Processing Workshop (UNLP), pp. 54-61. 2023.

40. Erdem, Erkut, Menekse Kuyu, Semih Yagcioglu, Anette Frank, Letitia Parcalabescu, Barbara Plank, Andrii Babii et al. "Neural natural language generation: A survey on multilinguality, multimodality, controllability and learning." Journal of Artificial Intelligence Research 73 (2022): 1131-1207.