ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
Навчально-науковий інститут економіки, управління, права та інформаційних технологій
Кафедра інформаційних систем та технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавр

на тему: **«Модель пошуково-підкріпленої генерації на зображеннях»**

Виконав: здобувач вищої освіти
за освітньою програмою
Інформаційні управляючі системи
спеціальності 126 Інформаційні
системи та технології
ступеня вищої освіти бакалавр
групи 126ІСТ_бд_2022
Будьомко Володимир Богданович
Керівник: Уткін Юрій Вікторович
Рецензент: Яхін Сергій Валерійович

Полтава – 2026 року

ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
Навчально-науковий інститут економіки, управління, права та інформаційних технологій
Кафедра інформаційних систем та технологій

Освітня програма Інформаційні управляючі системи
Спеціальність 126 Інформаційні системи та технології
Рівень вищої освіти перший (бакалаврський)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Юрій УТКІН
«10» червня 2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Будьомка Володимира Богдановича

1. Тема кваліфікаційної роботи:
«Модель пошуково-підкріпленої генерації на зображеннях»,
Керівник роботи: к. т. н., доцент, доцент кафедри інформаційних систем та технологій Уткін Юрій Вікторович.
Затверджено наказом закладу вищої освіти від «10» квітня 2026 року № 383-ст

2. Строк подання здобувачем вищої освіти роботи «08» червня 2026 року.

3. Вихідні дані до роботи: науково-технічна література, аналітичні джерела мережі Інтернет, що містять інформацію про архітектури LLM, VLM, RAG, підходи до побудови мультимодальних систем RAG та сучасні фреймворкі для них

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Розділ 1 Теоретичні основи пошуково-підкріпленої генерації на зображеннях
Розділ 2. Розробка моделі пошуково-підкріпленої генерації на зображеннях
Розділ 3. Рекомендації щодо реалізації пошуково-підкріпленої генерації на зображеннях

5. Перелік графічного матеріалу: схеми, рисунки, діаграми за темою та об’єктом дослідження

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання отримав
Оцінювання економічної ефективності результатів дослідження	Дивнич О. Д., к. е. н., доцент, доцент кафедри економіки та публічного управління	01.04.2026 р.	29.05.2026 р.

7. Дата видачі завдання «10» червня 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Вибір і затвердження теми роботи	02.06.2025 р. – 04.06.2025	
2	Складання і затвердження розгорнутого плану та завдання на кваліфікаційну роботу	05.06.2025 р. – 10.06.2025 р.	
3	Опрацювання джерел інформації	11.06.2025 р. – 15.06.2025 р.	
4	Збір, вивчення і обробка інформації, необхідної для виконання роботи	16.06.2025 р. – 20.06.2025 р.	
5	Виконання теоретико-методологічного розділу роботи	08.09.2025 р. – 01.11.2025 р.	
6	Виконання дослідницько-аналітичного розділу роботи	19.01.2026 р. – 14.02.2026 р.	
7	Виконання проєктно-рекомендаційного розділу роботи	16.02.2026 р. – 31.03.2026 р.	
8	Оцінювання економічної ефективності результатів дослідження	01.04.2026 р. – 29.05.2026 р.	
9	Оформлення тексту роботи	01.06.2026 р. – 06.06.2026р.	
10	Попередній захист роботи на кафедрі	08.06.2026 р.	
11	Доопрацювання роботи з урахуванням зауважень і пропозицій	09.06.2026 р. – 10.06.2026 р.	
12	Нормоконтроль	11.06.2026 р. – 15.06.2026 р.	
13	Захист кваліфікаційної роботи	16.06.2026 р. - 18.06.2026 р.	

Здобувач вищої освіти **Володимир БУДЬОМКО**

Керівник роботи **Юрій УТКІН**

ЗМІСТ

ВСТУП

Актуальність теми кваліфікаційної роботи підтверджується необхідністю підвищення точності, обґрунтованості та актуальності відповідей інтелектуальних систем, що працюють з великими обсягами корпоративної інформації. Технологія пошуково-підкріпленої генерації (RAG), дає змогу поєднати можливості мовних моделей з пошуком у зовнішніх джерелах релевантної інформації, що зменшує ризик помилкових або необґрунтованих відповідей. Проте класичні RAG-системи орієнтовані на роботу з текстом. Це створює потребу в застосуванні мультимодальної RAG для врахування візуального контексту. Однак реалізація RAG на зображеннях потребує оптимізації моделей та відповідного програмного забезпечення. Все це свідчить про актуальність теми роботи.

Метою кваліфікаційної роботи є підвищення ефективності пошуку, аналізу та генерації відповідей за зображеннями шляхом розробки моделі RAG з використанням мультимодальних підходів.

Завданнями кваліфікаційної роботи є:

– обґрунтувати вибір архітектури нейронної мережі для роботи з зображеннями;

– визначити основні підходи щодо побудови RAG на зображеннях;

– сформувати рекомендацій щодо реалізації RAG на зображеннях;

– виконати економічне обґрунтування прийнятих рішень.

Об'єктом дослідження є процес пошуково-підкріпленої генерації відповідей на основі зображень та мультимодальних даних.

Предметом дослідження є методи, моделі та програмні засоби побудови мультимодальної системи пошуково-підкріпленої генерації на зображеннях.

Методами дослідження є аналітичний – для дослідження теоретичних основ RAG, мультимодальних RAG-систем та підходів до обробки зображень; порівняльний аналіз – для зіставлення варіантів архітектури мультимодальної

RAG та методів індексування інформації; метод системного аналізу – для визначення структури моделі RAG на зображеннях, її основних компонентів і взаємозв'язків між ними; моделювання – для побудови узагальненої архітектури системи обробки зображень, створення моделі RAG на зображенні; метод узагальнення – для формування рекомендацій щодо реалізації пошуково-підкріпленої генерації на зображеннях.

Інформаційна база кваліфікаційної роботи сформована з Інтернет-ресурсів, що місять інформацію про архітектури LLM, VLM, RAG, підходи до побудови мультимодальних систем RAG та сучасні фреймворкі для них.

Практична значущість роботи полягає у розробці моделі RAG на зображеннях та рекомендацій щодо практичної реалізації мультимодальної RAG-системи, – можуть бути використані для подальших досліджень за даною тематикою та при проектуванні корпоративних RAG-систем.

Апробація результатів відбувалася в рамках XXI щорічної студентської наукової конференції «Сучасні інформаційні технології та інноваційні методики в економіці, менеджменті та бізнесі» Полтавського державного аграрного університету (22 квітня 2026 р., м. Полтава).

За результатами досліджень здійснено публікацію тез доповідей.

Структура кваліфікаційної роботи логічно пов'язана з завданнями досліджень і містить вступ, три розділи основної частини, висновки, список використаних джерел, додатки. Загальний обсяг пояснювальної записки кваліфікаційної роботи складає 58 сторінок формату А4. Вона містить 13 рисунків і 3 таблиці.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ ПОШУКОВО-ПІДКРІПЛЕНОЇ ГЕНЕРАЦІЇ НА ЗОБРАЖЕННЯХ

1.1 Обґрунтування доцільності використання мультимодальної пошуково-підкріпленої генерації

На даний час, корпоративні інтелектуальні системи використовують пошуково-підкріплену генерацію (Retrieval-Augmented Generation, RAG) [1] з метою формування відповіді великих мовних моделей (Large Language Models, LLM) [2] не лише на основі власних попередньо навчених знань, а й на основі актуальних корпоративних даних: документів, інструкцій, баз знань, звітів, регламентів, технічної документації тощо. Основне призначення RAG полягає в тому, щоб перед генерацією відповіді система спочатку виконувала пошук релевантної інформації у корпоративному сховищі даних, а потім передавала знайдені фрагменти до LLM як контекст. Завдяки цьому відповідь стає більш точною, обґрунтованою та прив’язаною до конкретних джерел, а ризик так званих «галюцинацій» моделі зменшується [3]. У корпоративному середовищі це особливо важливо, оскільки інформація часто є спеціалізованою, постійно оновлюється і може бути недоступною у відкритих навчальних даних LLM.

На практиці впровадження RAG стикається з низкою обмежень, пов’язаних із неоднорідністю реальних даних. Класична RAG добре працює з відносно структурованими текстовими документами, однак її ефективність знижується, коли інформація подана у складній формі. Одним із проблемних джерел є корпоративні wiki-сторінки зі складною розміткою та вкладеною структурою. Такі сторінки часто містять службові блоки, посилання, меню, коментарі, ієрархічні розділи та додаткові елементи навігації. Під час підготовки даних для індексації необхідно видалити інформаційний «шум», але водночас зберегти смислові зв’язки між розділами, підрозділами та

пов’язаними матеріалами. Перетворення ієрархічної структури у плоский формат, придатний для індексації у векторному сховищі, є складним завданням, оскільки при спрощенні можна втратити контекст документа. Окрему складність становить рольова модель доступу до даних. У корпоративному середовищі різні користувачі можуть мати різні права на перегляд документів, розділів або окремих фрагментів інформації. Тому RAG-система має не лише знаходити релевантні дані, а й враховувати, чи має конкретний користувач право отримати відповідь на основі цих даних. Якщо механізм доступу реалізовано неправильно, система може або не показати користувачу потрібну інформацію, або, навпаки, розкрити дані, які не повинні бути доступними. Ще однією проблемою є документи з таблицями та числовими даними. Великі мовні моделі не завжди стабільно працюють із числовими обчисленнями, агрегацією, порівнянням значень і аналізом таблиць без додаткових спеціалізованих інструментів. Наприклад, один і той самий запит до таблиці Excel може давати різні результати залежно від способу вилучення даних, формату таблиці або контексту, який був переданий до моделі. Тому для роботи з табличними даними часто потрібні додаткові модулі попередньої обробки, структурного аналізу та перевірки обчислень. Суттєві обмеження класичної RAG проявляються також під час обробки сторінок із візуальною структурою. До таких джерел належать схеми, блок-діаграми, організаційні структури, карти процесів та інші графічні представлення зв’язків між елементами. Звичайна текстова індексація не дозволяє повноцінно передати просторові та логічні відношення між об’єктами. Наприклад, після індексації організаційної структури мовна модель може плутати департаменти, підрозділи та керівників, оскільки для неї втрачаються візуальні зв’язки, які були очевидними для людини під час перегляду схеми. Особливо проблемними є PDF-документи та скановані матеріали. У таких файлах текст може бути поданий не як машинозчитуваний шар, а як частина зображення. Крім того, PDF-документи часто містять складну графіку, діаграми, таблиці, підписи,

колонки та нестандартне форматування. Базовий парсинг у таких випадках може давати спотворений результат: втрачати порядок читання, неправильно розпізнавати таблиці, пропускати підписи до зображень або змішувати різні блоки тексту. Через це якість подальшого пошуку та генерації відповіді суттєво знижується.

Таким чином, основною проблемою класичної RAG є те, що вона орієнтована переважно на текстову інформацію та не завжди здатна коректно працювати зі складними мультимодальними джерелами. Традиційні системи RAG стикаються з фундаментальним обмеженням: вони сліпі до приблизно 30% критичної інформації, що зберігається у вигляді зображень, діаграм, діаграм і таблиць. Раніше організаціям доводилося впроваджувати окремі системи та БД для різних типів даних, що не давало змоги здійснювати змішані пошукові запити. Сучасні підсистеми штучного інтелекту (Artificial Intelligence, AI) [4] вимагають багатшого контексту, і мультимодальна RAG надає його шляхом плавної інтеграції візуальної та текстової інформації для безпрецедентної точності та розуміння. Як відомо, звичайна RAG працює так: запит користувача → пошук релевантних фрагментів → передача цих фрагментів у LLM → відповідь. Мультимодальна RAG робить те саме, але джерелами знань можуть бути не лише тексти, а й: зображення; скриншоти; графіки; таблиці; сторінки PDF; діаграми; слайди; іноді аудіо або відео. Тобто система повинна вміти знайти не тільки текстовий фрагмент, а й, наприклад, потрібний графік, схему або скриншот, а потім використати його для відповіді. Мультимодальна RAG, яка може обробляти різні типи файлів (від тексту до зображень та відео), спирається на моделі вбудовування, які перетворюють дані на числові уявлення, які читають AI-моделі. Вбудовування, здатні обробляти всі види файлів, дозволяють компаніям знаходити інформацію з фінансових графіків, каталогів продукції або будь-яких інформаційних відеоматеріалів, надаючи цілісне уявлення про організацію. Щоб відповісти на питання: коли мультимодальна RAG реально потрібна, можна сформувати наступні чіткі тригери.

1. Картинки у документах дуже різні (графіки, фото, схеми, таблиці).

2. Користувач може шукати візуальну ознаку (наприклад, «як виглядає звіт» або «якого кольору індикатор»).

3. Картинка несе власний зміст, навіть якщо відірвати її від тексту.

При цьому необхідно дотримуватись кількох положень: все розгортається всередині компанії; ніяких зовнішніх послуг для чутливих даних; права доступу мають збігатися з вихідними системами; дані повинні зберігатись, а інфраструктура – бути під контролем. У міру того, як компанії починають експериментувати з мультимодальною RAG, постачальники таких мультимодальних систем радять підприємствам починати з малого, освоюючи включення зображень і відео.

1.2 Пріоритетні напрями використання мультимодальної RAG

Мультимодальна пошуково-підкріплена генерація є перспективним напрямом розвитку AI-систем, оскільки дає змогу працювати не лише з текстовою інформацією, а й із зображеннями, таблицями, схемами, діаграмами та іншими візуальними матеріалами. На відміну від класичної RAG-моделі, яка переважно здійснює пошук і генерацію відповідей на основі текстових документів, мультимодальна RAG здатна аналізувати комплексні джерела даних, де зміст подано у різних формах. Це особливо важливо для сфер, у яких значна частина інформації міститься не в суцільному тексті, а у вигляді сканованих документів, креслень, медичних зображень, графіків, таблиць або технічних схем.

Одним із пріоритетних напрямів використання мультимодальної RAG є юридична сфера та напрям відповідності нормативним вимогам. Юридичні документи часто мають складну структуру, містять не лише текстові положення, а й скановані сторінки, блок-схеми, таблиці, додатки, довідкові матеріали або візуальні дерева прийняття рішень. У таких випадках звичайна

текстова система пошуку може втратити частину важливої інформації, оскільки вона не аналізує візуальні елементи документа. Мультимодальна RAG дає змогу автоматизувати перевірку відповідності, поєднуючи аналіз текстових норм із розпізнаванням і тлумаченням схем, таблиць та інших графічних компонентів. Наприклад, така система може допомогти визначити, чи відповідає певний документ встановленим вимогам, використовуючи як текстові фрагменти, так і візуальні пояснення процесів.

Важливим напрямом застосування мультимодальної RAG є охорона здоров'я та медичні дослідження. Медична документація часто поєднує текстові описи стану пацієнта, результати лабораторних аналізів, таблиці показників, рентгенівські знімки, МРТ, КТ, ультразвукові зображення або анатомічні схеми. У цьому контексті мультимодальна RAG може бути використана для комплексного аналізу клінічного випадку. Система здатна зіставляти текстові симптоми та діагностичні висновки з візуальними ознаками, що містяться на медичних зображеннях. Це може підвищити якість інформаційної підтримки лікаря, полегшити пошук релевантних даних у медичній документації та допомогти швидше отримати узагальнену відповідь на складний запит. Водночас у медичній сфері такі системи мають використовуватися як допоміжний інструмент, а остаточне рішення повинно залишатися за фахівцем.

Ще одним перспективним напрямом є виробництво та інженерія. У технічній документації часто поєднуються текстові описи, специфікації, креслення, схеми складання, каталоги деталей, інструкції з експлуатації та ремонту. Для фахівця важливо швидко знаходити не лише текстову інформацію, а й пов'язані з нею візуальні матеріали. Мультимодальна RAG може використовуватися для пошуку відповідей на технічні запитання з урахуванням як описових фрагментів, так і інженерних креслень або схем. Наприклад, система може відповісти на запит щодо параметрів компонента, умов монтажу, послідовності складання або технічних обмежень, посилаючись одночасно на інструкцію та відповідну візуальну схему. Це

робить мультимодальну RAG корисною для технічної підтримки, навчання персоналу, обслуговування обладнання та аналізу складної інженерної документації.

Окрему роль мультимодальна RAG відіграє у фінансовому аналізі. Фінансові та інвестиційні звіти зазвичай містять не лише текстові коментарі, а й графіки динаміки показників, діаграми ефективності, порівняльні таблиці, інфографіку та візуалізації тенденцій. Класичний текстовий аналіз не завжди дає змогу коректно врахувати інформацію, представлену у графічній формі. Мультимодальна RAG може аналізувати фінансові документи комплексно, поєднуючи текстові пояснення з даними, що містяться у таблицях і діаграмах. Наприклад, користувач може поставити запитання щодо порівняння доходу за певний квартал із прогнозними значеннями, а система сформує відповідь на основі текстового аналізу звіту та даних із відповідного графіка або таблиці.

Таким чином, пріоритетні напрями використання мультимодальної RAG охоплюють ті сфери, де інформація має змішаний характер і представлена одночасно у текстовому та візуальному вигляді.

1.3 Архітектура мультимодальної системи пошуково-підкріпленої генерації

Архітектура мультимодальної системи RAG має забезпечувати обробку різнорідних джерел інформації, зокрема тексту, зображень, таблиць, схем, діаграм та інших візуальних елементів. На відміну від класичної RAG-системи, яка переважно працює з текстовими фрагментами, мультимодальна RAG повинна мати додаткові компоненти для вилучення, інтерпретації та подальшого використання візуального контенту. Саме тому її архітектура зазвичай будується як багатоступеневий конвеєр, у якому кожен етап відповідає за окрему частину обробки даних: від аналізу документа до генерації відповіді на запит користувача.

У загальному вигляді архітектурний стек мультимодальної системи RAG можна подати як 4-тупеневий процес (рис. 1.1). Він включає обробку документів, аналіз зображень за допомогою vision-language моделей (VLM) [5], створення уніфікованих векторних представлень та інтелектуальний пошук релевантної інформації. Такий підхід дозволяє перетворити різні типи медіа на єдиний формат знань, придатний для пошуку, порівняння та подальшої генерації відповіді мовною моделлю.

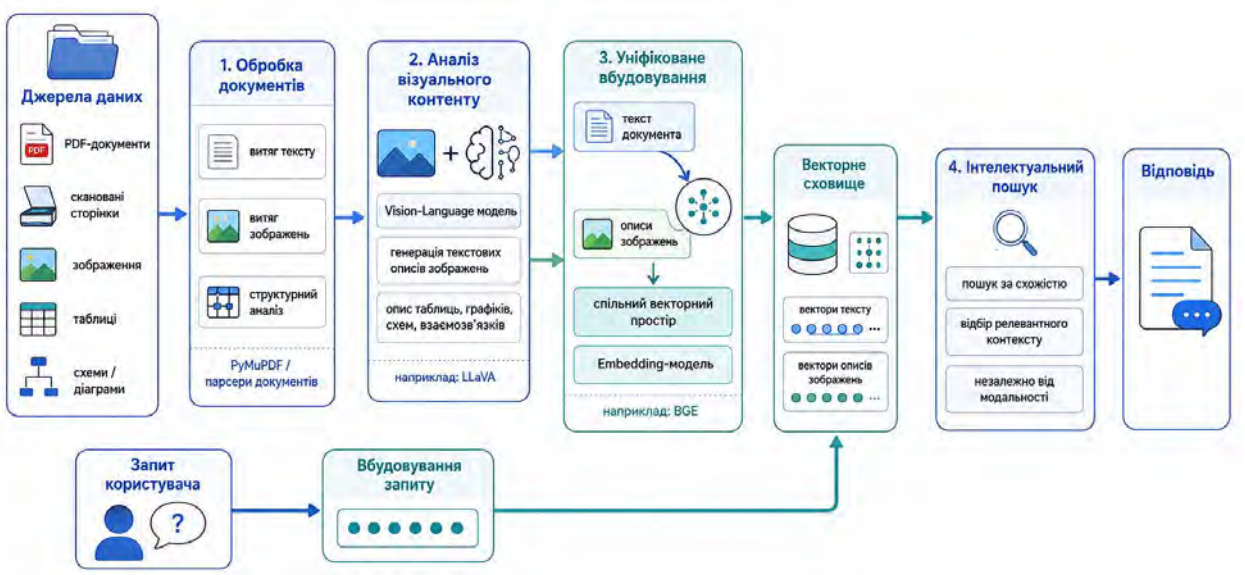

Рисунок 1.1 – Архітектура мультимодальної системи RAG

Першим етапом є обробка документів. На цьому рівні система приймає вхідні файли, наприклад PDF-документи, технічні інструкції, звіти, презентації або скановані матеріали, і виконує їх структурний аналіз. Для цього можуть використовуватися спеціалізовані інструменти парсингу документів, зокрема PyMuPDF [6]. Такі засоби дозволяють виділяти з документа не лише текстові блоки, а й вбудовані зображення, таблиці, схеми та інші графічні елементи. Наприклад, у PDF-документах методи на зразок page.get_images(full=True) можуть застосовуватися для виявлення зображень на сторінці, а extract_image(xref) – для вилучення відповідних графічних об’єктів у вигляді двійкових даних. Це створює основу для подальшої обробки як текстової, так і візуальної інформації.

Другим етапом є інтерпретація візуального контенту за допомогою моделей типу Vision-Language. Такі моделі поєднують можливості комп'ютерного зору та обробки природної мови. Їх завдання полягає не тільки у розпізнанні об'єкту на зображенні, а й формуванні змістовного текстового опису цього зображення. Наприклад, модель LLaVA-1.5-7B може аналізувати вилучені з документа зображення та генерувати розгорнуті описи діаграм, таблиць, схем, графіків або взаємозв'язків між елементами [7]. Важливо, що такі описи не є простими короткими підписами до зображень. Вони можуть містити пояснення структури, логіки, призначення елементів та контексту їх використання. Завдяки цьому візуальна інформація перетворюється на текстову форму, придатну для семантичного пошуку.

Третім етапом є уніфіковане вбудовування даних. Після вилучення тексту з документів і створення текстових описів зображень система повинна представити всю інформацію у спільному векторному просторі. Для цього як звичайні текстові фрагменти, так і згенеровані описи зображень передаються до однієї моделі вбудовування. Прикладом може бути BAAI/bge-en-icl [8] або інша embedding-модель, призначена для побудови семантичних векторних представлень. У результаті кожен фрагмент документа отримує числове векторне представлення, яке відображає його зміст. Основна перевага цього підходу полягає в тому, що текстова та візуальна інформація починають співіснувати в одному семантичному просторі. Тобто система може однаково ефективно шукати як фрагменти тексту, так і інформацію, яка спочатку була подана у вигляді зображення.

Четвертим етапом є інтелектуальний пошук релевантної інформації. Коли користувач ставить запитання до системи, його запит також перетворюється на векторне представлення. Далі система виконує пошук за схожістю у векторному сховищі, порівнюючи запит із раніше збереженими векторними представленнями текстових фрагментів та описів зображень. Завдяки цьому пошуковий механізм може знайти найбільш релевантний контент незалежно від того, у якій формі він був поданий у початковому

документі. Наприклад, відповідь на запит користувача може бути знайдена не у звичайному текстовому абзаці, а в описі схеми, таблиці або графіка, який був попередньо проаналізований VLM.

Після етапу пошуку відібрані релевантні фрагменти передаються до великої мовної моделі, яка формує кінцеву відповідь. При цьому генерація відповіді ґрунтується не лише на внутрішніх знаннях моделі, а й на знайденому контексті з документа. Це знижує ризик некоректних або вигаданих відповідей і дозволяє прив’язати результат до конкретних джерел інформації. У випадку мультимодальної RAG таким джерелом може бути як текстовий фрагмент, так і зміст зображення, перетворений у текстовий опис.

Архітектурний стек мультимодальної RAG-системи можна описати як поєднання кількох ключових компонентів: модуля обробки документів, модуля аналізу зображень, embedding-моделі, векторного сховища, механізму семантичного пошуку та LLM для генерації відповіді. Така архітектура дозволяє створювати системи, здатні працювати зі складними документами, у яких важлива інформація міститься не лише в тексті, а й у візуальних елементах. Для теми пошуково-підкріпленої генерації на зображеннях це суттєво, оскільки саме здатність поєднувати текстові та візуальні джерела даних є основою ефективної мультимодальної RAG.

1.4 Vision Transformer як основа обробки зображень у мультимодальних системах

VLM як і LLM будуються на основі архітектури transformer [9]. Це модель глибокого навчання на основі механізму самоуваги (self-attention), який по-різному зважує важливість кожної частини вхідних даних. Вони використовуються для обробки природної мови (Natural Language Processing, NLP) [10], комп'ютерного зору (Computer Vision, CV) [11]. До ключових переваги трансформерів відносяться: паралелізація (обробка всіх елементів

послідовності одночасно); довгострокові залежності (ефективне моделювання зв'язків між віддаленими елементами); масштабованість (навчання на великих датасетах); універсальність (застосовність до різних типів даних). До цього часу у галузі CV досить актуальними були згорткові нейронні мережі (Convolutional Neural Network, CNN) [12] – рис. 1.2. Згортки популярні через їх ефективність та масштабованість під час навчання з використанням графічних процесорів. Так само, як 2D-згортки можуть отримувати ознаки із зображення, ці моделі використовують 1D-фільтри для вилучення інформації з текстів, які представлені у вигляді 1D-послідовності. Однак CNN мають кілька обмежень.

1. Обмежене рецептивне поле – складність захоплення довгострокових залежностей.

2. Фіксовані фільтри – неадаптивність до різних контекстів.

3. Ієрархічна обробка – необхідність глибоких мереж для великих рецептивних полів.

4. Локальність – фокус на локальних патернах, а чи не глобальних зв'язках.

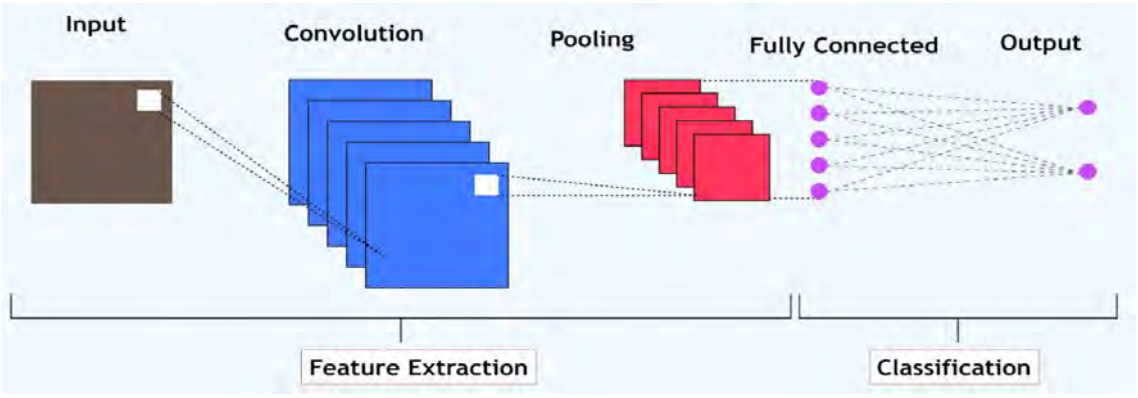

Рисунок 1.2 – Узагальнена архітектура CNN

Рецептивне поле цих типів CNN залежить від розміру їх фільтрів і кількості використовуваних згорткових шарів. Збільшення значення цих гіперпараметрів збільшує складність моделі, що може призвести до зникнення градієнтів і навіть неможливості навчання моделей.

Те, що трансформери добре показали себе в мовних моделях, не гарантувало їхнього успіху в інших сферах. Для цього необхідно було створити Vision Transformers (ViT), використовуючи архітектуру енкодера трансформера з найменшою кількістю можливих модифікацій, та застосувати його до завдань, наприклад, класифікації зображень (табл. 1.1).

Таблиця 1.1 – Порівняння ViT і CNN

Характеристики	ViT	CNN
Витяг ознак	Використовує механізм самоуваги між патчами зображення	Використовує згорткові фільтри для витягу локальних ознак
Глобальний контекст	Враховує глобальні зв’язки між патчами вже на одному шарі	Потребує глибших шарів для формування глобального розуміння на основі локальних ознак
Обробка зображень	Розділяє зображення на неперекривні патчі	Сканує зображення за допомогою перекривних рецептивних полів
Позиційне кодування	Потребує явного позиційного кодування для врахування розташування патчів	Неявно кодує позицію завдяки згортковій структурі
Складність моделі	Має вищу обчислювальну вартість через використання механізму уваги	Зазвичай є менш обчислювально затратною порівняно з ViT
Вимоги до даних	Потребує великих наборів даних для досягнення високої якості роботи	Добре працює як на малих, так і на великих наборах даних
Масштабованість	Добре масштабується зі збільшенням обсягу даних і розміру моделі	Також добре масштабується, але може бути схильною до перенавчання на малих наборах даних
Індуктивні зміщення	Має мінімальні індуктивні зміщення та вивчає зв’язки безпосередньо з даних	Має сильні індуктивні зміщення до локальних ознак, зокрема країв і текстур
Продуктивність на малих даних	Схильна до перенавчання на малих наборах даних	Краще працює на менших наборах даних завдяки вбудованим індуктивним зміщенням
Інтерпретованість	Складніша для інтерпретації через особливості механізму уваги	Легше інтерпретується через навчені фільтри

ViT (рис. 1.3) навчаються на досить великих обсягах даних (> 100M зображень) з набагато меншими обчислювальними ресурсами (вчетверо менше) ніж сучасні CNN (ResNet), і при перенесенні на кілька середніх або невеликих тестових наборів для розпізнавання зображень досягають

відмінних результатів. Щоб краще зрозуміти архітектуру, давайте розділимо її на 3 основні компоненти: вбудовування (Embedding), енкодер трансформера, голова MLP.

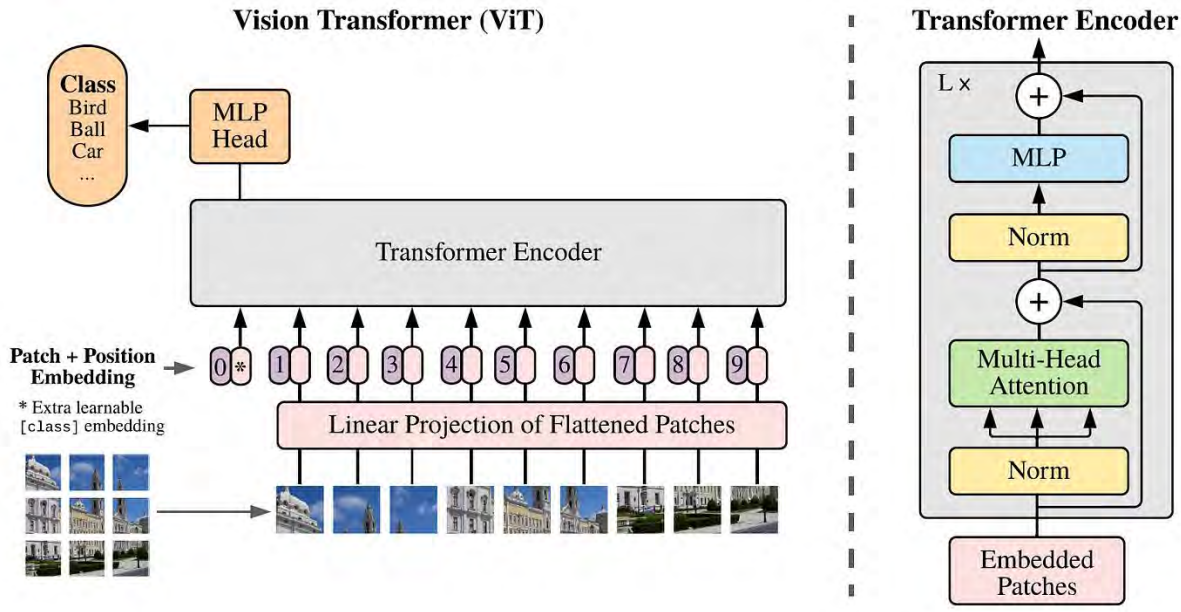

Рисунок 1.3 – Архітектура Vision Transformer

На етапі вбудовування поділяємо вхідне зображення на ділянки (patches) фіксованого розміру та лінійно згладжуємо їх шляхом об'єднання каналів. Зображення ділиться на квадратні ділянки, що не перекриваються. Кожен patch перетворюється на вектор фіксованої розмірності. Потім додається спеціальний токен для агрегації інформації. Нарешті, виконується позиційне кодування (збереження інформації про просторове розташування). Останній крок обумовлений наступним. Трансформери не здатні запам'ятовувати порядок чи послідовність входів. Якщо patches зображення переупорядковуються, значення вихідного зображення втрачається. Отже, додаємо позиційне вбудовування до лінійно вбудованих patches зображення, щоб відстежувати послідовність. Енкодер трансформера складається з кількох стеків однакових блоків (рис. 1.4).

У моделі ViT після обробки зображення трансформерними блоками формується вектор контексту C, який містить представлення всіх токенів. Для

задачі класифікації використовується не весь набір токенів, а спеціальний контекстний токен C_0, який накопичує узагальнену інформацію про все зображення. Саме цей токен передається до класифікаційної голови MLP, яка формує кінцеве передбачення класу.

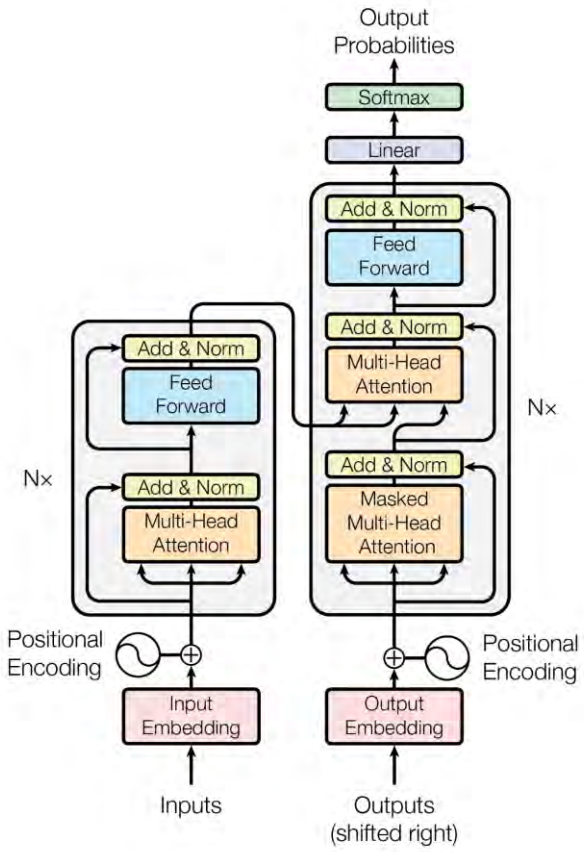

Рисунок 1.4 – Архітектура блоку енкодера трансформера

Голова MLP виконує перетворення отриманого векторного представлення у набір значень, що відповідають можливим класам. Спочатку до токена C_0 застосовується нормалізація шару, яка стабілізує значення ознак. Далі лінійний шар перетворює вектор розмірності D у вектор розмірності num_classes, де кожен елемент відповідає певному класу. На завершальному етапі може застосовуватися функція Softmax [13], яка перетворює ці значення на ймовірності належності зображення до кожного з класів. Таким чином, MLP-голова виконує роль класифікатора, який на основі узагальненого представлення зображення визначає найбільш імовірний клас об’єкта.

РОЗДІЛ 2
РОЗРОБКА МОДЕЛІ ПОШУКОВО-ПІДКРІПЛЕНОЇ ГЕНЕРАЦІЇ НА ЗОБРАЖЕННЯХ

2.1 Варіанти архітектури пошуково-підкріпленої генерації

Для реалізації системи RAG можна використовувати два підходи.

1. Векторна RAG [14]. В ній документи діляться на фрагменти – chunks. Кожен фрагмент перетворюється на числовий вектор – embedding. Потім запит користувача також перетворюється на вектор, і система шукає найближчі за змістом фрагменти у векторній базі. Далі ці фрагменти передаються у LLM як контекст. Такий підхід описується як стандартний RAG-процес: ingestion → retrieval → generation, де retrieval знаходить семантично схожі документи за допомогою vector search. Тобто, головна ідея – знайти текст, який за змістом найближчий до запиту (рис. 2.1). Наприклад, якщо користувач питає: «Які метрики використовують для оцінювання мультимодального RAG?», система знайде фрагменти, де є слова або близькі за змістом поняття: Recall@k, Precision@k, MRR, NDCG, faithfulness, groundedness, answer relevance тощо.

2. Графова RAG [15]. У ній система не лише зберігає фрагменти тексту, а й будує граф знань: сутності, поняття, об’єкти та зв’язки між ними. Microsoft GraphRAG описує цей підхід як структурований ієрархічний варіант RAG, де з тексту витягується knowledge graph, будуються спільноти/кластери сутностей, створюються їхні резюме, а потім ці структури використовуються під час генерації відповіді (рис. 2.2) [16]. Головна суть – знайти не тільки схожий текст, а й зв’язки між поняттями. Наприклад, у документах є наступні факти:

1. RAG використовує пошук.
2. Векторний пошук базується на embeddings.
3. Recall@k оцінює якість пошуку.

4. GraphRAG використовує knowledge graph.
5. Knowledge graph зберігає сутності та зв’язки.

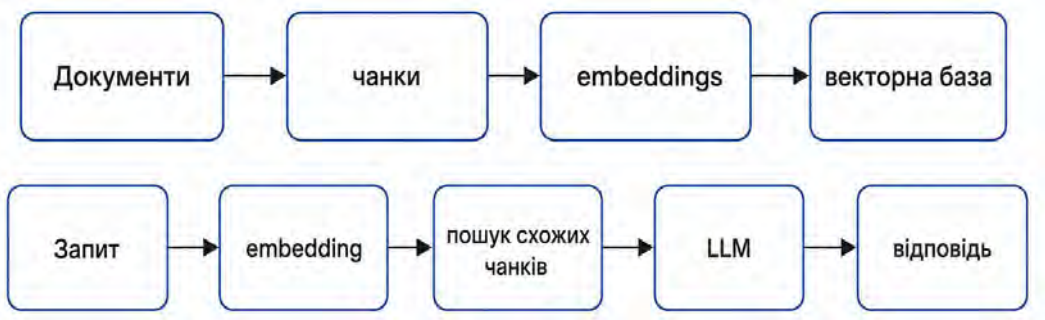

Рисунок 2.1 – Схема роботи векторної RAG

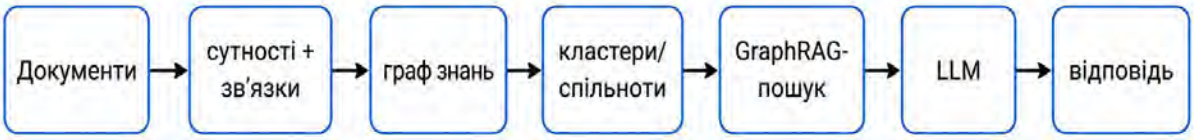

Рисунок 2.2 – Схема роботи графової RAG

Тоді графова RAG може побудувати наступні зв’язки:

– RAG → має компонент → retrieval;
– retrieval → оцінюється метрикою → Recall@k;
– GraphRAG → використовує → knowledge graph;
– knowledge graph → містить → сутності;
– knowledge graph → містить → зв’язки.

По суті, векторна і графова RAG мають одну мету: дати LLM зовнішні знання, щоб відповідь була не лише з «пам’яті моделі», а з конкретної бази документів. Але вони по-різному організовують ці знання (табл. 2.1). Векторна RAG працює приблизно так: «Знайди фрагменти тексту, які найбільше схожі на запит користувача». Графова RAG працює так: «Знайди сутності, зв’язки між ними, пов’язані поняття і на основі цього побудуй відповідь». Тобто векторна RAG краще відповідає на питання типу: «Де в документах сказано про X?» А графова RAG краще підходить для питань типу: «Як X пов’язаний з Y і які наслідки це має?»

Таблиця 2.1 – Порівняння векторної та графової RAG

Ознака	Векторна RAG	Графова RAG
Основна одиниця знань	Текстовий фрагмент / chunk	Сутність, зв’язок, спільнота сутностей
Як шукає інформацію	За семантичною схожістю embeddings	Через зв’язки між об’єктами у графі
Що добре знаходить	Локальні відповіді з конкретного фрагмента	Складні відповіді, де треба «з’єднати факти»
Типова база	Vector DB: FAISS, Chroma, Qdrant, Milvus, MongoDB Vector Search	Graph DB / knowledge graph: Neo4j, NetworkX, RDF-графи, GraphRAG pipeline
Складність реалізації	Нижча	Вища
Вартість індексації	Зазвичай менша	Часто більша (треба витягувати сутності, зв’язки, будувати граф і резюме)
Типові проблеми	Може знайти схожий, але неповний chunk	Може помилятися при витягуванні сутностей і зв’язків
Найкраще підходить для	FAQ, пошуку по документації, простих питань	Аналітики, досліджень, великих корпусів, зв’язків між поняттями

Векторна RAG – це пошук за змістовою схожістю. Вона проста, швидка й добре підходить для більшості базових задач. Векторна RAG може погано працювати, коли потрібно «з’єднувати точки» між різними частинами інформації або робити узагальнення за великим набором документів. Графова RAG якраз орієнтований на такі складніші запити. Векторну RAG доцільно використовувати, якщо потрібно швидко зробити пошук по документах, PDF, статтях, інструкціях, базі знань або FAQ. Це простіший і більш практичний стартовий варіант. Векторний пошук добре підходить для пошуку релевантних документів за змістовою близькістю. Графову RAG доцільно використовувати, якщо документи великі, між поняттями багато зв’язків, треба аналізувати структуру знань, робити висновки з різних джерел або відповідати на складні питання. Але такий підхід складніший і дорожчий: навіть у репозиторії Microsoft GraphRAG є попередження, що індексація може бути дорогою операцією. Графова RAG – це пошук і генерація на основі структури знань. Вона краще підходить для складних питань, де потрібно враховувати зв’язки між сутностями, поняттями та документами. На практиці найкращим рішенням часто є гібридний підхід: векторна RAG

використовується для швидкого пошуку релевантних фрагментів, а графова RAG – для складного аналізу зв'язків між поняттями.

2.2 Систематизація підходів до побудови мультимодальних систем

Надалі варто визначити основні підходи до побудови мультимодальної RAG. На даний час, виділяють наступні підходи.

CLIP-style – мультимодальні embeddings + передача сирих зображень у мультимодальну LLM [17]. У цьому підході зображення не перетворюється попередньо в текстовий опис. Замість цього система одразу створює для нього мультимодальне вбудовування – embedding. Тобто і текст, і зображення переводяться в один спільний векторний простір (рис. 2.3).

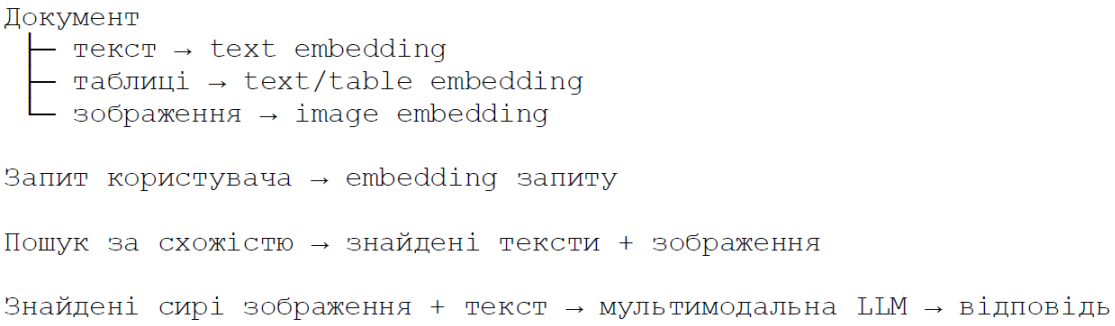

Рисунок 2.3 – CLIP-style

Для такого підходу використовуються моделі на кшталт CLIP (OpenCLIP), які можуть представляти текст і зображення у спільному векторному просторі [18]. Уявімо, що в базі є PDF-документ про локального AI-агента. У ньому є: текстовий опис архітектури; таблиця з моделями; схема «структура локальної системи»; графік продуктивності. Користувач питає: «Який модуль відповідає за генерацію аудіо»? При цьому система робить наступне.

1. Перетворює запит у embedding.
2. Порівнює його з embeddings текстів і зображень.

3. Може знайти не лише текстовий фрагмент, а й схему системи.
4. Передає цю схему як зображення до мультимодальної LLM.
5. LLM дивиться на схему і формує відповідь.

Саме зображення зазвичай не зберігають безпосередньо у векторній БД. Там зберігається embedding і посилання на файл. Сире зображення лежить окремо – у файловому сховищі, docstore, object storage або локальній папці. Головна перевага – система працює з зображенням як із повноцінним джерелом інформації. Якщо на зображенні є: схема; графік; фотографія; компоненти інтерфейсу; просторове розташування об’єктів; візуальні залежності, то мультимодальна LLM може безпосередньо їх проаналізувати. Цей варіант добре підходить, коли відповідь залежить саме від візуальних деталей, наприклад: «Що показано на графіку»; «Який блок на схемі з’єднаний із модулем пам’яті «Який елемент інтерфейсу розташований у правому верхньому куті». Проблема в тому, що мультимодальні embeddings не завжди добре передають складний зміст зображення. Наприклад, якщо на графіку є підписи осей, дрібні числа, технічні позначення або таблиця всередині скриншота, embedding може відобразити загальний зміст, але втратити точні деталі. Ще один недолік – потрібна мультимодальна LLM на етапі відповіді. Тобто звичайної текстової LLM недостатньо, бо в prompt передаються зображення. Цей варіант варто використовувати, якщо: у документах багато схем, графіків, креслень, скриншотів; потрібно відповідати на питання, де важливі візуальні деталі; є доступ до мультимодальної LLM; система має працювати не лише з текстом, а й із реальним візуальним змістом. Для локального агента цей варіант можливий, але складніший, бо потрібні локальні мультимодальні моделі або зовнішній API. Таким чином, запит користувача перетворюється на вектор і система шукає схожі вектори-картинки. Це простий та дешевий підхід, що пішов від моделі [19]. Недолік полягає у тому, що нейронна мережа розуміє картинку дуже грубо та втрачає дрібні деталі. Сучасними прикладами CLIP-style є Cohere Embed Multimodal [20], Jina CLIP [21], ImageBind [22].

2. ColPali [23]. У цьому підході зображення спочатку аналізується мультимодальною моделлю, яка створює текстове резюме зображення. Після цього система працює вже не з оригінальним зображенням, а з його текстовим описом (рис. 2.4). Це дорожче, але розкішно працює на складних документах: діаграмах, таблицях, графіках [24]. Тобто на етапі генерації відповіді LLM отримує не зображення, а тільки текст. Згідно [25], цей варіант як підхід, де мультимодальна LLM спочатку створює текстові резюме для зображень, а далі ці резюме індексуються та використовуються для пошуку текстовою embedding-моделлю. Розглянемо наступний приклад. Є зображення з схемою локальної TTS-системи. Мультимодальна LLM заздалегідь створює опис – рис. 2.5.

```
Зображення → мультимодальна LLM → текстовий опис / резюме

Текст документа → embedding
Таблиця → текстове резюме → embedding
Резюме зображення → embedding

Запит користувача → embedding

Пошук → знайдені текстові фрагменти

Текстові фрагменти → звичайна LLM → відповідь
```

Рисунок 2.4 – Схема ColPali

```
На схемі показано локальну TTS-систему. Вхідним елементом є текст
користувача. Далі текст проходить попередню обробку, нормалізацію,
токенізацію. Після цього він передається до TTS-моделі, яка генерує аудіо.
Результат зберігається у WAV-файл і може бути використаний локальним AI-
агентом.
```

Рисунок 2.5 – Опис від мультимодальної LLM

Цей текстовий опис зберігається у векторній БД. Коли користувач питає: «Які етапи має локальна TTS-система», система знаходить цей текстовий опис і передає його у звичайну LLM. Але в самому процесі відповіді оригінальне зображення може взагалі не використовуватися. Цей

варіант простіший для реалізації. Після етапу попередньої обробки вся база стає текстовою: текст документа; текстові резюме таблиць; текстові описи зображень; текстові описи графіків. Це означає, що можна використовувати звичайну текстову LLM. Не обов'язково мати мультимодальну модель на етапі відповіді. Це дуже зручно для локальних RAG-систем, де, наприклад, використовується: локальна embedding-модель; Chroma [26], FAISS [27], Milvus [28] або Qdrant [29]; локальна LLM через Ollama [30], LM Studio [C31], KoboldCpp [C32] або llama.cpp [C33]. Також цей підхід часто краще працює для семантичного пошуку, бо текстове резюме може прямо містити потрібні слова. Наприклад, якщо на схемі зображено «модуль генерації аудіо», то в резюме можна явно записати: «модуль генерації аудіо». Головний мінус – це втрата частини візуальної інформації. Якщо резюме було неточним або неповним, система не зможе відновити пропущені деталі. Наприклад, на графіку є: точні числові значення; кольорові лінії; підписи осей; дрібні позначення; взаємне розташування елементів. Якщо мультимодальна модель описала графік занадто загально, то RAG-система матиме тільки цей загальний опис. Наприклад, замість: «На графіку модель Piper TTS має затримку 120 мс, а Supertonic TTS – 250 мс», модель може створити слабке резюме: «На графіку порівнюється продуктивність двох TTS-моделей». Тоді точна відповідь буде неможливою. Цей варіант варто використовувати, якщо: відповідь може будуватися на текстових описах; зображення не потрібно повторно аналізувати під час відповіді; потрібно зменшити вартість і складність системи; немає мультимодальної LLM для генерації відповіді; система має працювати локально; важлива швидкість.

3. VLM-as-captioner [34]. Це гібридний підхід. Він поєднує переваги другого і першого варіантів. На етапі пошуку система використовує текстове резюме зображення, бо за текстом легше шукати. Але після того, як потрібне зображення знайдено, у мультимодальну LLM передається вже оригінальне зображення, а не тільки його опис (рис. 2.6). Згідно [25], такий варіант описується як підхід, де текстові резюме використовуються для retrieval, але

для синтезу відповіді передаються raw images і текстові фрагменти. Розглянемо простий приклад. Є графік, який показує залежність швидкодії TTS-моделей від розміру моделі.

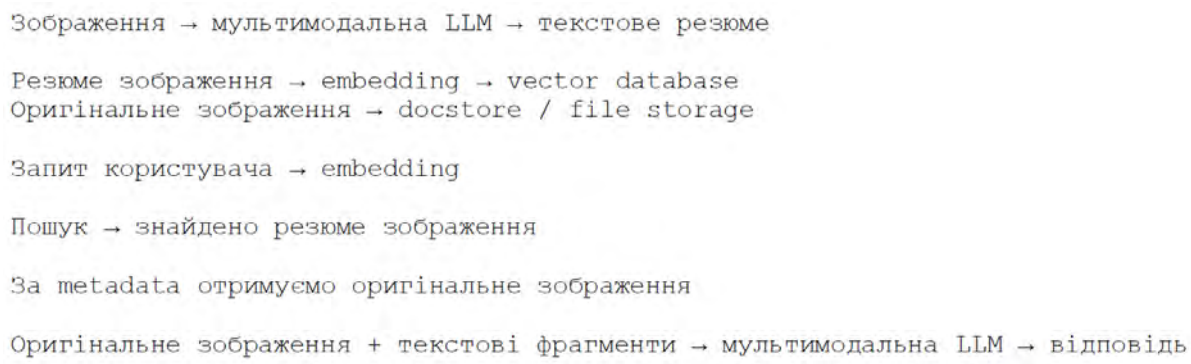

Рисунок 2.6 – Схема VLM-as-captioner

Під час індексації система створює резюме: «Графік порівнює швидкість генерації аудіо для Piper TTS, Supertonic TTS і Qwen3-TTS залежно від розміру моделі». Користувач питає: «Яка модель має найменшу затримку на графіку»? Тоді система робить наступне.

1. Шукає за текстовим резюме.
2. Розуміє, що потрібний саме цей графік.
3. Дістає оригінальне зображення графіку.
4. Передає його у мультимодальну LLM.
5. Модель дивиться на сам графік і відповідає точніше.

Це часто найсильніший підхід, який має дві важливі переваги. По-перше, пошук виконується по текстовому опису, тому система краще знаходить релевантні зображення за змістом. По-друге, відповідь генерується на основі оригінального зображення, тому мультимодальна LLM може перевірити деталі: числа, підписи, структуру, взаємне розташування блоків. LangChain у benchmark-прикладах показує, що пошук за summaries не потребує мультимодальних embeddings і може краще працювати для візуально або семантично схожого, але кількісно різного контенту, наприклад слайдів або графіків [35]. При цьому, недоліком є більша складність. Тобто

потрібно мати: мультимодальну модель для створення резюме; текстову embedding-модель; векторну БД; окреме сховище оригінальних зображень; мультимодальну LLM для генерації відповіді. Він може бути дорожчим або повільнішим, бо на етапі відповіді мультимодальна LLM обробляє зображення. Цей варіант використовують якщо: у документах багато важливих схем, графіків, креслень або діаграм; потрібно давати точні відповіді по зображеннях; текстового опису може бути недостатньо; є доступ до мультимодальної LLM; система має бути якіснішою, навіть якщо вона складніша. Для корпоративної мультимодальної RAG-системи це часто найкращий компроміс. Результати порівняння розглянутих варіантів узагальнено у табл. 2.2.

Таблиця 2.2 – Узагальнення варіантів побудови мультимодальних RAG-систем

Ознака	Варіант 1	Варіант 2	Варіант 3
Як шукаються зображення	Через мультимодальні embeddings	Через текстові summaries	Через текстові summaries
Що передається в LLM	Текст + сирі зображення	Тільки текст	Текст + сирі зображення
Чи потрібна мультимодальна LLM для відповіді	Так	Ні	Так
Чи потрібні image embeddings	Так	Ні	Ні
Чи зберігається оригінальне зображення	Так	Може зберігатися, але не обов’язково використовується	Так
Складність реалізації	Середня/висока	Найнижча	Найвища
Точність по візуальних деталях	Добра	Обмежена summary	Найкраща
Підходить для локальної системи	Можливо, але складно	Найкраще	Можливо, але вимогливо

Таким чином, найпростіший для реалізації – варіант 2; найточніший для справжнього аналізу зображень – варіант 3; найпряміший мультимодальний підхід – варіант 1.

2.3 Використання фреймворку RAG-Anything для побудови моделі пошуково-підкріпленої генерації на зображеннях

Побудова моделі RAG на зображеннях потребує такого інструментарію, який здатний працювати не лише з текстовими фрагментами, а й з візуальними об’єктами, таблицями, формулами, схемами та іншими елементами документа. Звичайна RAG-система зазвичай орієнтована на текст: документ розбивається на фрагменти, ці фрагменти перетворюються на векторні подання, після чого за запитом користувача виконується пошук найбільш релевантних частин тексту. Такий підхід є ефективним для текстових джерел, але він втрачає значну частину інформації, коли у документі присутні зображення, діаграми, графіки або скріншоти. Саме тому для теми цієї роботи доцільно розглядати мультимодальний підхід, у якому зображення стає повноцінним джерелом знань, а не лише допоміжним ілюстративним матеріалом. RAG-Anything є фреймворком для побудови мультимодальних RAG-систем, який орієнтований на наскрізну обробку документів із різнорідним вмістом.

З погляду архітектури, RAG-Anything можна розглядати як розширений варіант мультимодального RAG, у якому зберігається зв’язок між текстовим описом візуального об’єкта та його оригінальним зображенням [36]. Це наближає його до підходу, за якого система спочатку створює текстове резюме зображення, індексує його, але під час відповіді може звертатися також до початкового мультимодального об’єкта. Водночас RAG-Anything виходить за межі простого опису зображень, оскільки використовує основу LightRAG і підтримує графово-векторний пошук. Це дозволяє враховувати не лише семантичну схожість між запитом і фрагментом документа, а й структурні зв’язки між текстом, таблицями, зображеннями та іншими елементами.

Фреймворк подається як універсальне рішення для мультимодального опрацювання документів, побудоване на основі LightRAG [37]. Основна ідея

полягає в тому, щоб об’єднати в межах одного процесу розпізнавання структури документа, аналіз текстових і візуальних елементів, формування семантичних зв’язків між ними, індексацію та подальше отримання релевантної інформації для генерації відповіді. Завдяки цьому користувач може ставити запитання до документів, у яких одночасно містяться текст, зображення, таблиці та математичні вирази, а система має враховувати не ізольовані фрагменти, а контекстні зв’язки між різними типами даних.

RAG-Anything не обмежується простим збереженням файлів або векторизацією тексту. Фреймворк перетворює візуальні елементи на структуровані одиниці знань: для зображень можуть формуватися підписи, метадані, опис візуального змісту, а також зв’язки із сусідніми текстовими блоками документа. Це дає змогу виконувати пошук не лише за прямими збігами у тексті, а й за змістом зображення, його роллю в документі та взаємозв’язком із розділом, таблицею або пояснювальним текстом. Узагальнену архітектуру наведено на рис. 2.7 [38].

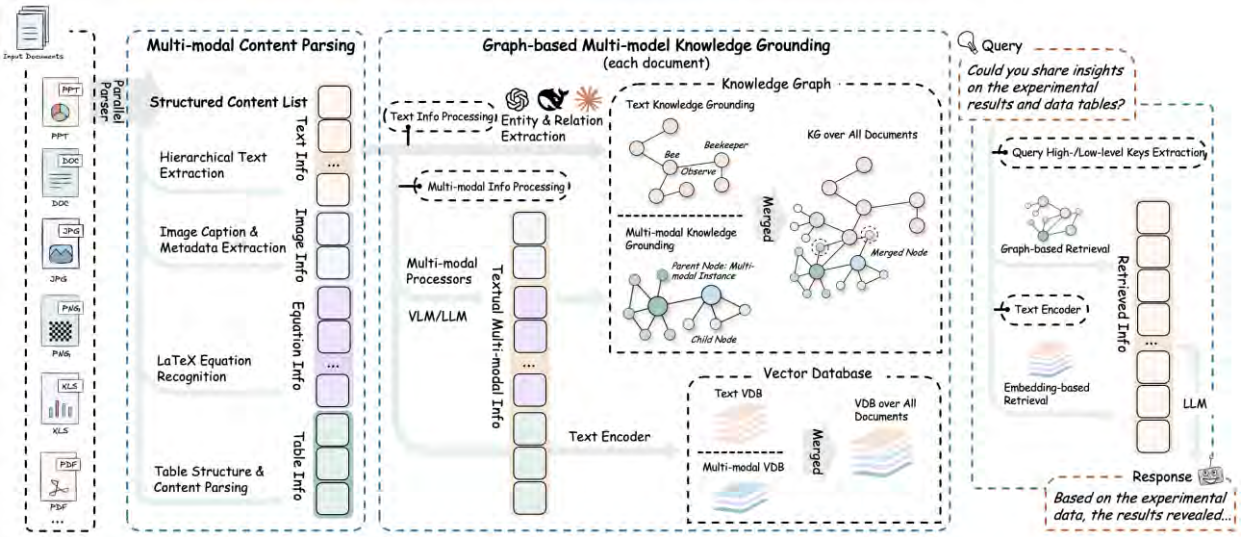

Рисунок 2.7 – Узагальнена архітектура фреймворку RAG-Anything

Робота RAG-Anything починається з надходження вхідних документів різних форматів. Це можуть бути PDF-файли, офісні документи, презентації, електронні таблиці, а також окремі графічні файли. На першому етапі здійснюється мультимодальний парсинг вмісту. Документ розкладається на

структурований список елементів, де окремо виділяються ієрархічні текстові блоки, зображення з підписами та метаданими, формули у форматі LaTeX, таблиці та інші спеціалізовані компоненти. Такий етап є принципово важливим, оскільки від якості початкового розділення документа залежить подальша точність пошуку. Якщо зображення або таблиця будуть втрачати зв'язок із заголовком, абзацом чи розділом, то згенерована відповідь може бути неповною або неточною.

Для високоякісного виділення структури документа RAG-Anything може використовувати MinerU та інші парсери. Їхнє завдання полягає не тільки у вилученні тексту, а й у збереженні логічної організації сторінки. Наприклад, під час обробки наукової статті система повинна розрізнити основний текст, підпис до рисунка, графік, формулу та таблицю результатів. У контексті роботи із зображеннями це означає, що кожен візуальний об'єкт має бути пов'язаний із відповідним текстовим оточенням. Саме такий підхід дозволяє надалі відповідати на запитання не лише про те, що написано в документі, а й про те, що показано на рисунку або схемі.

Після етапу парсингу виконується розуміння та обробка мультимодального вмісту. Система автоматично визначає тип кожного елемента документа та спрямовує його до відповідного каналу обробки. Текстові блоки можуть передаватися на звичайне мовне опрацювання та векторизацію, тоді як зображення аналізуються за допомогою мультимодальної або візуально-мовної моделі. Таблиці обробляються як структуровані дані, а математичні вирази розпізнаються як формалізовані елементи знань. Така маршрутизація дає змогу не змішувати різні типи інформації в один неструктурований потік, а обробляти кожну модальність найбільш придатним для неї способом.

Особливе місце в архітектурі займає мультимодальний аналітичний модуль. Для зображень він виконує функції візуального аналізатора: визначає основні об'єкти, описує семантику сцени, формує текстовий підпис і може враховувати просторові або ієрархічні зв'язки між елементами зображення.

Якщо у документі міститься схема, то система має не просто зберегти її як файл, а сформувати змістовний опис, наприклад вказати, які блоки присутні на схемі та як вони пов’язані між собою. Якщо зображення є графіком або діаграмою, то важливим стає виділення осей, позначень, трендів і загального висновку, який можна отримати з візуального представлення даних.

Крім зображень, RAG-Anything передбачає окрему обробку таблиць і формул. Інтерпретатор структурованих даних дає змогу аналізувати табличні набори, використовувати ці відомості для відповіді на запит. Парсер математичних виразів забезпечує збереження формул у придатному для подальшої обробки вигляді, зокрема у форматі LaTeX. У результаті модель може працювати з документом як з єдиним інформаційним простором, у якому текст, рисунки, таблиці та формули доповнюють одне одного.

Наступним ключовим етапом є побудова мультимодального графа знань. На цьому етапі значущі елементи документа перетворюються на сутності, між якими встановлюються зв’язки. Для тексту такими сутностями можуть бути поняття, терміни, об’єкти або події. Для зображень сутністю може бути сам візуальний об’єкт, його підпис, виділені елементи сцени або описані в ньому структури. Важливо, що RAG-Anything формує не лише незалежні записи, а й міжмодальні зв’язки: наприклад, рисунок належить до певного розділу, пояснює конкретне поняття, пов’язаний із таблицею результатів або деталізує твердження з абзацу. Такі зв’язки можуть зберігатися через відношення типу «належить до», «пояснює», «пов’язаний із» або інші семантичні залежності.

Графова складова є корисною для пошуково-підкріпленої генерації, оскільки вона доповнює звичайний векторний пошук. Векторна база дає змогу знаходити близькі за змістом фрагменти на основі embeddings, тобто числових подань тексту або мультимодального опису. Однак у складних документах релевантність часто визначається не лише семантичною схожістю, а й структурним положенням елемента. Наприклад, зображення може не містити в описі всіх ключових слів із запиту, але воно може бути

пов’язане з абзацом, у якому ці ключові слова є. Граф знань дозволяє перейти від знайденого текстового вузла до відповідного рисунка, таблиці або формули, зберігаючи цілісність контексту.

Під час індексації RAG-Anything використовує поєднання графового та векторного подання знань. У векторній базі можуть зберігатися текстові фрагменти, описи зображень, табличні пояснення та інші перетворені в текстову або мультимодальну форму елементи. У графі знань зберігається структура документа та зв’язки між об’єктами. Така подвійна організація дає змогу здійснювати гібридний пошук: з одного боку, система знаходить семантично подібні фрагменти, а з іншого – розширює результати через пов’язані вузли графа. Для задачі RAG на зображеннях це означає, що відповідь може спиратися не тільки на опис самого зображення, а й на весь контекст, у якому це зображення використовується.

На етапі обробки запиту користувача система визначає, які ключові елементи потрібні для відповіді, і запускає механізми пошуку. У схемі RAG-Anything поєднуються graph-based retrieval та embedding-based retrieval. Перший механізм орієнтований на обхід графа знань і пошук пов’язаних сутностей, а другий – на порівняння векторних подань запиту та елементів бази знань. Отримані результати ранжуються з урахуванням модальності: якщо запит явно стосується рисунка, схеми, графіка або зображення, то візуальні елементи та їх описи повинні отримувати вищу вагу. Якщо ж запит стосується пояснення поняття, система може надати перевагу текстовим фрагментам, але все одно зберегти пов’язані з ними рисунки як допоміжний контекст.

RAG-Anything можна розглядати як основу програмної архітектури, у якій вхідні зображення та документи проходять послідовні етапи: завантаження, структурний аналіз, формування описів зображень, створення мультимодальних сутностей, індексація у векторній базі та графі знань, пошук релевантного контексту й передавання його до LLM для генерації відповіді. Така послідовність дає змогу реалізувати не простий чат-бот над

текстовими документами, а систему, яка здатна відповідати на запитання щодо візуального змісту. Наприклад, користувач може запитати, що зображено на схемі, які компоненти входять до архітектури, як пов’язані блоки на рисунку або які висновки можна зробити з графіка.

Практична перевага RAG-Anything полягає також у тому, що він зменшує потребу в поєднанні багатьох окремих інструментів. У традиційній реалізації розробнику довелося б окремо налаштовувати парсер документів, OCR-модуль, модель опису зображень, інструмент обробки таблиць, векторну базу, граф знань і логіку об’єднання результатів. RAG-Anything пропонує узгоджену модель, у якій ці елементи поєднуються в єдиний мультимодальний конвеєр. Це спрощує експериментальну реалізацію, дає змогу зосередитися на якості пошуку та генерації відповіді, а також робить архітектуру більш прозорою для подальшого опису та оцінювання.

Разом з тим ефективність такого підходу залежить від якості проміжних представлень. Якщо VLM сформує неточний опис зображення, то помилка може потрапити до індексу та вплинути на відповідь. Аналогічно, некоректне розпізнавання таблиці або формули знижує якість пошуку. Тому під час реалізації моделі доцільно передбачати перевірку результатів парсингу, аналіз якості підписів до зображень, оцінювання релевантності знайдених елементів і порівняння відповідей системи з очікуваними результатами.

РОЗДІЛ 3
РЕКОМЕНДАЦІЇ ЩОДО РЕАЛІЗАЦІЇ ПОШУКОВО-ПІДКРІПЛЕНОЇ ГЕНЕРАЦІЇ НА ЗОБРАЖЕННЯХ

3.1 Формування оцінки мультимодальної моделі пошуково-підкріпленої генерації на зображеннях

Оцінювання мультимодальної системи RAG на зображеннях має комплексний характер, оскільки така система поєднує кілька етапів: індексацію зображень або їх текстових описів, пошук релевантного контексту, аналіз візуальної інформації мультимодальною моделлю та формування відповіді. Тому необхідно окремо оцінювати: релевантність контексту, повноту знайденої інформації, відповідність відповіді запиту та вірність відповіді відносно джерел [39].

Першу групу становлять метрики якості пошуку. Вони визначають, наскільки правильно система знаходить потрібні зображення, фрагменти документа, підписи до зображень або OCR-текст. До таких метрик належать Precision@k, Recall@k, Hit Rate@k, середній обернений ранг (Mean Reciprocal Rank, MRR), нормалізований дисконтований кумулятивний виграш (Normalized Discounted Cumulative Gain, NDCG). Знак @ позначає «на перших k позиціях». Метрика Precision@k показує, яка частка з перших k знайдених елементів є релевантною. Наприклад, якщо система повернула 5 зображень, а 3 з них справді відповідають запиту, то $\text{Precision@5}=0,6$. Метрика Recall@k показує, яку частину всіх релевантних об’єктів вдалося знайти серед перших k результатів. Precision і Recall не враховують порядок результатів, тоді як MRR та NDCG беруть до уваги позицію релевантних елементів у списку видачі [40].

Для RAG на зображеннях ці метрики можна застосовувати до різних типів пошуку: текстовий запит → зображення, текстовий запит → підпис до зображення, зображення → схожі зображення, зображення → текстовий опис.

Наприклад, якщо користувач запитує: «Який об’єкт показано на схемі?», система повинна знайти саме те зображення або фрагмент документа, де цей об’єкт зображений. Якщо релевантне зображення знаходиться на першій позиції, система має кращу якість ранжування, ніж у випадку, коли правильний результат з’являється лише на 5-ій або 10-ій позиції.

Другою групою є метрики релевантності контексту. У RAG-системі недостатньо просто знайти будь-яке схоже зображення; знайдений контекст має бути корисним саме для відповіді на запит користувача. Для цього використовують метрики Context Precision та Context Recall. Context Precision показує, наскільки знайдені фрагменти є справді потрібними, а Context Recall – чи містить знайдений контекст усю інформацію, необхідну для відповіді. Наприклад, якщо система знайшла зображення з потрібним об’єктом, але не знайшла підпис або пояснювальний текст до нього, відповідь може бути неповною. Саме тому для мультимодального RAG важливо оцінювати не лише схожість зображення з запитом, а й достатність візуального та текстового контексту.

Третю групу становлять метрики якості генерації відповіді: Answer Relevancy, Faithfulness, Factual Correctness та Semantic Similarity. Answer Relevancy оцінює відповідність відповіді поставленому запиту. Наприклад, якщо користувач просить пояснити, що зображено на рисунку, відповідь повинна описувати саме зміст рисунка, а не давати загальну інформацію про тему. Faithfulness оцінює, чи всі твердження у відповіді можна підтвердити знайденим контекстом. Це важливо, оскільки система повинна не вигадувати відповідь, а формувати її на основі знайдених джерел.

Для мультимодальних систем окремо застосовуються метрики Multimodal Faithfulness та Multimodal Relevance. Multimodal Faithfulness визначає фактологічну узгодженість відповіді одночасно з текстовим і візуальним контекстом. Тобто відповідь вважається якісною лише тоді, коли її твердження можна вивести із зображення або супровідного тексту. Multimodal Relevance оцінює, наскільки відповідь узгоджується із запитом

користувача та знайденим мультимодальним контекстом. Обидві метрики зазвичай подаються у шкалі від 0 до 1, де більше значення означає кращу якість відповіді.

Також треба оцінювати розуміння зображення. Для системи RAG на зображеннях важливо, щоб модель не лише знаходила потрібне зображення, а й правильно інтерпретувала його зміст. Наприклад, якщо на зображенні наведено графік, схема, таблиця або технічний рисунок, модель повинна правильно визначити основні елементи, підписи, зв’язки між об’єктами та числові значення. У таких випадках можуть використовуватися метрики точності розпізнавання об’єктів, точності OCR, правильності опису зображення, а також ручне експертне оцінювання [41].

У практичних дослідженнях мультимодального RAG доцільно використовувати комбінований підхід до оцінювання. Наприклад, спочатку перевіряється, чи система знайшла правильне зображення за допомогою Recall@k або Hit Rate@k. Далі оцінюється, чи достатній знайдений контекст для відповіді, тобто застосовуються Context Precision і Context Recall. Після цього аналізується якість згенерованої відповіді за допомогою Answer Relevancy, Faithfulness, Multimodal Faithfulness та Multimodal Relevance. Такий підхід дає змогу визначити, на якому саме етапі виникає помилка: під час пошуку, інтерпретації зображення або генерації відповіді [42].

Для систем, орієнтованих на роботу з зображеннями, також можуть використовуватися спеціалізовані тестові набори та сценарії. Наприклад, бенчмарки мультимодальної RAG перевіряють системи на завданнях опису зображень, мультимодального question answering, фактологічної перевірки та повторного ранжування зображень. Це показує, що оцінювання має охоплювати не лише кінцеву текстову відповідь, а й здатність моделі використовувати зображення як джерело доказової інформації.

3.2 Реалізація моделі на основі фреймворку RAG-Anything

Згідно п. 2.4, для практичної реалізації моделі RAG на зображеннях доцільно використати фреймворк RAG-Anything. Його основна перевага у можливості працювати не лише з текстовими фрагментами документів, а й із зображеннями, таблицями, графіками, схемами та іншими елементами, які часто містяться у PDF-файлах, презентаціях, звітах та ін. (Додаток А).

Така реалізація системи на основі RAG-Anything передбачає кілька послідовних етапів. Спочатку виконується встановлення фреймворку через pip та необхідних залежностей для роботи з мультимодальними даними. Оскільки система повинна обробляти не тільки текст, а й зображення, до складу середовища входять бібліотеки для роботи з графічними файлами та CV. Після цього створюється об’єкт RAG-Anything, у якому задаються параметри доступу до мовної моделі, адреса API-сумісного endpoint, каталог для збереження результатів обробки, а також парсер документів. У прикладах фреймворку часто використовується OpenAI. Однак архітектура дозволяє підключати й інші сумісні сервіси. У локальному варіанті замість хмарного API може бути використано платформу LM Studio, якщо в ній запущено відповідні мовні, embedding- та vision-моделі.

Важливим компонентом реалізації є мультимодальний парсер, який відповідає за попередню обробку документів. У RAG-Anything для цього може використовуватися MinerU. Його завдання полягає у тому, щоб розібрати вхідний документ на окремі структурні елементи: текстові абзаци, заголовки, таблиці, зображення, формули та інші об’єкти. Завдяки цьому документ не сприймається системою як суцільний текст, а перетворюється на набір взаємопов’язаних інформаційних одиниць. Це має значення для мультимодального RAG, оскільки зображення або таблиця часто набувають повного змісту лише у зв’язку з пояснювальним підписом, абзацом або розділом, у якому вони розміщені. Після підготовки середовища виконується індексація документів. На цьому етапі до системи передаються файли, які

потрібно додати до бази знань. Це можуть бути PDF-документи, офісні файли, презентації, електронні таблиці або окремі зображення. Кожен документ отримує власний ідентифікатор, що надалі дозволяє відстежувати джерело знайденої інформації. Під час індексації документ розбивається на сторінки та окремі змістові фрагменти. Текстові частини перетворюються на векторні представлення, а мультимодальні елементи проходять додаткову обробку, під час якої для них формуються описи та метадані.

Особливістю RAG-Anything є те, що зображення не ігноруються і не розглядаються як другорядні елементи документа. Для них може використовуватися спеціальний обробник, який передає зображення до мультимодальної мовної моделі та отримує його змістовий опис. Такий опис надалі індексується разом з іншими фрагментами документа. Наприклад, якщо у звіті міститься графік, схема або рисунок з результатами експерименту, фреймворк може сформувати текстове представлення цього зображення, зберегти його зв’язок з оригінальним файлом, сторінкою документа, підписом і додатковими поясненнями. Це дає змогу надалі знаходити релевантні візуальні матеріали за текстовим запитом користувача.

Порівняно зі звичайною текстовою RAG-системою, такий підхід є більш придатним для роботи зі складними навчальними, науковими та технічними документами. У традиційному RAG зображення, графіки або схеми часто втрачаються під час попередньої обробки, оскільки система орієнтується переважно на текст. У RAG-Anything, навпаки, кожен мультимодальний елемент може бути включений до єдиної бази знань.

Після завершення індексації користувач може виконувати мультимодальні запити до системи. Запит подається у природній мові, після чого RAG-Anything здійснює пошук релевантного контексту у створеній базі знань. Для цього використовується гібридний режим пошуку, який поєднує переваги векторного пошуку та графового представлення знань. Векторний пошук дає змогу знаходити фрагменти, близькі до запиту за змістом, а графовий підхід допомагає враховувати зв’язки між різними елементами

документа. Наприклад, якщо запит стосується результатів експерименту, система може знайти не тільки текстовий опис експерименту, а й відповідну таблицю, графік або зображення, пов’язане з цим фрагментом.

Результатом роботи системи є згенерована відповідь, побудована на основі знайденого контексту. Крім самого тексту відповіді, система може повертати інформацію про використані джерела: тип фрагмента, його розташування у документі, сторінку, шлях до файлу, а також посилання на зображення, якщо воно було використане під час формування відповіді. Така можливість є важливою з погляду перевірки достовірності результатів. Користувач може не лише отримати готову відповідь, а й побачити, на яких саме даних вона ґрунтується. Це підвищує прозорість роботи моделі та дозволяє оцінювати якість пошуку й генерації.

Окрему увагу в реалізації моделі слід приділити роботі з зображеннями. У межах RAG-Anything зображення може мати не лише файл або шлях до нього, а й підпис, примітку, номер сторінки, зв’язок з основним документом та іншу службову інформацію. Завдяки цьому система може краще інтерпретувати візуальний зміст. Наприклад, одне й те саме зображення має різне значення (залежить від контексту): графік може відображати результати експерименту, динаміку зміни показника або порівняння кількох моделей. Якщо зберігаються підпис, примітка і зв’язок з текстом, система отримує більше інформації для правильної інтерпретації.

RAG-Anything також підтримує додавання до бази знань не лише цілих документів, а й окремих мультимодальних елементів. Це корисно у випадках, коли зображення вже попередньо підготовлені або витягнуті з документів. Наприклад, до бази знань можна додати набір рисунків, графіків або схем із зазначенням їхніх підписів, пояснень і сторінок, на яких вони розташовані. У результаті такі об’єкти стають частиною загального індексу й можуть бути знайдені під час запиту користувача. Це дає можливість створити базу знань, у якій візуальні дані є повноцінними джерелами інформації. Ще однією перевагою фреймворку є підтримка пакетної обробки документів. Це означає,

що система може індексувати не один файл, а цілий набір документів. Такий підхід є практичним для реальних сценаріїв, коли база знань формується з великої кількості звітів, статей, презентацій або навчальних матеріалів. Після обробки всі елементи зберігаються в єдиному індексі, що дозволяє виконувати пошук одразу по всій колекції документів. Крім того, у фреймворку передбачено кешування результатів обробки за хешем вмісту. Це означає, що під час повторної індексації тих самих файлів система не виконує зайві звернення до мовної моделі, що зменшує витрати часу та обчислювальних ресурсів.

RAG-Anything підтримує роботу не тільки з PDF-файлами, а й з документами Microsoft Office, зокрема текстовими документами, електронними таблицями та презентаціями. Це розширює сферу застосування системи, оскільки в реальних умовах інформація рідко зберігається в одному форматі. Наприклад, опис системи може бути поданий у текстовому документі, експериментальні дані – в електронній таблиці, а архітектура моделі – у презентації або у вигляді окремої схеми. Підтримка таких форматів дозволяє формувати більш повну мультимодальну базу знань.

Практична реалізація моделі на основі RAG-Anything може бути побудована як послідовний конвеєр. На першому етапі користувач завантажує документи або зображення до системи. На другому етапі парсер виконує структурний аналіз і виділяє текстові та візуальні елементи. На 3-му етапі для тексту створюються embeddings, а для зображень, таблиць і графіків формуються змістові описи. На 4-му етапі всі елементи зберігаються в єдиній базі знань з зазначенням метаданих і зв’язків між ними. На 5-му етапі користувач задає запит, після чого система знаходить релевантні фрагменти, передає їх до LLM і формує відповідь.

У локальному сценарії реалізації важливим є питання вибору моделей. Якщо система не повинна залежати від хмарних сервісів, замість OpenAI API можна використовувати локальний OpenAI-сумісний endpoint, наприклад LM Studio. У такому випадку для генерації відповідей використовується локальна

мовна модель, для створення embeddings – окрема embedding-модель, а для аналізу зображень – vision-language model. Це дає змогу побудувати більш автономну систему, яка може працювати на локальному комп'ютері або сервері. Такий підхід є особливо актуальним у випадках, коли документи містять конфіденційну або службову інформацію.

Отже, реалізація моделі на основі RAG-Anything передбачає створення мультимодальної бази знань, у якій текстові та візуальні елементи документа обробляються як взаємопов'язані джерела інформації. Фреймворк забезпечує парсинг документів, індексацію тексту, опис зображень, збереження метаданих, гібридний пошук і генерацію відповіді на основі знайденого контексту. Завдяки цьому RAG-Anything є придатним інструментом для побудови моделі пошуково-підкріпленої генерації на зображеннях, оскільки дозволяє поєднати можливості векторного пошуку, мультимодального аналізу та мовної генерації в єдиній системі.

3.3 Техніки індексування для мультимодальної пошуково-підкріпленої генерації

Мультимодальну RAG часто розглядають як проблему вибору або налаштування моделі. Проте на практиці це зазвичай проблема індексації, яка лише зовні виглядає як модельна проблема. Якщо таблиці поділяються на фрагменти так само, як звичайний текст, скриншоти перетворюються на слабко розпізнаний OCR-текст, інтерфейс користувача втрачає свою структуру, а координати зникають ще на етапі завантаження даних, то генеративна модель фактично не має достатніх умов для якісної відповіді. Найкращі сучасні системи поступово переходять до гібридної індексації, обробки таблиць з урахуванням їхньої схеми, візуального пошуку та використання розширених контекстних метаданих, оскільки саме на цих етапах забезпечується реальна якість пошуку та витягування релевантної

інформації. Можна створити візуально привабливу демонстрацію мультимодального RAG, але все одно зазнати невдачі саме в тому, про що насправді запитують користувачі: «На якій інформаційній панелі було показано спад?»; «Знайди таблицю з цінами»; «Яка кнопка розташована поруч із Export?»; «Покажи екран із червоним попереджувальним банером». Саме на цьому етапі багато систем дають збій. Не під час генерації відповіді. І навіть не під час повторного ранжування результатів. Проблема виникає на етапі індексації.

Причина полягає в тому, що коли корпус даних містить таблиці, скриншоти, форми, інформаційні панелі та насичені елементи інтерфейсу користувача, звичайного поділу тексту на фрагменти вже недостатньо. Зміст документа більше не зберігається лише в реченнях. Він також міститься у структурі сторінки, заголовках, стовпцях, кольорах, іконках, візуальному групуванні та дрібних деталях, які людина сприймає миттєво, але які прості конвеєри обробки часто втрачають. Сучасні дослідження в галузі мультимодального RAG чітко підкреслюють цю проблему: візуально насичені документи містять важливу інформацію в макеті, рисунках, таблицях і структурі сторінки, яку системи, побудовані лише на OCR, часто не здатні повноцінно врахувати. Таким чином, індексація виглядає як справжнє обмеження.

Багато команд розглядають індексацію лише як технічний етап завантаження даних: розпізнати PDF-документ, виконати OCR сторінки, поділити текст на фрагменти, створити векторні представлення та запустити систему. Такий підхід працює доти, доки відповідь не міститься в заголовку таблиці, легенді діаграми або в розташуванні кнопки в інтерфейсі програмного продукту. Саме з цієї причини ColPali став важливим напрямом розвитку. Його основна ідея є простою, але ефективною: замість того щоб покладатися лише на ненадійне витягування тексту, сторінки документів подаються як зображення та індексуються з використанням механізму пізньої взаємодії. Це дає змогу зберігати візуальні ознаки, зокрема таблиці, структуру

сторінки та типографіку. На тестових наборах для візуально насичених документів такий підхід показав кращі результати порівняно з традиційними конвеєрами, орієнтованими переважно на текст. Отже, якщо розробляється мультимодальна RAG-система для роботи з таблицями та інтерфейсами користувача, найбільш важливими є кілька рішень щодо індексації.

1. Визначте, що саме індексується: текст, сторінки або обидва варіанти. Індексація лише тексту є дешевшою та простішою. Вона добре працює тоді, коли OCR виконується якісно, а запит користувача має переважно семантичний характер. Однак індексація сторінок як зображень дає змогу зберегти макет, просторову близькість елементів, візуальне групування та нетекстові ознаки, які зникають під час звичайного витягування тексту. У рекомендаціях щодо мультимодального RAG це прямо визначається як одне з ключових архітектурних рішень: система може або аналізувати й перетворювати мультимодальний контент у текст для подальшої індексації, або використовувати мультимодальні представлення, які краще зберігають структуру документа. Для нормативних документів, інструкцій і PDF-файлів, у яких переважає звичайний текст, зазвичай достатньо підходу, орієнтованого насамперед на текст. Для інформаційних панелей, рахунків, таблиць, інтерфейсів програмних продуктів, презентацій і сканованих звітів доцільно використовувати подвійний індекс:

– один текстовий індекс – для семантичного пошуку;

– один візуальний індекс на рівні сторінки або окремої області – для пошуку, чутливого до структури та макета.

Такий гібридний підхід стає дедалі поширенішим, оскільки жоден із цих варіантів окремо не є достатнім для всіх типів запитів.

2. Використовуйте індексацію на рівні сторінки, коли макет має змістове значення. Варто визнати: у деяких випадках відповідь має «форму сторінки». Фінансова таблиця з примітками, екран налаштувань із боковою навігацією або KPI-панель із фільтром у правому верхньому куті залежать не лише від тексту, а й від просторового розташування елементів. Якщо

перетворити їх лише на суцільний текст, слова збережуться, але «карта» документа буде втрачена. ColPali та подальші дослідження в галузі пошуку у візуально насичених документах демонструють, чому представлення на рівні сторінки є корисними: модель може зіставляти намір запиту з фактичною візуальною поверхнею документа, а не лише з витягнутими текстовими рядками. Це особливо важливо, коли користувачі формулюють запити на кшталт «екран із…» або «таблиця під…». Індексація на рівні сторінки добре підходить для: скриншотів; форм; презентацій; сканованих документів; інформаційних панелей; таблиць з об'єднаними комірками або складним форматуванням.

3. Розбивайте таблиці на структурно усвідомлені елементи, а не на випадкові фрагменти. Це, ймовірно, одна з найсерйозніших помилок індексації. Таблиця – це не просто абзац із лініями сітки. Вона має власну схему, заголовки, групи рядків, одиниці вимірювання та зв'язки між комірками. Якщо ділити її на фрагменти кожні 500 токенів, структура, про яку запитує користувач, фактично руйнується. Сучасні дослідження, присвячені RAG для роботи з таблицями, наголошують на необхідності такого пошуку, який окремо враховує схему таблиці та її комірки, а не просто повністю додає таблицю до контексту запиту. Наприклад, TableRAG використовує розширення запиту, а також пошук за схемою та комірками, щоб знаходити важливі частини дуже великих таблиць перед виконанням логічного аналізу. THoRR також підкреслює особливу важливість заголовків як пошукових ознак під час знаходження та уточнення таблиць. Кращий підхід до індексації таблиць. Доцільно індексувати щонайменше три рівні (рис. 3.1). Такий підхід виглядає складнішим. Однак саме завдяки цьому пошук перестає пропускати очевидну для користувача інформацію.

4. Зберігайте координати та метадані областей. Користувачі не завжди ставлять лише семантичні запитання. Часто вони формулюють просторові запити: «Верхня таблиця на сторінці»; «Діаграма поруч із легендою»; «Кнопка під розділом billing»; «Попереджувальна мітка на правій панелі».

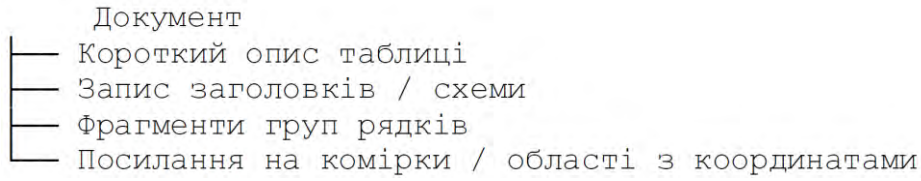

Рисунок 3.1 – Підхід до індексації таблиць

Це запити до системи пошуку, а не лише до модуля генерації відповіді. Якщо під час індексації втрачаються координати, номери сторінок, обмежувальні рамки, назви розділів або ієрархія DOM, модель змушена пізніше самостійно здогадуватися про просторові зв’язки на основі неповних даних. Саме цю слабкість намагаються усунути новіші підходи, орієнтовані на роботу з окремими областями документа. Пошук на рівні сторінки є ефективним, однак повернення цілої сторінки може бути занадто грубим для точних і добре обґрунтованих відповідей, особливо в RAG-системах, яким потрібні конкретні області, а не лише загальна релевантність сторінки.

5. Додайте контекстні метадані під час індексації, а не після пошуку. Однією з найцікавіших сучасних ідей у сфері пошуку є контекстний пошук. Компанія Anthropic повідомляла, що додавання спеціального контексту для кожного фрагмента та поєднання контекстних векторних представлень із контекстним BM25 зменшило кількість невдалих пошукових результатів на 49 %, а в поєднанні з повторним ранжуванням – на 67 %. Ця ідея має ще більше значення для таблиць та інтерфейсів користувача. Якщо індексувати область разом із метаданими, наприклад: назвою документа; розділом сторінки; найближчими заголовками; частиною продукту; підписом до таблиці; попередніми та наступними панелями, то пошук стає значно менш вразливим до помилок.

6. Індексуйте скриншоти інакше, ніж інтерфейси, створені у цифровому вигляді. Скриншот інтерфейсу програмного продукту та живий HTML-експорт – це не один і той самий об’єкт для пошуку. Скриншоти зберігають точний візуальний вигляд, але втрачають чітку внутрішню структуру. Натомість інтерфейси, створені у цифровому вигляді, можуть містити

DOM-структуру, accessibility-мітки, назви компонентів та ієрархію взаємодії. Якщо обробляти обидва типи даних однаково, зазвичай втрачаються переваги кожного з них. Для скриншотів доцільно індексувати області зображення, OCR-текст, виявлені елементи керування та візуальні описи. Для живих UI-експортів варто індексувати DOM-шляхи, видимі мітки, ролі компонентів, їхній стан і мініатюри скриншотів.

7. Використовуйте гібридний пошук, оскільки запити за своєю природою є змішаними. Реальні запити користувачів часто є складними та неоднорідними. В якості приклади можна вказати таке: «Покажи екран, де після квітня різко зростає відтік користувачів»; «Знайди таблицю з тарифами для корпоративних клієнтів»; «На якій інформаційній панелі є теплова карта затримок?». Такі запити поєднують текстовий намір, візуальний шаблон, посилання на структуру розміщення елементів і предметне значення. Тому індекс також має підтримувати таке поєднання. Ефективна архітектура зазвичай має вигляд, що наведений на рис. 3.2.

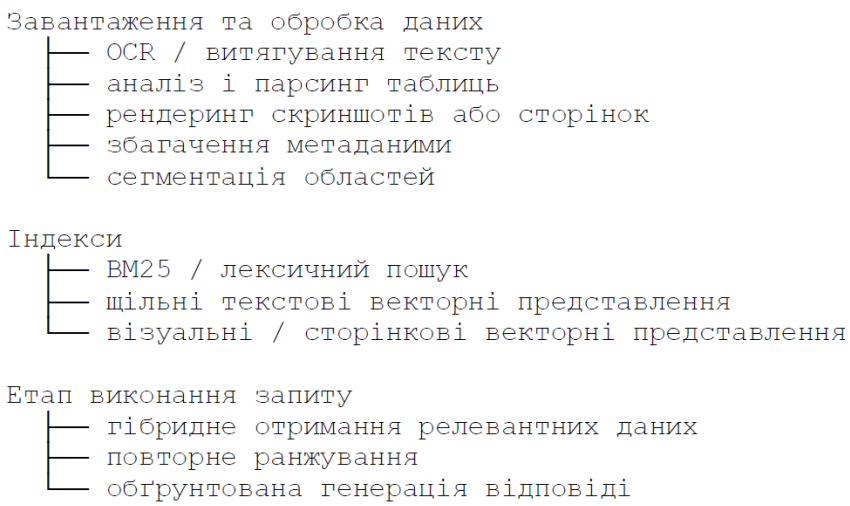

Рисунок 3.2 – Архітектура для мультимодального запиту

Такий підхід загалом відповідає сучасній практиці мультимодального RAG як у дослідженнях, так і в промислових архітектурних рішеннях: різнорідний контент завантажується, нормалізується, індексується у кількох представленнях, а потім об’єднується на етапі пошуку.

8. Індексуйте дані з урахуванням окремих сценаріїв оцінювання, а не лише для повного покриття корпусу. Саме цей аспект команди часто пропускають. Корпус може бути «повністю проіндексований», але система все одно може не справлятися саме з тими типами даних, які є важливими для користувачів: таблицями з об’єднаними комірками; скриншотами низької якості; інтерфейсами в темному режимі; багатомовними інформаційними панелями; сторінками з іконками, але з мінімальною кількістю тексту; таблицями, що займають кілька сторінок.

Бенчмарк ViDoRe, який лежить в основі оцінювання ColPali, є важливим зокрема тому, що перевіряє пошук у візуально насичених документах у різних предметних галузях і умовах, а не припускає, що один тип документа може репрезентувати всі реальні випадки. Отже, потрібно оцінювати не лише успішність завантаження та індексації даних. Важливо також вимірювати якість пошуку для окремих типів контенту. Інакше система може виглядати завершеною, але все одно не знаходити саме той екран або документ, який потрібен користувачу.

3.4 Економічне обґрунтування прийнятих рішень

Основна економічна перевага використання RAG-Anything полягає в тому, що фреймворк уже містить готові засоби для парсингу документів, індексації текстових і візуальних фрагментів, створення описів зображень, збереження метаданих та виконання гібридного пошуку. Це зменшує трудомісткість розроблення, оскільки немає потреби створювати з нуля окремі програмні компоненти для OCR-обробки, аналізу зображень, формування embeddings, організації бази знань та пошуку релевантного контексту. Застосування готового фреймворку дозволяє скоротити витрати на проєктування, а потім на впровадження системи. Для підтвердження економічної доцільності прийнятих рішень виконаємо орієнтовний

розрахунок витрат та очікуваного економічного ефекту від використання моделі RAG на зображеннях. Для розрахунку приймемо такі дані: витрати на розробку, налаштування системи становлять 14000 грн, амортизаційні витрати на обладнання – 2000 грн, витрати на ПЗ – 0 грн, оскільки використовуються відкриті або безкоштовно доступні інструменти, а витрати на супровід і тестування – 3000 грн. Тоді. загальні витрати становитимуть:

$$C_{заг} = C_{розр} + C_{обл} + C_{ПЗ} + C_{супр} = 14000 + 2000 + 0 + 3000 = 19000 \text{ грн}, \quad (3.1)$$

де $C_{заг}$ – загальні витрати на створення системи;

$C_{розр}$ – витрати на розроблення ПЗ;

$C_{обл}$ – витрати на використання або амортизацію обладнання;

$C_{ПЗ}$ – витрати на ПЗ;

$C_{супр}$ – витрати на супровід, тестування та налаштування системи.

Основним джерелом економічного ефекту є скорочення часу, необхідного для пошуку інформації. За ручного пошуку користувач витрачає час на перегляд сторінок, аналіз тексту, таблиць, графіків і зображень. Використання RAG дозволяє автоматизувати цей процес і швидше отримувати відповідь на запит. Приймемо, що ручний пошук відповіді займає 15 хв, тобто 0,25 год, а отримання відповіді за допомогою RAG-системи – 3 хв, тобто 0,05 год. Кількість запитів за рік становить 500, а вартість однієї години роботи фахівця – 150 грн. Тоді економія часу становитиме:

$$E_{час} = \left(t_{руч} - t_{RAG}\right) \cdot N_{зап} \cdot S_{год} = \left(0,25 - 0,05\right) \cdot 500 \cdot 150 = 15000 \text{ грн}, \quad (3.2)$$

де $E_{час}$ – економія за рахунок скорочення часу пошуку;

$t_{руч}$ – середній час ручного пошуку відповіді на один запит;

t_{RAG} – середній час отримання відповіді за допомогою RAG-системи;

$N_{зап}$ – кількість запитів за певний період;

$S_{год}$ – вартість однієї години роботи фахівця.

Перевагою локального варіанта є зменшення витрат на використання хмарних API-сервісів. Наприклад, якщо використання хмарного API коштує

≈6000 грн на рік, то у разі локального розгортання ці витрати суттєво зменшені. Приймемо, що економія на API-запитах становить 6000 грн на рік, а експлуатаційні витрати на підтримку системи – 2500 грн на рік. Тоді річний економічний ефект становитиме:

$$E_{річ} = E_{час} + E_{API} - C_{експл} = 15000 + 6000 - 2500 = 18500 \text{ грн}, \quad (3.3)$$

де $E_{річ}$ – річний економічний ефект;

E_{API} – економія на використанні хмарних API-сервісів;

$C_{експл}$ – експлуатаційні витрати на підтримку роботи системи.

Для оцінки запропонованого рішення визначимо коефіцієнт економічної ефективності:

$$K_{еф} = \frac{E_{річ}}{C_{заг}} = \frac{18500}{19000} = 0,97 .$$

Отримане значення є близьким до одиниці, що свідчить про достатній рівень економічної доцільності рішення. При збільшенні кількості запитів або ширшому використанні системи економічний ефект зростатиме. Строк окупності системи визначається за формулою:

$$T_{окуп} = \frac{C_{заг}}{E_{річ}} = \frac{19000}{18500} = 1,03 \text{ року.}$$

Орієнтовний строк окупності запропонованої системи становить ≈1 рік. Це свідчить про економічну доцільність використання моделі RAG на зображеннях, оскільки вона дозволяє скоротити час пошуку інформації, зменшити залежність від хмарних API-сервісів і підвищити ефективність роботи з мультимодальними документами.

ВИСНОВКИ

Класична RAG має суттєві обмеження під час роботи з такими джерелами, оскільки не завжди здатна коректно враховувати візуальні та структурні зв’язки між елементами документа. Мультимодальна RAG дає змогу об’єднати текстовий і візуальний контекст (інформація подана не лише у вигляді тексту, а й у формі зображень, схем, таблиць, графіків та сканованих документів), що підвищує точність пошуку релевантної інформації та якість згенерованої відповіді.

Мультимодальна RAG має широкий спектр практичного застосування у сферах, де дані представлені у змішаній текстово-візуальній формі. Найбільш перспективними напрямами її використання є юридична діяльність, медицина, інженерія, виробництво та фінансовий аналіз.

Одним з компонентів мультимодальної RAG є засоби аналізу зображень. Для цього може застосуватись ViT, щоб коректно інтерпретувати візуальний контент і використовувати його під час пошуку та генерації відповіді.

Векторна та графова RAG є двома основними підходами до організації зовнішніх знань для LLM. Векторна RAG краще підходить для швидкого пошуку релевантних фрагментів за змістовою схожістю, тоді як графова RAG є ефективнішою для складних запитів, де потрібно враховувати зв’язки між поняттями, сутностями та документами. Для RAG на зображеннях доцільним є комбінування цих підходів, реалізуючи гібридну архітектуру системи.

існує кілька підходів до побудови мультимодальних RAG-систем: CLIP-style, ColPali та VLM-as-captioner. Основний акцент зроблений на дослідження останнього підходу. Хоча він найскладніший з усіх, але є найточнішим. Він передбачає виконання пошуку за текстовим описом, а генерація відповіді може спиратися на оригінальне зображення.

Для побудови моделі RAG на зображеннях у роботі запропоновано використовувати фреймворк RAG-Anything. Його архітектура поєднує

парсинг документів, аналіз зображень, обробку таблиць і формул, побудову мультимодального графа знань та гібридний пошук. Його перевагою RAG-Anything є збереження зв’язків між текстовими блоками, зображеннями, таблицями та іншими елементами документа, що підвищує якість контексту для генерації відповіді. Ефективність залежить від якості попередньої обробки, точності описів зображень і коректності індексації мультимодальних даних.

Оцінювання мультимодальної RAG на зображеннях має охоплювати не лише якість кінцевої відповіді, а й ефективність пошуку, релевантність контексту та правильність інтерпретації візуальної інформації. Для цього доцільно використовувати як класичні метрики пошуку, так і спеціалізовані, наприклад: Answer Relevancy, Faithfulness, Multimodal Faithfulness та Multimodal Relevance.

Якість роботи мультимодальної RAG-системи значною мірою залежить від правильно обраних технік індексування. Найбільш ефективним є гібридний підхід, який поєднує текстову індексацію, візуальні представлення, метадані, координати областей і структурно усвідомлену обробку таблиць.

Економічне обґрунтування показує доцільність використання моделі RAG на зображеннях, особливо у випадках роботи з великими обсягами мультимодальних документів. Основний економічний ефект досягається завдяки скороченню часу пошуку інформації, зменшенню потреби у ручному аналізі документів і можливості локального розгортання без постійних витрат на хмарні API-сервіси.

Таким чином, результатами роботи є модель RAG на зображеннях, рекомендації щодо практичної реалізації мультимодальної RAG-системи. Вони можуть бути використані для подальших досліджень за даною тематикою та при проектуванні корпоративних RAG-систем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Maheshus M. Retrieval-Augmented Generation (RAG): Real Advances in 2025. Medium.com. URL: https://medium.com/@maheshus007/ retrieval-augmented-generation-rag-real-advances-in-2025-8a0933ce20c2 (дата звернення: 05.05.2026)

2. Why Large Language Models are the future of manufacturing. *Weforum*. URL: https://www.weforum.org/stories/2024/04/why-large-language-models-are-so-important-for-the-future-of-the-manufacturing-industry/ (дата звернення: 05.05.2026)

3. Kamath U., Keenan K., Somers G., Sorenson S. Large Language Models: A Deep Dive: Bridging Theory and Practice. Springer, 2024.

4. Slyusar V., Sliusar I. Leveraging pre-trained neural networks for image classification in audio signal analysis for mobile applications of home automation. *Research Tendencies and Prospect Domains for AI Development and Implementation*. River Publishers, 2024. P. 109-128.

5. What Are Vision Language Models. *Nvidia*. URL: https://www.nvidia.com/en-us/glossary/vision-language-models (дата звернення: 05.05.2026)

6. Welcome to PyMuPDF. *PyMuPDF*. URL: https://pymupdf.readthedocs.io/en/latest/ (дата звернення: 05.05.2026)

7. LLaVA-1.5-7B. *Ollama*. URL: https://ollama.com/library/llava:7b (дата звернення: 05.05.2026)

8. BAAI/bge-en-icl. *Huggingface*. URL: https://huggingface.co/BAAI/bge-en-icl (дата звернення: 05.05.2026)

9. Tunstall L., von Werra L., Wolf T. Natural Language Processing with Transformers. Revised Edition. Sebastopol: O'Reilly Media, 2022.

10. Jurafsky D., Martin J. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models. 3rd ed. *Online manuscript*, 2026. URL: https://web.stanford.edu/ ~jurafsky/slp3/ (дата звернення: 05.05.2026)

11. Slyusar V., Sliusar I., Bihun N., Piliuhin V. Segmentation of analogue meter readings using neural networks. *In 4th Int. Workshop on Modern Machine Learning Technologies and Data Science MOMLET&DS2022*, Lviv, Ukraine, Nov. 2022. 11 p.

12. Zhang A., Lipton Z., Li M., Smola A. Dive into Deep Learning. Cambridge: Cambridge University Press, 2023. 548 p.

13. Будьомко В.Б. RAG на зображеннях у локальних AI-агентах. *Збірник XXI щорічної студентської наукової конференції «Сучасні інформаційні технології та інноваційні методики в економіці, менеджменті та бізнесі» Полтавського державного аграрного університету* (квітень 2026 р., м. Полтава). Полтава: ПДАУ, 2026 р. С. 71, 72.

14. Knollmeyer S., Caymazer O., Grossmann D. Document GraphRAG: Knowledge Graph Enhanced Retrieval-Augmented Generation for Complex Document Understanding. *Electronics*, 2025. URL: https://doi.org/10.3390/electronics14112102 (дата звернення: 05.05.2026)

15. GraphRAG. *Microsoft*. URL: https://microsoft.github.io/graphrag/ (дата звернення: 05.05.2026)

16. Turn long videos into viral clips. *OpenCLI*. URL: https://openclip.app (дата звернення: 05.05.2026)

17. Multi-Vector Retriever for RAG on tables, text, and images. *LangChain*. URL: https://www.langchain.com/blog/semi-structured-multi-modal-rag (дата звернення: 05.05.2026)

18. Radford A., Kim J., Hallacy C. et al. Learning Transferable Visual Models From Natural Language Supervision. *arXiv preprint*. arXiv:2103.00020. URL: https://arxiv.org/abs/2103.00020 (дата звернення: 05.05.2026)

19. Unlocking the Power of Multimodal Embeddings. *Cohere*. URL: https://docs.cohere.com/docs/multimodal-embeddings (дата звернення: 05.05.2026)

20. Koukounas A., Mastrapas G., Eslami S. Jina-clip-v2: Multilingual Multimodal Embeddings for Text and Images. *arXiv preprint*. arXiv:2412.08802. URL: https://arxiv.org/abs/2412.08802 (дата звернення: 05.05.2026)

21. ImageBind: a new way to ‘link’ AI across the senses. *Meta AI*. URL: https://imagebind.metademolab.com (дата звернення: 05.05.2026)

22. Faysse M., Sibille H., Wu T. ColPali: Efficient Document Retrieval with Vision Language Models. *arXiv preprint*. arXiv:2407.01449. URL: https://arxiv.org/abs/2407.01449 (дата звернення: 05.05.2026)

23. Faysse M., Sibille H., Wu T. ColPali: Efficient Document Retrieval with Vision Language Models. *arXiv preprint*. arXiv:2407.01449. URL: https://arxiv.org/abs/2407.01449 (дата звернення: 05.05.2026)

24. LangGraph overview. *LangChain*. URL: https://docs.langchain.com/oss/python/langgraph/overview (дата звернення: 05.05.2026)

25. ChromaDB. Chroma. URL: https://www.trychroma.com/products/chromadb (дата звернення: 05.05.2026)

26. FAISS – Welcome to Faiss Documentation. *FAISS*. URL: https://faiss.ai/index.html (дата звернення: 05.05.2026)

27. The High-Performance Vector Database. Built for Scale. *Milvus*. URL: https://milvus.io (дата звернення: 05.05.2026)

28. Qdrant. Efficient, Scalable, Fast. *Qdrant*. URL: https://qdrant.tech/qdrant-vector-database/ (дата звернення: 05.05.2026)

29. Ollama. URL: https://ollama.com (дата звернення: 05.05.2026)

30. LM Studio. URL: https://lmstudio.ai (дата звернення: 05.05.2026)

31. Koboldcpp installation package. URL: https://kcpptools.concedo.workers.dev/ (дата звернення: 05.05.2026)

32. Llama.cpp – Run LLM Inference in C/C++. *Llama-cpp*. URL: https://llama-cpp.com (дата звернення: 05.05.2026)

33. Cai Z., Ke F., Jahangard S. et al. NAVER: A Neuro-Symbolic Compositional Automaton for Visual Grounding with Explicit Logic Reasoning. *arXiv*. arXiv:2502.00372v1. URL: https://arxiv.org/html/2502.00372v1 (дата звернення: 05.05.2026)

34. Sharma A. Getting Started with Multimodal RAG: A Hands-On Guide. *Medium*. URL: https://medium.com/%40ashutoshsharmaengg/revolutionizing-ai-a-deep-dive-into-building-multimodal-rag-3cb4aa756dcc (дата звернення: 05.05.2026)

35. Guo Z., Ren X., Xu L. RAG-Anything: All-in-One RAG Framework. *arXiv preprint*. arXiv:2510.12323. URL: https://arxiv.org/abs/2510.12323 (дата звернення: 05.05.2026)

36. LightRAG. URL: https://github.com/hkuds/lightrag (дата звернення: 05.05.2026)

37. RAG-Anything. URL: https://github.com/hkuds/rag-anything (дата звернення: 05.05.2026)

38. Es S., James J., Espinosa-Anke L., Schockaert S. Ragas: Automated Evaluation of Retrieval Augmented Generation. *arXiv preprint*. arXiv:2309.15217. URL: https://arxiv.org/abs/2309.15217 (дата звернення: 05.05.2026)

39. Evaluation Metrics for Search and Recommendation Systems. *Weaviate*. URL: https://weaviate.io/blog/retrieval-evaluation-metrics (дата звернення: 05.05.2026)

40. MultiModalFaithfulness. *Ragas*. URL: https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/multi_modal_faithfulness/ (дата звернення: 05.05.2026)

41. Liu Z., Zhu X., Zhou T. Et al. Benchmarking Retrieval-Augmented Generation in Multi-Modal Contexts. *arXiv preprint*. arXiv:2502.17297. URL: https://arxiv.org/abs/2502.17297 (дата звернення: 05.05.2026)