*DOI: https://doi.org/10.25140/2411-5363-2025-3(41)-284-295*
*УДК 004.75*

***Вячеслав Іванович Чимшир***

аспірант кафедри інформаційних систем та технологій
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського» (Київ, Україна)
**E-mail:** vchimshir@gmail.com. **ORCID:** https://orcid.org/0009-0005-2555-5373

# ІНТЕЛЕКТУАЛЬНИЙ ДВИГУН І МОДЕЛІ ТА МЕТОДИ ПЛАТФОРМИ КОМПЛЕКСНОЇ ПІДТРИМКИ ПРОЦЕСІВ ЖИТТЄВОГО ЦИКЛУ ІНФОКОМУНІКАЦІЙНИХ СЕРВІСІВ

*Розглянута проблема розроблення інтелектуального двигуна як основи інтеграції засобів автоматизації процесів життєвого циклу сервісів моделей і методів для розроблення інструментів ініціації та бізнес-аналізу для платформи підтримки життєвого циклу сервісів у інформаційних системах провайдерів. Запропонований загальний підхід до підтримки процесів усіх етапів життєвого циклу сервісів. Інтелектуальний двигун виступає в ролі асистента команди підтримки життєвого циклу сервісів. Інтелектуальний двигун реалізований на основі відповідно структурованої і навченої великої мовної моделі. Для комунікації з користувачами в контексті проблем велика мовна модель інтегрується із зовнішніми базами знань за допомогою системи генерації з доповненим пошуком знань. Структурована відповідно до поставленої проблеми інформація допомагає великій мовній моделі формувати правильну і зрозумілу користувачам відповідь у контексті проблеми. Для етапів ініціації та бізнес-аналізу, виявлення вимог та побудови архітектури сервісів запропоновані відповідні моделі та методи, реалізовані в рамках загального підходу, порядок і особливості застосування яких підказує асистент.*

*Для реалізації прототипу рішення використані сучасні технології. Для наповнення бази знань застосовані структура DataFrame і бібліотека Pandas. Векторизація представлень і семантичний пошук релевантного контексту виконані за допомогою моделі all-MiniLM-L6-v2. Формування підказки з контекстом для великої мовної моделі, генерація відповіді реалізовані засобами відкритої моделі Llama 2 у вигляді Docker-сервісу.*

*Виконане експериментальне дослідження прототипу з порівнянням моделей all-MiniLM-L6-v2, paraphrase-MiniLM-L3-v2 і all-mpnet-base-v2 за такими прийнятими метриками як час генерації векторних представлень, час перетворення одного запиту користувача на вектор, час обчислення схожості між запитом та всіма векторами бази, косинусна і середня подібність підтвердило працездатність запропонованого рішення. Зріс відсоток релевантних відповідей на запити користувачів. Реалізований варіант досить ефективний у проектах з невеликою або середньою кількістю записів. Його використання у проєктах з великими базами вимагатиме для швидкого пошуку розбиття бази на тематичні блоки.*

***Ключові слова:*** *сервісний підхід; інфокомунікаційні сервіси; життєвий цикл сервісів; штучний інтелект; Large Language Model (LLM).*

*Табл.: 1. Бібл.: 31.*

**Актуальність теми дослідження.** Сервісний підхід широко впроваджується у всіх галузях діяльності людини, зокрема в галузі інфокомунікацій. Інформаційні системи (ІС) провайдерів надаються ІТ-компаніями за моделлю End-to-End (E2E) і орієнтовані на підтримку системи бізнес-процесів. Досягненню бізнес-цілей провайдерів сприяють нові методи роботи. У таких умовах інструменти підтримки діяльності провайдерів надаються за моделлю платформи підтримки життєвого циклу (ЖЦ), що дозволяє розробляти, впроваджувати і розвивати ІС-провайдерів інфокомунікаційних послуг (ІКП) [1].

Виникає потреба в ефективних моделях та методах підтримки окремих процесів ЖЦ сервісів та інтелектуальному асистенті для служб ІТ-компанії і провайдерів ІКП, здатному допомагати їм підтримувати, контролювати й поліпшувати діяльність провайдера.

Для цього доцільно використовувати велику мовну модель – Large Language Model (LLM), яка підтримуватиме ЖЦ сервісів, орієнтуючи його на досягнення цілей провайдерів шляхом використання моделей та методів підтримки окремих процесів, наприклад, запропонованих у праці [2]. Це перетворить зазначену платформу в ефективний інструмент вирішення проблем діяльності провайдера, але породжує важливу наукову проблему.

Стаття присвячена актуальній темі – розробленню інтелектуального двигуна, здатного перетворити комплекс відповідних моделей та методів процесів ЖЦ сервісів у інструмент вирішення проблем діяльності провайдера. Для цього генеративні можливості LLM треба доповнити здатністю міркувати, приймати рішення, вирішувати проблеми.

**Постановка проблеми.** Необхідно LLM і відповідні моделі та методи спрямувати на підкріплення можливостей працівників ІТ-компаній і провайдера. Інтелектуальний двигун, комунікуючи з користувачами, повинен розуміти проблеми й допомагати їх вирішувати, залучаючи відповідні інструменти (LLM, нейронні мережі, програмні компоненти). LLM структурується в рамках платформи підтримки ЖЦ сервісів, дозволяючи поєднувати її традиційні можливості з підходами до вирішення проблем шляхом їх декомпозиції.

При розв'язанні проблеми необхідно врахувати специфіку галузі інфокомунікацій.

По-перше, необхідно розробити моделі, методи та інструменти підтримки множини взаємопов'язаних процесів, згрупованих в етапи ЖЦ сервісів та їх інтеграції з врахуванням бізнесового та операційного рівнів, виділенням бізнес-процесів, додатків і технологій [3].

По-друге, оскільки ресурси зосереджені в ІТ-інфраструктурі, надання сервісів визначеного рівня вимагає моделей та методів оптимального використання її ресурсів.

По-третє, при використанні моделі E2E ІТ-компанії повинні постійно удосконалювати підтримувані ними ІС-провайдерів, щоб надавати найбільш дешевим способом вчасно якісні послуги і запобігти втратам у всьому ланцюжку – клієнт-провайдер-ІТ-компанія.

По-четверте, орієнтований на потреби користувачів розвиток, зростання конкуренції і вимог замовників вимагають забезпечення певного рівня рентабельності діяльності.

Нові умови породжують нові проблеми. Універсальним рішенням є платформа підтримки ЖЦ сервісів. Але набір її інструментів треба впорядкувати і створити відкрите середовище, комфортне для тих, хто відповідає за надання сервісів користувачам.

Вихід полягає у створенні інтелектуального двигуна – асистента працівників ІТ-компанії і провайдерів ІКП, який зрозуміє користувача і його проблему та допоможе її вирішити, використовуючи інструменти платформи підтримки ЖЦ сервісів.

**Аналіз останніх досліджень та публікацій.** Дослідженню інтелектуальних інструментів підтримки процесів ЖЦ сервісів приділяється все більше уваги. Предметом дослідження найчастіше є використання LLM у різних галузях для природомовної комунікації [4]. Однак у багатьох джерелах, наприклад [5], LLM вже використовують як засіб комунікації з користувачем, який розуміє проблеми й допомагає їх вирішити.

З метою надати LLM здатність виконувати багатоетапне міркування, планувати дії та звертатися до зовнішніх джерел даних запропоновано ряд реалізацій агентного підходу, наприклад, Chain-of-Thought, ReAct, Graph-of-Thought, AutoGPT, BabyAGI, Plan-and-Solve Prompting, Toolformer, Reflexion. Агентним рішенням притаманні багатокроковість, явне насамперед ітеративне планування наступних дій, взаємодія з різними інструментами [6].

Набуває популярності генерація з доповненим пошуком знань – Retrieval-Augmented Generation (RAG), орієнтована на один цикл отримання контексту з наступною генерацією відповіді на запит користувача [7]. Результати семантичного пошуку потрібних рядків у базі знань допомагають LLM формувати відповідь. Зовнішні знання залучаються у вигляді наперед індексованого CSV-вмісту. LLM з RAG надає правильніші відповіді на питання.

Розробленню математичних моделей та методів підтримки ЖЦ сервісів присвячені багато праць, які охоплюють як окремі процеси ЖЦ сервісів, наприклад [8], так і методологію бізнесової діяльності з погляду концепції сервісів, наприклад [9; 10]. Популярним предметом досліджень також є моделі та методи визначення параметрів рівня обслуговування, забезпечення визначеного рівня показників надання сервісів, безпеки й довіри, управління змінами [11], підтримки найважливіших етапів ЖЦ сервісів, насамперед проєктування, реалізації, надання [12], управління навантаженням і ресурсами, моніторингу та аналізу функціонування ІТ-інфраструктури з врахуванням параметрів сервісів.

**Виділення недосліджених частин загальної проблеми.** Не всі проблеми, пов'язані з впровадженням сервісного підходу, вирішено. Так, вимагає ефективного вирішення комплексна проблема підтримки процесів ЖЦ сервісів, а також проблема розроблення

ефективних інструментів – платформ, ІТ, компонентів для проєктування, реалізації і надання сервісів у галузі інфокомунікацій. При цьому перспективною видається інтеграція в рамках LLM інструментів підтримки процесів ЖЦ сервісів, побудованих на основі моделей ШІ, прийняття рішень, математичного програмування та ін.

**Метою дослідження** є встановлення ефективності RAG-підходу для інтеграції баз знань з можливостями LLM. Створене рішення при роботі з реальною базою знань повинне надавати точні відповіді на запити користувачів, мінімізуючи «галюцинації». Для досягнення цієї мети будуть перевірені такі гіпотези:

*Гіпотеза 1*. RAG-підхід забезпечить не менше 90 % правильних відповідей на питання користувачів, тоді як точність LLM без RAG-системи є набагато нижчою.

*Гіпотеза 2*. Семантичний пошук на основі векторних представлень забезпечить вищу релевантність відповідей у порівнянні з простим ключовим пошуком. У якості KPI тут будемо використовувати частку запитів, для яких знайдено правильний запис у базі знань.

*Гіпотеза 3*. Рішення на основі RAG забезпечить прийнятний час відповіді для інтерактивного використання (KPI – середній час відповіді на один запит має залишатися в інтервалі 1-5 с із врахуванням часу індексації даних, важливого для масштабування).

Сформульована мета направлена на досягнення загальної мети створення ІС-провайдерів ІКП – збільшення ефективності діяльності працівників ІТ-компаній і провайдерів ІКП, пов'язаної з підтримкою процесів ЖЦ сервісів.

Постає проблема створення інтелектуального асистента для підтримки процесів ЖЦ сервісів у ІС провайдерів інфокомунікацій з врахуванням наведеної вище специфіки галузі.

**Виклад основного матеріалу.**

**Концепція пропонованого рішення.** Структура інтелектуального двигуна наведена на рисунку. Оскільки задачі підтримки ЖЦ сервісів дуже різноманітні, для надання двигунові здатності до розв'язання проблем тут використано два популярних на сьогодні підходи – інтелектуальні агенти і RAG. Перші зручні для розв'язання традиційних комплексних проблем підтримки процесів окремих етапів. Дійсно, оркестратор виконує декомпозицію комплексної проблеми, а інтелектуальні агенти (сервіси) призначені для розв'язання окремих підпроблем. З іншого боку, змінність умов діяльності провайдерів та ІТ-компаній, поява нових регуляторних документів, методик, узагальнень досвіду вимагають розуміння проблем користувачів і пошуку релевантних відповідей на природномовні запити користувачів. Тут доцільним видається використання LLM з RAG-системою. Реалізації саме цього підходу присвячена стаття.

Стандартна LLM буде додатково структурована і навчена, щоб знати всі процеси, їх зв'язки, входи й виходи, комунікувати з користувачем природною мовою, розуміти його наміри та проблеми, генерувати релевантні відповіді на його запити. Структурування LLM доцільне з погляду ЖЦ сервісів, охоплюючи цілісну систему (LLM кореневого рівня), процеси етапів (LLM першого рівня) і т. ін.

В ІС-провайдерів ІКП LLM повинні надавати точні відповіді на велику множину специфічних запитів користувачів, які вирішують складні проблеми. Для генерування релевантних відповідей у контексті підтримки ЖЦ сервісів, сучасним LLM треба прищепити можливості спілкуватися в термінах проблем, підпроблем, причин, наслідків, передумов, варіантів рішення, вибору варіантів рішення і т. ін. Ці моделі мають бути навчені розуміти призначення кожних груп процесів, процесу і підпроцесу, їх входи, виходи, зв'язки, стан об'єктів контролю, управління, оцінювання, проблеми контролю, управління, оцінювання, засоби вирішення проблем.

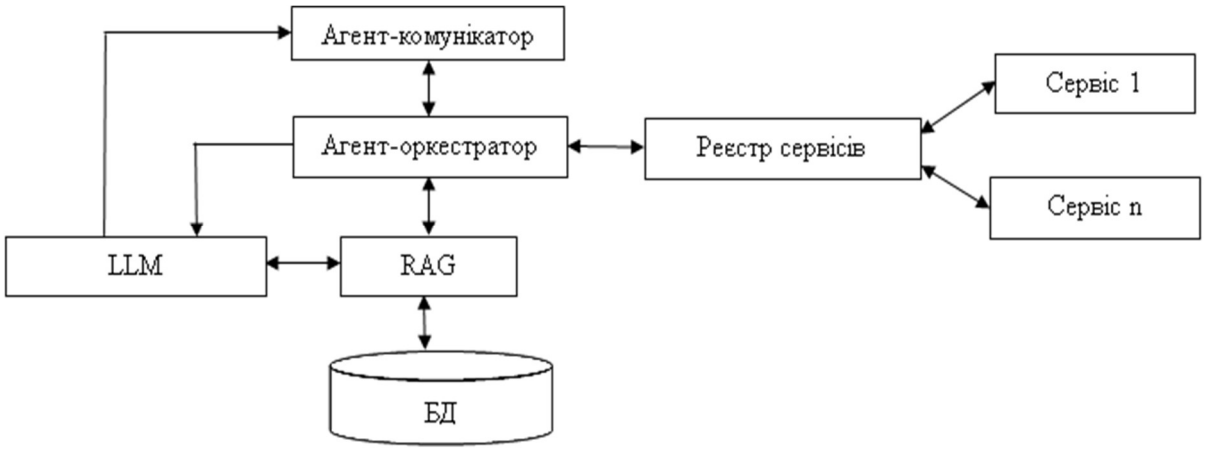


*Рис. 1. Структура інтелектуального двигуна* (посилання!)
Джерело: розроблено автором.

Щоб LLM підтримувала діалог, виконувала роль асистента у реалізації відповідного процесу чи їх групи будемо використовувати RAG-системи. Для співпраці з користувачами в термінах вирішення проблем будемо інтегрувати LLM із зовнішніми базами знань. Перед генеруванням відповіді LLM отримує від RAG-системи адекватний фрагмент інформації відповідно до поставленої проблеми структурованої і заповненої бази знань [7]. LLM формує відповідь у контексті проблеми на фактичних даних про базові сценарії і шаблони операцій для надання сервісів. Це зменшує ймовірність «галюцинацій» моделі й користувачі інтерпретують відповіді в термінах проблеми.

При такому підході універсальне розв'язання проблем через поєднання міркувань, планування та виклик зовнішніх інструментів поступається простішим методам отримання контексту при роботі з CSV-даними за рахунок точного «витягання» потрібних фактів із заданого набору даних і генерації відповіді винятково на основі отриманої підказки. Безітераційність зменшує гнучкість рішення, не дозволяє поділити складне питання на підпитання, коригувати план, якщо прямої відповіді не знайдено. Водночас гарантується вища надійність, зменшуються ризики неправильних відповідей. У такий спосіб, інтегруючи генеративні можливості LLM з можливостями RAG, інтелектуальний двигун забезпечує підтримку процесів ЖЦ сервісів у взаємодії з працівниками IT-компанії і провайдера, приводячи в дію засоби платформи підтримки сервісів.

**Реалізація перспективного інтелектуального двигуна.** Підсумовуючи викладене вище, інтегральне рішення для комплексної підтримки процесів ЖЦ сервісів за рахунок створення інтелектуального двигуна доцільно будувати на концепціях прототипування, структурування, керованого швидкого пошуку розв'язків проблем, навчання з підкріпленням, інтелектуальних агентів і RAG-систем. З використанням усіх сучасних концепцій створення інтелектуального двигуна у перспективі доцільно виконувати, реалізуючи в ітераційний спосіб підготовчий і п'ять конструктивних кроків.

Підготовчий крок полягає у виборі інструментів. Найчастіше вибирають LangChain – для побудови ланцюжка RAG, Pinecone – для векторного пошуку, Sentence-BERT – для векторних представлень та ін. Прототипування передбачає перевірку різних варіантів.

Перший крок – це побудова LLM, позбавленої відомих обмежень. Так, запобіганню нестійкості моделі під час додавання нових навчальних матеріалів сприятиме концепція самовдосконалення з поєднанням дослідження (exploration) та експлуатації (exploitation) і стратегія зовнішніх винагород. Генеративні можливості LLM розширимо засобами розв'язання проблем за рахунок динамічного налаштування конфігурацій параметрів моделі для підтримки балансу, впровадження відповідних метрик, методів оцінювання і оптимізації [13]. Щоб використовувати LLM для інтерпретації, розуміння і використання

графічних елементів, які визначають порядок дій, вхідні і вихідні дані, іншу корисну для підтримки ЖЦ сервісів інформацію навчимо LLM перетворювати графічні зображення у структуровані машинозчитувані дані, які визначають схеми розв'язання проблем.

На другому кроці LLM інтегрується з RAG, компонентами якої є база знань та пошуковий і генеративний модулі. Щонайменше варто перевірити векторні бази даних Pinecone, Chroma або Milvus. Для пошуку в них релевантних документів і перетворення запитів користувачів у векторне представлення доцільно перевірити принаймні Sentence-BERT та OpenAI embeddings. Для створення текстових відповідей на основі знайдених пошуковим модулем даних варто перевірити такі сучасні моделі, як GPT-4, LLaMA 2, Anthropic Claude. Побудова бази знань, збір релевантних документів, їх індексація у векторній базі платформи підтримки ЖЦ сервісів мають враховувати всі відомі джерела даних – технічну документацію, бізнес-звіти, історичні дані, специфікації тощо. Адаптація моделі до доменної області ослабить обмеження у розумінні контексту. Для підвищення якості вмісту бази знань використовуватимемо регулярні оновлення та очищення. Зменшенню затримки пошуку сприятиме кешування для повторюваних запитів.

На третьому етапі створюється система інтелектуальних агентів для підтримки здатності інтелектуального двигуна розв'язувати проблеми користувачів. Перспективним вбачається підхід, побудований на основі моделей математичної логіки.

На четвертому етапі виконується навчання моделей системи, налаштування компонентів. Для пошукового компонента будемо виконувати оптимізацію пошукових запитів, ранжувати результати з використання метрик семантичної схожості. Для генеративного варто перевірити техніки fine-tuning і Промпт Інжинірінгу, інтегрувати механізми валідації результатів.

На п'ятому етапі виконується тестування системи. Щонайменше варто оцінювати точність відповіді, наприклад, з використанням метрики *Precision@k, продуктивність та час відповіді, перевірити працездатність концепцій і прийняти рішення про наступні кроки реалізації інтелектуального двигуна в ітеративному процесі прототипування.

Отже, концепція прототипування, покладена в основу сучасних методологій створення ІС дозволить реалізацію інтелектуального двигуна виконувати поступово, перевіряючи крок за кроком працездатність концепцій та їх ефективність. Спочатку доцільно створити прототип, інтегруючи прості моделі. Потім необхідно перевіряти й поліпшувати ключові здатності двигуна, використовуючи інші сучасні моделі та їх різні поєднання.

Перейдемо до перевірки працездатності прототипу інтелектуального двигуна. Розвиток прототипу буде предметом дослідження у наступних статтях.

**Побудова й експериментальне дослідження прототипу.**

*Основні концепції*. Використовується модель LLaMA 2, взаємодія LLM і RAG базується на процесах перетворення даних CSV у текстову базу знань, семантичного пошуку для отримання релевантного контексту, надання зазначеного контексту LLM для генерації відповіді. Основні кроки роботи системи можна представити у такий спосіб:

1. *Підготовка та завантаження даних у форматі CSV* – вміст CSV-файлу імпортується у структуру даних (DataFrame) із використанням бібліотеки Pandas.

2. *Обчислення семантичних векторних представлень* – для текстових фрагментів бази знань генерується векторне представлення, що відображає семантику тексту.

3. *Семантичний пошук релевантного контексту* для відповіді на запитання користувача. На основі косинусної міри подібності між векторним представленням тексту запитання користувача і векторами всіх записів бази знань записи упорядковуються за релевантністю. Вибирається запис із найбільшою косинусною мірою подібності.

4. *Формування підказки з контекстом для LLM* – вибрані текстові фрагменти (контекст) об'єднуються у підказку, яка передається LLM для формування відповіді.

5. *Виклик локальної мовної моделі для генерації відповіді* – підказка надсилається до LLM. Модель опрацьовує підказку, генерує підсумкову відповідь лише на основі наданих даних і повертає користувачеві. Передбачена ситуація, коли потрібна інформація відсутня.

*Дані*: CSV-файл з даними про продукти IBM, етапи їх ЖЦ. Таблиця містить 21 колонку з характеристиками (назва продукту, версія, дати їх анонсу, зняття з продажу, припинення підтримки тощо) і 13 506 рядків. Щоб зберегти контекст, кожен рядок CSV було перетворено у текстовий документ формату «Назва колонки: значення». База знань – список текстових фрагментів, кожен з яких відповідає одному продукту з його атрибутами.

*Модель та інфраструктура*: модель Llama 2 (версія від Meta), що в якості LLM генерує відповіді, розгорнута локально у режимі чату через Docker-сервіс з API. Інтерфейс сумісний з OpenAI, запити формуються як до OpenAI API, але обробляються локально. Це забезпечує конфіденційність – дані CSV і запити не надсилаються стороннім сервісам. Семантичний пошук реалізовано за допомогою моделі перетворення тексту у векторні представлення – з бібліотеки Sentence Transformers. Для порівняння протестовано три моделі представлення: all-MiniLM-L6-v2 (384-вимірна модель, збалансована за швидкістю та якістю); paraphrase-MiniLM-L3-v2 (полегшена модель з доступною швидкістю, яка поступається в точності семантичних збігів); all-mpnet-base-v2 (більша модель із високою точністю визначення семантичної схожості, але суттєво повільніша за попередні).

Усі 13,5k записів CSV одноразово трансформовані у векторну форму на початку експерименту. З 384-вимірних векторів формується векторний простір, семантично подібні тексти якого відповідають близьким векторним точкам. Модель ембеддінгів вибрано на основі балансу між швидкодією та точністю пошуку. Модель all-MiniLM-L6-v2 генерує вектори для всього корпусу за 4,8 с, тоді як all-mpnet-base-v2 – за 12,6 с. Час оброблення одного запиту (векторизація і пошук) моделлю all-MiniLM-L6-v2 становив 0,20 с (0,12 с на представлення запиту і 0,08 с на пошук), що майже вдвічі швидше, ніж моделлю all-mpnet-base-v2 (0,36 с сумарно). Водночас якість семантичного пошуку в all-mpnet-base-v2 дещо вища: Top-1 схожість – 0,91 проти 0,88 у all-MiniLM-L6-v2 (за шкалою 0-1). З огляду на ці результати було обрано модель all-MiniLM-L6-v2 як така, що забезпечує кращий баланс.

*Процедура тестування*: Для оцінювання точності було підготовлено набір із 30 тестових запитань користувачів таких типів:

- отримання конкретного атрибута продукту (наприклад, *«Яка дата закінчення підтримки сервера IBM Power модель X?»* на основі значень колонки End of Support);
- пошук продукту за назвою або кодом (наприклад, *«Коли вийшла версія 1.1.0 IBM Storage Suite for Cloud Paks?»* – потрібно знайти рядок з цією назвою та версією і вибрати дату);
- узагальнюючі питання, що вимагають опрацювання кількох релевантних записів (наприклад, *«Які продукти були зняті з продажу у 2020 році?»* – у відповіді очікується перелік або пояснення на основі знайдених даних);
- запитання *поза базою знань* для перевірки поведінки системи (наприклад, питання про неіснуючий продукт або про інформацію, якої гарантовано немає в таблиці).

Кожен запит тестувався у двох режимах: *RAG-режим* із повним циклом генерації відповіді (семантичний пошук, контекст і LLM); *Базовий режим* з генерацією відповіді LLM без надання контексту на основі власних знань (це базова лінія (baseline) для порівняння точності). Відповіді моделі фіксувалися для оцінювання коректності.

*Умови тестування*: автономне середовище без доступу до інтернету. Llama 2 працювала на апаратному забезпеченні із GPU NVIDIA (Tesla T4) для пришвидшення генерації тексту та обчислення векторних представлень. Семантичний пошук здійснювався

в оперативній пам'яті. Щоб успішність відповідей у RAG-режимі визначалася лише релевантним контекстом, а не «знаннями» моделі, перед початком тестування було перевірено, що модель не була попередньо навчена на цьому датасеті.

Для оцінювання точності результати перевірялися вручну: відповідь правильна, якщо відповідала фактам з таблиці CSV; відповідь неправильна, якщо містила фактологічні помилки або відсилала до даних, яких немає у CSV. Якщо модель відповіла на запити четвертого типу, що даних немає, відповідь теж приймалася як правильна (відсутність галюцинації). Після отримання усіх відповідей, за формулою Вілсона для біноміальної пропорції, обчислено відсоток правильних відповідей та 95 % довірчий інтервал. Для відокремлення помилок пошуку від помилок генерації для кожного запиту фіксувалося, чи був знайдений правильний запис – топ-3 контекст містив необхідну інформацію.

**Результати експерименту**

*Точність відповідей*: Із 30 запитань 27 були правильно оброблені системою, що становило 90 %. З них 27 випадків модель у 25 відповіла фактично правильно, використовуючи надані фрагменти даних, а у 2 випадках коректно повідомила, що потрібної інформації немає (наприклад, на питання про продукт, відсутній у таблиці, модель відповіла приблизно: «На жаль, даних про такий продукт немає в базі» – що ми вважаємо правильною поведінкою). Три помилки (10 %) сталися на запитах, де модель або некоректно інтерпретувала знайдені дані, або пошук повернув близькі, але не ті записи. Наприклад, на питання про дату зняття з підтримки одного з продуктів модель помилилася на 1 рік, імовірно через те, що в контекст потрапили дані про схожий продукт іншої версії. Отримана точність підтверджує гіпотезу 1, адже LLM без контексту не впоралася з жодним із цих запитів. Обчислений для показника 90 % точності 95 % довірчий інтервал становив приблизно 10 % при n = 30. Це означає, що з імовірністю 0,95 точність лежить у межах 80-100 %. Це доволі широкий інтервал через відносно невеликий обсяг тестів. При розширенні тестового набору статистична похибка зменшиться. Отже, беручи до уваги нульову гіпотезу повної випадковості (0 % успіху), можна стверджувати, що точність перевищує поріг у 50 %.

*Рівень галюцинацій*: галюцинації не зафіксовано. Модель давала відповіді строго на основі знайдених фактів або правильно вказувала на відсутність даних. Отже, RAG-підхід значно знижує частоту вигаданих відповідей. Для порівняння, в режимі без контексту LLM «вигадувала» відповіді – наприклад, на питання про дату закінчення підтримки вигаданого продукту модель спершу дала ймовірну, але неправильну відповідь (галюцинацію).

*Ефективність семантичного пошуку*. В усіх 30 випадках потрібна інформація, якщо вона існувала в базі, була серед перших трьох фрагментів контексту. Тобто повернення по базі знань становило 100 % для тестових запитів і жоден запит з відповіддю, що наявна у CSV, не залишився без релевантного фрагмента в контексті. При цьому у 26 випадках правильний запис стояв першим (Top-1), ще у 1 випадку – другим. Для обраної моделі all-MiniLM-L6-v2 середній показник Top-1 similarity для запитів склав близько 0,87, а Top-3 average similarity близько 0,84. Тобто найрелевантніший документ має дуже високий косинусний перетин із запитом, а два наступні теж доволі близькі. На практиці такі значення підтверджують надійність – у векторному просторі запит знаходився близько до правильного рядка і той суттєво відрізнявся від нерелевантних (схожість інших записів різко спадала). Для ілюстрації, семантичний пошук зміг правильно зіставити запит «BladeCenter HS22» із записом "BladeCenter HS22 (7809-H22) – Withdrawal notification", навіть якби у питанні не було точного збігу за деякими полями. Ключова перевага – стійкість до формулювань запиту: система знайде потрібний запис навіть якщо користувач використовує синоніми чи неповну назву, чого не гарантує ключовий пошук.

*Часові результати*. Середній загальний час отримання відповіді прототипом становив близько 2-3 секунд. Цей час включає близько 0,2 с на семантичний пошук (0,12 с на перетворення запиту і 0,08 с на пошук по більш ніж 13k записах) та додатково близько 2 с на генерацію тексту LLM (довжина відповіді зазвичай становить 1-2 речення). Затримка знаходиться в комфортних межах для користувача. Щодо часу індексації – 13 506 рядків CSV перетворюються на вектори за близько 5 секунд на GPU для обраної моделі. Тобто навіть при оновленні бази знань чи запуску прототипу на новому датасеті оброблення 10-15 тисяч записів займе лічені секунди, що цілком прийнятно. Часові оцінки свідчать, що прототип можна масштабувати на більші об'єми даних, проте варто врахувати лінійне зростання часу пошуку. Наприклад, для 135 000 записів (у 10 разів більше) пошук займав би близько 0,8 с, що все ще допустимо. За потреби можна використати спеціалізовану векторну базу, наприклад, ChromaDB, Faiss для швидшого пошуку на мільйонах документів. Але при 10-15 тисяч записів прототип продемонстрував добру продуктивність.

**Порівняння з базовими підходами**

Для повноти оцінки результати RAG-методу були зіставлені з двома підходами.

LLM без доступу до знань (закрита книга) без наданого контексту не відповіла правильно на питання. Лише 2 відповіді з 30 можна було вважати частково правильними. Наприклад, на питання про рік анонсу деякого старого продукту модель назвала 2015, що збіглося з реальною датою. Решта 28 відповідей цієї моделі були неправильними або занадто загальними. Таким чином, точність LLM без доступу до знань становила близько 7 % проти 90 % для прототипу. Це узгоджується з висновками інших дослідників.

Ключовий пошук по тексту таблиці CSV за допомогою індексування всіх текстових даних і пошуку рядків, які містять слова запиту, на тих же запитаннях продемонстрував кращі показники, ніж LLM без доступу до знань, але значно гірші ніж прототип. У простих випадках (назва продукту або код точно згадані в питанні) він дозволяє знайти відповідний запис. Проте його ефективність різко падає, якщо запит не збігається буквально з текстом у таблиці. Наприклад, запит *«Чи є в назві продукту X слово Storage?»* вимагав би розпізнати синонімію чи логічний зв'язок, що не притаманне лексичному пошуку. Ключовий підхід відпрацював некоректно приблизно 20-25 % запитів, тоді як семантичний пошук дав релевантні результати для всіх. Цим підтверджено *гіпотезу 2*: використання семантичних векторів дозволяє «розуміти» запитання користувача і знаходити інформацію, навіть якщо слова не збігаються точно.

Вибір моделі перетворення визначають вимоги до швидкості і точності. Модель all-MiniLM-L6-v2 в експерименті була найкращою – у 5 разів швидшою за all-mpnet-base-v2 в індексації і вдвічі швидшою в обробленні запиту за мінімальної втрати точності пошуку (схожість Top-1 становила 0.88 проти 0.91). Отже, для середніх обсягів даних можна обрати легшу модель без суттєвого погіршення якості відповіді. Більш повільні моделі доцільно використовувати лише якщо пріоритетом є максимальна точність і дозволяє час.

Дані експериментів наведені у таблиці 1.

Використані для експерименту моделі – all-MiniLM-L6-v2, paraphrase-MiniLM-L3-v2, all-mpnet-base-v2 – описані вище. Вони оцінюються за такими показниками:

1) час генерації векторних представлень (Embedding Time (Corpus)) для всіх рядків CSV у секундах. Цей показник важливий для оцінки масштабованості рішення;

2) час перетворення одного запиту користувача на вектор (Embedding Time (Query)) у секундах. Це ключовий параметр, якщо користувач очікує швидкої відповіді;

3) час обчислення схожості між запитом та всіма векторами бази (Search Time). Фактично цей показник репрезентує швидкість пошуку релевантного контексту.

4) значення косинусної подібності (Top-1 Similarity) для найбільш релевантного запису. Чим ближче його значення до верхньої границі інтервалу (0, 1), тим більший збіг;

5) середня подібність (Top-3 Avg Similarity) для трьох найрелевантніших записів. Показник свідчить про один сильний збіг, чи кілька добрих, що важливо для оцінювання стабільності та надійності результатів.

*Таблиця 1 – Експериментальні дані дослідження прототипу*

| Назва моделі | Час генерації векторних представлень, с | Час перетворення запиту на вектор, с | Search Time, с | Top-1 Similarity | Top-3 Avg Similarity |
|---|---|---|---|---|---|
| all-MiniLM-L6-v2 | 4,8 | 0,12 | 0,08 | 0,88 | 0,85 |
| paraphrase-MiniLM-L3-v2 | 3,2 | 0,09 | 0,07 | 0,82 | 0,79 |
| all-mpnet-base-v2 | 12,6 | 0,24 | 0,12 | 0,91 | 0,89 |

Джерело: розроблено автором.

Дані експериментів свідчать, що ефективність взаємодії LLM і RAG-системи залежить від вибраних векторних представлень і метрики подібності. Модель all-MiniLM-L6-v2, використана для семантичного кодування, підтримує баланс між швидкістю та якістю. Вона демонструє вищу, ніж інші моделі швидкість при високих семантичних оцінках. Підтверджено, що реалізований варіант взаємодії LLM і RAG є ефективним у проєктах з CSV-файлами, які мають невелику або середню кількість записів. Його використання у проєктах з великими CSV-файлами вимагатиме для швидкого пошуку використання спеціалізованих векторних баз даних або розбиття даних на тематичні блоки.

Прототип рішення з використанням даних у форматі CSV підтвердив переваги підходу, коли на запит користувача система надає релевантну відповідь з генеративним поясненням, спираючись на результати семантичного пошуку у зовнішньому джерелі в контексті проблеми користувача. Це уможливлює створення інтелектуальних асистентів, які допоможуть розв’язувати проблеми користувачів із використанням внутрішніх даних провайдера, забезпечуючи вимоги як до точності, так і конфіденційності системи.

Продемонстрована ефективна інтеграція взаємодії LLM і RAG-системи забезпечує підвищення ефективності використання накопичених даних і розширення сфери застосування LLM і RAG-систем для підтримки процесів ЖЦ сервісів.

**Висновки.** У статті розглянута проблема розроблення інтелектуального двигуна як основи інтеграції засобів автоматизації процесів ЖЦ сервісів, побудови моделей та методів для розроблення інструментів платформи підтримки ЖЦ сервісів у ІС-провайдерів ІКП.

Запропоновано загальний підхід до вирішення проблеми. Інтелектуальний двигун, реалізований на основі відповідно структурованої і навченої LLM та RAG-системи, виступає в ролі асистента команди підтримки ЖЦ сервісів.

Проведене дослідження кількісно продемонструвало переваги інтеграції зовнішніх табличних даних з LLM через RAG-підхід. Операціоналізована мета досягнута – отримано близько 90 % точних відповідей при нульових випадках галюцинацій. Ключові показники ефективності – точність відповідей, повнота пошуку, затримка – знаходяться у задовільних межах. Це підтверджує висновок – зниження частки помилкових відповідей та вигаданих фактів до нуля є наслідком надання актуального контексту моделі. Підтверджено, що навіть відносно простий семантичний пошук (на основі MiniLM) здатен забезпечити повне покриття інформаційних запитів по базі знань, тоді як традиційні підходи поступаються.

Без доповнення знаннями LLM генерувала неправильні відповіді у приблизно 93 % випадків. Поєднання LLM із зовнішньою базою знань різко підвищило продуктивність, узгоджуючись із сучасними дослідженнями в цій галузі. Підтверджено як гіпотезу 1, так і гіпотезу 2 – завдяки векторному представленню досягнуто 100 % пошуку необхідних даних у таблиці, незалежно від формулювань запиту, що було б складно для простого пошуку.

Окрім того, експериментальне дослідження дозволило зробити висновок про те, що ефективність LLM з RAG-системою дуже залежить від якості векторних представлень і вибору відповідної моделі, при цьому швидкодія системи відповідає інтерактивним вимогам – затримка пошуку складає менше ніж 0,5 с при 13k документів, генерація відповіді вимагає близько 2 с, що є цілком прийнятним. Встановлено, що збалансована модель MiniLM забезпечує компроміс, але в інших доменах можливо варто пробувати кілька варіантів перетворень та метрик схожості. Базова реалізація показала добру масштабованість для середніх датасетів, але при зростанні обсягів даних доцільним буде використання розвинених рішень – векторних баз даних, шардування даних тощо.

Отже, експериментальна перевірка підтвердила працездатність запропонованого рішення, перспективи його розвитку і практичне значення інтеграції RAG-системи з можливостями LLM. Це допомагає LLM розв'язати більше проблем підтримки процесів усіх етапів ЖЦ сервісів, при цьому значно рідше генеруючи неправильну або неактуальну інформацію. Отримані кількісні результати точності та інших показників, підтверджені гіпотези, зручність подачі інформації користувачу підтверджують, що інтеграція структурованої бази знань і LLM відкриває можливості створення приватних помічників та аналітичних систем, які здатні відповідати на вузькоспеціалізовані питання, спираючись на внутрішні дані організацій, забезпечуючи при цьому і точність, і конфіденційність.

Запропонований підхід дозволить працівникам ІТ-компанії і провайдера аналізувати проблеми і визначати вимоги до ІС, будувати архітектуру сервісів, виконувати завдання, пов'язані з підтримкою процесів ЖЦ сервісів.

## Список використаних джерел

1. Чимшир, В., Теленик, С., Ролік, О. & Жаріков, Е. (2023). Платформа підтримки життєвого циклу сервісів в інформаційних системах провайдерів інформаційно-комунікаційних послуг. *Міжвідомчий науково-технічний збірник Адаптивні системи автоматичного управління*, *1 (42)*, 65-78. https://doi.org/10.20535/1560-8956.42.2023.279172

2. Chymshyr, V., Zhdanova, O., Havrylenko, O., Nowakowski, G. & Telenyk, S. (2024). Models and methods for forming service packages for solving of the problem of designing services in information systems of providers. *Inf. Comput. and Intell. syst. j., 2*, 29-54. https://doi.org/10.20535/2786-8729.5.2024.316432

3. Morelli, N., De Götzen, A., & Simeone, L. (2020). Service design capabilities (Springer Series in Design and Innovation, Vol. 10). Springer. https://doi.org/10.1007/978-3-030-56282-3.

4. Zubiaga, A. (2024). Natural language processing in the era of large language models. Frontiers in Artificial Intelligence, 6, 1-5. https://doi.org/10.3389/frai.2023.1350306.

5. Nagavalli, S. P., Tiwari, S., & Sarma, W. (2024). Large language models and NLP: Investigating challenges, opportunities, and the path to human-like language understanding. International Research Journal of Engineering and Technology, 12(11), 571-584.

6. Lu, P., Chen, B., Liu, S., Thapa, R., Boen, J., & Zou, J. (2025). OctoTools: An agentic framework with extensible tools for complex reasoning. In *ICLR 2025 Workshop on Foundation Models in the Wild*. https://doi.org/10.48550/arXiv.2502.11271.

7. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems, 33*, 9459-9474. https://doi.org/10.48550/arXiv.2005.11401.

8. Tanko Ishaya. (2012). A service oriented approach to Business Intelligence in Telecoms industry. *Telematics and Informatics*, *3(29)*, 273-285. https://doi.org/10.1016/j.tele.2012.01.004.

9. Daidj, N. (2022). Towards renewed Business-IT alignment models in the digital era: The impact of data inclusion. *Information Resources Management Journal, 35*(1), 1-13. https://doi.org/10.4018/IRMJ.298972.

10. MacLean, D. & Titah, R. (2023). Implementation and impacts of IT Service Management in the IT function. *International Journal of Information Management*, *70*, 1-53. https://doi.org/10.1016/j.ijinfomgt.2023.102628.

11. Sun, Y., White, J., Eade, S., & Schmidt, D. C. (2016). ROAR: A QoS-oriented modeling framework for automated cloud resource allocation and optimization. *Journal of Systems and Software, 116*, 146-161. Elsevier BV. https://doi.org/10.1016/j.jss.2015.08.006.

12. Tuunanen, T., Salo, M., & Li, F. (2023). Modular service design of information technology-enabled services. *Journal of Service Research*, *26(2)*, 270-282.

13. Wan, J., Song, S., Yu, W., Liu, Y., Cheng, W., Huang, F., et.al. (2024). Omniparser: A unified framework for text spotting key information extraction and table recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 15641-15653).

## References

1. Chymshyr, V., Telenyk, S., Rolik, O., Zharikov, E. Platforma pidtrymky zhyttievoho tsyklu servisiv v informatsiinykh systemakh provaideriv informatsiino-komunikatsiinykh posluh [Platform for supporting the life cycle of services in information systems of information and communication service providers]. *Mizhvidomchyi naukovo-tekhnichnyi zbirnyk Adaptyvni systemy avtomatychnoho upravlinnia – Interdepartmental scientific and technical collection Adaptive systems of automatic control, 1 (42)*, 65-78. DOI: https://doi.org/10.20535/1560-8956.42.2023.279172.

2. Chymshyr, V., Zhdanova, O., Havrylenko, O., Nowakowski, G. & Telenyk, S. (2024). Models and methods for forming service packages for solving of the problem of designing services in information systems of providers. *Inf. Comput. and Intell. syst. j., 2*, 29-54. DOI: 10.20535/2786-8729.5.2024.316432.

3. Nicola Morelli, Amalia de Götzen & Luca Simeone. (2020). Service Design Capabilities, Springer Series in Design and Innovation, vol. 10, 89 p. https://doi.org/ 10.1007/978-3-030-56282-3_8.

4. Arkaitz Zubiaga. (2024). Natural language processing in the era of large language models / Frontiers in Artificial Intelligence, 6, 1-5. doi.org/10.3389/frai.2023.1350306.

5. Sudarshan Prasad Nagavalli, Sundar Tiwari, Writuraj Sarma. (2024). Large Language Models and NLP: Investigating Challenges, Opportunities, and the Path to Human-Like Language Understanding. *International Research Journal of Engineering and Technology*, *12(11)*, 571-584.

6. Lu, P., Chen, B., Liu, S., Thapa, R., Boen, J. & Zou, J. (2025). OctoTools: An Agentic Framework with Extensible Tools for Complex Reasoning. *Proceedings of the ICLR 2025 Workshop on Foundation Models in the Wild.* https://doi.org/10.48550/arXiv.2502.11271.

7. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., et. al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems, 33. 9459-9474. https://doi.org/10.48550/ arXiv.2005.11401.

8. Tanko Ishaya. (2012). A service oriented approach to Business Intelligence in Telecoms industry. *Telematics and Informatics*, *3(29)*, 273-285. https://doi.org/10.1016/j.tele.2012.01.004.

9. Nabyla Daidj. (2022). Towards Renewed Business-IT Alignment Models in the Digital Era:The Impact of Data Inclusion. *Information Resources Management Journal*, *1(35)*, 1-13. DOI: 10.4018/IRMJ.298972.

10. Don MacLean, Ryad Titah. (2023). Implementation and impacts of IT Service Management in the IT function. *International Journal of Information Management*, *70*, 1-53. https://doi.org/10.1016/ j.ijinfomgt.2023.102628.

11. Sun, Y., White, J., Eade, S., Schmidt, D. C. (2016). ROAR: A QoS-oriented modeling framework for automated cloud resource allocation and optimization. *Journal of Systems and Software*, *116*, 146-161. Elsevier BV. https://doi.org/10.1016/j.jss.2015.08.006.

12. Tuunanen, T., Salo, M., Li, F. (2023). Modular service design of information technology-enabled services. *Journal of Service Research*, *26(2),* 270-282.

13. Wan, J., Song, S., Yu, W., Liu, Y., Cheng, W., Huang, F., et.al. (2024). Omniparser: A unified framework for text spotting key information extraction and table recognition. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 15641-15653).

Отримано 04.09.2025

TECHNICAL SCIENCES AND TECHNOLOGIES

*UDC 004.75*

***Viacheslav Chymshyr*[1]**

[1]PhD student, Department of Information Systems and Technologies
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute" (Kyiv, Ukraine)
**E-mail:** vchimshir@gmail.com. **ORCID:** https://orcid.org/0009-0005-2555-5373

# INTELLIGENT ENGINE AND MODELS AND METHODS OF THE PLATFORM FOR COMPREHENSIVE SUPPORT OF THE LIFE CYCLE PROCESSES OF INFORMATION AND COMMUNICATION SERVICES

*The problem of developing an intelligent engine as a basis for integrating tools for automating service life cycle processes, models and methods for developing initiation and business analysis tools for a service life cycle support platform in information systems of providers is considered. The general approach to supporting processes at all stages of the service life cycle is proposed. The intelligent engine acts as an assistant to the service life cycle support team. The intelligent engine is implemented based on the suitably structured and trained large language model. To communicate with users in the context of problems, the large language model is integrated with external knowledge bases using a generation system with augmented knowledge search. Information structured in accordance with the problem helps the large language model to form a correct and understandable answer for users considering the problem. For the stages of initiation and business analysis, requirements identification and service architecture construction, appropriate models and methods are proposed, implemented within the framework of the general approach, the procedure and features of which are prompted by the assistant.*

*Modern technologies were used to implement the solution prototype. The DataFrame structure and Pandas library were used to populate the knowledge base. Vectorization of representations and semantic search of relevant context were performed using the all MiniLM L6 v2 model. The formation of a prompt with context for a large language model, response generation were implemented using the open Llama 2 model in the form of a Docker service.*

*An experimental study of the prototype was performed with a comparison of the all-MiniLM-L6-v2, paraphrase-MiniLM-L3-v2 and all-mpnet-base-v2 models according to these accepted metrics as the time of generation of vector representations, the time of conversion of one user query into a vector, the time of calculation of similarity between the query and all vectors of the base, cosine and average similarity confirmed the operability of the proposed solution. The percentage of relevant responses to user queries increased. The implemented option is quite effective in projects with a small or medium number of records. Its use in projects with large databases will require dividing the database into thematic blocks for quick searching.*

***Keywords:*** *service approach; infocommunication services; service life cycle; artificial intelligence; Large Language Model (LLM).*

*Table: 1. References: 31.*

Чимшир В. І. Інтелектуальний двигун і моделі та методи платформи комплексної підтримки процесів життєвого циклу інфокомунікаційних сервісів. *Технічні науки та технології,* 2025. № 3(41). С. 284-295. DOI: https://doi.org/10.25140/2411-5363-2025-3(41)-284-295.