УДК 004.8:004.912 **doi: 10.26906/SUNZ.2026.2.110**

І. М. Івасенко, Т. В. Філімончук, С. О. Партика, Д. І. Пивоварова

Харківський національний університет радіоелектроніки, Харків, Україна

МОДЕЛЬ РОЗРОБКИ ТЕМАТИЧНИХ ЧАТ-БОТІВ З ВИКОРИСТАННЯМ ПІДХОДУ RAG

Анотація. Актуальність дослідження зумовлена стрімким розвитком великих мовних моделей (LLM), таких як GPT-4, Llama 3 та Claude, революціонізував сферу обробки природної мови (NLP). LLM демонструють виняткові здібності до генерації зв'язного тексту, узагальнення інформації та ведення діалогу. Однак, при їх застосуванні у спеціалізованих доменах (юриспруденція, медицина, технічна підтримка, корпоративні бази знань) виникають критичні обмеження. По-перше, параметричні знання моделей обмежені датою завершення їх навчання (knowledge cutoff), що унеможливлює роботу з актуальною інформацією. По-друге, моделі схильні до "галюцинацій" – генерації правдоподібних, але фактично невірних тверджень, особливо коли запит стосується вузькоспеціалізованих даних, відсутніх у навчальному наборі. Архітектура RAG стала стандартом де-факто для вирішення цих проблем, поєднуючи генеративні можливості LLM із точним пошуком у зовнішніх базах знань. Проте, практика показує, що "наївна" реалізація RAG (Vanilla RAG) часто є недостатньою для побудови надійних систем. Втрата контексту під час пошуку, нездатність обробити складні запити користувача та відсутність механізмів верифікації призводять до нерелевантних відповідей. У зв'язку з цим, актуальною науково-практичною задачею є не просто імплементація RAG, а розробка та дослідження методів оптимізації кожного етапу генерації. **Об'єктом дослідження** є процеси інформаційного пошуку та генерації природної мови в інтелектуальних діалогових системах, побудованих на базі великих мовних моделей. **Предметом дослідження** є методи та алгоритми підвищення точності, релевантності та контекстуальної узгодженості відповідей у системах архітектури RAG, а саме гібридний пошук, покращення запитів користувача. **Висновки.** Практична цінність дослідження полягає у створенні моделі архітектури, яку можна адаптувати для будь-якої предметної області (від технічної документації до нормативно-правових баз), забезпечуючи при цьому вищу метричну точність відповідей порівняно з базовими рішеннями.

Ключові слова: тематичні чат-боти, великі мовні моделі, штучний інтелект, RAG, NLP, попередня обробка, покращення запиту, гібридний пошук, розумна генерація тексту, механізм переранжування, релевантність відповіді, точність контексту.

Постановка проблеми

Попри суттєві досягнення у розвитку великих мовних моделей (LLM), їх пряме застосування для побудови спеціалізованих інформаційних систем супроводжується низкою серйозних обмежень.

Ключовою проблемою є нездатність моделей працювати з актуальною інформацією, що з'явилася після завершення процесу навчання, а також підвищений ризик так званих "галюцинацій" – формування переконливих, проте помилкових відповідей.

Архітектура Retrieval Augmented Generation (RAG) була запропонована як підхід до усунення цих недоліків, оскільки забезпечує інтеграцію мовної моделі із зовнішніми джерелами знань. Однак, класична реалізація RAG (так званий "Vanilla RAG"), що базується на простому розбитті тексту на фрагменти (chunking) та векторному пошуку (dense retrieval), часто виявляється недостатньою для забезпечення високої якості обслуговування користувачів у тематичних доменах.

Основна проблема полягає у невідповідності (misalignment) між запитом користувача та збереженими даними. Користувачі схильні формулювати короткі, нечіткі або контекстно-залежні запитання, які семантично можуть не перетинатися із відповідними фрагментами документації. Векторний пошук, хоч і є ефективним для знаходження загальних концепцій, часто демонструє низьку точність при пошуку специфічних термінів, власних назв або абревіатур, що призводить до вибірки нерелевантного контексту.

Додатковою проблемою є обмеження самої генеративної моделі при роботі з контекстом. Навіть за умови успішного пошуку, LLM може ігнорувати надану інформацію, якщо вона суперечить її внутрішнім вагам, або ж втрачати важливі деталі, якщо обсяг контексту занадто великий (проблема "Lost in the Middle"). Таким чином, науково-технічна проблема полягає у необхідності вдосконалення архітектури RAG для мінімізації помилок на етапах пошуку та генерації. Необхідно вирішити задачу підвищення точності ретрівалу (retrieval accuracy) та узгодженості відповідей (faithfulness) шляхом впровадження методів гібридного пошуку, семантичної трансформації запитів та алгоритмічного керування увагою моделі через інженерію запитів. Вирішення цієї проблеми дозволить створити тематичний чатбот, здатний надавати верифіковані та точні відповіді, що є критичною вимогою для професійного застосування.

Огляд досліджень і розробок

Проблематика покращення генерації тексту за допомогою зовнішніх джерел знань (RAG) є однією з найбільш динамічних у сучасній комп'ютерній лінгвістиці. Аналіз наукового доробку за останні роки дозволяє виділити фундаментальні праці, що сформували архітектуру сучасних RAG-систем та визначили вектори їх оптимізації. Нижче проаналізовано ключові дослідження, що стосуються як базової архітектури, так і методів покращення пошуку та генерації.

У роботі [1] автори запропонували архітектуру, яка об'єднує попередньо навчену модель "seq2seq"

(BART) з механізмом щільного векторного пошуку (Dense Passage Retrieval). Дослідження показало, що моделі RAG перевершують звичайні параметричні моделі у задачах Open Domain Question Answering, навіть маючи значно меншу кількість параметрів. Головний висновок роботи полягає в тому, що відокремлення "знань" (база даних) від "вміння міркувати" (нейромережа) дозволяє легко оновлювати знання системи без необхідності перенавчання моделі.

У роботі [2] було продовжено розвиток попередньої архітектури. Дослідники зосередилися на етапі переднавчання (pre-training) та запропонували підхід, де ретрівер (пошуковик) навчається одночасно з генератором. Було доведено, що такий спільний процес навчання (end-to-end training) значно покращує здатність моделі знаходити релевантні документи, оскільки ретрівер отримує зворотний зв'язок від мовної моделі щодо корисності знайденої інформації.

Робота [3] стала переломною для відмови від ключових слів на користь семантичного пошуку. Автори довели неефективність традиційного алгоритму BM25 для складних запитань та запропонували модель DPR, що використовує два окремі енкодери (для питань та для контексту). Висновком дослідження є те, що щільні векторні представлення (dense embeddings) значно краще вловлюють семантичну близькість, ніж перетин слів.

У роботі [4] було розроблено механізм "пізньої взаємодії" (late interaction), який зберігає контекстуальні вектори токенів та порівнює їх лише на фінальному етапі. Дослідження показало, що цей метод (який лежить в основі сучасних Re-ranker моделей) дозволяє досягти точності повноцінних трансформерів (BERT) при швидкості, співставній з векторним пошуком. Результати дослідження обґрунтовують необхідність використання етапу переранжування (Reranking) у тематичних ботах.

У роботі [5] було запропоновано архітектуру Fusion-in-Decoder, яка замість об’єднання всіх знайдених документів в один довгий текст і подання його на вхід моделі, обробляє кожен документ окремо. Злиття інформації відбувається лише на етапі генерації відповіді, що дозволяє ефективніше обробляти знайдені документи та надає моделі можливість самій вирішувати, на що звертати увагу при генерації.

Робота [6] підтвердила гіпотезу про те, що доступ до гігантської бази даних (трильйони токенів) дозволяє зменшити розмір самої нейромережі у 25 разів без втрати якості. Наведені результати свідчать про те, що для побудови якісного чат-боту важливіша якість бази знань та пошуку, аніж розмір самої LLM.

У роботі [7] автори дослідили вплив інжинірингу запитів на якість відповідей. Хоча робота не фокусувалася виключно на RAG, вона довела, що спонукання моделі до покрокового міркування ("Let's think step by step") значно покращує її здатність аналізувати складний контекст та зменшує кількість логічних помилок.

Аналіз наведених джерел дозволяє стверджувати, що "Vanilla RAG" (простий векторний пошук та генерація) є пройденим етапом. Сучасні дослідження вказують на необхідність впровадження гібридних методів пошуку (поєднання DPR та BM25), використання механізмів Re-ranking та просунутих технік інжинірингу запитів для створення надійних систем.

На основі проведеного аналізу можливо представити стандартну модель архітектури RAG у вигляді спеціалізованого набору складових (1):

$$M = \{МП, БЗ, ВММ\}, \quad (1)$$

де: МП – модуль пошуку, який знаходить релевантні документи або фрагменти тексту (наприклад, Retriever); БЗ – зовнішнє сховище знань, компонент, що містить інформацію, недоступну моделі під час її навчання (приватні корпоративні дані, свіжі новини, специфічна технічна документація); ВММ – велика мовна модель (LLM), яка попередньо навчена на необхідних масивах тексту для розуміння та генерації природньої мови.

Наведена модель, попри свою концептуальну простоту, має низку критичних архітектурних вразливостей, які суттєво обмежують її використання у професійних сферах:

- семантичний розрив – проблема неспівпадіння лексики, коли користувач формулює запити побутовою мовою, описуючи симптоми або загальні ситуації, тоді як технічна документація оперує суворою професійною термінологією. Оскільки базовий модуль пошуку (МП) часто спирається на косинусну близькість векторів, він може не виявити зв'язку між цими фразами, оскільки в багатовимірному просторі їхні вектори можуть бути ортогональними (дивитися в різні боки), попри спільний зміст;

- низька точність пошуку специфічних сутностей, що пов’язано з втратами інформації під час стиснення. Коли модель перетворює текст у набір чисел, вона добре зберігає загальний зміст ("це текст про комп'ютери"), але часто втрачає точні деталі. Через це системи часто плутають схожі за написанням, але критично різні за суттю ідентифікатори (наприклад, артикули "Part-A100" та "Part-A200" або коди помилок). Для нейромережі ці токени семантично майже ідентичні, що призводить до видачі неправильних інструкцій;

- "Lost in the Middle" – обмеження архітектури, на якій базуються сучасні великі мовні моделі, призводять до того, що механізм уваги (Attention Mechanism) працює нерівномірно: модель найкраще сприймає інформацію, розташовану на початку та в кінці вхідного тексту. Якщо релевантний фрагмент, знайдений модулем пошуку, потрапляє в середину довгого списку контексту, модель схильна ігнорувати його при генерації відповіді, що нівелює саму ідею пошуку;

- схильність до "галюцинацій" – якщо модуль пошуку не знайшов релевантної інформації (або знайшов нерелевантну), чат-бот все одно намагається дати відповідь, заповнюючи прогалини знань вигадками або застарілими даними зі своєї тренувальної

пам'яті. Без примусового посилання на факти, така відповідь може виглядати переконливо, але бути фактично хибною.

Мета дослідження

Метою роботи є підвищення точності, фактологічної достовірності та контекстуальної релевантності відповідей тематичних діалогових систем шляхом дослідження, вдосконалення та програмної реалізації розширеної архітектури RAG (Advanced RAG).

Для досягнення поставленої мети необхідно вирішити наступні задачі:

- проаналізувати обмеження базової архітектури та існуючих підходів до їх подолання;

- модифікувати математичну модель архітектури RAG шляхом додавання складових, які реалізують поєднання методів семантичної трансформації запитів (Query Rewriting), гібридного пошуку інформації (Hybrid Search) та керованої генерації тексту;

- розглянути практичні сценарії використання вдосконаленої архітектури та провести оцінку її ефективності.

Викладення основного матеріалу

Стандартний підхід до побудови RAG-систем працює за лінійною схемою: користувач ставить питання, система шукає схожий текст у базі за допомогою векторної подібності та передає його нейромережі для генерації відповіді.

Цей метод демонструє високу ефективність для загальних запитів, проте часто виявляється недостатнім у спеціалізованих доменах (юриспруденція, технічна підтримка, медицина).

Основна проблема полягає в семантичному розриві: користувач і база даних "говорять різними мовами". Користувачі використовують побутову лексику, тоді як документи містять суху професійну термінологію. Для вирішення цієї проблеми та підвищення метрик якості (Accuracy, Recall, Precision) існує низка методів вдосконалення архітектури на етапах попередньої обробки, пошуку та генерації. У роботі пропонуються впровадження цих методів (як окремих модулів) в існуючу архітектуру (1), які нададуть "розумних посередників" на кожному етапі:

- модуль покращення запиту, який перетворює нечітке питання користувача на професійний пошуковий запит (МПЗ);

- модуль гібридного пошуку, який буде шукати інформацію декількома способами одночасно (за змістом та за ключовими словами), щоб нічого не пропустити (МГП);

- модуль розумної генерації, який аналізує знайдене та формує відповідь, спираючись виключно на факти (МРГ).

Після додаванням цих модулів, покращена модель буде мати вигляд (2):

$$М = \{МПЗ, МГП, ВММ, БЗ, МРГ\}. \quad (2)$$

Логіка руху інформації в представленій моделі (2) трансформується з лінійної у багаторівневу, де вихід кожного попереднього модуля стає збагаченим входом для наступного.

Модуль покращення запиту (МПЗ) виправляє найслабше місце будь-якого пошуку – постановку питання. Користувачі часто пишуть коротко, з помилками або використовують займенники, зрозумілі тільки з контексту діалогу (наприклад, "А скільки він коштує?", де "він" стосується товару, згаданого три повідомлення потому). Якщо відправити такий запит у базу даних, система не знайде нічого релевантного, бо в документах немає слова "він", є конкретні назви товарів.

Вирішити цю проблему можливо за допомогою методу Query Rewriting, який здійснює переписування запиту. Перед тим як шукати інформацію, використовується допоміжна велика мовна модель (LLM), що "перекладає" питання користувача на мову пошукової системи. Це працює наступним чином: система бере історію діалогу та останнє питання користувача і на їх основі створює новий, самодостатній запит:

- було: "Як мені це підключити, якщо зникло світло?";

- стало після обробки: "Інструкція з підключення пристрою [назва моделі] до джерела безперебійного живлення або генератора в умовах відсутності електропостачання".

Такий переписаний запит містить всі необхідні ключові слова та контекст і гарантує, що пошуковий алгоритм шукатиме саме інструкцію, а не випадкові тексти зі словом "світло".

Окрім простого переписування одного запиту, існує більш просунута стратегія, яка називається розширенням запиту, або генерацією множинних варіантів. Часто буває так, що навіть ідеально переписане питання може не знайти потрібний документ просто тому, що автор документа використав зовсім інші слова для опису тієї ж проблеми. Наприклад, користувач питає про "ціну", а в документах використовується термін "вартість тарифного плану". Щоб вирішити цю проблему, мовна модель генерує не один, а три або п’ять варіантів одного й того ж запитання під різними кутами. Система автоматично створює синонімічні конструкції, розбиває складні питання на простіші підзадачі та формує векторні представлення для кожного з них. Далі пошук відбувається одночасно за всіма згенерованими варіантами. Потім система збирає всі знайдені за різними запитами документи та видаляє дублікати, залишаючи лише унікальний набір даних для аналізу. Такий підхід особливо ефективний у ситуаціях, коли база знань наповнювалася різними авторами в різний час та не має єдиного стандарту.

Після отримання переписаного запиту, модуль гібридного пошуку (МГП) починає пошук інформації. У звичайних системах використовують лише один метод, але для кращого результату необхідно оперувати декількома різними підходами (3):

$$МГП = \{ВП, ПКС, ОР, ПР\}, \quad (3)$$

де: ВП – векторний пошук; ПКС – пошук за ключовими словами; ОР – механізм об’єднання результату; ПР – механізм переранжування.

Перший підхід – це семантичний або векторний пошук (ВП). Уявімо, що комп'ютер перетворює кожну частину тексту в набір координат (чисел) у багатовимірному просторі. Тексти, які мають схожий

зміст, будуть знаходитися поруч у цьому просторі, навіть якщо вони написані різними словами.

Наприклад, якщо користувач шукає "проблеми з мотором", векторний пошук знайде документ про "несправності двигуна", тому що для нейромережі слова "мотор" та "двигун", "проблема" та "несправність" є синонімами і знаходяться поруч, що дозволяє знаходити відповіді, які людина могла б пропустити через розбіжність у термінології. Сучасні моделі пропонують вектори розмірністю 384-3072 чисел. Існує пряма кореляція: чим більша розмірність, тим більше семантичних нюансів може "запам'ятати" вектор, але тим повільніше працює пошук і тим більше оперативної пам'яті потребує база даних. Для тематичних чат-ботів оптимальним балансом вважається розмірність 768 або 1024.

Однак саме лише використання векторів не гарантує успіху, якщо модель, яка перетворює текст на числа, не розуміє специфіки предметної області. Слід зазначити, що більшість стандартних моделей навчені на текстах з інтернету, новинах та художній літературі. Вони чудово розуміють загальні поняття, але можуть губитися у вузькоспеціалізованих темах, що створює проблему зміщення домену.

Уявімо ситуацію в медичному чат-боті. Для звичайної моделі слова "серце" та "любов" можуть знаходитися поруч у векторному просторі, оскільки в художній літературі вони часто вживаються разом. Проте для лікаря "серце" має бути семантично пов'язане з термінами "міокард", "кардіологія", "шлуночок", а не з романтичними почуттями. Якщо модель цього не розуміє, пошук буде видавати нерелевантні результати.

Для вирішення цієї проблеми використовується вже існуюча велика мовна модель з додатковим її тренуванням на документах необхідної конкретної сфери. Цей процес нагадує підвищення кваліфікації для фахівця: мовна модель читає тисячі прикладів з бази знань і вчиться розуміти, що в контексті компанії слово "коса" – це не зачіска і не інструмент для трави, а, наприклад, елемент географічного ландшафту або деталь механізму. Після такого донавчання вектори стають значно точнішими, і система починає розрізняти найдрібніші нюанси професійної термінології. Однак векторний пошук має недолік, він іноді надто узагальнює: може сплутати "Модель X-100" та "Модель X-200", бо семантично це майже одне й те саме (обидва – назви моделей). Тут на допомогу приходить другий підхід – метод пошуку за ключовими словами (наприклад, алгоритм BM25), який працює подібно до функції "Знайти на сторінці", але розумніше. Він не просто рахує співпадіння слів, а оцінює їх важливість:

- якщо слово зустрічається в кожному документі (наприклад, "і", "що", "бути"), алгоритм ігнорує його;

- якщо слово рідкісне та унікальне (наприклад, конкретний код помилки "Error-404" або прізвище автора), алгоритм надає йому більшої ваги.

Завдяки роботі цього методу (ПКС) гарантується, що якщо користувач ввів точний артикул або специфічний термін, система знайде саме той документ, де він згадується.

Після попередніх дій у користувача є два списки знайдених документів: один сформований на основі розуміння змісту, інший – на основі точних слів. Наступним кроком є їх злиття в один фінальний перелік, тому що пряме порівняння результатів цих алгоритмів неможливе через різні критерії оцінки. Для об’єднання результатів використовується метод Reciprocal Rank Fusion (RRF), принцип роботи якого ґрунтується на врахуванні позиції (рангу) кожного документа у відповідних списках результатів пошуку:

- якщо певний документ займає високі позиції одночасно і у векторному, і у ключовому пошуку, він отримує найвищий пріоритет у фінальному списку;

- якщо документ знайдений лише одним із методів або знаходиться в кінці списків, він автоматично опускається нижче.

Такий підхід дозволяє об'єднати переваги обох технологій, коли система виводить у топ документи, які є найбільш релевантними як за змістом (вектором), так і за наявністю конкретних термінів, що створює надійну основу для генерації відповіді.

Завершальним етапом роботи модуля МГП є переранжування. Попередній список кандидатів (наприклад, 50 фрагментів) може містити "шум". Однак передавати весь цей масив даних безпосередньо у велику мовну модель (LLM) є помилкою з двох причин:

- економічна: більшість сучасних LLM тарифікують послуги за кількість токенів (частин слів) на вході. Обробка зайвої інформації збільшує вартість кожного запиту;

- якісна: існує феномен "розмиття уваги", коли модель отримує занадто багато тексту, в якому корисна інформація змішана з "шумом" (нерелевантним текстом) та ймовірність правильної відповіді падає. Модель може "загубитися" у великому обсязі даних.

Саме тут вступає в дію етап переранжування (Reranking) – процес, який працює як фільтр: він бере широку вибірку документів та залишає лише декілька (зазвичай 3-5) найбільш влучних. Щоб зрозуміти важливість переранжування, слід детальніше розглянути, як працює первинний пошук.

Векторний пошук використовує архітектуру, яку називають Bi-Encoder, що працює наступним чином: на першому кроці система заздалегідь створює "портрет" (вектор) кожного документа в базі; коли приходить запит, система створює його "портрет"; далі вона просто порівнює ці портрети на відстані.

Цей метод неймовірно швидкий (можна порівняти мільйони документів за секунду), але він поверхневий. Оскільки вектор документа створюється до того, як система дізнається запит користувача, він є узагальненим тому що у ньому "стиснуто" весь зміст абзацу в один набір чисел. Але при такому стисненні втрачаються тонкі нюанси зв'язків між словами, тому на етапі переранжування використовується інший інструмент – Cross-Encoder. На відміну від Bi-Encoder, який обробляє запит та документ окремо, Cross-Encoder приймає на вхід пару "Запит + Документ" одночасно, що дозволяє нейромережі використовувати механізм "уваги" (Self-Attention) на повну потужність. Модель переглядає кожне слово у запиті користувача та безпосередньо порівнює його з кожним

словом у документі, аналізуючи їх взаємодію в контексті.

Наприклад: припустимо, користувач питає: "Що робити, якщо пристрій погано працює взимку?". Векторний пошук (Bi-Encoder) може знайти документ: "Взимку компанія не працює по вихідних". Для нього слова "не працює" та "взимку" семантично близькі до запиту, хоча зміст зовсім інший. Через переранжування (Cross-Encoder) мовна модель "прочитає" цю пару разом і побачить, що в запиті словосполучення "не працює" стосується технічного стану, а в документі – графіку роботи офісу. Завдяки глибшому аналізу зв'язків слів, буде відкинуто цей документ або поставлено йому дуже низьку оцінку. Нажаль цей процес вимагає значно більше обчислювальних ресурсів та часу, саме тому не можна застосувати Cross-Encoder до всієї бази даних. У даному випадку слід використовувати двоступеневу схему: швидкий, але грубий (Bi-Encoder) пошук відбирає топ-50 кандидатів; повільний, але дуже розумний (Cross-Encoder) сортує ці 50 кандидатів та видає топ-5 ідеальних результатів. Такий сімбіоз дозволяє досягти балансу: система працює швидко (бо перевіряє детально лише мало документів) та надзвичайно точно.

Після завершення збору всієї необхідної інформації, програма переходить на фінальний етап обробки інформації, а саме оформлення відповіді в модулі розумної генерації (МРГ).

Навіть у випадку знаходження ідеальних документів, робота ще не завершена. Останній крок – змусити LLM правильно використати цю інформацію. Багато розробників помилково вважають, що достатньо просто скопіювати текст документів у чат та попросити "відповісти". Однак без спеціальних технік це призводить до неструктурованих або помилкових відповідей, для уникнення яких використовується комплексний підхід до конструювання запитів, що базується на науково обґрунтованих методиках:

$$МРГ = \{СР, ЛД, НП, ЦД\}, \quad (4)$$

де СР – сортування по релевантності; ЛД – створення "ланцюжка думок"; НП – навчання по прикладах; ЦД – цитування документів.

Однією з цікавих вад в роботі великих мовних моделей є так звана проблема "Lost in the Middle" (загублені посередині). Експерименти показали, що LLM найкраще запам'ятовують інформацію, яка знаходиться на самому початку тексту і в самому кінці запиту. Інформація, що розміщена всередині довгого тексту, часто ігнорується або "забувається" під час формування відповіді.

У покращеній системі це враховується при формуванні контексту. Після етапу переранжування, коли отримано 5 найкращих документів, вони не просто додаються у випадковому порядку, а спочатку сортуються таким чином, щоб найбільш релевантний документ (з найвищим балом Cross-Encoder) знаходився в кінці списку контексту – безпосередньо перед самим питанням користувача, а другий за важливістю документ ставився на початок. Менш важливі документи розміщуються всередині. Така структура подачі інформації гарантує, що модель зверне максимальну увагу на найважливіші дані.

Наступним кроком, перш ніж надати відповідь, модель форматує її за відповідними правилами (ця методика має назву "Chain of Thought"). У системний запит додається інструкція: "Перш ніж надати відповідь користувачу, проаналізуй наданий контекст крок за кроком. Випиши ключові факти, перевір, чи немає суперечностей між різними документами, і лише після цього сформулюй висновок". Це працює тому, що LLM є авторегресійною моделлю: вона передбачає наступне слово на основі попередніх. Коли модель спочатку генерує ланцюжок логічних міркувань, ці міркування стають частиною контексту для генерації фінальної відповіді, тобто чат-бот сам собі створює "чернетку", на яку потім спирається, що радикально зменшує кількість логічних помилок у відповідях.

Замість того, щоб писати довгі інструкції про те, як має виглядати відповідь (наприклад, "будь ввічливим, не використовуй жаргон, пиши стисло"), велика мовна модель отримує приклади. У запит включається кілька пар "Питання – Ідеальна Відповідь". Наприклад, користувач пише запит "Як скинути пароль?". Бот надає відповідь "Згідно з інструкцією безпеки (розділ 4), для скидання пароля перейдіть за посиланням... Зверніть увагу, що посилання дійсне 15 хвилин". Коли чат-бот бачить ці приклади, він автоматично копіює стиль, тон та структуру відповіді (In-Context Learning), що особливо важливо для тематичних ботів, які мають дотримуватися корпоративного стандарту спілкування.

Однією із найбільших проблем штучного інтелекту є "галюцинації". Щоб боротися з цією проблемою, впроваджується механізм суворої перевірки джерел. Мовній моделі надається жорстка інструкція: "Кожне твердження у відповіді має супроводжуватися посиланням на номер документа, з якого взято цю інформацію (наприклад, [документ №1]). Якщо інформації немає в наданих документах ти повинен чесно відповісти: "Я не знаю" і не намагатися вигадати відповідь". Це перетворює процес генерації тексту на задачу компіляції фактів. Якщо модель не може поставити посилання на джерело, вона з меншою ймовірністю згенерує це твердження, що критично важливо для таких сфер, як право або медицина, де вигадана порада може мати серйозні наслідки.

Окрім правильних інструкцій та цитування, сучасні системи все частіше використовують механізми самокорекції та рефлексії. У стандартній схемі система працює в один прохід: знайшла інформацію, згенерувала відповідь та віддала її користувачу. Але якщо знайдена інформація була неповною або помилковою, користувач отримає неякісний результат. Щоб цьому запобігти, може бути впроваджений додатковий етап перевірки. Після того, як модель сформулювала чернетку відповіді, вона сама виступає в ролі критика. Система аналізує власну відповідь на предмет логічних суперечностей та відповідності знайденим документам. Якщо модель виявляє, що відповідь неповна або базується на

слабких доказах, вона не показує її користувачу. Замість цього запускається повторний цикл пошуку з уточненими параметрами. Наприклад, якщо бот зрозумів, що йому не вистачає інформації про конкретну характеристику товару, він самостійно формує новий пошуковий запит саме по цій характеристиці, знаходить відсутні дані, і лише потім генерує фінальну відповідь. Такий ітеративний підхід перетворює лінійний процес на замкнений цикл, який триває доти, доки не буде досягнуто прийнятного рівня якості. Хоча це займає трохи більше часу, результат стає надійнішим.

Слід зазначити, що якість роботи будь-якої RAG-системи починається задовго до того, як користувач задасть питання. Вона починається з того, як обробляються вхідні документи. Якщо просто завантажити у базу цілу книгу одним шматком, система не зможе ефективно знайти конкретний абзац. Якщо ж спочатку розділити текст на частини занадто дрібно, втратиться контекст.

Для вирішення цієї проблеми пропонується застосовувати стратегію "Recursive Character Text Splitter" (Рекурсивний поділ тексту) з перекриттям (overlap). Найпростіший метод – розділяти текст через кожні 500 символів. Але це "сліпий" метод, який може розірвати речення посередині або відділити питання від відповіді. Уявіть, що одне речення каже: "Не натискайте червону кнопку", а наступне (яке потрапило в інший шматок): "якщо система не увімкнена". Якщо знайти тільки першу частину, інструкція стане небезпечною.

Рекурсивний метод працює розумніше. Спочатку він намагається розділити текст по великих логічних блоках (параграфах): якщо блок все ще завеликий, він ділить його по реченнях; якщо і це забагато – по словах. Використання такого підходу дозволяє зберегти цілісність думки.

Також точність відповіді можна покращити використанням техніки перекриття: кінець одного шматка тексту (chunk) дублюється на початку наступного. Наприклад, якщо ми маємо текст А-Б-В-Г, ми ріжемо його не як [А-Б] та [В-Г], а як [А-Б-В] та [Б-В-Г], що створює "страховку": якщо важлива думка знаходиться на стику двох шматків, вона не загубиться, а потрапить в обидва. Наприклад, система може використовувати розмір сегменту тексту 1000 токенів із перекриттям у 200 токенів.

Ще однією серйозною проблемою при підготовці даних є конфлікт між пошуком та генерацією. Для точного пошуку найкраще підходять маленькі шматочки тексту, які містять одну конкретну думку або факт. Їх вектори дуже чіткі, і системі легко знайти відповідність із запитом користувача. Проте для генерації якісної відповіді маленького шматочка часто недостатньо. Великій мовній моделі потрібен широкий контекст, щоб зрозуміти передісторію, умови та наслідки. Якщо просто віддати чатботу одне речення, він може не зрозуміти його суті без навколишнього тексту. Щоб вирішити цю дилему, рекомендується використовувати стратегію батьківського документа. Суть методу полягає в розриві зв'язку між тим, що шукає користувач, і тим, що надається для відповіді. Під час індексації документ розбивається на дуже дрібні фрагменти для пошуку, але зберігається зв'язок кожного дрібного фрагмента з його великим "батьківським" блоком. Коли система знаходить маленький фрагмент, який відповідає запиту, вона не передає його одразу на генерацію. Замість цього вона звертається до бази даних та витягує повний батьківський блок, до якого належав цей фрагмент. Таким чином, пошук відбувається з хірургічною точністю по дрібних деталях, а велика мовна модель отримує повноцінний розгорнутий контекст для формування зв'язної та обґрунтованої відповіді.

Також варто згадати про метод семантичного поділу тексту. Традиційний поділ на частини фіксованого розміру часто розриває логічні ланцюжки. Семантичний підхід пропонує ділити текст не за кількістю символів, а за змістом. Спеціальний алгоритм аналізує текст речення за реченням та вимірює, наскільки сильно змінюється тема розмови. Поки вектори сусідніх речень схожі, система вважає, що триває опис однієї думки, і об'єднує їх в один блок. Як тільки вектор різко змінюється, це слугує сигналом, що автор перейшов до нової теми, і тут потрібно зробити розріз. Такий підхід дозволяє створювати смислові блоки, які є цілісними та завершеними, що значно полегшує роботу нейромережі на етапі генерації.

Для побудови вище описаної системи (Advanced RAG) недостатньо стандартних інструментів, необхідно обрати набір технологій, які підтримують гібридний пошук та складну логіку обробки.

Оркестратор LangChain – це "скелет" застосунку, який дозволяє легко з'єднувати різні компоненти: завантажувачі файлів, моделі та бази даних. Саме він керує логікою: "спочатку перепиши запит, пошукай, відфільтруй".

Векторна база даних Qdrant, як і Weaviate "з коробки" підтримують гібридний пошук. Вони вміють зберігати і вектори (для розуміння змісту), і звичайний текст (для пошуку ключових слів), а також самостійно виконувати злиття результатів (Fusion), що значно спрощує розробку. Вибір бази даних – це не лише питання зберігання, а й питання алгоритмів індексації. Простого порівняння запиту з кожним документом у базі недостатньо, коли кількість документів сягає тисяч або мільйонів – це буде надто повільно.

Сучасні векторні бази даних використовують алгоритм наближеного пошуку найближчих сусідів – HNSW (Hierarchical Navigable Small World), який спочатку робить "широкий огляд" і визначає лише загальний напрямок пошуку, моментально відсіюючи все зайве (наприклад, визначає, що шукати треба в групі "юридичні договори"). Зрозумівши, де приблизно знаходиться відповідь, він заглиблюється в деталі і шукає вже всередині цієї меншої групи (наприклад, фокусується на "договорах оренди"), поступово звужуючи коло пошуку до конкретного файлу. Такий метод дозволяє швидко знаходити потрібну інформацію серед мільйонів записів.

Ще одним критичним компонентом Advanced RAG, який часто ігнорують у базових реалізаціях, є Hybrid Filtering (Фільтрація за метаданими). Тематичні документи завжди мають атрибути: рік видання, автор, категорія, рівень доступу. Векторний пошук сліпий до цих атрибутів і може знайти документ, який ідеально підходить за змістом, але є застарілим (наприклад, наказ, який вже скасовано).

Важливим елементом технологічного стека також є система маршрутизації діалогу. Не кожне повідомлення користувача вимагає запуску складного та дорогого процесу пошуку в базі знань. Якщо користувач просто вітається або дякує, немає сенсу навантажувати векторну базу даних. Для оптимізації роботи використовується легкий класифікатор на вході системи, який миттєво аналізує намір користувача та вирішує, куди спрямувати запит. Якщо це проста розмова, запит йде до звичайної великої мовної моделі. Якщо це питання по темі бази знань, запускається повний цикл RAG. Якщо ж питання стосується теми, яку бот не повинен обговорювати, система активує модуль безпеки мовної моделі та ввічливо відмовляє у відповіді. Така архітектура дозволяє суттєво економити обчислювальні ресурси та гроші, оскільки найважчі алгоритми задіються лише тоді, коли це дійсно необхідно. Крім того, це дозволяє підключати до системи різні джерела даних. Наприклад, для фінансових питань маршрутизатор може направити запит до бази зі звітами, а для технічних питань – до бази з інструкціями, що робить чат-бота універсальним помічником.

Для аналізу запиту, формування відповіді та її аналізу можна використовувати різні великі мовні моделі, деякі з яких запропоновано нижче:

- Embeddings (для створення векторів): text-embedding-3-small від OpenAI або bge-m3. Ключова особливість цих моделей – підтримка технології Matryoshka Representation Learning (MRL), що дозволяє динамічно скорочувати розмірність вектору (наприклад, з 1536 до 512 або 256 чисел) без суттєвої втрати якості пошуку, що критично важливо для оптимізації витрат на зберігання векторів у базі даних. Модель bge-m3 особливо цікава тим, що вона мультимовна і краще працює зі специфічними мовами, ніж стандартні моделі;

- LLM (для генерації): вибір на користь моделей серії GPT у RAG-системах може бути зумовлений не лише їхніми "знаннями", а передусім архітектурною особливістю, яка називається Instruction Following (слідування інструкціям). У RAG критично важливо, щоб модель ігнорувала свої попередньо отримані (вивчені) знання та відповідала виключно на основі наданого контексту. Моделі GPT демонструють високу стабільність у дотриманні системного запиту (System Prompt Adherence) та формати виведення;

- Re-ranker: bge-reranker-v2-m3 – спеціалізована програма-суддя, що приймає на вхід пару "Запит + Документ" як єдиний текст і пропускає їх через шари уваги (Self-Attention) разом та оцінює, як кожне слово запиту впливає на кожне слово документу. На виході вона дає не вектор, а одне число від 0 до 1, що відображає релевантність.

Для доведення, що "розумний" RAG працює краще за звичайний, потрібні конкретні цифри результатів. Традиційні метрики для перекладу тексту (як BLEU) тут не підходять, бо бот може сформулювати правильну думку іншими словами та BLEU покаже поганий результат. Для цього краще використовувати фреймворк RAGAS (Retrieval Augmented Generation Assessment). Ідея RAGAS геніальна: він використовує одну сильну LLM (наприклад, GPT-4), щоб вона оцінювала роботу запропонованої системи як екзаменатор. В ході аналізу вимірюються три ключові показники: вірність (Faithfulness), релевантність відповіді (Answer Relevance), точність контексту (Context Precision).

Вірність (Faithfulness) – ця метрика відповідає на питання: "Чи не вигадав бот щось від себе?" Модель-екзаменатор перевіряє кожне твердження у відповіді бота та шукає підтвердження у знайдених документах. Якщо бот сказав "Гарантія 3 роки" і в документах написано "Гарантія 3 роки" – бал 1.0, якщо бот сказав "Гарантія 5 років", а в документах "Гарантія 3 роки" або інформації немає – бал падає. Використання такого підходу дозволяє математично виміряти рівень "галюцинацій".

Релевантність відповіді (Answer Relevance) – ця метрика відповідає на питання: "Чи відповів бот саме на те, про що його питали?". Бот може сказати чисту правду, але не по темі. Наприклад, на питання "Як вимкнути прилад?" він може відповісти "Прилад був винайдений у 1990 році". Це "вірно" (Faithfulness висока), але "нерелевантно". RAGAS оцінює, наскільки відповідь корисна для користувача.

Точність контексту (Context Precision) – ця метрика оцінює роботу пошуковика із базової реалізації (1). Вона перевіряє, чи є "золоті" (ідеальні) документи на вершині списку знайденого. Якщо запропонована система Re-ranking працює правильно, найкорисніші документи завжди мають бути в топі (ранг 1 або 2), а не десь на 10-му місці. Запропонований підхід трансформує RAG з простого пошуковика на складну когнітивну систему, яка не просто шукає слова – вона трансформує думки (Query Rewriting), використовує різні типи пошуку, критично оцінює знайдене (Re-ranking) і змушує модель думати логічно.

Результати та їх обговорення

Ефективність запропонованої архітектури ілюструється на прикладі реальних бізнес-кейсів, де вартість помилки є високою, а обсяг даних перевищує можливості контекстного вікна стандартних моделей. Далі наведено впровадження Advanced RAG для створення спеціалізованих асистентів.

Приклад 1: фінансові аналітики витрачають години на пошук інформації серед сотень тисяч внутрішніх PDF-документів, звітів про ринки. При використанні RAG здійснюються наступні дії: створення внутрішнього чат-боту, який має доступ до понад 100000 документів банку. Як результат – система використовує векторний пошук для миттєвого знаходження відповідного розділу у звіті. Застосування гібридного пошуку тут критичне, оскільки бот повинен розрізняти схожі назви фінансових інструментів

(наприклад, "Class A Shares" та "Class B Shares"), що для чистого векторного пошуку може виглядати ідентично. Це дозволяє скоротити час пошуку інформації на 90%, надаючи відповідь з прямим посиланням на джерело.

Приклад 2: юридичні фірми працюють з тисячами контрактів та прецедентів. Використання звичайної мовної моделі LLM (наприклад, ChatGPT) є небезпечним через ризик "галюцинацій" – вигадування неіснуючих законів. При використанні Advanced RAG здійснюються наступні дії: система індексує базу законодавства та судових рішень. Як результат – ключовим елементом тут є Prompt Engineering з вимогою цитування. Бот не генерує юридичну пораду "з голови", а компілює її на основі знайдених статей кодексу. Якщо закон змінився вчора, покращена Advanced RAG-система знайде оновлення, тоді як звичайна LLM буде оперувати застарілими даними з моменту останнього тренування.

Приклад 3: користувачі ставлять технічні питання щодо API або налаштувань, використовуючи непрофесійну лексику ("побутовий опис" симптомів), тоді як документація написана складною технічною мовою та часто оновлюється, що збільшує шанс неточної відповіді. Для вирішення цієї проблеми використовується чат-бот з Advanced RAG, який має доступ не лише до документації, а й архіву виконаних задач (наприклад, з Jira). Коли користувач пише: "кнопка не працює", LLM аналізує контекст діалогу або схожі вирішені задачі в базі Jira, і формулює гіпотезу про причину збою. Вона трансформує запит у технічний формат: "помилка методу POST при збереженні форми" або "відомі помилки UI при підтвердженні дії". Таким чином пошуковий алгоритм за цим технічним запитом знаходить конкретну інструкцію для розробника, чи статтю з бази знань, яку неможливо було б знайти просто за словом "кнопка".

Приклад 4: лікарям необхідний швидкий доступ до протоколів лікування та досліджень (PubMed), обсяг яких зростає експоненційно. При обранні Advanced RAG застосовується система, що допомагає аналізувати симптоми та пропонує протоколи на основі доказової медицини. Оскільки медичні терміни дуже схожі, системі важливо відфільтрувати 50 знайдених статей та залишити лише 3 найбільш релевантні для конкретного випадку пацієнта, щоб не перевантажити лікаря зайвою інформацією.

Для оцінки ефективності запропонованого підходу було проведено порівняння трьох типів систем на наборі даних зі складних тематичних запитань (Open-Domain QA dataset) (табл. 1):

LLM – стандартна модель (наприклад, GPT-4) без доступу до зовнішніх даних;

Vanilla RAG – базова реалізація (простий векторний пошук);

Advanced RAG – запропонована в роботі система (Hybrid Search + Re-ranking + Query Rewriting).

Таблиця 1 – **Порівняння метрик якості відповідей діалогових систем**

Метрика	Опис показника	LLM (GPT-4)	Vanilla RAG	Advanced RAG
Точність фактів, %	частка тверджень, що не суперечать джерелам/реальності	65-75	85	92-95
Рівень "галюци-націй", %	частота вигадування неіснуючих фактів	~20	~10	<3
Актуальність знань, %	здатність відповідати на події, що сталися після навчання моделі	0	100	100
Релевантність відповіді, %	відповідність заданому питанню	80	75	90
Цитування джерел	можливість перевірити джерело інформації	відсутнє	присутнє (часто неточне)	точне (на рівні речення)

У сучасних системах, які побудовано на базі LLM, існує низка фундаментальних обмежень, що ускладнюють їх використання в реальних бізнес-сценаріях. Зокрема, такі моделі мають обмеження щодо актуальності знань, можуть генерувати недостовірну інформацію та демонструють нестабільну релевантність відповідей. Саме тому в практичних застосуваннях дедалі частіше використовують підхід RAG, який поєднує можливості мовної моделі з механізмами пошуку релевантної інформації. Аналіз досліджень (таблиця 1) показує, що застосування RAG та його розширених варіацій дозволяє значно підвищити точність, актуальність та надійність відповідей системи. Основні аспекти цієї проблематики можна узагальнити наступним чином:

- проблема "Knowledge Cutoff" (точність фактів): звичайна LLM нездатна повністю працювати з актуальними даними, у тематичних ботах (наприклад, новини компанії або зміна цін) використання RAG є безальтернативним, оскільки показник актуальності для чистої LLM дорівнює нулю;

- зменшення "галюцинацій": впровадження базового RAG вже знижує їх рівень удвічі (з 20% до 10%). Однак, саме використання методів Advanced RAG (зокрема Re-ranking та чіткі системні запити) дозволяє досягти критично низького рівня помилок (<3%), що робить систему придатною для використання в бізнесі;

- парадокс релевантності: цікаво відзначити, що Vanilla RAG іноді показує гіршу релевантність (75%), ніж чиста LLM (80%), що пояснюється тим, що поганий пошук може знайти "сміттєві" документи, які зіб'ють чат-бот з пантелику. Використання же підходів Query Rewriting та Hybrid Search підвищує релевантність шляхом подачі лише якісного контексту.

Таким чином можна стверджувати, що експериментальні дані підтверджують гіпотезу, що покращення та інвестування в попередню обробку запиту

та постобробку пошуку (Advanced RAG) є виправданими та забезпечують значний приріст якості наданих користувачу відповідей.

Висновки

У рамках даної роботи було проведено комплексне дослідження проблематики побудови тематичних діалогових систем на основі великих мовних моделей (LLM) з використанням архітектури Retrieval Augmented Generation (RAG). Метою роботи було не лише відтворення базового алгоритму пошуку та генерації (1), але й вдосконалення існуючого підходу шляхом розширення складових базової моделі. Запропонована модель (Advanced RAG) орієнтована на підвищення точності, релевантності та фактологічної вірності відповідей у вузькоспеціалізованих доменах. Аналіз теоретичної бази та сучасного стану розвитку NLP-технологій підтвердив, що архітектура RAG є гарним рішенням для створення професійних чат-ботів. Стандартні LLM (GPT-4, Llama 3), попри їхні високі когнітивні здібності, страждають від проблеми "Knowledge Cutoff" (відсутність актуальних знань) та схильності до "галюцинацій" при роботі з незнайомими фактами. Відокремлення бази знань від нейромережі дозволяє вирішити ці проблеми, забезпечуючи динамічне оновлення інформації без необхідності дороговартісного донавчання моделі.

У ході дослідження було виявлено, що однією з головних причин низької якості роботи стандартних RAG-систем є "семантичний розрив" між запитом користувача та текстом документів. Користувачі схильні формулювати нечіткі, короткі або контекстнозалежні запитання. Запропонований та імплементований метод Query Rewriting (переписування запиту) з використанням LLM довів свою ефективність. Трансформація вхідного питання у повноцінний пошуковий запит, насичений синонімами та уточненою термінологією, дала змогу значно підвищити якість векторних представлень, що забезпечило зростання повноти пошуку, надаючи можливість системі знаходити релевантні документи навіть тоді, коли оригінальний запит користувача не містив прямих ключових слів. Експериментально доведено, що покладання виключно на векторний пошук може бути недостатнім для тематичних чат-ботів, де важлива точність термінології (артикули, назви моделей, специфічні абревіатури), тому що векторні моделі часто "розмивають" ці деталі. Реалізація гібридного пошуку, що поєднує векторну подібність та лексичне співпадіння (BM25) з алгоритмом злиття Reciprocal Rank Fusion (RRF), дозволила досягти синергетичного ефекту.

Система демонструє високу здатність розуміти контекст (завдяки векторам) та водночас знаходити точні входження термінів (завдяки BM25), що мінімізувало кількість випадків, коли бот надавав загальну відповідь замість конкретної інструкції. Впровадження етапу переранжування (Reranking) з використанням моделей Cross-Encoder стало ключовим фактором підвищення точності (Precision). Виявлено, що первинний пошук часто повертає документи, які є семантично близькими, але фактично нерелевантними. Використання Cross-Encoder, який виступає в ролі "суворого фільтра", ефективно відсіює інформаційний шум, залишаючи для генерації лише найбільш цінні фрагменти контексту, що дозволило вирішити проблему "Lost in the Middle", коли LLM втрачає увагу через надмірну кількість вхідних даних. Дослідження також довело, що якість генерації залежить не лише від знайденого контексту, але й від інструкцій, наданих моделі. Застосування техніки Chain of Thought (CoT) дозволило структурувати "мислення" моделі. А вимога до обов'язкового цитування джерел перетворила процес генерації з "творчого написання" на "аналітичну компіляцію", що знизило рівень "галюцинацій" до <3%.

Порівняльний аналіз модифікованої архітектури (Advanced RAG) з базовою реалізацією (Vanilla RAG) та "чистою" LLM продемонстрував суттєву перевагу запропонованого підходу. Показники достовірності та релевантності відповідей зросли на 15-20%. Розроблений програмний підхід є універсальним і може бути адаптований для будь-якої предметної області: від юридичного консалтингу до технічної підтримки, що підтверджує практичну цінність роботи.

Незважаючи на досягнуті результати, тема залишається відкритою для подальшого вивчення. Перспективними напрямками розвитку є:

- GraphRAG: інтеграція графів знань (Knowledge Graphs) для покращення розуміння складних взаємозв'язків між сутностями, які важко вловити через простий текст;
- Agentic RAG: перехід від пасивної архітектури "питання-відповідь" до агентної моделі, де система може самостійно вирішувати, чи достатньо їй інформації, і за потреби виконувати додаткові запити або уточнювати деталі у користувача;
- Multimodal RAG: розширення системи для роботи не лише з текстом, але й таблицями та схемами, що важливо для технічної документації.

Підсумовуючи, можна стверджувати, що мета роботи досягнута: розроблено та обґрунтовано методологію створення високоефективного тематичного чат-боту, який вирішує основні проблеми генеративних моделей і готовий до впровадження у реальні бізнес-процеси.

Конфлікт інтересів. Автори декларують, що не мають конфлікту інтересів стосовно даного дослідження, в тому числі фінансового, особистісного характеру, авторства чи іншого характеру, що міг би вплинути на дослідження та його результати, представлені в даній статті.

Використання засобів штучного інтелекту. Автори підтверджують, що не використовували технології штучного інтелекту при створенні представленої роботи.

Список літератури

1. Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, Douwe Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *arXiv preprint arXiv:2005.11401v4*. 2021. doi: https://doi.org/10.48550/arXiv.2005.11401

2. Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, Ming-Wei Chang. REALM: Retrieval-Augmented Language Model Pre-Training. *arXiv preprint arXiv:2002.08909v1*. 2020. doi: https://doi.org/10.48550/arXiv.2002.08909
3. Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, Wen-tau Yih. Dense Passage Retrieval for Open-Domain Question Answering. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020, pp. 6769-6781. doi: https://doi.org/10.18653/v1/2020.emnlp-main.550
4. Omar Khattab, Matei Zaharia. Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. SIGIR '20: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 2020, p. 39-48. doi: https://doi.org/10.1145/3397271.3401075
5. Gautier Izacard, Edouard Grave. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume. 2021, pages 874-880. doi: https://doi.org/10.18653/v1/2021.eacl-main.74
6. Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark. Improving language models by retrieving from trillions of tokens. *arXiv preprint arXiv:2112.04426v3*. 2022. doi: https://doi.org/10.48550/arXiv.2112.04426
7. Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *arXiv preprint arXiv:2201.11903v6*. 2023. doi: https://doi.org/10.48550/arXiv.2201.11903

Received (Надійшла) 11.01.2026
Accepted for publication (Прийнята до друку) 08.04.2026
Publication date (Дата публікації) 22.05.2026

ВІДОМОСТІ ПРО АВТОРІВ / ABOUT THE AUTHORS

Івасенко Ілля Михайлович – магістрант кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна;
Illia Ivasenko – master's student of the Department of Electronic Computers, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine;
e-mail: illia.ivasenko@nure.ua; ORCID Author ID: https://orcid.org/0009-0000-2588-279X.

Філімончук Тетяна Володимирівна – кандидат технічних наук, доцент, доцент кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна;
Tetiana Filimonchuk – PhD, Associate Professor, Associate Professor of the Department of Electronic Computers, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine;
e-mail: tetiana.filimonchuk@nure.ua; ORCID Author ID: http://orcid.org/0000-0002-4380-504X;
Scopus Author ID: https://www.scopus.com/authid/detail.uri?origin=resultslist&authorId=57190949991.

Партика Станіслав Олександрович – старший викладач кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна;
Stanislav Partyka – Senior lecturer of the Department of Electronic Computers, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine;
e-mail: stanislav.partyka@nure.ua; ORCID Author ID: http://orcid.org/0000-0002-7376-8980;
Scopus Author ID: https://www.scopus.com/authid/detail.uri?origin=resultslist&authorId=57204560890.

Пивоварова Дар'я Ігорівна – асистент кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна;
Daria Pyvovarova– Assistant of the Department of Electronic Computers, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine;
e-mail: daria.pyvovarova@nure.ua; ORCID Author ID: http://orcid.org/0000-0002-7251-994X;
Scopus Author ID: https://www.scopus.com/authid/detail.uri?origin=resultslist&authorId=57224191788.

Using the rag architecture to build thematic applications

Illia Ivasenko, Tetiana Filimonchuk, Stanislav Partyka, Daria Pyvovarova

Abstract. The relevance of the study is due to the rapid development of large language models (LLM), such as GPT-4, Llama 3 and Claude, which have revolutionized the field of natural language processing (NLP). These models demonstrate exceptional abilities to generate coherent text, generalize information and conduct dialogue. However, when applying LLM in specialized domains (law, medicine, technical support, corporate knowledge bases), critical limitations arise. First, the parametric knowledge of the models is limited by the date of completion of their training (knowledge cutoff), which makes it impossible to work with current information. Second, models are prone to "hallucinations" - the generation of plausible, but actually false statements, especially when the query concerns highly specialized data that is not in the training set. The RAG architecture has become the de facto standard for solving these problems, combining the generative capabilities of LLM with accurate search in external knowledge bases. However, practice shows that the "naive" implementation of RAG (Vanilla RAG) is often insufficient for building reliable systems. Loss of context during search, inability to process complex user queries and lack of verification mechanisms lead to irrelevant answers. In this regard, the current scientific and practical task is not just the implementation of RAG, but the development and research of methods for optimizing each stage of generation. **The object of research** is the processes of information search and natural language generation in intelligent dialog systems built on the basis of large language models. **The subject of the study** is methods and algorithms for increasing the accuracy, relevance and contextual consistency of responses in RAG architecture systems, namely hybrid search, improving user queries. **Conclusions**. The practical value of the study lies in creating an architecture model that can be adapted for any subject area (from technical documentation to regulatory frameworks), while ensuring higher metric accuracy of responses compared to basic solutions.

Keywords: thematic chatbots, LLM, AI, RAG, NLP, preprocessing, query improvement, hybrid search, intelligent text generation, reranking mechanism, faithfulness, answer relevance, context precision.