DOI 10.31891/2307-5732-2025-347-12
УДК 004.94

ВИШНЕВСЬКИЙ ОЛЕКСАНДР
Національний університет "Львівська політехніка"
https://orcid.org/0009-0005-4857-9669
e-mail: oleksandr.k.vyshnevskyi@lpnu.ua
ЖУРАВЧАК ЛЮБОВ
Національний університет "Львівська політехніка"
https://orcid.org/0000-0002-1444-5882
e-mail: liubov.m.zhuravchak@lpnu.ua

РОЗРОБЛЕННЯ ПРОГРАМНОЇ СИСТЕМИ АНАЛІЗУ ЕНЕРГОВИТРАТ З ВИКОРИСТАННЯМ ОНТОЛОГІЧНОГО ПІДХОДУ ТА ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ

У статті розглянуто потенціал використання великих мовних моделей для взаємодії людини з цифровим двійником, яким змодельовано систему енергоспоживання будівлі. Огляд наявних наукових праць показав, що застосування великих мовних моделей до вивчення енергоощадності будівель висвітлено недостатньо, хоч кількість досліджень в цій галузі швидко зростає. Розроблено підхід для побудови цифрової моделі енергоспоживання будівлі, який ґрунтується на використанні запропонованої нами раніше онтології, графової бази даних та великих мовних моделей, що дозволило отримувати дані про ефективність використання енергії через інтерфейс прямого спілкування користувача з базою знань.

У рамках дослідження побудовано базу знань з доменними знаннями про взаємозв'язки між будівлями, лічильниками, показниками споживання, метеостанціями і кліматичними даними, ґрунтуючись на графовій базі Neo4j і онтологічній моделі. Розроблено прототип чат-боту, побудованого з використанням фреймворку Lang Chain, агентного підходу і технології Retrieval Augmented Generation, проаналізовано здатність моделі Gpt-4o-mini відповідати на запити, пов'язані з енергоспоживанням, використовуючи семантичні зв'язки в схемі побудованої бази знань, автоматично генерувати запити Cypher до бази Neo4j, виконувати їх і давати відповіді, враховуючи додатковий контекст за основі сформованих підказок для агента.

Запропонований для побудови системи аналізу енерговитрат онтологічний підхід, який базується на знаннях про особливості доменної області, має формалізовану структуру і описує, як подати знання придатними для машинного читання. Він може допомогти людям зрозуміти цю систему за допомогою взаємодії з нею природною мовою, забезпечити краще планування та ухвалення рішень. Встановлено, що цифрові двійники моделюють таку систему та допомагають в керуванні нею, визначаючи аномалії, здійснюючи прогнози та знаходячи неполадки в її роботі.

Ключові слова: енергоспоживання, онтологія, граф знань, цифровий двійник, графова база даних, neo4j.

VYSHNEVSKYY OLEKSANDR, ZHURAVCHAK LIUBOV
Lviv Polytechnic National University

CONSTRUCTION OF AN ENERGY CONSUMPTION ANALYSIS SOFTWARE SYSTEM USING AN ONTOLOGICAL APPROACH AND LARGE LANGUAGE MODELS

This study aims to investigate the potential of using large language models for interaction between human and a digital twin that models a building's energy consumption management system. A review of existing works has shown that the literature on the application of large language models to the field of buildings and energy is currently quite limited, but research in this area is growing rapidly.

The work develops an approach for building a buildings energy usage digital twin based on the use of the proposed ontological model, a graph database, and large language models, which allows obtaining results on energy efficiency through a direct user communication interface with the knowledge database.

As part of the study, a knowledge database with domain knowledge about the relationships between buildings, meters, meter readings and consumption, weather stations, and climate data was built using Neo4j graph database and the ontological model we proposed earlier. A prototype of a chatbot built using the Lang Chain framework, an agent approach, and Retrieval Augmented Generation technology was developed, and the ability of the Gpt-4o-mini model to respond to queries related to energy consumption using semantic relationships in the schema of the built knowledge database, to automatically generate Cypher queries to the Neo4j database, execute them, and respond considering additional context, using defined prompts for the agent, was analysed.

The paper presents the results of the research, the analysis showed that the proposed ontological approach for building an energy consumption analysis system, which is based on knowledge about the features of the domain area, offers a formalized structure and description of the system for representing knowledge, suitable for machine reading, can help people understand the system through natural language interaction, providing better planning and decision-making. It was established that digital twins can model, simulate, help manage the energy consumption system, identify anomalies, predict and identify problems in the system.

Keywords: energy, ontology, knowledge graph, digital twin, graph database, neo4j.

Вступ

Онтології стали основою при моделюванні процесів у різних галузях, фундаментально трансформуючи спосіб опису фізичних систем. За рахунок структурованого представлення онтології полегшують взаємодію, інтеграцію даних і автоматизацію різних завдань управління будівлею. Семантичні онтології пропонують формалізовану структуру для представлення знань, придатну для машинного читання, уможливлюючи структурований опис складних систем [1]. У галузі енергоспоживання будівель впровадження онтологій змінило спосіб моделювання процесів у системах будівель, забезпечивши стандартизовану сумісну мову. Складність навчання, пов'язана з розробкою

семантичних моделей і запитами, є суттєвою проблемою внаслідок трудоємності процесу і вимоги наявності спеціалізованого досвіду. Семантичні моделі спрощують розробку та розгортання додатків, надаючи стандартизовану структуру для розуміння просторових і функціональних зв'язків між обладнанням, що дозволяє робити узагальнення щодо виявлення несправностей чи діагностику й оптимізацію управління. Деякі дослідники використовували вимірювання часових рядів для точнішої класифікації певних даних датчиків. Ці підходи добре працюють для визначення класу кожного сенсора та надання інформації про зв'язки між сенсором та відповідним обладнанням. Набагато менше досліджено підходи на основі даних до визначення просторових і функціональних зв'язків між обладнанням для надання контекстної інформації для елементів керування та аналітики.

Генеративний штучний інтелект (Generative Artificial Intelligence, GAI) – неконтрольована або частково контрольована структура машинного навчання, яка створює контент, використовуючи статистичні дані, отримані в ході навчання з наявного цифрового контенту (наприклад, текст, відео, зображення та аудіо). Велика мовна модель (Large Language Model, LLM) – статистична математична модель розподілу токенів у великому загальнодоступному обсязі створеного людиною тексту, внаслідок навчання вона може створювати людиноподібну мову. Генеративний попередньо навчений трансформатор (Generative Pre-trained Transformer, GPT) – система на базі LLM, призначена для генерування або статистичного прогнозування послідовності слів, коду або інших даних, починаючи з вихідного введення, яке називається підказкою. GPT базується на архітектурі трансформерів, яка паралельно обробляє великі обсяги загальнодоступних даних. В останніх дослідженнях показано, що потенціал LLM можна використовувати для створення онтології з неструктурованого тексту, розширювати наявні онтології для представлення нових концепцій і валідації онтологій.

Підхід до генерації з доповненням через пошук (Retrieval-Augmented Generation, RAG) став багатообіцяючим для підвищення продуктивності LLM у завданнях, пов'язаних із певною галуззю (доменом). RAG поєднує широкі знання попередньо підготовлених мовних моделей з можливістю отримання та інтеграції відповідної зовнішньої інформації. У побудові семантичних моделей RAG пропонує такі переваги: покращена точність, адаптація домену, включення сучасних знань і зменшення «галюцинацій» (відповіді, що містять фактично неправдиву інформацію). Однак його успіх залежить від актуальності отриманої інформації, збереження послідовності і контекстуального розуміння в онтологіях. Інтелектуальні чат-боти на базі великих мовних моделей набули надзвичайної популярності в усьому світі для широкого спектру промислових застосувань, особливо з кінця 2022 року. Лише через два місяці після випуску вдосконаленого інструменту ChatGPT охопив 100 мільйонів користувачів із 590 мільйонами відвідувань. З січня 2023 р. ця тенденція постійно зростала, і станом на кінець 2023 р. було понад 1,5 млрд відвідувань на місяць. Інтелектуальні чат-боти на базі LLM є частиною розробки генеративного штучного інтелекту (ШІ). Навіть враховуючи наслідки пандемії коронавірусної хвороби (COVID-19) і регіональних воєн, прогнозується, що генеративний штучний інтелект підтримуватиме щорічні темпи зростання на рівні 24,4% з 2023 по 2030 рік, що відповідає збільшенню обсягу ринку з 44,9 до 207 мільярдів доларів США. У цій шаленій конкуренції з'явилися подібні інтелектуальні чат-боти, зокрема, Google Bard, LLaMA, Alpaca, PaLM, Bing Chat, Copilot X, а також глобальні передові технологічні компанії та дослідницькі установи, включаючи Google AI, Microsoft OpenAI, Amazon, Meta AI, що призводить до дуже великої кількості варіантів вибору допомоги ШІ.

До появи ChatGPT з вбудованими LLM розмовні чат-боти вже широко застосовувалися в різних сферах освіти, виробництва, охорони здоров'я, державних послуг. Існує три основних типи розмовних чат-ботів: чат-боти на основі правил, живі чат-боти та базові інтелектуальні чат-боти на основі штучного інтелекту. Перші два типи чат-ботів спілкуються, дотримуючись заздалегідь визначених правил і, відповідно, включають програмне забезпечення чат-ботів і людські розмови для надання послуг клієнтам. Третій тип, базові чат-боти на основі ШІ, сприяють спілкуванню за межами попередньо визначених команд без взаємодії з людьми, закладаючи основу для просунутих інтелектуальних чат-ботів. Інтелектуальні чат-боти на базі LLM, включаючи ChatGPT, мають величезну кількість параметрів моделі. Тенденція до зростання кількості параметрів моделі в 12-и репрезентативних LLM, випущених з 2019 по 2021 рік, починаючи від DistilBERT з 60 мільйонами параметрів і закінчуючи Switch-C з 1500 мільярдами параметрів, демонструє значне збільшення (в 23 000 разів) протягом трьох років. Щоб підтримувати LLM та інтелектуальні сервіси чат-ботів, потрібно збирати та керувати величезними даними. Для прикладу наведемо такі джерела даних, які використовуються для навчання GPT-3: Common Crawl (410 мільярдів токенів), WebText2 (19 мільярдів токенів), Books1 (12 мільярдів токенів), Books2 (55 мільярдів токенів) і Wikipedia (3 мільярди токенів) [2].

Великі мовні моделі на базі архітектури GPT показали значні перспективи в інтерпретації завдань енергетики за допомогою вхідних даних на основі природної мови [3]. У цьому дослідженні перевірено можливості та обмеження LLM у застосуванні до сектору електроенергетики. Обговорювалась ефективність LLM у відповідях на загальні запити енергосистеми, у створенні коду та аналізі даних. Крім того, завдяки підходу RAG великі мовні моделі можуть слугувати базою знань документації та допомагати в таких завданнях, як навчання операторів. Мультимодальні можливості LLM можуть бути корисними для діагностики несправності обладнання та віддаленого моніторингу. LLM продемонстрували сильні можливості у виявленні кореляції між об'єктами (текст, зображення,

дані), проте вони все ще слабкі у вирішення проблем, пов'язаних із фізикою, які, зазвичай, включають складні математичні принципи.

Постановка проблеми у загальному вигляді
та її зв'язок із важливими науковими чи практичними завданнями

Щоб досягти цілей щодо стримування глобального потепління, необхідно зменшити споживання енергії будівлями, на які припадає близько 30 % глобальних викидів парникових газів. Моделювання енергоефективності будівлі (Building Energy Performance Simulation, BEPS) є усталеним методом оцінки її енерговитрат на ранніх стадіях проєктування. Інформаційні моделі будівель (Building Information Models, BIM) надають геометричну та семантичну інформацію для створення точних енергетичних моделей будівель (Building Energy Modeling, BEM) на ранніх стадіях проєктування. Проте ручне збагачення відсутньої семантичної інформації все ще є дуже трудомістким процесом. У дослідженні [4] запропоновано нову методологію для автоматичного збагачення відсутньої інформації в BIM за допомогою семантичної текстової подібності (Semantic Textual Similarity, STS) і покращеного налаштування LLM. Автори зіставляли типи приміщень і конструкції з відсутніми тепловими властивостями, використовуючи семантично найбільш схожі пари в моделі BIM і у відповідних базах даних. Вони використали три практичні приклади для покращеного налаштування LLM. Різні стратегії покращеного налаштування (використання різних функцій втрати, додавання протилежних пар слів або скорочень, що стосуються домену) значно підвищили точність зіставлення. Результати показали, що семантична відповідність на основі багатомовної тонко налаштованої LLM працює гірше, ніж в одномовної тонко налаштованої LLM.

У роботі [5] досліджено, як великі мовні моделі можуть допомогти у вирішенні проблем розробки семантичних моделей, основна увага зосереджена на їхній ролі в побудові запитів до семантичних моделей, зокрема, з використанням онтології Brick Schema. Зазначено, що хоча вже докладено багато зусиль, щоб охопити тонкощі побудови систем, проте інструменти, які дозволяють менеджерам будівель та розробникам додатків ефективно створювати ці моделі та робити запити, не були належним чином розроблені. Таким чином, відсутність необхідних, простих у використанні, інструментів обмежує користувачів з передовими знаннями програмування та інформаційних систем, оскільки вони також повинні мати глибоке розуміння побудови систем енергоспоживання, їхніх компонентів і варіантів моделювання.

Генерація запитів дозволяє користувачам отримувати структуровану інформацію від семантичної моделі, що є вирішальним для виявлення несправностей у діагностиці і аналітиці. SPARQL (Protocol and RDF Query Language) – мова запитів, яку використовують для запиту даних на основі графів. Переклад запитань природною мовою в запити SPARQL, який часто називають Text-to-SPARQL, має вирішальне значення для підвищення зручності використання графа знань (Knowledge Graph), оскільки користувачам часто бракує технічної експертизи для безпосереднього написання запитів SPARQL. Цей процес перекладу є основою системи відповідей на запитання в графі знань (KGQA, Knowledge Graph Question Answer), де методи семантичного аналізу перетворюють запитання природною мовою на формальні мови запитів SPARQL. Дослідження LLM показали перспективу створення запитів SPARQL з мінімальним навчанням, пропонуючи більшу адаптивність між доменами. Введено метод SGPT (SPARQL GPT), який обходить потребу в ручному створенні SPARQL, використовуючи LLM для вивчення шаблонів графів і створення запитів. Спираючись на це, підхід SPARQLGEN (SPARQL Generation) використовував GPT-3 у одноразовій структурі генерації SPARQL, де надання відповідного контексту значно покращило якість створених запитів. AUTO-KGQAGPT (Autonomous Knowledge Graph Question Answer GPT) розширив ці дослідження, провівши експерименти з GPT-3.5 і продемонструвавши, що вибіркова подача фрагментів графа знань T-Box і A-Box може покращити переклад запитань природною мовою в запити SPARQL [6].

Огляд наявних робіт показав, що літератури на тему застосування LLM до галузі енергоощадності будівель зараз досить мало, проте кількість досліджень в цій галузі швидко зростає. Ця робота відображає розроблений авторами підхід із застосуванням сучасних інструментів, зокрема, онтологій [7] та LLM, для створення семантичних запитів до цифрової моделі будівлі.

Аналіз досліджень та публікацій

У роботі [8] розглянуто можливості одержання знань великими мовними моделями в галузі опалення, вентиляції та кондиціонування (Heating, Ventilation, and Air Conditioning , HVAC), досліджено продуктивність моделі, її обмеження та майбутні напрямки використання у ній LLM. Показано ефективність LLM щодо володіння знаннями та навичками, пов'язаними з індустрією HVAC, дозволяючи користувачам скласти авторитетний іспит у галузі HVAC – ASHRAE Certified HVAC Designer. LLM показали можливості на рівні людини у вирішенні різних складних завдань, мислення та навчання як професіоналів у галузі HVAC. Досліджено три ключові властивості знань: пригадування, аналіз і застосування – на дванадцяти типових моделях: LLM: LLaMA, GPT-4, GPT-3.5, Alpaca, Koala, Vicuna, Dolly, Oasst-pythia, FastChat-T5, ChatGLM, StableLM-tuned-alpha, ERNIE Bot. Згідно з результатами GPT-4 пройшла іспит ASHRAE Certified HVAC Designer з балами від 74 до 78, що вище, ніж середній результат у людей. Крім того, модель GPT-3.5 здала іспит двічі з п'яти. Це продемонструвало, що деякі LLM, такі як GPT-4 і GPT-3.5, мають великий потенціал для допомоги або

заміни людей у проєктуванні та експлуатації систем HVAC, однак вони все одно іноді роблять деякі помилки через брак знань, погані здібності до міркування та незадовільні здібності щодо розв'язування рівнянь. Запропоновано майбутні напрямки досліджень, щоб виявити, як використовувати та вдосконалювати LLM у галузі HVAC: навчання LLM використовувати інструменти проєктування або програмне забезпечення в HVAC промисловості, дозволяючи LLM зчитувати та аналізувати робочі дані з систем HVAC, розроблення індивідуальних моделей для промисловості HVAC, а також оцінка продуктивності LLM у реальних сценаріях проєктування та експлуатації HVAC.

У роботі [9] досліджено потенціал генеративних попередньо навчених трансформерів в управлінні енергією будівлі на основі аналізу даних. Досліджено потенціал моделі GPT-4 у трьох сценаріях аналізу даних споживання енергії будівлі: прогнозування енергетичного навантаження, діагностика несправностей і виявлення аномалій. Запропоновано підхід для оцінки продуктивності можливостей GPT-4 щодо створення програмного коду прогнозування енергетичного навантаження, діагностики несправностей пристрою та виявлення шаблонів аномальної роботи системи. Продемонстровано, що GPT-4 може автоматично вирішувати більшість завдань аналізу даних в галузі HVAC.

У роботі [10] представлено дослідження прототипу агента віртуальної служби підтримки університету, побудованого на моделі LLM, для вирішення запитів студентів, викладачів і співробітників. Досліджено інтеграцію генеративного штучного інтелекту та природних властивостей мови, властивих LLM, щоб подолати недоліки обслуговування клієнтів. Як наслідок, агент підтримки університету забезпечив життєздатний інтерфейс запитань і відповідей для студентів, викладачів і адміністраторів, щоб дізнатися про керівні принципи та політику університету. Автори використали комбінований підхід поодинокого навчання та бібліотек ланцюжків думок на етапі навчання, щоб пом'якшити сприйнятливість LLM до «галюцинацій».

У роботі [11] зазначено, що наразі малим і середнім підприємствам не вистачає інтегрованої системи, яка об'єднує дані з різноманітних медіа-джерел у централізовану інформаційну систему, що обмежує їхню здатність ефективно орієнтуватися в ініціативах зі сталого розвитку енергії. Щоб усунути цю прогалину, представлено Energy Chatbot – систему, що використовує LLM, інтегровану з доповненням через пошук з кількох джерел. Ця система охоплює різноманітні медіа-джерела, зокрема новини, урядові звіти, галузеві публікації, наукові дослідження та соціальні мережі. Енергетичний чат-бот розроблено для покращення процесу ухвалення рішень для малого і середнього бізнесу, надаючи комплексну інформацію про енергетичний сектор через систему відповідей на запитання. Цей підхід зменшує витрати завдяки використанню моделей з відкритим кодом. Energy Chatbot надає доступ до актуальної інформації, що дозволяє визначати довгострокові стратегії сталого розвитку та підтримувати конкурентну перевагу в енергетичному сегменті, що розвивається. Автори вибрали моделі Llama (Llama2, Llama3 і Llama3.1), розроблені Meta, які є оновленими версіями, навченими на великих наборах даних і відомими своєю високою продуктивністю порівняно з іншими моделями з відкритим кодом. Щоб оцінити відповіді чат-бота, вибрано такі ключові метрики: відповідність (precision), запам'ятовування (recall), точність (accuracy) і оцінку F1 (поєднує відповідність і запам'ятовування). Зазначено, що існують й інші метрики оцінки: достовірність (faithfulness), релевантність контексту (context relevance), релевантність відповіді (answer relevance) та правильність відповіді (answer correctness), представлені різними дослідниками. Зазначено, що ці показники варто дослідити, щоб досягти більш точної та відповідної оцінки інформації, наданої системою, порівняно з використаним підходом. Визначено перспективу майбутніх досліджень щодо аналізу продуктивності інших мовних моделей, зокрема, GPT, щоб визначити, яка з них найкраще працює в сфері енергетики.

У роботі [12] досліджено вдосконалення ресурсів чат-бота для медичних послуг, що використовує LLM, ефективне тонке налаштування (Boundary Efficient Fine-Tuning) і ефективні стратегії обробки інформації. Для розробки структури чат-боту використано бібліотеку Lang Chain та методологію двовимірного пошуку даних для залучення чат-бота до великого сховища інформації про медичні послуги, включно зі складанням записів (record stacking) і частин поруч (piecing close) за допомогою векторних сховищ. Використано метрики BLEU і ROUGE, побудовані на оцінках точності, дослідженнях задоволення клієнтів та оцінках клінічних експертів. Дослідження показали обнадійливі результати щодо можливостей чат-бота як важливого інструменту для навчання пацієнтів і внутрішніх ресурсів організації. Зазначено, що розкриття потенціалу LLM для розробки чат-ботів вимагає ретельного розгляду обмежень ресурсів, особливо в таких середовищах, як Google Colab. У дослідженні розглянуто стратегічні методи завантаження та конфігурації моделі Tiny Llama 1.1B Chat v1.0 від Hugging Face. Проєкт TinyLlama зосереджений на створенні компактних і ефективних LLM шляхом попереднього навчання з 1.1 мільярдом параметрів. Механізми уваги дозволяють LLM вибірково зосереджуватися на відповідних частинах вхідної послідовності, фіксуючи довгострокові залежності та покращуючи розуміння контексту моделлю. Оптимізовано стек моделі та пам'ять, ретельно застосовуючи квантування, дезактивацію резервування та продуману конфігурацію маркерів (tokeniser configuration). Ці методи забезпечують розвиток ефективних чат-ботів, навіть в умовах з обмеженими обчислювальними ресурсами.

У роботі [13] запропоновано підхід, який використовує унікальні здібності LLM для генерації

SQL коду трансформацій даних на базі підказок (prompt) з доменними знаннями і шаблонами історичних даних. Розроблено бібліотеку SQLMorpher, що продемонструвала ефективність використання LLM в складних задачах, які стосуються конкретної області, підкреслюючи їхній потенціал для стимулювання стійких рішень в галузі енергоефективності.

У роботі [14] представлено розробку та реалізацію застосунку GreenIFTTT (Green If This Then That), яка базується на моделі GPT4, для створення та контролю процедур домашньої автоматизації. Агент допомагає користувачам зрозуміти, які процедури оптимізації енергоспоживання можна створити та застосувати, щоб зробити їхню побутову техніку більш екологічною, зменшити загальне споживання енергії та спростити виконання рутинних робіт за допомогою розумних технологій. Система зосереджена на створенні послідовності процедур автоматизації в домашньому середовищі на основі послідовного виконання певних дій, викликаних різними умовами. Основною парадигмою взаємодії є розмова, яка використовує можливості LLM, щоб допомогти користувачам знаходити та контролювати їхні розумні пристрої. Система також інтегрує дані з під'єднаних датчиків, надаючи користувачам інформацію про їхні повсякденні справи в реальному часі.

У роботі [15] досліджено потенціал використання технології доповненої генерації пошуку для відповіді на питання щодо вимірювання споживання електроенергії в будівлі, враховуючи аспекти енергетичного цифрового двійника (digital twin), заснованого на доменних знаннях [16]. Автори використали моделі ChatGPT, Gemini і Llama для відповіді на запитання, базуючись на графі знань щодо споживання електроенергії будівлею. Їхній граф знань створений у форматі RDF (Resource Description Framework), зберігається в базі даних Blazegraph і може приймати запити через мову SPARQL. Технологія RAG перетворює природну мову людини на відповідний запит SPARQL, а LLM отримує питання природної мови, запит SPARQL і триплети (triple), отримані з графу знань. Автори порівняли відповіді, отримані LLM і методами RAG, які використовують наявний цифровий двійник, заснований на знаннях про електроенергію. Їхні висновки показали, що підхід RAG не лише зменшує кількість неправильної інформації, згенерованої LLM, але й значно покращує якість результату, обґрунтовуючи відповіді даними, які можна перевірити.

Формулювання цілей статті

Метою роботи є моделювання системи енергоспоживання будівлі цифровим двійником на основі онтологічного підходу, графової бази даних та великих мовних моделей, що дозволить підвищити енергоощадність будівлі та забезпечить через інтерфейс легкодоступне пряме спілкування користувача з базою знань.

Об'єкт дослідження – споживання енергії будівлями.

Предмет дослідження – онтологічний підхід і агентний метод генерації з доповненням через пошук за допомогою великих мовних моделей.

Виклад основного матеріалу

Дослідження проведено на локальному комп'ютері з такими характеристиками: 8-ядерний процесор AMD Ryzen 7 6800H 3,20 ГГц і 12-ядерний графічний процесор AMD Radeon 680M 2200 МГц. Використано графову базу даних Neo4j, середовище розробки PyCharm, мову програмування Python з бібліотеками LangChain, FastApi, Streamlit.

Онтологія, розроблена нами раніше в роботі [7], описує низку енергетичних і будівельних термінів та відношень. Онтологія – форма представлення знань у моделі предметної області, щоб зробити дані розумнішими, перенести частину логіки програми та створювати нові дані. Прикладом представлення знань може бути публікація структурованих даних, самоописаних даних або даних, описаних у термінах чітко визначеної семантики. Онтології охоплюють представлення знань (representation) і міркування (reasoning), природну мову, машинне або автоматизоване навчання, мову, зір, робототехніку та вирішення проблем.

Розглянемо зв'язок між онтологіями і графовими базами даних. Станом на 2024 рік згідно з [17] перше місце за популярністю серед інших графових сховищ даних займає база Neo4j, тенденцію можна побачити на рис. 1. У Neo4j існує два основних способи використання онтологій: інтероперабельність (визначення спільного словника) та логічний висновок (inferencing), який є результатом знання фрагментів словника.

Neo4j – нативна графова база даних, яка реалізує справжню графову модель аж до рівня зберігання. Вона використовує Cypher, декларативну мову запитів, схожу на SQL, але оптимізовану для графів. З 2007 року Neo4j перетворилася на багату екосистему інструментів, програм і бібліотек. Екосистема Neo4j дозволяє інтегрувати технології графів у робоче середовище різноманітними способами. Neo4j наразі підтримує драйвери для клієнтів на мовах Python, .NET, Java, JavaScript, Go, GraphQL, Spring, також низку бібліотек, зокрема, Neo4j Graph Data Science (GDS), яка забезпечує реалізацію загальних алгоритмів графів і пайплайнів машинного навчання для навчання прогнозних керованих моделей. Вона має хмарний сервіс Neo4j AuraDB, її можна запускати в контейнері Docker та кластері Kubernetes. У цьому дослідженні ми працювали з локальним розгортанням і використовували Neo4j Desktop версії 1.6.1.

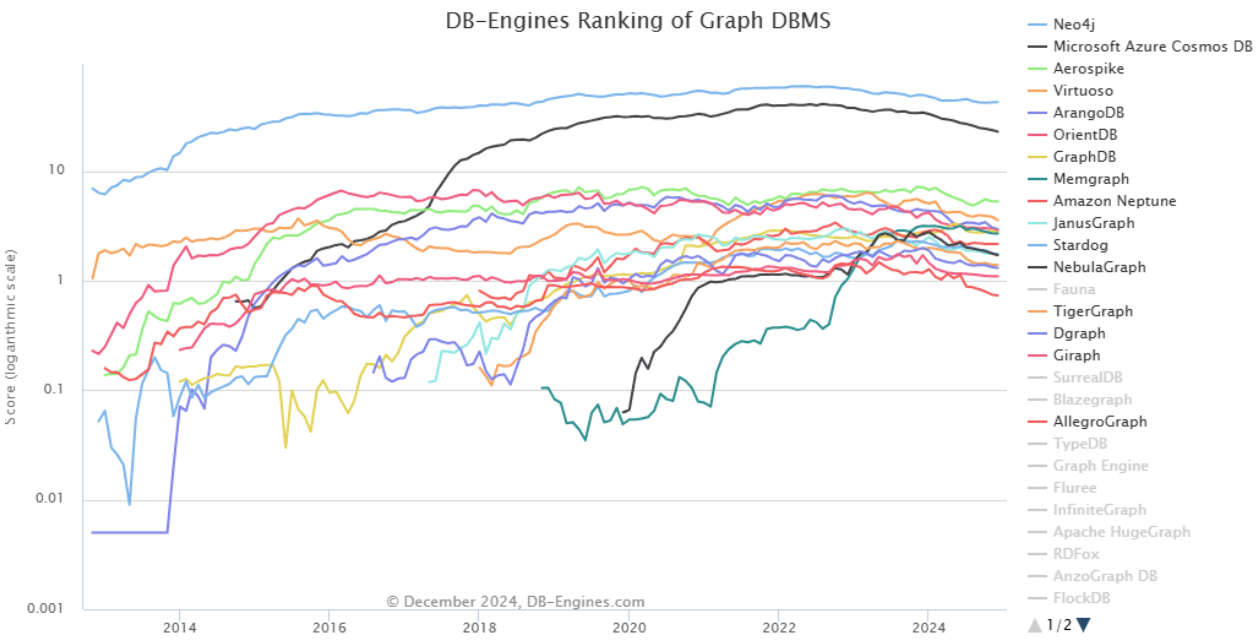

Рис. 1. Рейтинг популярності графових баз даних

Neo4j може завантажувати та записувати онтологію у форматі RDF. Тому згідно рис. 2 розроблену нами онтологію можна завантажити в базу знань Neo4j за допомогою розширення Neosemantics (n10s). Дотепер міркування в RDF і OWL (Web Ontology Language) відносили лише до повноцінних сховищ триплетів або спеціальних механізмів міркування. Neo4j можна розширити за допомогою унікальної технології міркування, щоб створити дуже виразний і конкурентоспроможний механізм міркування для RDF. Графова база даних Neo4j не є сховищем триплетів, тому вона не підтримує механізму запитів SPARQL. Однак Neo4j пропонує мову запитів Cypher і для багатьох семантичних запитів можна перекладати запити SPARQL у Cypher. З точки зору користувача Neo4j об'єднує дві технології в одну платформу – транзакційну аналітичну систему графів та механізм міркування RDF/OWL, здатний забезпечувати складні семантичні Cypher запити завдяки матеріалізованому графу у Neo4j. Дослідження показали, що отримання всіх виведених фактів (так звана матеріалізація) за допомогою Neo4j показує набагато нижчий коефіцієнт зростання часу міркування зі збільшенням кількості даних, ніж будь-яка інша система, таким чином матеріалізація є ключем до ефективних запитів в реальному світі [18]. Без попередньої матеріалізації сховище триплетів з обґрунтуванням має тимчасово отримувати всі відповіді та релевантні факти для кожного окремого запиту на вимогу.

Рис. 2. Відображення OWL онтології в графовій базі знань Neo4j

Згідно розробленої нами раніше онтології [7] створено такі вузли і зв'язки в графовій базі даних: Customer (клієнт) поєднаний з Building (будівля) зв'язком HAS, Building – з Apartment (помешканням) зв'язком LOCATED_AT, Meter (лічильник) – з Apartment зв'язком MEASURES. Зразок зав'язків між однією будівлею, помешканнями, які в ній знаходяться, та лічильниками, які вимірюють споживання їхньої енергії, зображено рис. 3, тут наведено Cypher-запит для пошуку всіх лічильників, які роблять виміри для всіх помешкань, що знаходяться в будівлі.

Вузол Meter поєднаний з Consumption (споживання) зв'язком HAS_CONSUMPTION та з MeterReading (показник лічильника) зв'язком HAS_READING. Оскільки збір показників лічильників є постійним процесом з певним інтервалом (наприклад, нові значення зчитуються з інтервалом 1 год), то важливим є семантичне представлення послідовного зв'язку показників: вузол MeterReading за момент часу *t* з'єднаний зв'язком NEXT з вузлом MeterReading за момент часу *t*+1. Тому проблема часових рядів перетворюється на графову проблему і може бути ефективно вирішена за допомогою наявних алгоритмів для роботи з графами. Оскільки споживання за одиницю часу – це різниця між показником лічильника за момент часу *t* і за момент часу *t*+1, то це відображено семантичним зв'язком показників: вузол Consumption за момент часу *t* з'єднаний зв'язком START_READING з вузлом MeterReading за момент часу *t* і зв'язком STOP_READING з вузлом MeterReading за момент часу *t*+1. Споживання також є послідовним процесом, тому вузол Consumption за момент часу *t* з'єднаний зв'язком NEXT з вузлом Consumption за момент часу *t*+1. Зразок послідовних зв'язків між показниками лічильника та споживанням за кілька послідовних годин зображено на рис. 4.

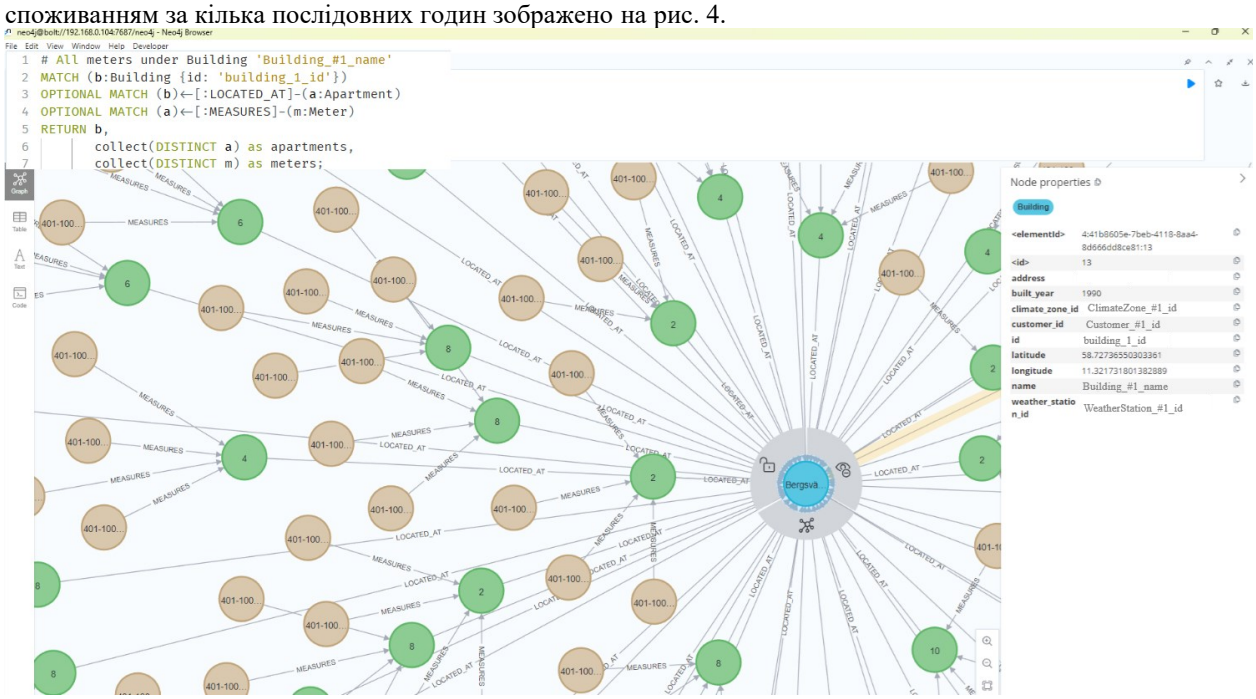

Рис. 3. Представлення будівель, лічильників і зв'язків графом

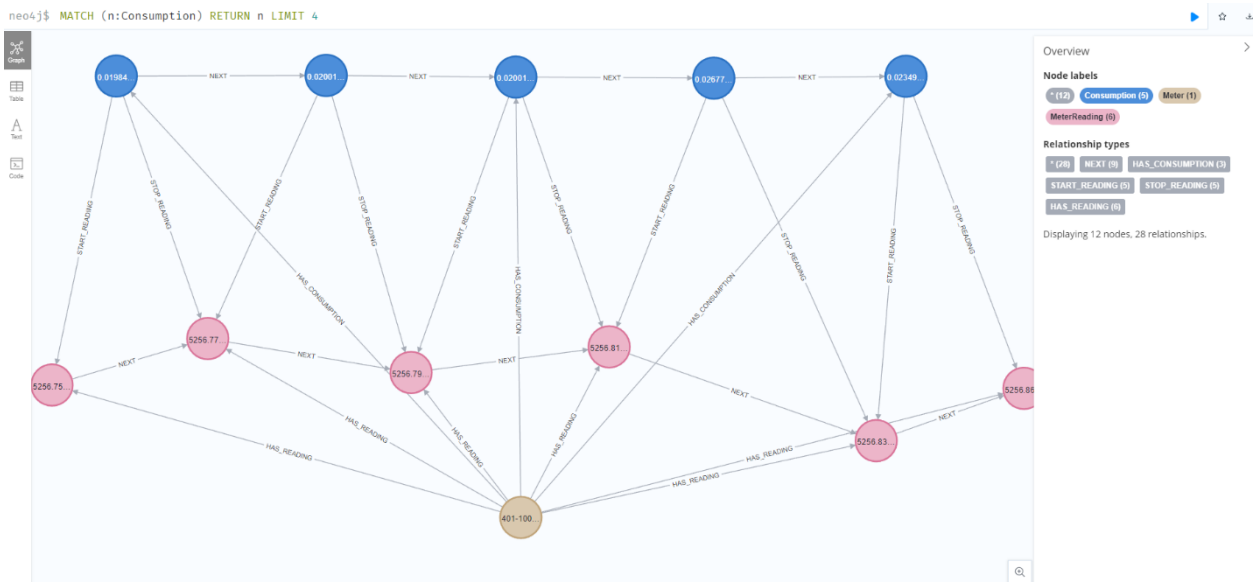

Рис. 4. Представлення даних про показники лічильників і споживання як послідовності вузлів у графі

Вузол WeatherStation (станція прогнозу погоди) поєднаний з вузлом Temperature (температура навколишнього середовища) зв'язком HAS_TEMPERATURE. Покази станції прогнозу погоди є постійним процесом з певним інтервалом (наприклад, нові значення надходять з інтервалом 1 день), тому вузол Temperature за момент часу *t* з'єднаний зв'язком NEXT з наступним вузлом Temperature за момент часу *t*+1. Зразок послідовних зав'язків між показниками температури навколишнього середовища за кілька послідовних днів зображено на рис. 5. Вузли Consumption, MeterReading і Temperature мають індекс для поля timepoint для пришвидшення пошуку.

Зберігання даних у структурі графу має такі переваги:

1) простота – моделювання зв'язків у реальному світі між сутностями природнім чином, уникнення потреби у складних схемах з кількома операціями з'єднання для відповіді на запити;
2) ефективне опрацювання складних зв'язків за рахунок обходу графу, спрощення запитів та аналізу пов'язаних даних;
3) гнучкість – не містять схем, що дозволяє легко пристосовуватися до змін у структурі даних;
4) продуктивність – отримання пов'язаних даних відбувається швидше в графових базах даних, ніж у реляційних, особливо для сценаріїв, що включають складні запити з кількома зв'язками;
5) знаходження шаблонів – підтримка запитів на знаходження шаблонів полегшує пошук конкретних структур у даних.

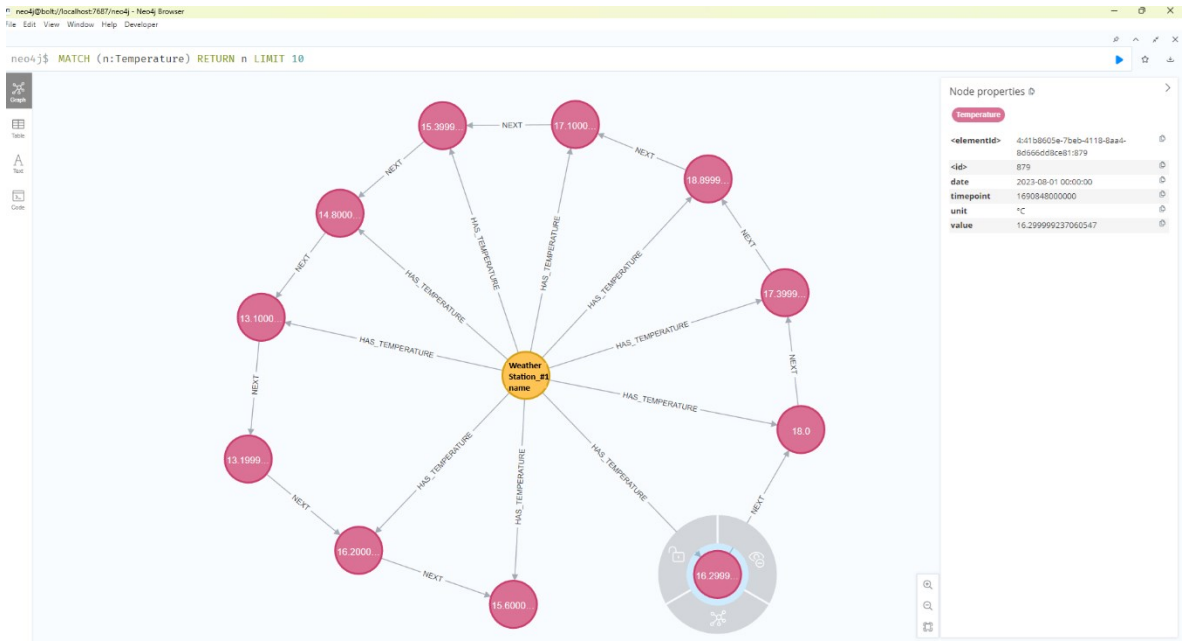

Рис. 5. Представлення кліматичних даних про навколишню температуру як послідовності вузлів у графі

Коли модель містить дані з багатьма складними зв'язками, простота та гнучкість графових баз даних полегшує їхнє проєктування та запити порівняно з реляційними. Визначення зв'язків у запитах до графової бази даних є лаконічним і не вимагає складних об'єднань таблиць (joins). Завдяки цьому гнучкому представленню даних LLM здатні генерувати запити до бази даних графів з меншою кількість помилок. Достатньо повідомити LLM лише про вузли, зв'язки та властивості графової бази даних. Порівняно з реляційними базами даних LLM повинні розбиратися з знаннями про схеми таблиць і зв'язки між ключами, що може збільшити кількість помилок у генерації запитів SQL.

У цьому дослідженні використано онтологічний підхід для побудови моделі знань про область (домен) енергоспоживання будівлею. Застосовано агентний метод генерації з доповненням через пошук і розроблено набір інструментів, які взаємодіятимуть з базою знань і кінцевим користувачем, використовуючи великі мовні моделі. Схема, зображена на рис. 6, показує принцип роботи розробленого чат-бот-застосунку та відображає, як запит від користувача на мові, зрозумілій людині, перетворюється на Cypher-запит, зрозумілий для графової бази даних Neo4j; результат виконання передається до агента, який вже повертає користувачеві фінальну відповідь. Використано модель gpt-4o-mini від OpenAI, що дозволяє ефективно вирішувати завдання завдяки низьким вартості та затримці.

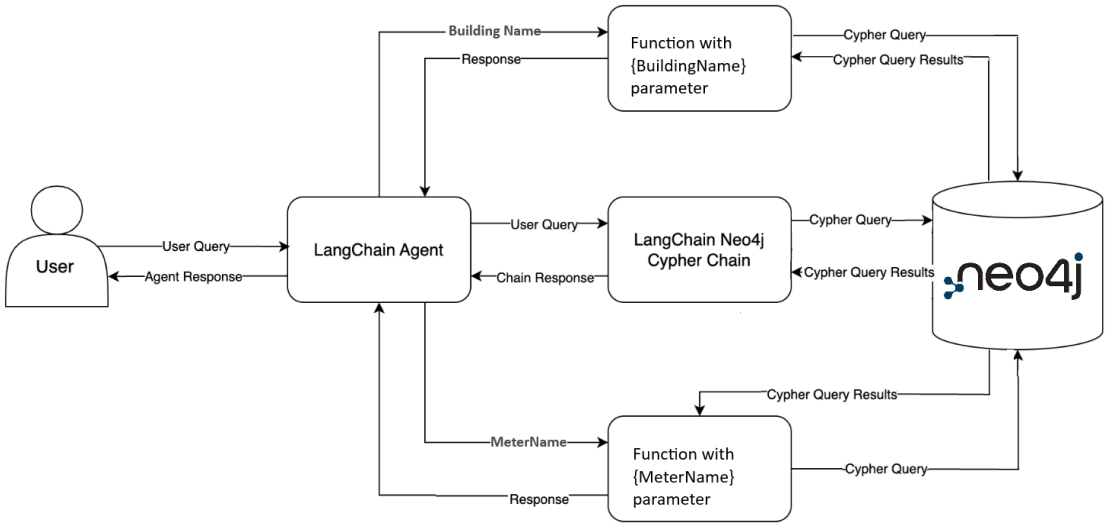

Рис. 6. Схема взаємодії користувача з базою знань з використанням великих мовних моделей

Нижче подано короткий опис кожного компонента:

1) *агент LangChain* – мозок чат-бота. За запитом користувача агент вирішує, який інструмент (tool) викликати та що надати інструменту як вхідні дані. Потім агент спостерігає за відповіддю інструмента та вирішує, що повернути користувачеві (agent response).
   Для створення агента використано функцію create_openai_functions_agent з такими параметрами:
   - llm=ChatOpenAI(model=gpt-4o-mini, temperature=0) – LLM, яка використовується для створення запитів Cypher;
   - prompt=hub.pull("hwchase17/openai-functions-agent") – агент, розроблений OpenAI для передачі вхідних параметрів функціям. Це здійснюється шляхом повернення у форматі JSON об'єктів, які зберігають вхідні дані функції та їх відповідне значення;
   - tools=tools – список інструментів, які використовує агент для вибору, дуже важливо надати детальний для кожного інструменту.

   Щоб запустити виконання агента, потрібно передати агент та інструменти в клас AgentExecutor з параметрами return_intermediate_steps=True і verbose=True. Це дозволяє відображати процес мислення агента та інструменти, які він викликає;
2) графова база даних Neo4j зберігає структуровані дані про будівлі, квартири, мешканців, лічильники та споживання;
3) *інструмент LangChain Neo4j Cypher Chain* перетворює запит на мові користувача на запит у форматі Cypher, зрозумілий для графової бази Neo4j. Інструмент відповідає на запит користувача, використовуючи результати запиту Cypher. Відповідь ланцюжка повертається до агента LangChain і надсилається користувачеві. Для цього використано клас GraphCypherQAChain з бібліотеки LangChain, що формує ланцюжок (chain) для запитань-відповідей на графі шляхом генерації запиту Cypher.

   Для створення ланцюжка використано функцію GraphCypherQAChain.from_llm з такими параметрами:
   - cypher_llm=ChatOpenAI(model=gpt-4o-mini, temperature=0) – LLM, яка використовується для створення запитів Cypher;
   - qa_llm=ChatOpenAI(model=gpt-4o-mini, temperature=0) – LLM, яка використовується для генерування відповіді за результатами запиту Cypher;
   - verbose=True – для виведення проміжних етапів, які виконує ланцюжок;
   - qa_prompt=qa_creation_prompt – шаблон підказки для відповідей на запитання;
   - cypher_prompt=cypher_creation_prompt – шаблон підказки для створення запитів Cypher;
   - validate_cypher=True – запит Cypher буде перевірено на наявність помилок і виправлено перед виконанням;
   - top_k=10 – кількість результатів запиту для включення в qa_prompt.

   Використання LLM для створення точних запитів Cypher може бути нелегким завданням, особливо для складного графа. В таких випадках потрібно розробити підказки (prompt engineering), щоб показати LLM структуру графа та варіанти використання запитів. Покращене налаштування (fine-tuning) LLM для генерування запитів також є варіантом, але для цього потрібно вручну підібрати та позначити дані;
4) *інструмент функція з параметром назви будівлі* використовується, коли агенту LangChain вдається отримати назву будівлі із запиту користувача. Назва будівлі передається як вхідний параметр для функції Python, що містить певну логіку, специфічну для конкретної будівлі, наприклад, робить запит до зовнішнього ресурсу, який надає певні метадані про будівлю, і цей результат повертається

агенту;

5) *інструмент функція з параметром назви лічильника* використовується, коли агенту LangChain вдається отримати назву лічильника із запиту користувача. Назва лічильника передається як вхідний параметр для функції Python, що містить логіку, яка надає прогноз споживання конкретного лічильника на наступні 24 години, і цей результат повертається агенту.

Для спілкування з агентом додано API-ендпойнт з POST методом, який приймає один параметр з запитанням користувача, запускає агент на виконання і повертає результат від LLM з використанням одного із визначених інструментів. Ми використали бібліотеку FastAPI – високопродуктивний вебфреймворк для створення API за допомогою мови Python. На рис. 7 зображено сторінку з документацією методів розробленого API.

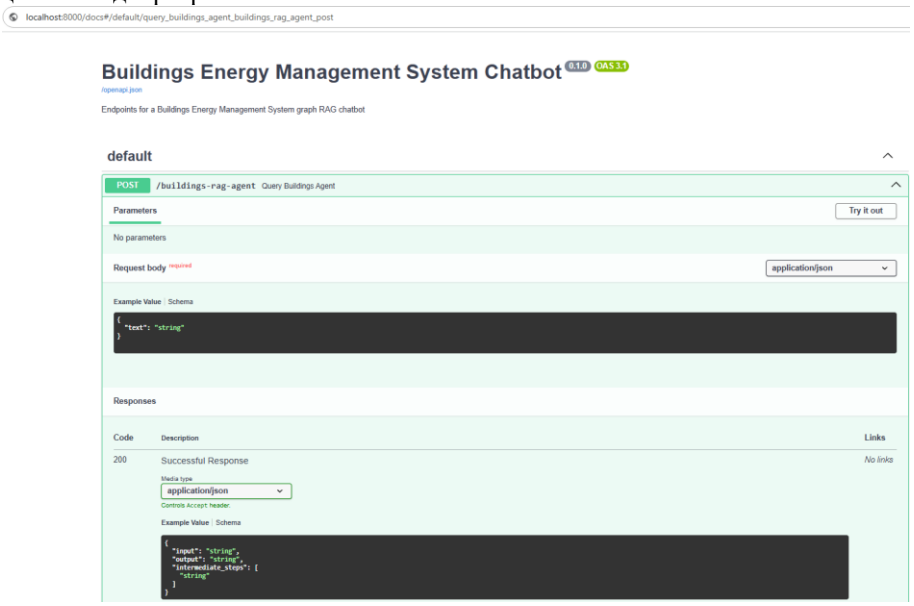

Рис. 7. Документація методів розробленого API

Для створення графічного вебінтерфейсу для взаємодії з кінцевим користувачем використано бібліотеку Streamlit. Розгортання розробленого застосунку реалізовано з використанням менеджера контейнерів Docker Compose з двома сервісами: для API і графічного інтерфейсу. Інтерфейс користувача приймає запитання користувача та надсилає запит POST до API-ендпойнта агента. Уся історія чату зберігається та відображається щоразу, коли користувач робить новий запит. Остання відповідь агента відображається внизу чату та додається до його історії. Користувачеві надається пояснення того, як агент згенерував відповідь, це можна використати під час тестування, оскільки можна побачити, чи агент викликав правильний інструмент і чи дав правильну відповідь. Результат пошуку даних про будівлі та лічильники в розробленому застосунку показано на рис. 8, тут користувач поставив кілька запитань: “Які лічильники недавно були приєднані і коли саме?”, “Які лічильники приєднані до будівлі {Назва будівлі}?”, “Яка будівля є найстаршою і скільки їй років?”

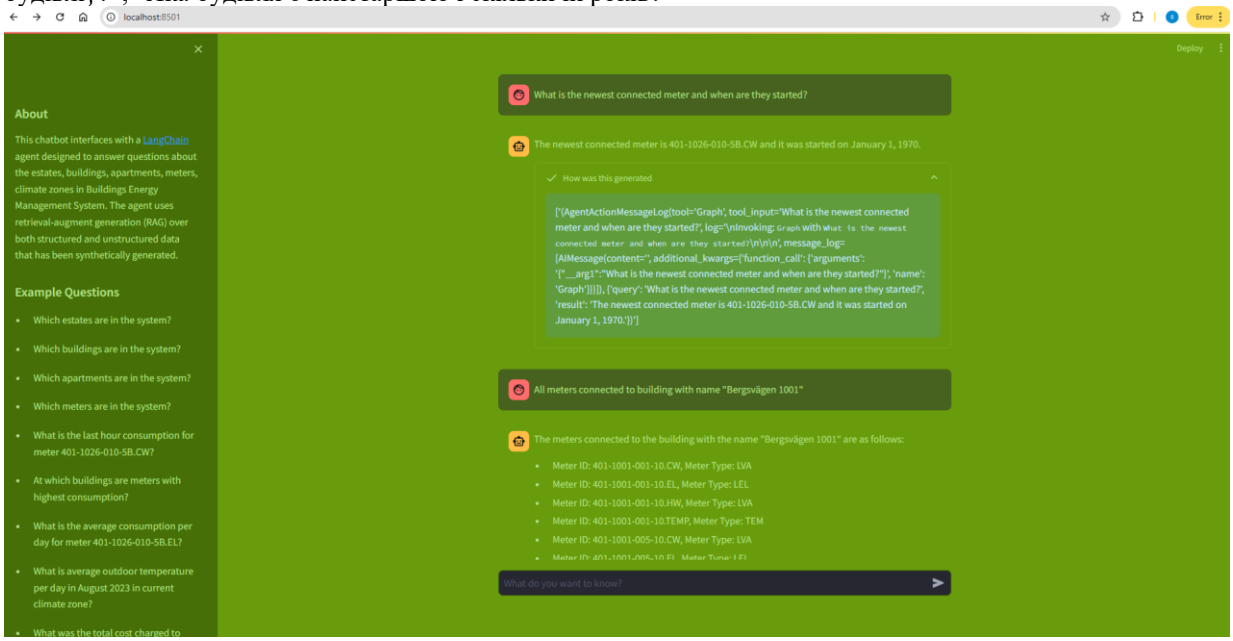

Рис. 8. Результат пошуку даних про будівлі та лічильники в розробленому застосунку

Деталі про запит “Яка будівля є найстаршою і скільки їй років?” бачимо в консолі API сервісу. Зазначимо, що агент вибрав правильний інструмент GraphCypherQAChain і згенерував Cypher-запит, показаний на рис. 9, який після виконання вернув значення “34”. Потім це значення передалося до агента, який сформував відповідь “Найстарішою будівлею є будівля {Назва будівлі} і їй 34 роки”, при цьому завдяки сформованим підказкам (prompt) агент знав, що одиниця вимірювання саме рік, а не день чи година.

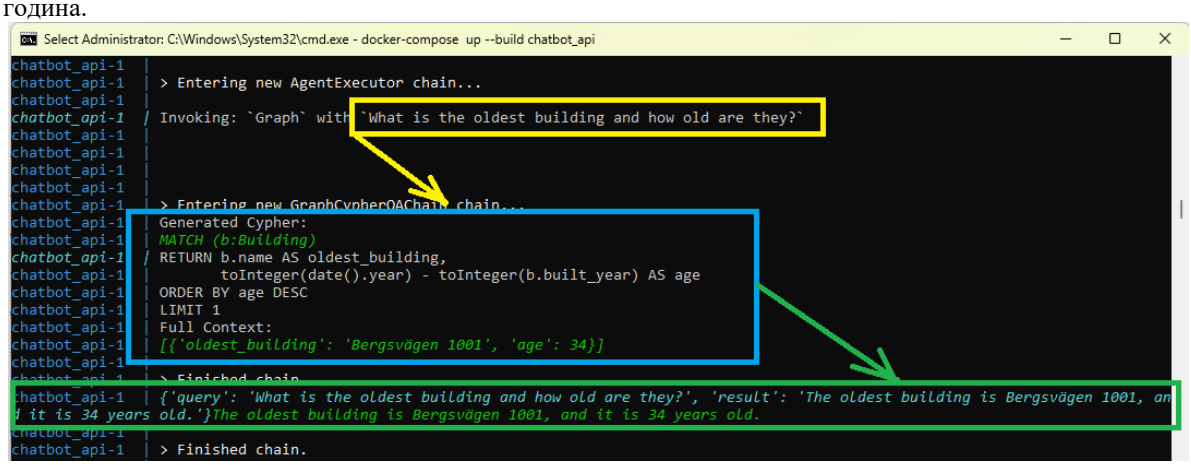

Рис. 9. Результат роботи LLM агента, генерація Cypher-запиту і відповіді

Розглянемо запит “Чи є аномалії для лічильника {НазваЛічильника}?”. В консолі API сервісу бачимо, що агент за допомогою інструменту GraphCypherQAChain згенерував Cypher-запит, показаний на рис. 10. Цей запит шукає, чи є споживання, в яких відношення різниці послідовних показників лічильника до різниці часу більше за 10. Цей запит до бази знань Neo4j повернув пустий масив “[]”, що передалося до агента, який сформував кінцеву відповідь “Аномалії для лічильника {НазваЛічильника} не знайдено”. Зазначимо, що не було попередньо сформованих підказок для пошуку аномалій, ланцюжок GraphCypherQAChain автоматично згенерував цей Cypher-запит, базуючись на структурі побудованої бази знань. Завдяки модифікації підказок можна додати інструкцію, щоб аномальне значення визначалося іншим чином, враховуючи особливості типу лічильника, так будуть застосовані різні параметри для електрики, води, газу.

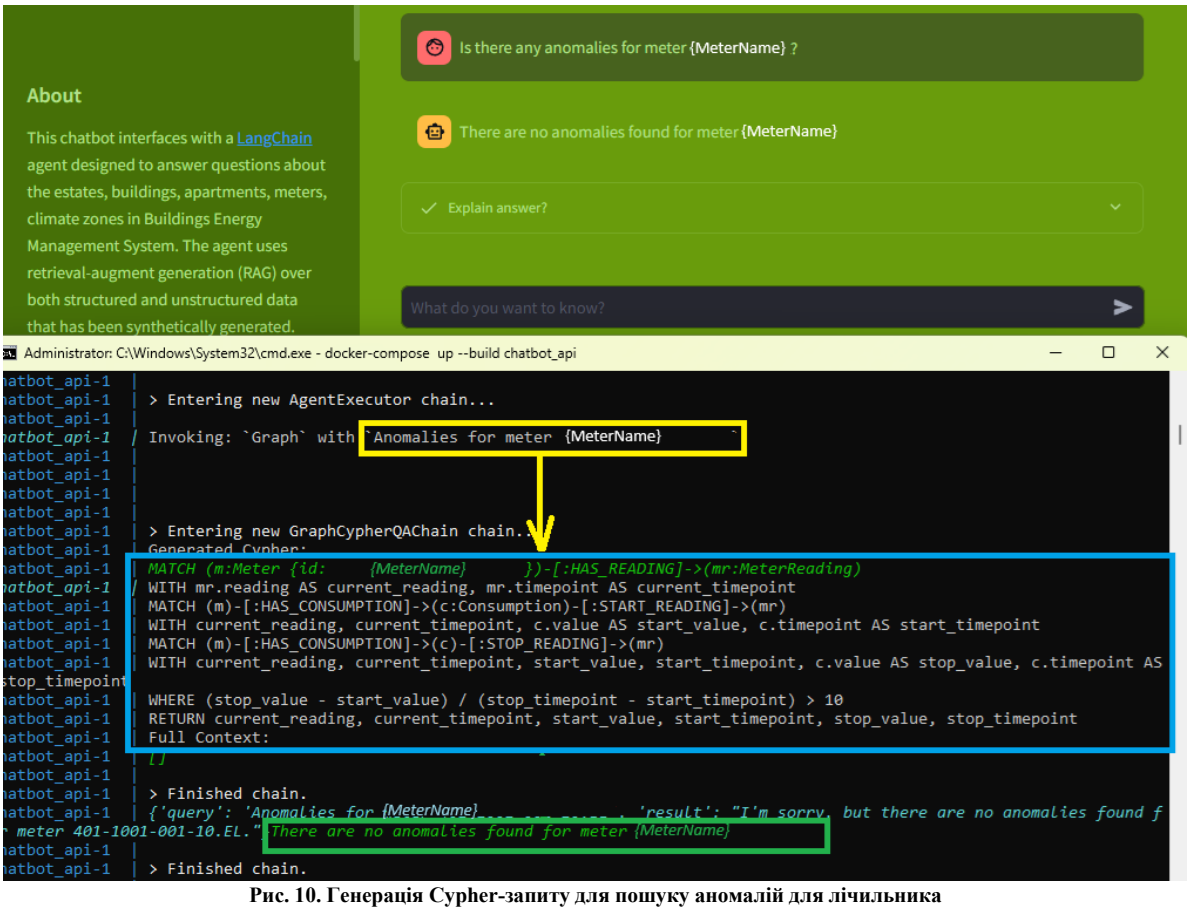

Рис. 10. Генерація Cypher-запиту для пошуку аномалій для лічильника

Розглянемо запит “Який є профіль споживання лічильника {НазваЛічильника} протягом літнього періоду?”. В консолі API сервісу бачимо, що агент за допомогою інструменту GraphCypherQAChain згенерував Cypher-запит, показаний на рис. 11.

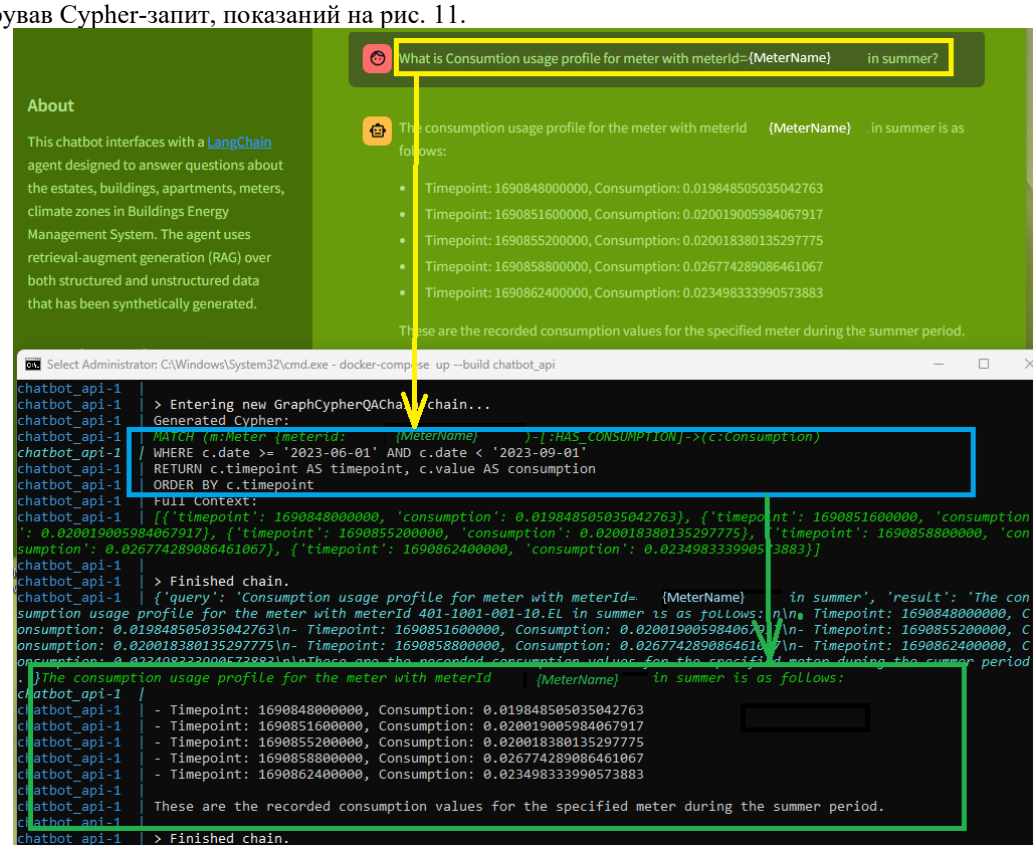

Рис.11. Генерація Cypher-запиту для профілю споживання лічильника протягом літнього періоду

Цей запит шукає споживання лічильника, в яких дата знаходиться в проміжку від 1 червня до 1 вересня. Цей запит до бази знань Neo4j повернув масив значень, який передався до агента, що сформував кінцеву відповідь у вигляді списку значень споживання за певні дати. Зазначимо, що не було попередніх підказок для формування профілю споживання, ланцюжок GraphCypherQAChain автоматично згенерував цей Cypher-запит, базуючись на структурі побудованої бази знань, і, автоматично відфільтрувавши дані, сформував умову для початку і кінця літнього періоду. Завдяки модифікації підказок, можна додати інструкцію, щоб дата відображалася в форматі “2023-08-01 04:00:00” (а не в epoch-форматі) і щоб споживання електрики відображалося в кіловатах.

Висновки з даного дослідження і перспективи подальших розвідок у даному напрямі

Дослідження, проведене в рамках цієї статті, підкреслює значний потенціал використання великих мовних моделей для взаємодії людини з системою енергоспоживання будівлі за рахунок побудови цифрового двійника системи на основі онтологічного підходу.

Експериментальним шляхом досліджено, як великі мовні моделі можуть допомогти перетворити запит про енергоспоживання на мові людини у відповідний Cypher-запит до бази знань. Навчаючись на величезному масиві даних, великі мовні моделі мають потенціал для узагальнення в різних областях, не вимагаючи обширного предметного навчання. Застосування гібридного підходу генерації з доповненням через пошук дало змогу збільшити точність генерування запитів за рахунок попереднього пошуку релевантної інформації, використовуючи семантичні зв’язки в схемі побудованої бази знань Neo4j.

Підхід з використанням запропонованої онтологічної моделі для побудови бази знань про будівлі показав, що він може бути ефективним інструментом для взаємодії з великими мовними моделями за рахунок формалізованої структури, придатної для машинного читання, і опису системи для представлення знань про особливості області застосування. Інтерактивний інтерфейс робить для людей такі підходи більш адаптивним і доступним інструментом, оскільки вони не обмежені попередньо визначеними правилами чи моделями, що дозволяє ширше застосовувати їх у сфері будівель та енергетики. Використовуючи великі мовні моделі, можна зменшити залежність від спеціальних знань, а створення моделей на основі онтології сприяє їхньому широкому застосуванню кінцевими користувачами.

Подальші дослідження можуть бути зосереджені на застосуванні підходу тонкого налаштування (fine tuning) для компактних і ефективних LLM моделей типу TinyLlama, що дозволить заощадити кошти та забезпечить можливість використання цих моделей для вбудованих пристроїв у режимі без мережі.

Література

1. Vyshnevskyy O., Zhuravchak L. Semantic Models for Buildings Energy Management. *2023 IEEE 18th International Conference on Computer Science and Information Technologies (CSIT)*, Lviv, Ukraine, 19–21 October 2023. DOI: 10.1109/csit61576.2023.10324108.

2. Jiang P. та ін. Preventing the Immense Increase in the Life-Cycle Energy and Carbon Footprints of LLM-Powered Intelligent Chatbots. *Engineering*. 2024. DOI: 10.1016/j.eng.2024.04.002.

3. Majumder S. та ін. Exploring the capabilities and limitations of large language models in the electric energy sector. *Joule*. 2024. Т. 8, № 6. С. 1544–1549. DOI: 10.1016/j.joule.2024.05.009.

4. Forth K., Borrmann A. Semantic enrichment for BIM-based building energy performance simulations using semantic textual similarity and fine-tuning multilingual LLM. *Journal of Building Engineering*. 2024. С. 110312. DOI: 10.1016/j.jobe.2024.110312.

5. Mulayim O. B. та ін. Large Language Models for the Creation and Use of Semantic Ontologies in Buildings: Requirements and Challenges. *BuildSys '24: The 11th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, м. Hangzhou China. New York, NY, USA, 2024. С. 312–317. DOI: 10.1145/3671127.3698792.

6. Avila C. V. S. та ін. Experiments with text-to-SPARQL based on ChatGPT. *2024 IEEE 18th International Conference on Semantic Computing (ICSC)*, м. Laguna Hills, CA, USA, 5–7 лют. 2024 р. DOI: 10.1109/icsc59802.2024.00050.

7. Вишневський О., Журавчак Л. Методи машинного навчання для підвищення енергоефективності будівель. *Вісник Національного університету «Львівська політехніка» «Інформаційні системи та мережі».* 2023. Т. 14. С. 189–209. DOI: 10.23939/sisn2023.14.189.

8. Lu J. et al. Evaluation of large language models (LLMs) on the mastery of knowledge and skills in the heating, ventilation and air conditioning (HVAC) industry. *Energy and Built Environment*. 2024. DOI: 10.1016/j.enbenv.2024.03.010.

9. Zhang C., Lu J., Zhao Y. Generative pre-trained transformers (GPT)-based automated data mining for building energy management: Advantages, limitations and the future. *Energy and Built Environment*. 2023. DOI: 10.1016/j.enbenv.2023.06.005.

10. Ilagan J. B., Ilagan J. R. A prototype of a conversational virtual university support agent powered by a large language model that addresses inquiries about policies in the student handbook. *Procedia Computer Science*. 2024. Vol. 239. P. 1124–1131. DOI: 10.1016/j.procs.2024.06.278.

11. Arslan M., Mahdjoubi L., Munawar S. Driving sustainable energy transitions with a multi-source RAG-LLM system. *Energy and Buildings*. 2024. Vol. 324. P. 114827. DOI: 10.1016/j.enbuild.2024.114827.

12. Vidivelli S., Ramachandran M., Dharunbalaji A. Efficiency-Driven Custom Chatbot Development: Unleashing LangChain, RAG, and Performance-Optimized LLM Fusion. *Computers, Materials & Continua*. 2024. P. 1–10. DOI: 10.32604/cmc.2024.054360.

13. Sharma A. et al. Automatic Data Transformation Using Large Language Model - An Experimental Study on Building Energy Data. *2023 IEEE International Conference on Big Data (BigData)*, Sorrento, Italy, 15–18 December 2023. 2023. DOI: 10.1109/bigdata59044.2023.10386931.

14. Giudici M. et al. Designing Home Automation Routines Using an LLM-Based Chatbot. *Designs*. 2024. Vol. 8, no. 3. P. 43. DOI: 10.3390/designs8030043.

15. Fortuna C., Hanžel V., Bertalanič B. Natural Language Interaction with a Household Electricity Knowledge-based Digital Twin. *2024 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, Oslo, Norway, 17–20 September 2024. P. 8–14. DOI: 10.1109/smartgridcomm60555.2024.10738062.

16. Fortuna C., Hanžel V., Bertalanič B. Towards Data-Driven Electricity Management: Multi-Region Harmonized Data and Knowledge Graph. URL: http://arxiv.org/abs/2405.18869.

17. Historical trend of graph DBMS popularity. DB-Engines - Knowledge Base of Relational and NoSQL Database Management Systems. URL: https://db-engines.com/en/ranking_trend/graph+dbms

18. Liebig, T. GraphScale: Adding Expressive Reasoning to Semantic Data Stores / Liebig, T., Vialard, V., Opitz, M., & Metzl, S. *Proceedings of the 14th International Semantic Web Conference (ISWC 2015)*. Vol 1486. 2015. URL: https://ceur-ws.org/Vol-1486/paper_117.pdf.

References

1. Vyshnevskyy O., Zhuravchak L. Semantic Models for Buildings Energy Management. *2023 IEEE 18th International Conference on Computer Science and Information Technologies (CSIT)*, Lviv, Ukraine, 19–21 October 2023. DOI: 10.1109/csit61576.2023.10324108.

2. Jiang P. et al. Preventing the Immense Increase in the Life-Cycle Energy and Carbon Footprints of LLM-Powered Intelligent Chatbots. *Engineering*. 2024. DOI: 10.1016/j.eng.2024.04.002.

3. Majumder S. et al. Exploring the capabilities and limitations of large language models in the electric energy sector. *Joule*. 2024. Vol. 8, no. 6. P. 1544–1549. DOI: 10.1016/j.joule.2024.05.009.

4. Forth K., Borrmann A. Semantic enrichment for BIM-based building energy performance simulations using semantic textual similarity and fine-tuning multilingual LLM. *Journal of Building Engineering*. 2024. P. 110312. DOI: 10.1016/j.jobe.2024.110312.

5. Mulayim O. B. et al. Large Language Models for the Creation and Use of Semantic Ontologies in Buildings: Requirements and Challengesю *BuildSys '24: The 11th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and*

Transportation, Hangzhou China. New York, NY, USA, 2024. P. 312–317. DOI: 10.1145/3671127.3698792.

6. Avila C. V. S. et al. Experiments with text-to-SPARQL based on ChatGPT / *2024 IEEE 18th International Conference on Semantic Computing (ICSC)*, Laguna Hills, CA, USA, 5–7 February 2024. 2024. DOI: 10.1109/icsc59802.2024.00050.

7. Vyshnevskyy O., Zhuravchak L. Machine Learning Methods to Increase the Energy Efficiency of Buildings. *Visnik Nacìonal'nogo unìversitetu "L'vìvs'ka polìtehnìka". Seria Informacijni sistemi ta merezi*. 2023. Vol. 14. P. 189–209. DOI: 10.23939/sisn2023.14.189.

8. Lu J. et al. Evaluation of large language models (LLMs) on the mastery of knowledge and skills in the heating, ventilation and air conditioning (HVAC) industry. *Energy and Built Environment*. 2024. DOI: 10.1016/j.enbenv.2024.03.010.

9. Zhang C., Lu J., Zhao Y. Generative pre-trained transformers (GPT)-based automated data mining for building energy management: Advantages, limitations and the future. *Energy and Built Environment*. 2023. DOI: 10.1016/j.enbenv.2023.06.005.

10. Ilagan J. B., Ilagan J. R. A prototype of a conversational virtual university support agent powered by a large language model that addresses inquiries about policies in the student handbook. *Procedia Computer Science*. 2024. Vol. 239. P. 1124–1131. DOI: 10.1016/j.procs.2024.06.278.

11. Arslan M., Mahdjoubi L., Munawar S. Driving sustainable energy transitions with a multi-source RAG-LLM system. *Energy and Buildings*. 2024. Vol. 324. P. 114827. DOI: 10.1016/j.enbuild.2024.114827.

12. Vidivelli S., Ramachandran M., Dharunbalaji A. Efficiency-Driven Custom Chatbot Development: Unleashing LangChain, RAG, and Performance-Optimized LLM Fusion. *Computers, Materials & Continua*. 2024. P. 1–10. DOI: 10.32604/cmc.2024.054360.

13. Sharma A. et al. Automatic Data Transformation Using Large Language Model - An Experimental Study on Building Energy Data. *2023 IEEE International Conference on Big Data (BigData)*, Sorrento, Italy, 15–18 December 2023. 2023. DOI: 10.1109/bigdata59044.2023.10386931.

14. Giudici M. et al. Designing Home Automation Routines Using an LLM-Based Chatbot. *Designs*. 2024. Vol. 8, no. 3. P. 43. DOI: 10.3390/designs8030043.

15. Fortuna C., Hanžel V., Bertalanič B. Natural Language Interaction with a Household Electricity Knowledge-based Digital Twin. *2024 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, Oslo, Norway, 17–20 September 2024. P. 8–14. DOI: 10.1109/smartgridcomm60555.2024.10738062.

16. Fortuna C., Hanžel V., Bertalanič B. Towards Data-Driven Electricity Management: Multi-Region Harmonized Data and Knowledge Graph. URL: http://arxiv.org/abs/2405.18869.

17. Historical trend of graph DBMS popularity. DB-Engines - Knowledge Base of Relational and NoSQL Database Management Systems. URL: https://db-engines.com/en/ranking_trend/graph+dbms.

18. Liebig, T. GraphScale: Adding Expressive Reasoning to Semantic Data Stores / Liebig, T., Vialard, V., Opitz, M., & Metzl, S. *Proceedings of the 14th International Semantic Web Conference (ISWC 2015)*. Vol 1486. 2015. URL: https://ceur-ws.org/Vol-1486/paper_117.pdf.