УДК 004.8:004.67
DOI https://doi.org/10.32689/maup.it.2025.4.20

Руслан МОРОЗОВ
здобувач вищої освіти за спеціальністю «Комп'ютерні науки»,
Харківський національний університет імені В. Н. Каразіна,
morozov2020ks11@student.karazin.ua
***ORCID:** 0009-0007-7793-1224*

Володимир ДОНЕЦЬ
доктор філософії, Харківський національний університет імені В. Н. Каразіна,
v.donets@karazin.ua
***ORCID:** 0000-0002-5963-9998*

РОЗРОБКА МЕТОДУ RAG ДЛЯ НАДАННЯ ІНФОРМАЦІЙНОЇ ПІДТРИМКИ КОРИСТУВАЧАМ

Анотація. *У статті розглядається проблема надання точної та релевантної інформаційної підтримки користувачам у роботі з великими обсягами текстових даних. Звичайні пошукові системи та окремо використані мовні моделі не завжди здатні правильно інтерпретувати зміст запитів і можуть створювати неточні відповіді або галюцинації. Для вирішення цієї проблеми запропоновано гібридний підхід – Retrieval-Augmented Generation (RAG), який поєднує точність інформаційного пошуку з генеративними можливостями LLM.*

Мета роботи. *Розробка та експериментальне тестування повного методу RAG, призначеного для автоматизованої інформаційної підтримки. Ключовою метою є забезпечення економії та високої ресурсної ефективності, що робить метод практично придатним для невеликих організацій, університетських факультетів або інших застосувань з обмеженими обчислювальними бюджетами.*

Методологія. *Запропоновано модульну архітектуру системи, реалізовану з використанням відкритих бібліотек: LangChain для узгодження процесів, ChromaDB як локальне векторне сховище, HuggingFace для доступу до моделей вбудовування та для швидкого та економного виконання запитів LLM використано Groq API. Проведено кілька етапів перевірки: тестування точності пошуку (Top-k) для різних моделей вбудовування на українськомовному наборі даних, тестування якості генерації від початку до кінця з використанням LLM-оцінювача; оптимізація параметрів (підказок, способи розбиття тексту) за допомогою фреймворку Ragas.*

Наукова новизна. *Проведено систематичне порівняння ефективності моделей вбудовування для семантичного пошуку на українськомовному масиві текстів. Експериментально ідентифіковано оптимальний баланс між вартістю та якістю генеративних моделей (LLM), доступних через API. Запропоновано та валідовано економна система RAG, оптимізований за допомогою метрик Ragas для досягнення високої точності відповідей при мінімальних витратах.*

Висновки. *Дослідження підтвердило життєздатність розробленого методу. Модель вбудовування intfloat/multilingual-e5-large-instruct продемонструвала найкращу точність пошуку, досягнувши 100% у Top-7. Генеративна модель meta-llama/llama-4-scout-17b-16e-instruct показала оптимальне співвідношення ціни та якості (88,3% коректних відповідей). Після налаштування підказок і параметрів фрагментації точність відповідей стала найвищою серед усіх випробуваних варіантів. Перспективи подальшої роботи включають тестування спеціалізованих українських моделей та впровадження системи в реальні чат-боти.*

Ключові слова: *Retrieval-Augmented Generation (RAG), великі мовні моделі (LLM), інформаційний пошук, економний, Ragas, моделі вбудовування.*

Ruslan MOROZOV, Volodymyr DONETS. DEVELOPMENT OF THE RAG METHOD FOR PROVIDING INFORMATION SUPPORT TO USERS

Abstract. *The article discusses the problem of providing accurate and relevant information support to users when working with large amounts of text data. Traditional search engines and isolated large language models (LLMs) have significant limitations, such as insufficient semantic understanding and hallucination generation. To solve this problem, a hybrid approach is proposed – Retrieval-Augmented Generation (RAG), which combines the accuracy of information retrieval with the generative capabilities of LLM.*

Purpose of the work. *Development and experimental testing of a complete RAG method designed for automated information support. The key goal is to ensure economical and high resource efficiency, making the method practical for small organizations, university departments, or startups with limited computing budgets.*

Methodology. *A modular system architecture is proposed, implemented using open libraries: LangChain for process coordination, ChromaDB as a local vector storage, HuggingFace for access to embedding models, and for high-speed and cost-effective LLM queries used Groq API. A multi-stage evaluation was conducted: testing search accuracy (Top-k) for different embedding models on a Ukrainian-language dataset, testing end-to-end generation quality using an LLM evaluator, and optimizing parameters (prompt templates, fragmentation strategies) using the Ragas framework.*

Scientific novelty. *A systematic comparison of the effectiveness of embedding models for semantic search in a Ukrainian-language text corpus was conducted. The optimal balance between the cost and quality of generative models (LLM) available through API was experimentally identified. A cost-effective RAG pipeline, optimized using Ragas metrics to achieve high response accuracy at minimal cost, was proposed and validated.*

Conclusions. *The study confirmed the viability of the developed method. The intfloat/multilingual-e5-large-instruct embedding model demonstrated the best search accuracy, reaching 100% in the Top-7. The meta-llama/llama-4-scout-17b-16e-instruct generative model showed the optimal price-quality ratio (88.3% correct answers). Optimization of prompts and fragmentation strategy (size 1000, overlap 150) allowed us to achieve the highest accuracy rates. Prospects for further work include testing specialized Ukrainian models and implementing the system in real chatbots.*

Key words: *Retrieval-Augmented Generation (RAG), large language models (LLM), information retrieval, economical, Ragas, embedding models.*

Вступ. У сучасному інформаційному просторі користувачі часто стикаються з серйозною проблемою пошуку релевантної та точної інформації серед величезних обсягів цифрових текстових даних. Ця проблема є особливо гострою в спеціалізованих галузях, таких як академічна та наукова діяльність, де величезний обсяг доступної інформації, від наукової літератури до обширних навчальних матеріалів, робить пошук конкретних відповідей надзвичайно складним і трудомістким завданням. Завдання ручного перегляду величезних сховищ для пошуку та синтезу інформації є серйозною перешкодою для ефективності, як для дослідника, який намагається бути в курсі останніх розробок, так і для студента, який шукає роз'яснення щодо змісту курсу [5; 6].

Ця проблема ускладняється обмеженнями існуючих технологій пошуку інформації. Традиційні пошукові системи, які часто покладаються на збіг ключових слів та системи на зразок базових FAQ часто виявляються недостатніми. Цим методам зазвичай бракує складного розуміння природної мови, необхідного для сприйняття конкретного контексту запиту користувача. Як правило вони повертають список документів або заздалегідь визначених відповідей, які можуть бути лише частково релевантними, перекладаючи когнітивне навантаження синтезу повної відповіді назад на користувача. Вони не надають прямої, контекстуальної та синтезованої інформаційної підтримки.

Одночасно з цим поява великих мовних моделей (LLM) запровадила потужні генеративні можливості, які дозволяють створювати плавний, схожий на людський текст у формі діалогу. Однак LLM, коли їх використовують ізольовано, мають критичні недоліки. Їхні знання є статичними та обмеженими даними, на яких вони були навчені, що означає, що вони не можуть отримати доступ до інформації в режимі реального часу, власницької або специфічної для певної галузі, яка не була частиною їхнього навчального корпусу. Це призводить до значної і добре задокументованої проблеми, відомої як «галюцинація», коли модель генерує правдоподібну, але фактично невірну або повністю вигадану інформацію. У наукових та освітніх контекстах, де точність і надійність є обов'язковими, такі галюцинації роблять автономні LLM непридатними для надійної інформаційної підтримки [6].

Ця розбіжність між обмеженнями традиційного пошуку та обмеженнями надійності автономних LLM підкреслює важливе наукове та практичне завдання: розробку системи, яка може надавати точні, контекстно-орієнтовані та надійні відповіді з конкретних, великомасштабних баз знань. Генерація з розширеним пошуком (RAG) стала важливим методом вирішення саме цієї проблеми. Практичне значення RAG полягає в її гібридній архітектурі, яка ефективно поєднує точність пошуку інформації з генеративною потужністю LLM [5].

Механізм RAG працює так що спочатку з певної зовнішньої бази знань (наприклад, бази даних наукових статей або навчальних матеріалів університету) вибирається набір відповідних документів або фрагментів тексту. Потім ця інформація передається до LLM як контекст, який модель використовує для генерації відповіді. Цей процес «ґрунтує» відповідь LLM на фактичних, перевірених даних, тим самим значно підвищуючи точність і надійність результатів та зменшуючи ризик галюцинацій. Тому розробка методів RAG є важливим науковим завданням, оскільки її метою є створення більш інтелектуальних систем інформаційної підтримки, які можуть надійно обслуговувати користувачів у таких важливих сферах, як освіта та наукові дослідження [5; 6].

Аналіз останніх досліджень і публікацій. Розвиток технології Retrieval-Augmented Generation (RAG) був обумовлений необхідністю вирішити основні обмеження великих мовних моделей (LLM), а саме їх статичні, не оновлювані знання та схильність до галюцинацій. У науковій літературі RAG представлено не як єдину модель, а як багатоступеневу архітектуру, що інтегрує два ключові компоненти: пошукову систему інформації та генератор [4; 7]. Така конструкція дозволяє системі отримувати доступ до зовнішніх, перевірених знань і базуючись на них давати свої відповіді.

ISSN 2786-5460 (Print), ISSN 2786-5479 (Online)

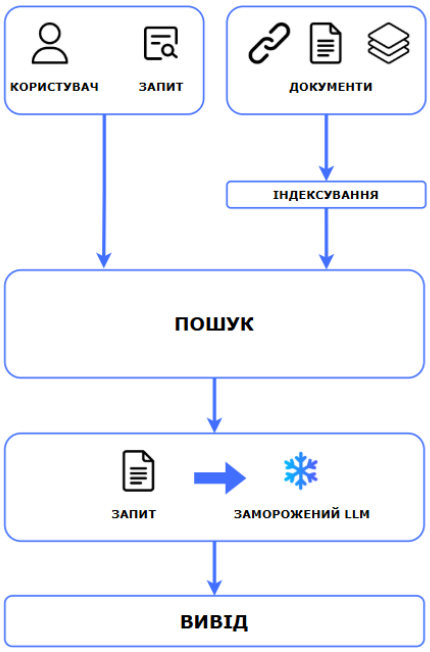

Рис. 1. Діаграма, що ілюструє типовий робочий процес RAG [4]

Компонент пошуку є центральним для успіху системи, оскільки він відповідає за пошук відповідного контексту з великого масиву. Нинішні методи еволюціонували від традиційних «розріджених» методів пошуку, таких як BM25, які в основному покладаються на збіг ключових слів. Хоча ці методи є ефективними, їм важко вловити семантичне значення запиту. Це обмеження призвело до розробки «щільних» моделей пошуку. Яскравим прикладом є Dense Passage Retrieval (DPR), який використовує архітектуру подвійного кодувальника на основі моделі BERT для відображення запитів і документів у спільному високорозмірному векторному просторі. Це дозволяє здійснювати семантичний пошук, при якому система виявляє фрагменти на основі концептуальної схожості, а не лише спільних слів. Ці векторні представлення управляються та запитуються за допомогою ефективних індексів, для чого використовуються такі інструменти, як Faiss, а сучасні векторні бази даних, такі як ChromaDB або Qdrant, служать подібній меті [8; 10].

Компонент генератора зазвичай є попередньо навченим LLM на основі Трансформера, таким як моделі з сімейств BERT або GPT. Ці моделі, багато з яких доступні через такі платформи, як HuggingFace, відповідають за синтез послідовної відповіді природною мовою на основі контексту, наданого пошуковою системою. У статті REALM, одній з перших фундаментальних робіт, було продемонстровано, як попереднє навчання як пошуку, так і генератора від початку до кінця дозволяє моделі навчитися, які знання потрібно отримувати і як ефективно їх використовувати [4; 7]. З того часу з'явилися такі фреймворки, як LangChain, які спрощують координацію цих складних систем, з'єднуючи компонент пошуку, векторне сховище та генератор.

Ці підходи успішно вирішили значні проблеми, такі як надання LLM можливості відповідати на питання про нову або конфіденційну інформацію та зменшення фактичних неточностей [4]. Однак аналіз наукової літератури виявляє кілька проблем, які ще не вирішені повністю.

По-перше, продуктивність і точність залишаються критичними проблемами. Загальна якість системи RAG значною мірою залежить від пошукової системи, якщо пошукова система не може знайти відповідні документи, генератор, незалежно від його потужності, не зможе виконати завдання [1]. Крім того, вибір між RAG і точним налаштуванням є складним; RAG чудово впорається з включенням нових фактів, але точне налаштування може бути кращим для наповнення моделі конкретними, неявними знаннями або стилями, що свідчить про необхідність гібридних підходів. Обчислювальні витрати на індексацію та пошук у великих колекціях документів також залишаються важливим фактором, що впливає на продуктивність [4; 10].

По-друге, інтерпретація результатів є постійною проблемою. Хоча RAG забезпечує певну прозорість, цитуючи свої джерела, внутрішня робота як dense retriever (чому він вважав уривок семантично подібним), так і генератора LLM (як він синтезував свою відповідь) залишається «чорною скринькою»

[1]. Ця відсутність справжньої інтерпретованості є перешкодою в науковій або медичній сферах, де розуміння міркувань моделі є надзвичайно важливим.

Нарешті, існує значна прогалина в багатомовній підтримці. Фундаментальні дослідження та домінуючі еталони, що використовуються для розробки та оцінки таких систем, як DPR та REALM, переважно орієнтовані на англійську мову [7; 10]. Як наслідок, продуктивність, надійність та культурні нюанси цих моделей в інших мовах залишаються в основному неперевіреними. Існує явна нестача високоякісних, маркованих наборів даних та спеціалізованих моделей для багатьох мов, включаючи українську. Це означає, що застосування цих методів RAG безпосередньо до українських текстів може призвести до значно нижчої точності, оскільки базові моделі пошуку та генерації не оптимізовані для її унікальних лінгвістичних структур. Це є критичною та відкритою областю для майбутніх досліджень.

Метою цієї статті є розробка та експериментальне тестування повного методу RAG, призначеного для автоматизованої інформаційної підтримки користувачів. Основним обмеженням і метою цієї роботи є створення рішення, яке працює швидко й не потребує дорогих обчислювальних ресурсів. Ця спрямованість має на меті зробити рішення недорогим і практично прийнятним для невеликих організацій, таких як незалежні фірми, університетські факультети, студентські організації або інші застосування, які часто не мають бюджету для великих моделей або чи необхідного обчислювального обладнання.

Виклад основного матеріалу. Точна перевірка будь-якої системи RAG залежить від прозорої презентації її архітектури та ретельної емпіричної оцінки її ефективності. Як показують комплексні дослідження в цій галузі, система RAG – це не єдине ціле, а модульна система, що зазвичай складається щонайменше з пошуку та генератора. Компонент пошуку відповідає за пошук відповідного контексту у зовнішній базі знань, а генератор (LLM) синтезує відповідь, яка базується виключно на цьому контексті. Успіх усього цього процесу залежить від якості компонентів, обраних для кожного етапу [3].

Сучасні фреймворки полегшують побудову цих систем, дозволяючи дослідникам координувати потік даних і «підключати» різні компоненти для векторного зберігання, вбудовування та генерації. Ця модульність є важливою для основного завдання цього дослідження – експериментальної оцінки.

Найбільш критичною оцінкою, як визначено в систематичних оглядах, є визначення ефективності пошукової системи. Якщо пошукова система не може знайти правильні документи, генератор неминуче видає неправильну або нерелевантну відповідь, навіть якщо сама LLM є потужною. Тому основний метод дослідження полягає в ізоляції та тестуванні цього етапу пошуку за допомогою точних, рангових показників точності (таких як точність Top-k) для вимірювання частоти знаходження правильної інформації. Згодом необхідна комплексна оцінка для перевірки якості та фактичної узгодженості остаточної згенерованої відповіді [2; 3].

На основі цих принципів система була розроблена та протестована. Архітектура системи, реалізована в цьому дослідженні, відповідає послідовному робочому процесу RAG, розробленому з урахуванням модульності та ощадливості. Весь процес від запиту користувача до остаточної відповіді, координується за допомогою комбінації спеціалізованих бібліотек. Робочий процес відбувається наступним чином:

1. Процес починається, коли користувач надсилає запит. Цей запит у вигляді простого тексту негайно передається до моделі вбудовування (embedding), яка перетворює текст у високорозмірний вектор. У цьому дослідженні використовувалися різні моделі з відкритим кодом, отримані з репозиторію HuggingFace.

2. Далі вектор запиту використовується для виконання пошуку схожості у попередньо заповненій векторній базі даних. Для цієї ролі було обрано ChromaDB (langchain_chroma.Chroma), легке сховище векторів з відкритим кодом. Ця база даних зберігає вбудовані векторні представлення фрагментів вихідного документа.

3. ChromaDB повертає «Top-k» найбільш семантично схожих фрагментів документа. Цей крок пошуку явно управляється, де база даних перетворюється в об'єкт пошуку і налаштовується для отримання k найбільш релевантних документів для тестування точності.

4. Витягнуті фрагменти («контекст») автоматично об'єднуються з оригінальним запитом користувача в структуровану підказку. Ця розширена підказка надсилається до генеративної великої мовної моделі. У цьому дослідженні використовується ChatGroq для інтеграції з Groq API, який забезпечує високошвидкісне виведення для моделей з відкритим кодом.

5. Мовна модель генерує відповідь природною мовою, засновану виключно на наданому контексті, яка потім надсилається користувачеві.

Для побудови та перевірки цієї архітектури використовується набір сучасних бібліотек з відкритим кодом.

Було використано LangChain (langchain, langchain_chroma, langchain_groq та інші). Вона служить основною платформою для координації. LangChain забезпечує зв'язок, що поєднує всі компоненти, пропонуючи високий рівень абстракції для завантаження документів, розділення тексту, інтеграції моделей вбудовування, операцій векторного зберігання та взаємодії з LLM. Його використання є центральним для всього дослідження.

Платформа HuggingFace є джерелом для всіх моделей вбудовування з відкритим кодом, протестованих у цьому дослідженні.

ChromaDB використовується як постійна локальна векторна база даних, що забезпечує ефективне зберігання та пошук вбудованих документів без необхідності використання потужної серверної інфраструктури.

Для перевірки ефективності запропонованої системи RAG було проведено багатоетапну оцінку з використанням набору тестів, створеного за допомогою PyTest. Такий підхід дозволяє незалежно виміряти точність пошукової системи та загальну якість генерованих відповідей.

Першим етапом є тестування компонента пошукової системи. Оскільки якість пошукової системи є основним вузьким місцем, її тестування треба проводити окремо. Спочатку створення тестового набір, де кожне питання зіставляється з ідентифікатором правильного фрагменту тексту(chunk). Далі цей тестовий набір тестується для чотирьох різних моделей вбудовування. Для вимірювання йде в три рівня точності: top-1, top-3, top-7. Правильний фрагмент присутній у k найкращих документах. Цей тест дозволяє провести пряме кількісне порівняння моделей вбудовування для набору даних українською мовою.

Другим етапом є тестування генерації від початку до кінця. Повна система RAG тестується на якість кінцевої відповіді з використанням моделі вбудовування з найкращими показниками. Набір оцінювання містить питання та вручну створену очікувану відповідь для кожного з них. Система обробляє кожне питання, щоб отримати контекст і згенерувати фактичну відповідь. Оскільки тестування проводиться вже на основі результатів тестування пошукової системи, то максимальний відсоток коректності приблизно буде дорівнювати отриманому на першому етапі. LLM слугує автоматизованим оцінювачем і відповідачем. Він отримує підказку(prompt) та визначає чи були відповіді семантично еквівалентними, повертаючи лише результат.

Останнім етапом є показники фреймворку Ragas, який створений для тестування RAG систем. Він надає спеціалізовані показники для оцінювання системи без повної залежності від відповідей, наприклад:

– faithfulness (метрика генератора) вимірює чи відповідь фактично підтверджується контекстом, кількісно оцінюючи галюцинацію;

– answer_relevancy (метрика генератора) вимірює чи відповідь безпосередньо стосується оригінального питання, а не лише контексту;

– context_precision (метрика пошукової системи) вимірює відношення сигнал/шум у відновленому контексті, штрафуючи нерелевантну інформацію;

– context_recall (метрика пошукової системи) вимірює чи знайшла пошукова система всю необхідну інформацію для відповіді на запитання;

– answer_correctness (метрика генератора) порівнює згенеровану відповідь з істинною відповіддю, поєднуючи фактичну та семантичну схожість для отримання загального балу правильності [9].

Результати. Першим і найбільш важливим експериментом було визначення найефективнішої моделі вбудовування для конкретного набору даних українською мовою. Чотири вибрані моделі, всі з яких були отримані з Hugging Face, були протестовані (табл. 1) на їхню здатність знаходити правильний фрагмент документа для набору з тестових питань.

Експеримент демонструють чітку ієрархію продуктивності та розкривають кілька ключових моментів щодо вибору ефективної моделі вбудовування.

Модель intfloat/multilingual-e5-large-instruct однозначно показала найкращі результати. Це була єдина модель, яка досягла ідеальної 100% точності в Top-7, а її точність Top-1 на рівні 90% була значно вищою, ніж у всіх конкурентів. Цю чудову продуктивність можна пояснити двома основними факторами. Ця модель використовує 1024-вимірні вектори, найбільші в тестовій групі. Більший розмір дозволяє їй захоплювати більш детальне та нюансоване семантичне представлення тексту. Це призводить до більш точного зіставлення схожості. Вона була спеціально налаштована на величезному наборі даних пар запитів та уривків. Це узгоджує її розуміння тексту з наміром запиту користувача, що робить її придатною для завдання пошукової системи RAG.

Модель paraphrase-multilingual-mpnet-base-v2 з її більшими 768-вимірними векторами перевершила меншу модель paraphrase-multilingual-MiniLM-L12-v2 за точністю Top-1 та Top-7. Це підтверджує

ідею, що вища розмірність може забезпечити багатший семантичний простір для пошуку, хоча це вимагає більших обчислювальних витрат. Однак стрибок моделі e5-large-instruct до 1024 вимірів у поєднанні з її навчанням демонструє більш значне підвищення якості.

Зважаючи на те, що вона була спеціально навчена на українській (а також англійській та російській) мовах, вона мала кращу точність ніж така сама mpnet з таким самим векторним простором у всіх трьох точностях. Це свідчить про те, що для складного семантичного пошуку якість і масштаб вузька лінгвістична спеціалізація можуть бути важливішими, ніж загальне багатомовне навчання.

На основі цих результатів вона була обрана єдиною моделлю вбудовування для системи RAG у всіх наступних тестах на генерацію від початку до кінця.

Наступним кроком було обрання заощадливого генератора (LLM) для остаточного синтезу відповіді. Метою було знайти оптимальний баланс між якістю відповіді та вартістю API, що є критичним фактором для невеликих організацій. Потім автоматизований оцінювач LLM порівняв відповідь, згенеровану з очікуваною відповіддю на предмет семантичної еквівалентності.

Як видно з таблиці 2, результати добре показують баланс між якістю та вартістю. Модель llama-3.1-8b-instant була набагато найдешевшою опцією, але її рівень успішності 83,3% був неідеальним. На диво, набагато дорожча openai/gpt-oss-20b показала найгірші результати, з рівнем успішності лише 75%. Максимальна вартість проти максимальної якості: моделі llama-4-maverick і qwen3-32b досягли найвищої якості з рівнем успішності 90% (неправильно відповіли лише на 6 питань). Однак вони також є найдорожчими моделями, вартість яких майже вдвічі перевищує вартість моделі scout. За результатами дослідження, найкраще співвідношення якості та вартості показала модель meta-llama/llama-4-scout-17b-16e-instruct. Вона досягла високого рівня успішності 88,3%, лише на одну неправильну відповідь поступаючись моделям з найкращими показниками, при цьому коштуючи значно менше (0,45 долара за 1 млн вхідних токенів та 1 млн вихідних токенів).

Останнім кроком було налаштування основних параметрів системи –підказок та стратегії розділення на фрагменти. Для цієї оцінки було використано фреймворк Ragas, який надає набір детальних метрик для вимірювання продуктивності компонентів та кінцевої продуктивності.

Було протестовано чотири різновиди підказок (табл. 3), зберігаючи фіксовану стратегію фрагментації. Підказки варіювалися від базового прикладу (№1) і короткої версії (№2) до повної, заснованої на правилах підказки англійською мовою (№3) та її українського перекладу (№4).

Таблиця 1

Результати точності top-1, top-3 і top-7

Назва моделі	top-1, %	top-3, %	top-7, %	Вимір векторного простір
uaritm/multilingual_en_ru_uk	80	96	98	768
sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2	72	86	94	384
sentence-transformers/paraphrase-multilingual-mpnet-base-v2	76	86	96	768
intfloat/multilingual-e5-large-instruct	90	98	100	1024

Таблиця 2

Результати правильності різних LLM

Генеративна модель (Groq API)	Ціна (сума цін 2M вхідних та вихідних токенів), $	Правильні відповіді	Неправильні відповіді	Відсоток здачі, %
llama-3.1-8b-instant	0.13	50	10	83.3
openai/gpt-oss-20b	0.60	45	15	75
meta-llama/llama-4-scout-17b-16e-instruct	**0.45**	**53**	**7**	**88.3**
meta-llama/llama-4-maverick-17b-128e-instruct	0.80	54	6	90
qwen/qwen3-32b	0.88	54	6	90

Таблиця 3

Метрики Ragas для набору підказок

Характеристики	Підказка № 1	Підказка № 2	Підказка № 3	Підказка № 4
context precision	0.9	0.9	0.9	0.9
context recall	0.7875	0.7	0.7	0.775
faithfulness	0.6148	0.2857	0.6066	0.4927
answer relevancy	0.8	0.6855	0.817	0.6249
answer correctness	0.4781	0.4852	0.564	0.4415

ISSN 2786-5460 (Print), ISSN 2786-5479 (Online)

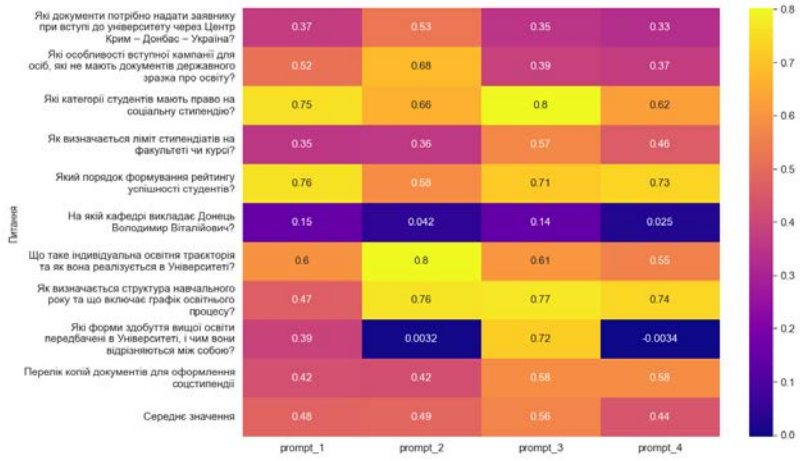

Рис. 2. Метрика answer correctness для набору підказок

Підказка не впливає на пошук, тому context_precision залишився незмінним. Ключовим висновком стала продуктивність підказки №3 (Повна), яка перевершила всі інші за найважливішим показником – answer_correctness (рис. 2). Цей показник, який порівнює згенеровану відповідь з істинним значенням є найкращим індикатором загальної якості. Мінімальні правила в підказці №2 (коротка) призвели до падіння точності, що підтверджує, що чіткі інструкції є життєво важливими для запобігання галюцинаціям. На основі вищої правильності відповідей та релевантності відповідей було обрано підказку №3.

Використовуючи найкращу підказку (підказка №3), було протестовано три стратегії розбиття на фрагменти:

– фрагмент №1 – розмір: 1000, перекриття: 150, кількість отриманих фрагментів (k): 4;
– фрагмент №2 – розмір: 1400, перекриття: 200, кількість отриманих фрагментів (k): 5;
– фрагмент №3 – розмір: 600, перекриття: 100, кількість отриманих фрагментів (k): 4.

Таблиця 4

Метрики Ragas для набору фрагментів

Характеристики фреймворку Ragas	chunk № 1	chunk № 2	chunk № 3
context precision	0.9	0.8889	0.8889
context recall	0.7	0.6847	0.775
faithfulness	0.6066	0.4527	0.5297
answer relevancy	0.817	0.7161	0.797
answer correctness	0.564	0.4536	0.5509

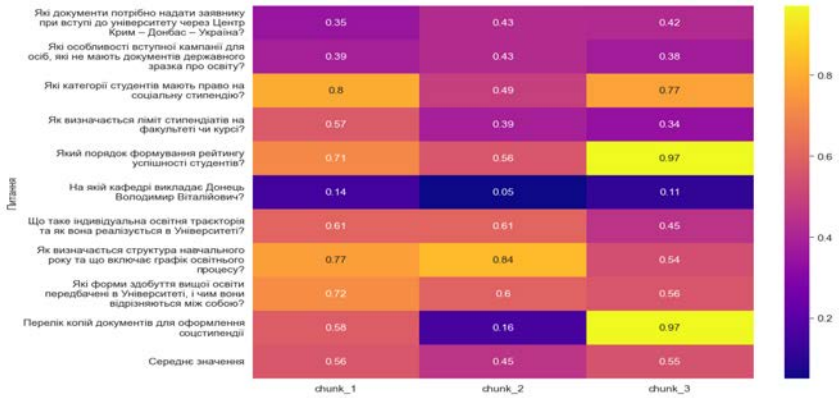

Рис. 3. Метрика answer correctness для набору фрагментів

Фрагмент №1 (середній) показав найкращі загальні результати, переконливо вигравши за правильністю відповідей (рис. 3) та релевантністю відповідей. Фрагмент №2 (великий і багато) показав найгірші результати, оскільки його великий і зашумлений контекст спричинив значне зниження як точності, так і правильності відповідей. Хоча фрагмент №3 (малий) досяг найкращого показника

відтворення контексту, його кінцеві показники по іншим показникам не перевершили збалансований підхід фрагмента №1.

Висновки. У цьому дослідженні було успішно розроблено та протестовано заощадливий метод RAG з використанням LangChain, ChromaDB та Groq API, що підтвердило його життєздатність для невеликих організацій. Систематичний багатоетапний процес тестування підтвердив функціональність системи.

Ключові висновки показують, що точність пошуку має першочергове значення. Модель вбудовування intfloat/multilingual-e5-large-instruct досягла 100% точності Top-7 на українському наборі даних. Для генерації meta-llama/llama-4-scout-17b-16e-instruct LLM, керований правилами підказки №3 та стратегією розбиття на фрагменти по 1000 символів, дав найвищу точність відповідей, збалансувавши продуктивність та вартість.

Перспективи на майбутнє включають тестування спеціалізованих українських моделей вбудовування, тестування масштабованості архітектури на більших вибірках, впровадження системи в реальних додатках, таких як університетські та корпоративні чат-боти. А також постійне тестування нових компонентів RAG для підтримки найсучаснішої та фінансової продуктивності.

Список використаних джерел:

1. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions / L. Huang et al. ACM Transactions on Information Systems. 2024. https://doi.org/10.1145/3703155.

2. Evaluation of a retrieval-augmented generation system using a Japanese Institutional Nuclear Medicine Manual and large language model-automated scoring. Y. Fukui et al. Radiological Physics and Technology. 2025. https://doi.org/10.1007/s12194-025-00941-y.

3. Evaluating Retrieval-Augmented Generation Models for Financial Report Question and Answering / I. Iaroshev et al. *Applied Sciences*. 2024. Vol. 14, no. 20. P. 9318. https://doi.org/10.3390/app14209318.

4. Gao Y. et al. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv preprint. 2023. arXiv:2312.10997. https://doi.org/10.48550/arXiv.2312.10997.

5. Hersh W. Search still matters: information retrieval in the era of generative AI. *Journal of the American Medical Informatics Association*. 2024. https://doi.org/10.1093/jamia/ocae014.

6. Liu X. A Survey of Hallucination Problems Based on Large Language Models. *Applied and Computational Engineering*. 2024. Vol. 97, no. 1. P. 24–30. https://doi.org/10.54254/2755-2721/2024.17851.

7. Patrick L. et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv preprint. 2020. arXiv:2005.11401. https://doi.org/10.48550/arXiv.2005.11401.

8. Pokhrel S., K C B., Shah P. B. A Practical Application of Retrieval-Augmented Generation for Website-Based Chatbots: Combining Web Scraping, Vectorization, and Semantic Search. *Journal of Trends in Computer Science and Smart Technology.* 2024. Vol. 6, no. 4. P. 424–442. https://doi.org/10.36548/jtcsst.2024.4.007.

9. Ragas. Ragas. URL: https://docs.ragas.io/en/stable/. (дата звернення: 01.11.2025).

10. Vladimir K. et al. Dense Passage Retrieval for Open-Domain Question Answering. arXiv preprint. 2020. arXiv:2004.04906. https://doi.org/10.48550/arXiv.2004.04906.

Дата надходження статті: 18.11.2025
Дата прийняття статті: 10.12.2025
Опубліковано: 30.12.2025