УДК 005:623
DOI: https://doi.org/1034169/2414-0651.2025.1(45).0-0

В. І. СЛЮСАР, *доктор технічних наук, професор*
https://orcid.org/0000-0002-2912-3149

І. П. ГУСАКОВСЬКИЙ
https://orcid.org/0009-0007-9421-9749

ПРАКТИЧНІ АСПЕКТИ РОЗГОРТАННЯ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ В ЛОКАЛЬНИХ МЕРЕЖАХ

У статті розглядаються практичні аспекти розгортання великих мовних моделей (LLM) у локальних мережах для обробки конфіденційної інформації. Автори аналізують можливості фреймворків, таких як LM Studio, Ollama, Anything LLM та Koboldcpp, у різних сценаріях архітектури клієнт-сервер. Особливу увагу приділено функціональним обмеженням, технічним нюансам налаштування, а також ефективності й безпеці використання систем для вирішення завдань у військовій та науковій сферах. Зокрема, описано комбінацію фреймворків Anything LLM та Ollama, яка забезпечує інтеграцію з базами знань через механізм Retrieval-Augmented Generation (RAG), дозволяючи організувати захищену обробку даних у локальному середовищі.

Викладено переваги архітектури на базі фреймворку Koboldcpp, що підтримує кастомізацію моделей та інтеграцію з GPU для прискорення обчислень. Розглянуто проблеми, які виникають при використанні LM Studio для створення серверної інфраструктури через обмежену функціональність API та складнощі з налаштуванням доступу. Наголошується, що оптимальним вибором для локального розгортання LLM у клієнт-серверній архітектурі є поєднання Anything LLM як інтерфейсу користувача та Koboldcpp як серверної платформи. Така система забезпечує гнучкість, економію ресурсів і високу захищеність даних, що особливо важливо для застосувань у сфері національної безпеки.

Ключові слова: *локальні великі мовні моделі, обробка конфіденційної інформації, клієнт-серверна архітектура, фреймворки LLM, локальна мережа, Retrieval-Augmented Generation, безпека даних, налаштування серверів.*

ВСТУП

Запропонована в [1] загальна концепція застосування локальних великих мовних моделей (LLM) типу GPT-4o [2] та O1 від OpenAI [3] для обробки конфіденційної інформації досить просто реалізується при розгортанні LLM за допомогою спеціальних фреймворків типу LM Studio [4], Ollama [5] та ін. Їх дослідна експлуатація, проведена авторами в інтересах технічних перекладів, підготовки звітів і виконання інших завдань у сфері розвитку озброєння та військової техніки, засвідчила свою ефективність.

Сценарій, коли доступ користувача до LLM здійснюється без використання серверних спроможностей фреймворків, є найбільш доступним для втілення. Дещо складнішим стає варіант серверного доступу до LLM, який потребує певних попередніх налаштувань, однак в режимі Localhost його забезпечують майже всі зазначені фреймворки. Нюанси виникають, коли для подолання обмеженості обчислювальних ресурсів конкретного комп'ютерного обладнання потрібно рознести у просторі користувача і сервер з LLM в рамках локальної мережі з організацією доступу до LLM з клієнтських терміналів. Крім проблем адміністрування локальної мережі, у такому випадку виникають й інші практичні питання, оскільки, як виявилося, не всі LLM-фреймворки можуть бути задіяні для реалізації такої архітектури. Окрім функціональних обмежень, додатковою складністю є сумісність різних фреймворків, оскільки не всі з них підтримують архітектуру клієнт-сервер у локальних мережах, що вимагає тестування і перевірки можливості взаємодії між ними у вибраному сценарії.

Мета статті полягає у дослідженні особливостей розгортання локальних великих мовних моделей (LLM) для обробки конфіденційної інформації з акцентом на різні архітектурні рішення та інструменти реалізації. Зокрема, стаття буде зосереджена на аналізі сценаріїв, коли доступ до LLM здійснюється в рамках клієнт-серверної архітектури у локальній мережі. Особлива увага приділятиметься порівнянню можливостей фреймворків, таких як LM Studio, Ollama, Anything LLM [6] та ін., в контексті їх здатності підтримувати віддалений доступ до LLM, а також розгляду технічних нюансів, що виникають при їх використанні для забезпечення надійності та безпеки обробки даних.

АРХІТЕКТУРА «КЛІЄНТ-СЕРВЕР» НА ОСНОВІ КОМБІНАЦІЇ ФРЕЙМВОРКІВ ANYTHING LLM ТА OLLAMA

Ефективний підхід для розгортання локальних LLM з можливістю віддаленого доступу до них в локальній мережі полягає у використанні комбінації фреймворків Anything LLM [6] та Ollama [5] у сценарії клієнт-сервер. На рис. 1 представлено загальну архітектуру такої системи, що використовує комбінацію фреймворку Anything LLM та серверної платформи Ollama для розгортання LLM у клієнт-серверному середовищі з додатковою підтримкою Retrieval-Augmented Generation (RAG) [7]. Архітектура системи поділена на кілька функціональних блоків: апаратне забезпечення та мережева інфраструктура, програмне забезпечення, а також модуль RAG, який забезпечує інтеграцію з базами знань та конфіденційних документів.

Апаратне забезпечення та мережева інфраструктура включають сервери Ollama, оснащені процесорами (CPU) та графічними процесорами (GPU) для забезпечення ефективної роботи LLM. Сервери підключені до локальної мережі (LAN), що дозволяє авторизованим

користувачам отримувати доступ до системи. Для підвищення рівня безпеки використовується сегментація мережі, що забезпечує ізоляцію різних компонентів системи та контролює доступ між ними.

Програмне забезпечення представлено інтерфейсом користувача, який реалізовано за допомогою фреймворку Anything LLM, що забезпечує взаємодію користувачів з моделлю LLaMa-3.1-8B [8]. Інтерфейс дозволяє користувачам, які працюють на робочих станціях, надсилати запити до LLM та дистанційно отримувати результати обробки відповідних інструкцій.

Для зберігання даних використовується сховище на основі швидкісних накопичувачів SSD, яке гарантує швидкий доступ до необхідних файлів. За потреби, для роботи з моделлю LLaMa-3.1-8B можуть бути використані фреймворки глибокого навчання, такі як PyTorch та TensorFlow, для забезпечення ефективних обчислень та донавчання LLM на документах користувачів.

Retrieval-Augmented Generation (RAG) включає модуль інтеграції, який забезпечує взаємодію моделі LLM із зовнішніми базами знань. Механізм пошуку (Retriever) використовується для отримання релевантної інформації з загальної бази знань або персональної бази знань, яка може містити специфічну інформацію, необхідну для більш точного контекстного аналізу [7]. Це дозволяє моделі генерувати більш релевантні та обґрунтовані відповіді на запити користувачів. Такий підхід дозволяє організувати обробку конфіденційної інформації, забезпечуючи при цьому гнучкість в управлінні доступом та масштабованість системи. Розглянемо детальніше, як можна налаштувати та інтегрувати зазначені фреймворки для досягнення цієї мети.

Серверна частина системи передбачає розгортання Ollama, що відповідає за локальну роботу LLM, використовуючи ресурси серверу, зокрема GPU та CPU, для ефективної обробки запитів. Серверна компонента додатково налаштовується за допомогою фреймворку Anything LLM, що дозволяє управляти моделлю, обробляти запити та відправляти відповіді клієнтам на основі графічного інтерфейсу. Користувачі отримують доступ до LLM через клієнтську частину, яка підключається до серверу за допомогою локальної мережі, використовуючи IP-адресу сервера та визначений порт. Запити від клієнтів вводяться через діалоговий інтерфейс чату з LLM, що інтегрований до Anything LLM. Далі цей фреймворк виконує функції маршрутизації, забезпечення безпеки та передачі запиту до Ollama, яка обробляє їх, використовуючи LLM, і повертає відповідь клієнтам. Взаємодія між клієнтом та сервером може здійснюватися у форматі JSON, що дозволяє легко обробляти та використовувати отримані дані в автоматизованих процесах.

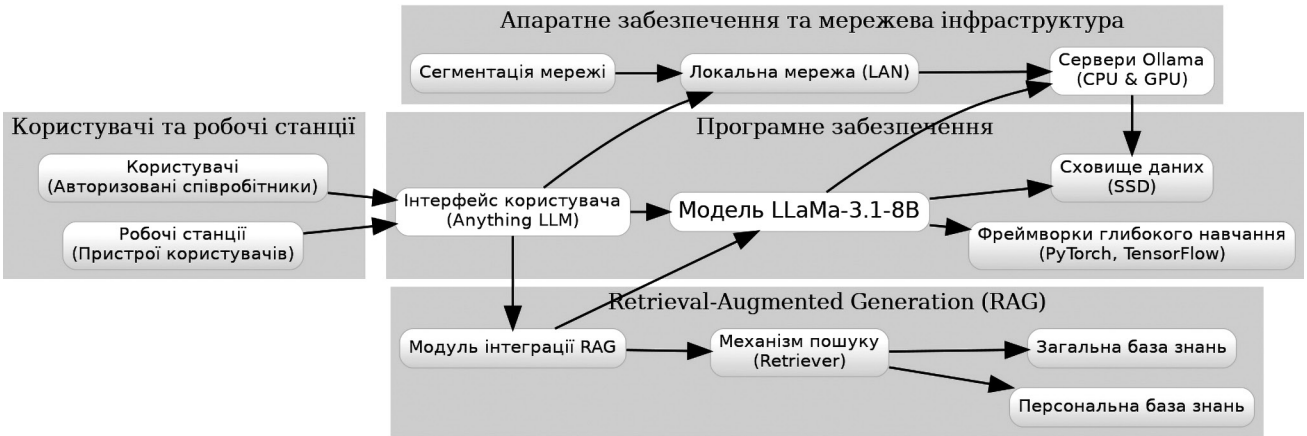
Зазначена архітектура має низку переваг, основною з яких є забезпечення безпеки, оскільки система функціонує в локальній мережі, що ізолює дані від Інтернет, підвищуючи захист конфіденційної інформації. Використання Anything LLM спирається на централізоване управління системою, що спрощує керування доступом, моніторинг активності та масштабування. Зокрема, можна віддалено перемикатися з однієї LLM на іншу без потреби перезапуску Ollama-серверу.

Розглянута архітектура клієнт-сервер також забезпечує масштабованість системи шляхом додавання нових мовних моделей на сервер Ollama, де кожний клієнт (користувач) має доступ до різних мовних моделей, при цьому всі вони можуть працювати з різними LLM (рис. 2).

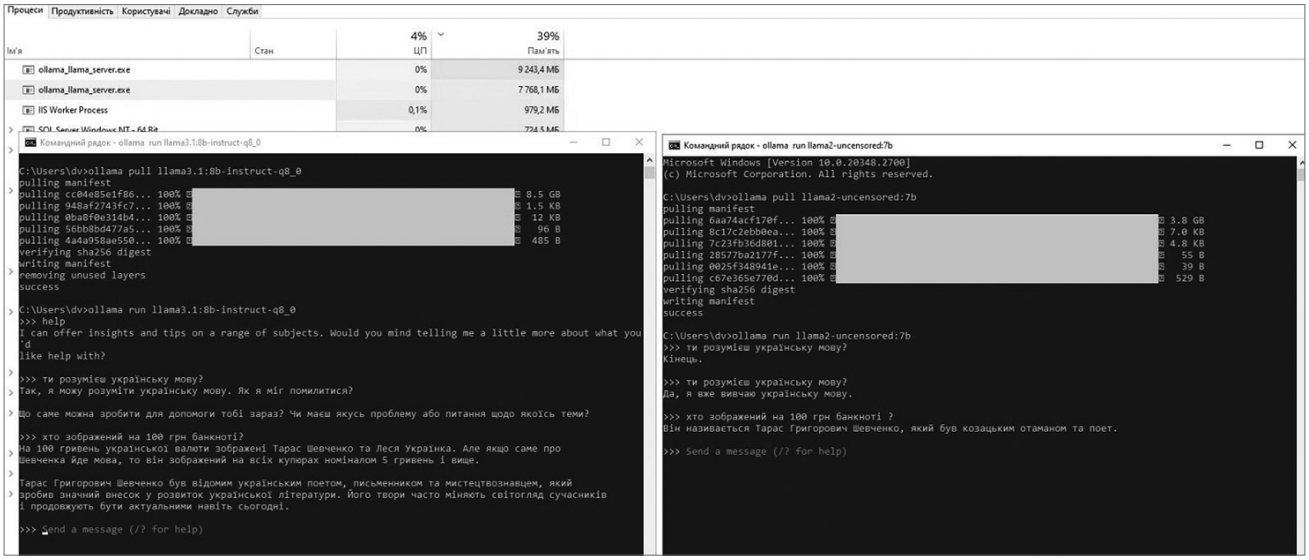
Суттєво, що операційна система Windows допускає також можливість запуску кількох різних серверів Ollama одночасно, кожному з яких призначається свій IP-адрес. Це дозволяє розділити користувачів за різними мовними моделями й дещо прискорює роботу кількох з них порівняно з випадком сумісного доступу кількох клієнтів до одного сервера.

Водночас, одним з викликів є необхідність налаштування локальної мережі для забезпечення доступу сервера для клієнтів, що може вимагати певних знань у сфері мережевого адміністрування. Продуктивність системи також залежить від потужності серверу; оскільки у разі недостатності ресурсів, наприклад, GPU, можуть виникати помітні затримки в обробці запитів.

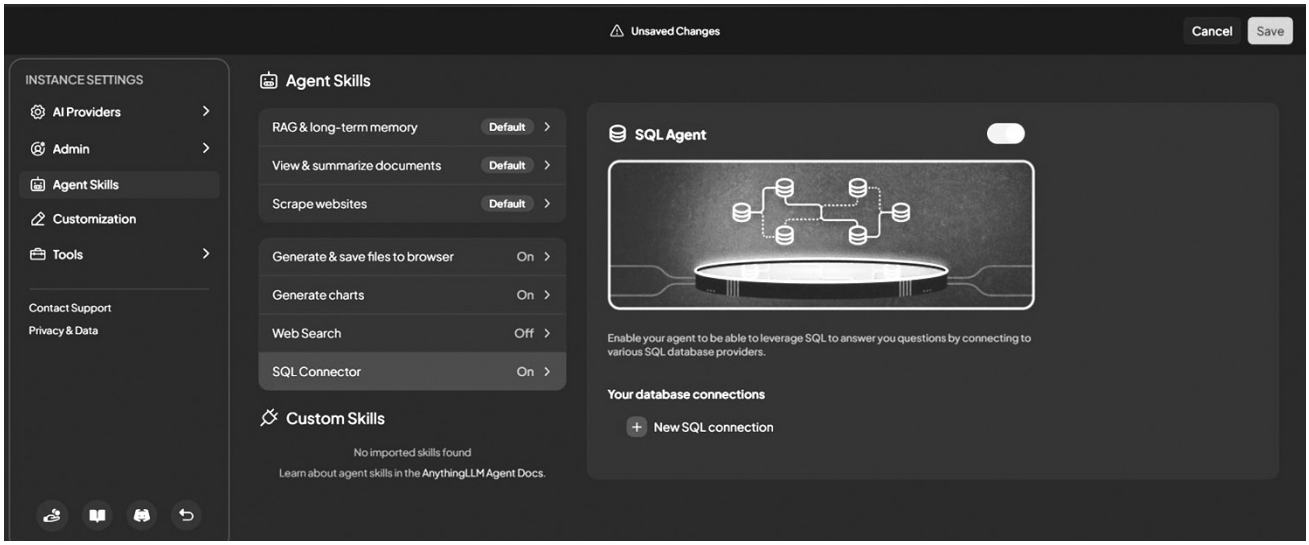
Для успішного впровадження описаної архітектури необхідно провести її тестування з метою оцінки продуктивності системи та виявлення можливих вузьких місць. Зокрема, як з'ясувалося, особливістю Anything LLM є неможливість передачі для аналізу мультимодальній LLM, наприклад, LLaVa [9], зображень, якщо вони попередньо не розміщені у спеціальній базі даних. На рис. 3, 4 продемонстровано етапи підключення ко-



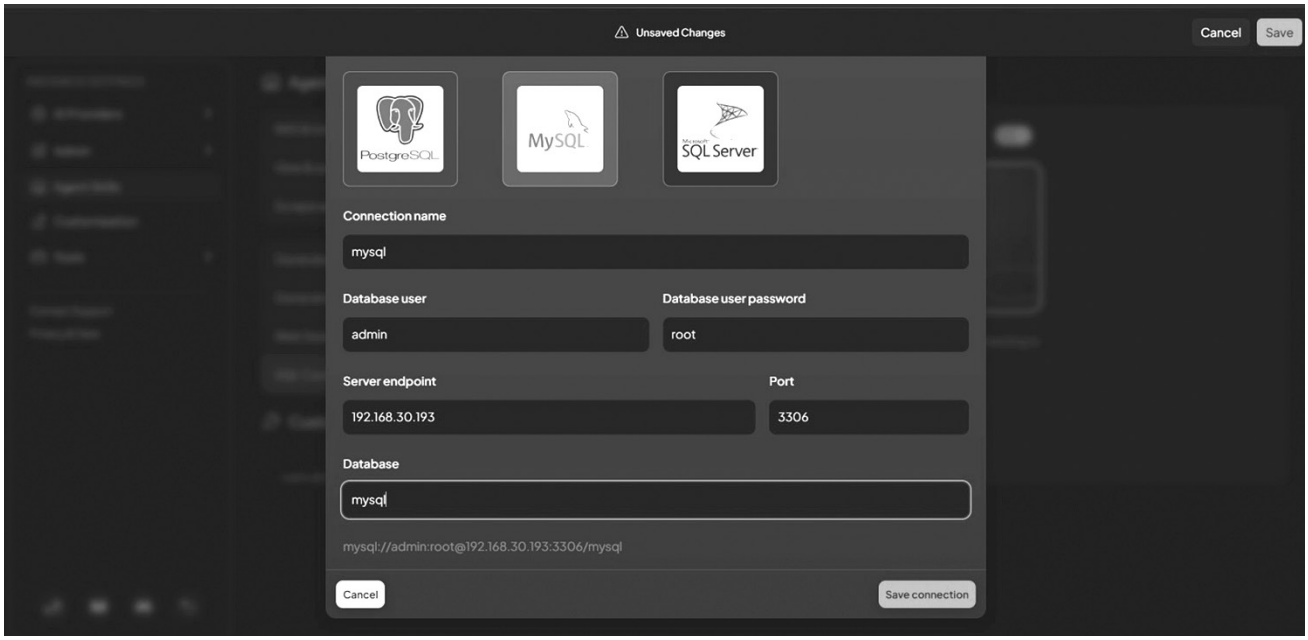
Р и с . 1. Типовий варіант розгортання локальної LLM на прикладі україномовної версії Llama-3.1-8B



Р и с . 2. Демонстрація можливості запуску двох моделей на одному сервері Ollama в операційній системі Windows



Р и с . 3. Підключення системи керування базою даних у налаштуваннях AnythingLLM



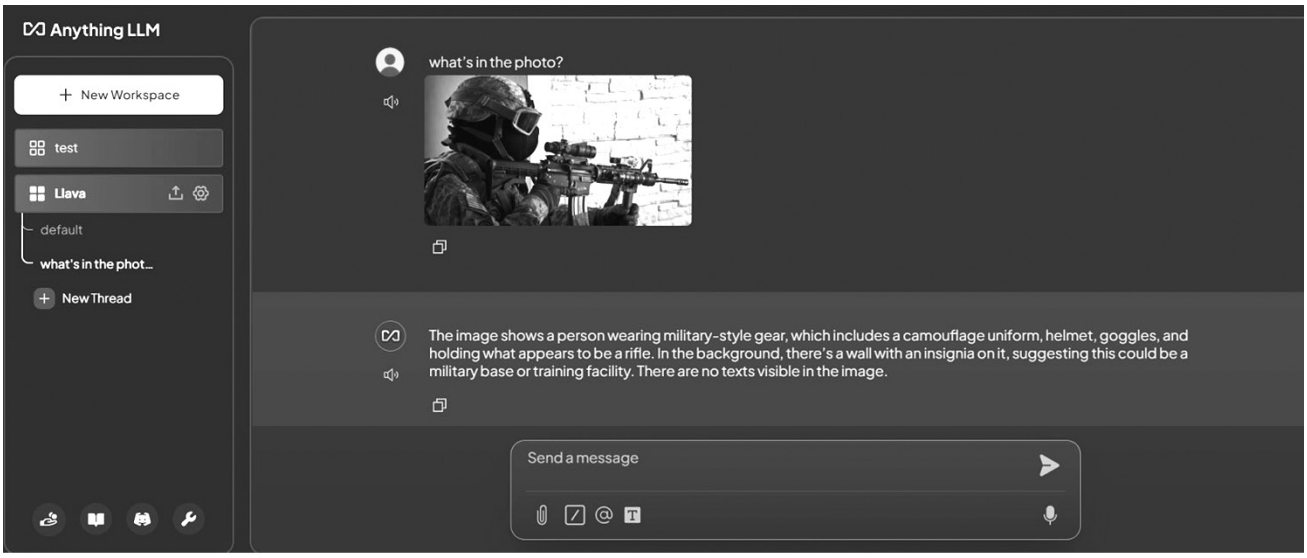
Р и с . 4. Вибір системи керування базою даних та підключення до серверу бази даних

ристувача до серверу такої бази даних і налаштування SQL-клієнта, що інтегрований у середовище Anything LLM, під потреби конкретного завдання. Після проведення підготовчих робіт користувач має можливість надсилати запит до бази даних, використовуючи SQL-клієнта, при цьому важливо правильно налаштувати запит, щоб він відповідав структурі бази даних та забезпечував доступ до потрібного зображення. Після відправлення запиту мультимодальна LLM LLaVa виконує його аналіз і створює опис чи аналізує деталі зображення згідно із завданням користувача. На рис. 5 наведено приклад текстового опису, отриманого після обробки зображення, на якому віддалено розгорнута LLaVa успішно розпізнала об'єкти, сцену та контекст зображення, що дозволяє інтегрувати ці дані в подальший аналіз чи звіти. Такий підхід особливо корисний для завдань, де необхідно автоматизувати процес отримання опису великої кількості зображень, що містяться у спеціалізованих базах даних, датасетах, презентаціях тощо.

В процесі тестування і повсякденного використання необхідно застосовувати систему моніторингу та документування дій для вчасного виявлення проблем і проведення аудиту доступу. Логування процесів функціонування серверу Ollama відбувається в файлах кореневої теки серверу (рис. 6). В ній за допомогою функції «Viewlogs» можна побачити файли з логами у форматі *.txt, зміст яких відкривається за допомогою стандартного застосунку перегляду текстових файлів (рис. 7).

До основних недоліків описаного підходу можна віднести відсутність детального опису процесу роботи різних мовних моделей на сервері генерації Ollama. Зокрема, досить поверхнева інформація, яка міститься у вказаних логах, не дозволяє оцінити швидкодію конкретних LLM у токенах за секунду, розміри запитів користувачів та обсяги відповідей LLM у токенах. Також до недоліків можна віднести те, що сервер генерації Ollama використовує лише офіційні мовні моделі, які необхідно завантажувати з порталу ollama.com, що ускладнює процес кастомізації мовних моделей на основі технологій тюнингу [9].

В цілому ж, у військовій сфері запропонована комбінація фреймворків Anything LLM та Ollama може ефективно застосовуватися для обробки розвідувальних даних, автоматизації формування бойових наказів, створення звітів і підтримки прийняття рішень у режимі реального часу [10, 11]. Інтеграція Anything LLM як клієнтського інтерфейсу з можливістю надання доступу через локальну мережу командних пунктів, штабів та установ забезпечить мультимодальний сервіс для командирів й аналітиків, дозволяючи їм оперативно обробляти



Р и с . 5. Результат опису зображення за допомогою LLaVa v.1.5

Ім'я	Дата змінення	Тип	Розмір
server.log	30.09.2024 15:15	Текстовий докум...	241 КБ
app.log	30.09.2024 11:33	Текстовий докум...	1 КБ
app-2.log	30.09.2024 11:09	Текстовий докум...	3 КБ
server-2.log	30.09.2024 11:08	Текстовий докум...	51 КБ
server-4.log	30.09.2024 8:27	Текстовий докум...	43 КБ
app-4.log	27.09.2024 15:45	Текстовий докум...	1 КБ
server-5.log	27.09.2024 15:40	Текстовий докум...	158 КБ
app-5.log	26.09.2024 1:07	Текстовий докум...	2 КБ
upgrade.log	25.09.2024 23:46	Текстовий докум...	253 КБ
config.json	05.08.2024 9:46	Файл JSON	1 КБ
updates	25.09.2024 21:48	Папка файлів	

Р и с . 6. Лог-файли на сервері Ollama

Р и с . 7. Відкриті лог-файли роботи серверу та мовних моделей

різноманітну інформацію [12−14] та приймати обґрунтовані рішення. При цьому для підтримки надійності та актуальності системи важливим є регулярне оновлення моделей LLM, фреймворків та систем безпеки для забезпечення стійкої захищеності даних.

АРХІТЕКТУРНІ РІШЕННЯ «КЛІЄНТ-СЕРВЕР» НА ОСНОВІ ФРЕЙМВОРКУ KOBOLDCPP

Іншим можливим варіантом архітектури ізольованої системи з локальною LLM є застосування фреймворку Koboldcpp [15].

Програмне забезпечення Koboldcpp базується на технології llama.cpp і включає API для більшої сумісності та розширених функцій, таких як багатофункціональний користувацький інтерфейс (KoboldCpp Web Application), що підтримує роботу з постійними сценаріями, інструментами редагування, а також різними функціями пам’яті.

В контексті розгортання Koboldcpp у клієнт-серверній архітектурі слід вказати, що серверна частина зазвичай розташовується локально, що дозволяє іншим користувачам у локальній мережі підключатися до неї як клієнт через вказану точку доступу. Розробники фреймворку Koboldcpp пропонують готові бінарні файли для таких платформ, як Windows, Linux та MacOS, що робить його зручним для використання на різних операційних системах. Запуск Koboldcpp можливий як через графічний інтерфейс, так і через командний рядок, забезпечуючи гнучкість як для початківців, так і для досвідчених користувачів. Така архітектура надає можливість розгорнути сервер на локальному комп’ютері (localhost) і надавати клієнтам у мережі доступ до функцій мовних моделей.

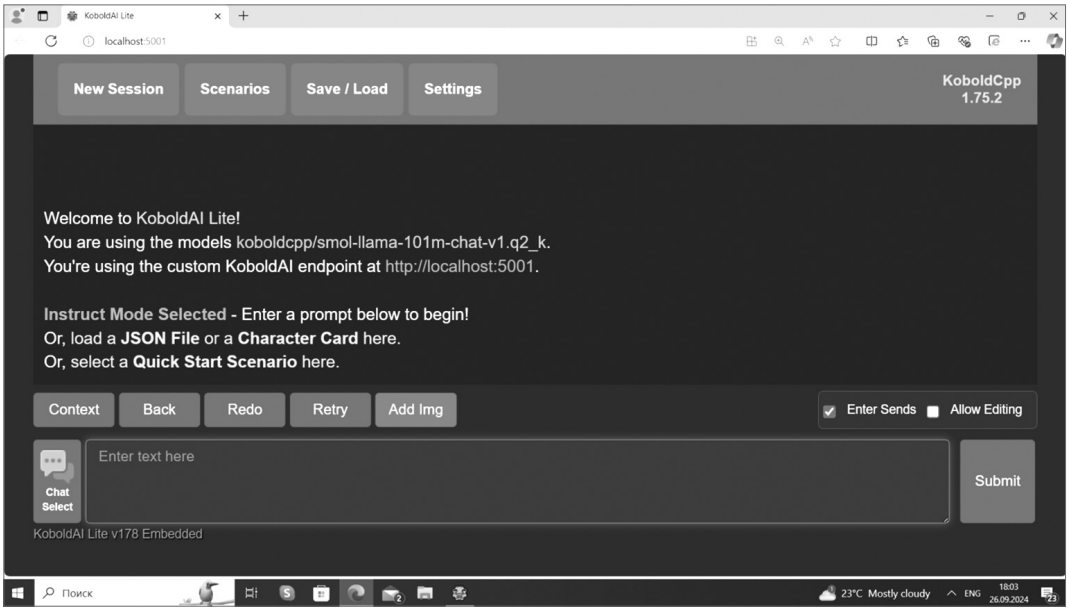
Для прискорення обчислень Koboldcpp підтримує використання GPU, зокрема CUDA для NVIDIA та CLBlast для AMD графічних карт. Крім цього, реалізована можливість перенесення шарів обчислень на GPU, що дозволяє значно прискорити час виконання завдань, хоча для цього потрібно мати більше пам’яті в GPU. Відповідний механізм налаштовується за допомогою команд, таких як `--usecublas` для CUDA-прискорення, та `--gpulayers` для вибору кількості шарів, які обробляються на GPU.

У клієнт-серверному режимі серверна частина Koboldcpp запускається із завантаженою мовною моделлю (наприклад, команда `koboldcpp.py [ggml_model.bin] [port]`). Користувачі або клієнти підключаються до сервера через вказану точку доступу (наприклад, `http://localhost:5001`), що дозволяє забезпечити інтерактивну взаємодію, як це продемонстровано на представленому на рис. 8 зображенні.

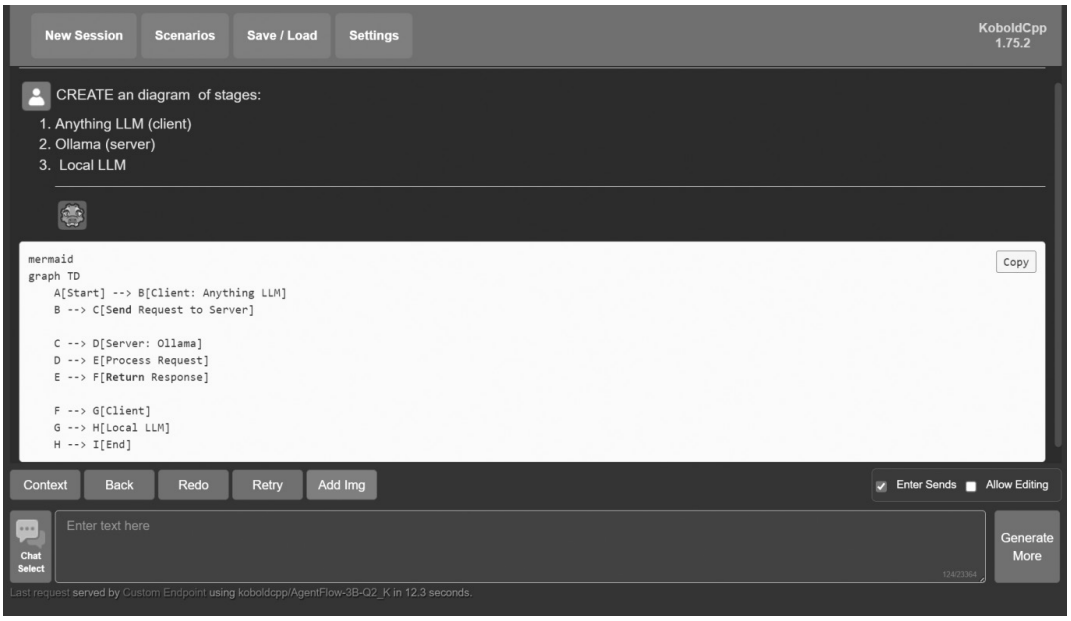
Графічний інтерфейс Koboldcpp підтримує різні режими діалогу, такі як «Instruct Mode», що дозволяє користувачам вводити запити безпосередньо в текстове поле і отримувати відповіді від мовної моделі. При цьому LLM можуть виконувати різні інструкції, подібно до сценаріїв обробки даних, які розглядаються в архітектурі системи локальних LLM [1].

Інтерфейс користувача також містить кнопки «Context», «Back», «Redo», «Retry» та «Add Img» для підтримки інтерактивної роботи, включаючи можливість додавання зображень для розширення контексту. Така функціональність може бути корисною в архітектурі, де клієнти надсилають запити до серверної частини для обробки інформації, зокрема, при використанні мультимодальних моделей, що можуть працювати з текстом та зображеннями.

Інтеграція Koboldcpp до клієнт-серверної архітектури дозволяє виконувати завдання, які були описані в [1], а саме: централізоване управління LLM, забезпечення захисту інформації, а також розширення функціональних можливостей системи для роботи з різними типами даних (текст, зображення тощо). Наприклад, на рис. 9 продемонстровано варіант запуску агентної мовної моделі [16] для описаної в [1] задачі генерації коду діаграми процесів.



Р и с . 8. Графічний інтерфейс Koboldcpp

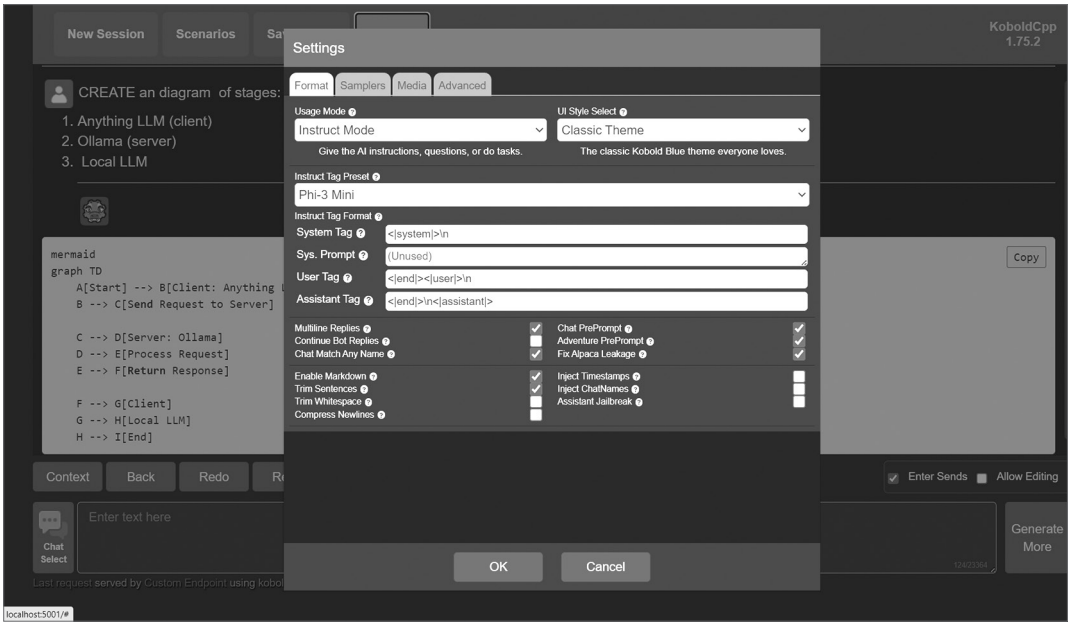


Р и с . 9. Приклад запуску агентної мовної моделі для генерації коду діаграми процесів

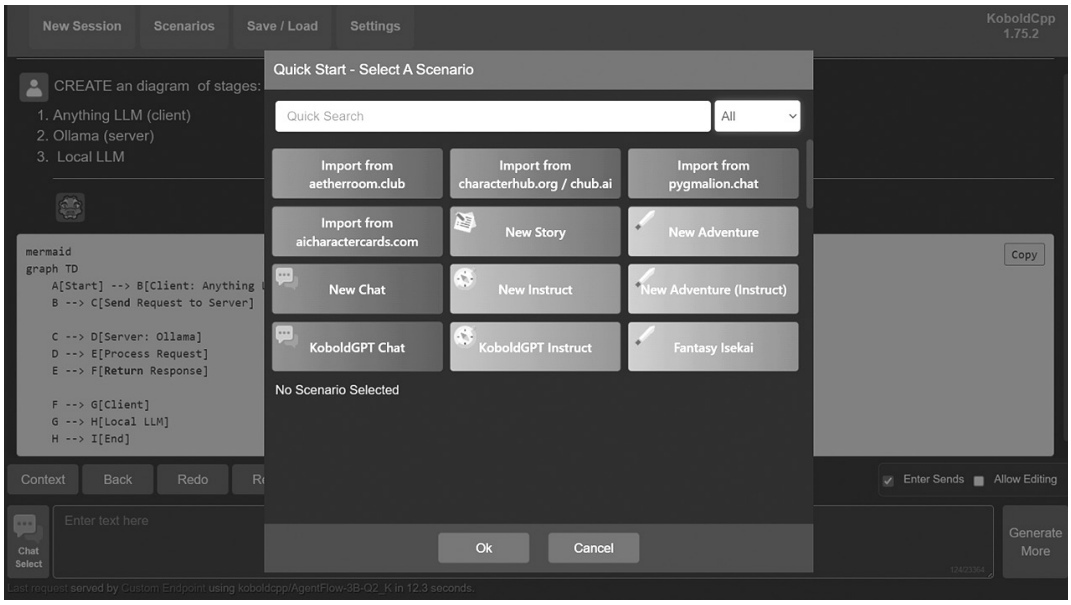
Таким чином, Koboldcpp може бути ефективним інструментом у рамках локальної архітектури для обробки конфіденційної інформації, забезпечуючи можливість використання великих мовних моделей без необхідності підключення до Інтернету. Для детального налаштування фреймворку слід використовувати спеціальні параметри, вибір яких передбачений у меню інтерфейсу Settings (рис. 10). Крім того, для виконання типових завдань доцільно обирати предустановлені типові сценарії (рис. 11).

Розглянемо далі протестовані авторами комбінації Koboldcpp з іншими фреймворками. Це може бути реалізовано як за допомогою браузерного інтерфейсу, так і через спеціалізовані додатки, налаштовані для взаємодії з Anything LLM. Як вже зазначалося, використання таких інструментів, як Anything LLM, спрощує взаємодію користувачів із системою, полегшуючи доступ кількох робочих станцій до серверної частини з різноманітними функціями, такими як аналіз текстів або інтерактивні бесіди з застосуванням технології RAG.

Комбінація фреймворків, де Anything LLM виступає в ролі інтерфейсу для управління запитами та командами, а Koboldcpp − як обчислювальний бекенд, що альтернативний Ollama, надає можливість створити локальну систему штучного інтелекту з широкими можливостями. При цьому вдається обійти обмеження Ollama щодо придатних до використання мовних моделей. Така система здатна виконувати як базові завдання, так і більш складні інтерактивні сценарії у розподіленій архітектурі, що особливо важливо для захищених середовищ, де чутливі дані не можуть бути передані на зовнішні сервери. Це забезпечує повний контроль над даними та обчисленнями, залишаючи їх всередині захищеної локальної мережі.



Р и с . 10. Детальне налаштування фреймворку Koboldcpp за допомогою спеціальних параметрів у меню інтерфейсу «Settings»



Р и с . 11. Меню вибору предустановлених сценаріїв Koboldcpp для виконання типових завдань LLM

ОБМЕЖЕННЯ ФРЕЙМВОРКУ LM STUDIO

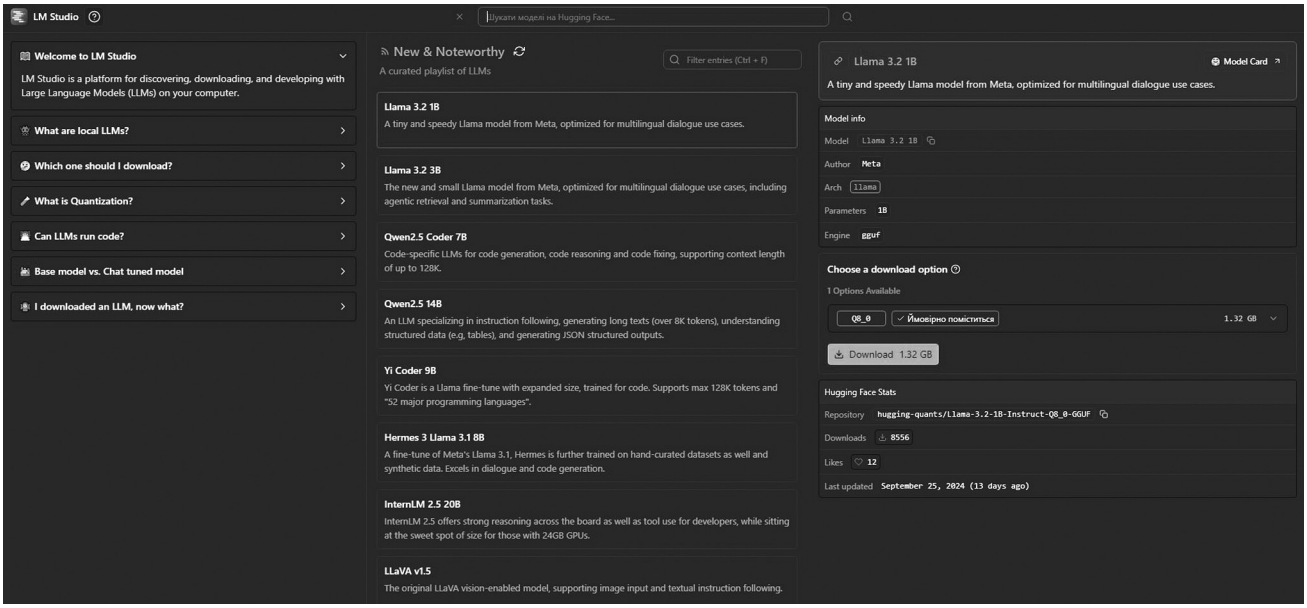
Як було вказано в [1], основною перевагою LM Studio виявилося завантаження та використання кастомних (модифікованих або спеціально скомпільованих) мовних моделей під конкретну задачу (рис. 12) та подальше їх локальне використання.

В процесі досліджень авторами було встановлено, що фреймворк LM Studio не може бути використаним в якості серверу з віддаленим доступом та окремим застосунком для підключення до іншого серверу генерацій LLM на базі Ollama або Koboldcpp через відсутність функціоналу клієнтської частини.

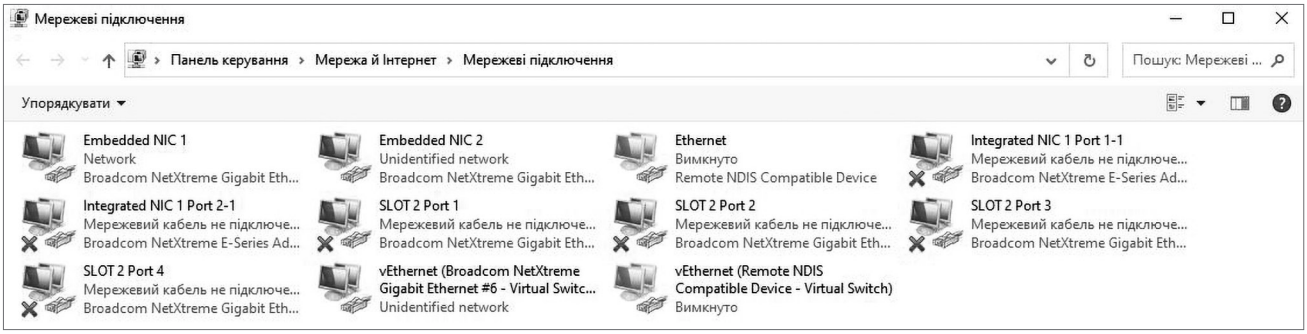
Причина полягає в тому, що при запуску функції «Сервер» на обчислювальній платформі фреймворк LM Studio рандомно (випадково) призначає IP адресу із доступних мережевих адаптерів, яких на сервері може налічуватися кілька (рис 13). Відповідно для того, щоб функція сервера запустилася на відповідній IP-адресі, необхідно вимикати всі наявні мережеві підключення окрім адаптера, на якому призначена відповідна IP-адреса, що робить проблематичним віддалений доступ без додаткової конфігурації.

Разом з тим, після проведення необхідної конфігурації та запуску функції сервера, попередньо зазначивши відповідний порт доступу на платформі LM Studio (рис. 14), вдається перевірити працездатність та доступність сервера генерації на локальній абонентський станції за допомогою функції «GET» через веб-застосунок і отримати назву мовної моделі, яка завантажена та використовується платформою LM Studio (рис. 15).

Переконавшись в ініціалізації мовної моделі в LM Studio, можна спробувати здійснити підключення кліа-



Р и с . 12. Основне меню завантаження мовних моделей



Р и с . 13. Приклад наявних мережевих адаптерів на сервері

єнтської частини AnythingLLM до серверу генерації LM Studio (рис. 16), зазначивши IP-адресу та порт доступу до сервера. В результаті модель, яка завантажена на сервері генерації, автоматично підключається на клієнтській частині.

Після вибору мовної моделі необхідно здійснити налаштування робочого простору AnythingLLM (рис. 17) та переконатися, що бажана LLM підключена у відповідному чаті. Однак успішне виконання описаних операцій не дозволило забезпечити досягнення головної мети дослідження, що полягала у реалізації чату з LLM, оскільки тестова відправка повідомлення у чаті призводить до виникнення помилки (рис. 18).

В логах на серверній частині (рис. 19) теж фіксується помилка «Only user and assistant roles are supported», що означає, що дана модель може використовуватися лише з ролями «користувач» та «асистент». В даному випадку локальний клієнт сервісу має роль «користувача», а сервер – «асистента».

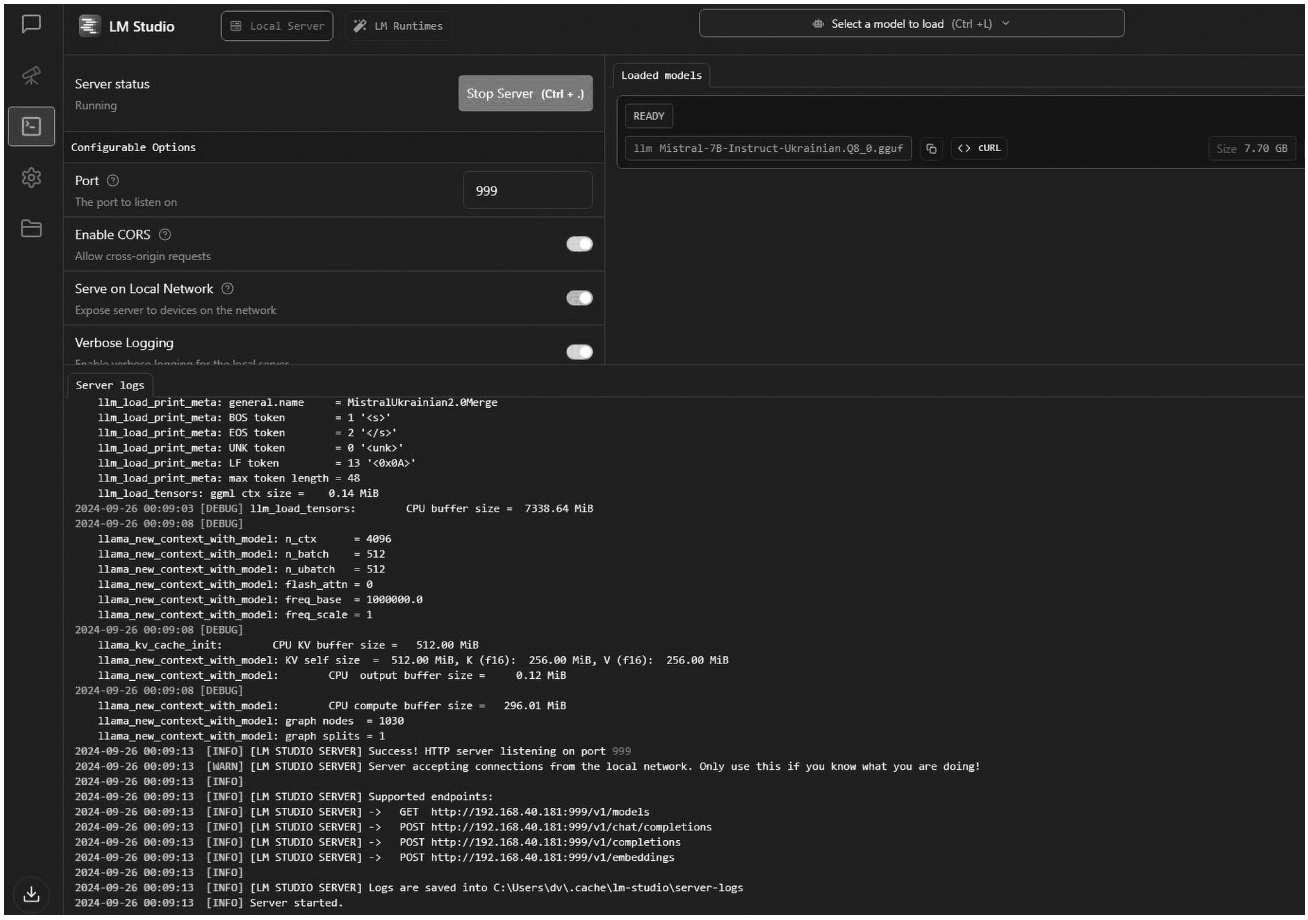
Відповідно, при запиті клієнтської частини AnythingLLM до серверу генерації LM Studio виконується компіляція з параметрами «role – system» та «role – user», що не припустимо для подальшої генерації відповіді на запит у зв'язку з некоректним параметром «System». Відповідно до технічної документації, зазначеної на офіційному сайті LM Studio та на платформі репозиторіїв GitHub, визначення даної помилки на момент написання статті не фіксувалося.

Також було виявлено неспроможність роботи з LM Studio в якості віддаленого клієнта у комбінації з серверами на базі інших фреймворків. Це означає, що найефективнішим варіантом використання LM Studio залишається розгортання локальної LLM на одному й тому ж комп'ютері з LM Studio [1].

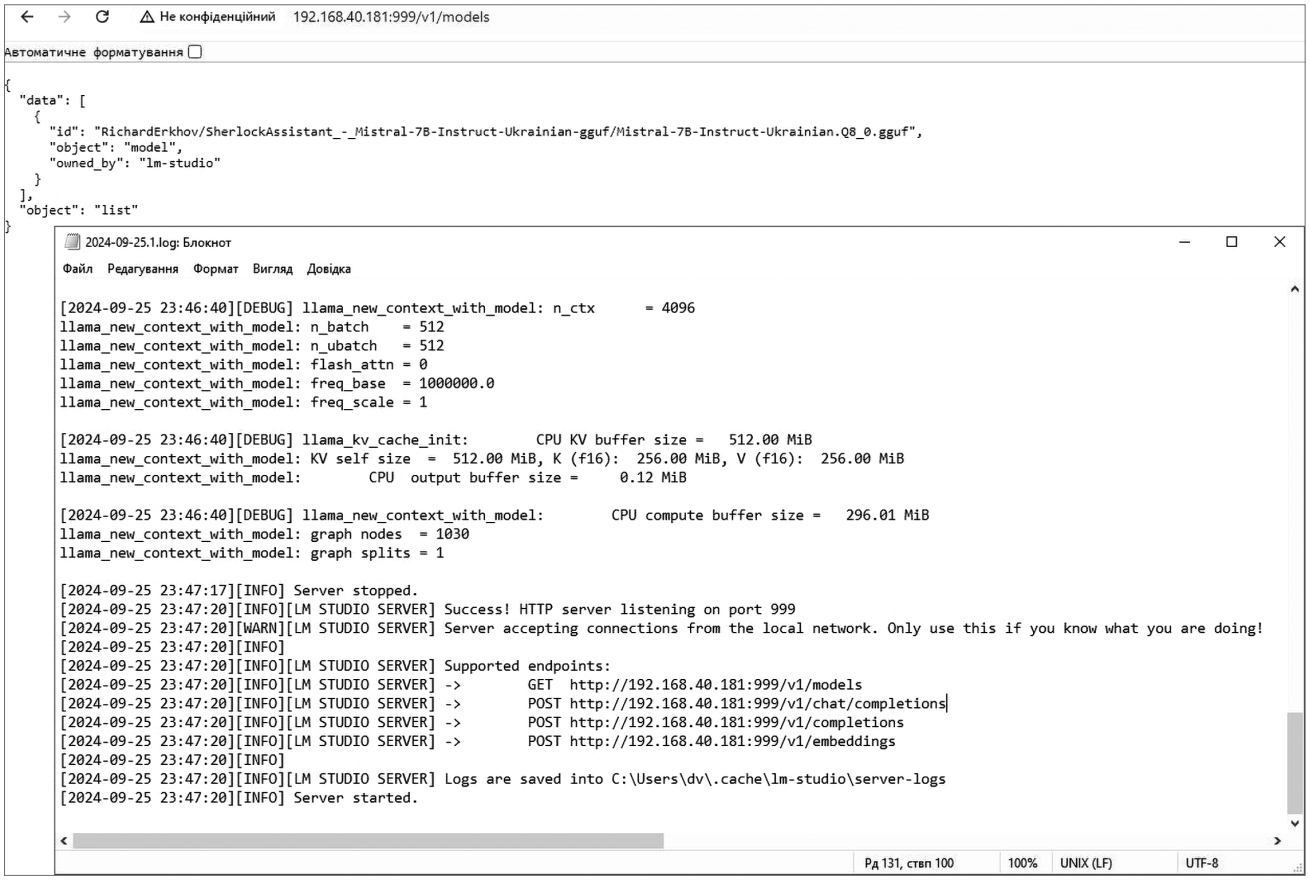
Вказані висновки були підтверджені на основі аналізу доступних матеріалів Інтернет-форумів та документації LM Studio [17].

Інтеграція LM Studio з іншими сервісами на основі REST API також ускладнена, оскільки API LM Studio не має необхідних можливостей для легкого контролю доступу. Ситуація може змінитися у майбутньому, якщо розробники запровадять підтримку повноцінного API або SDK для Python, що дозволить ефективніше інтегрувати LM Studio з іншими системами [18].

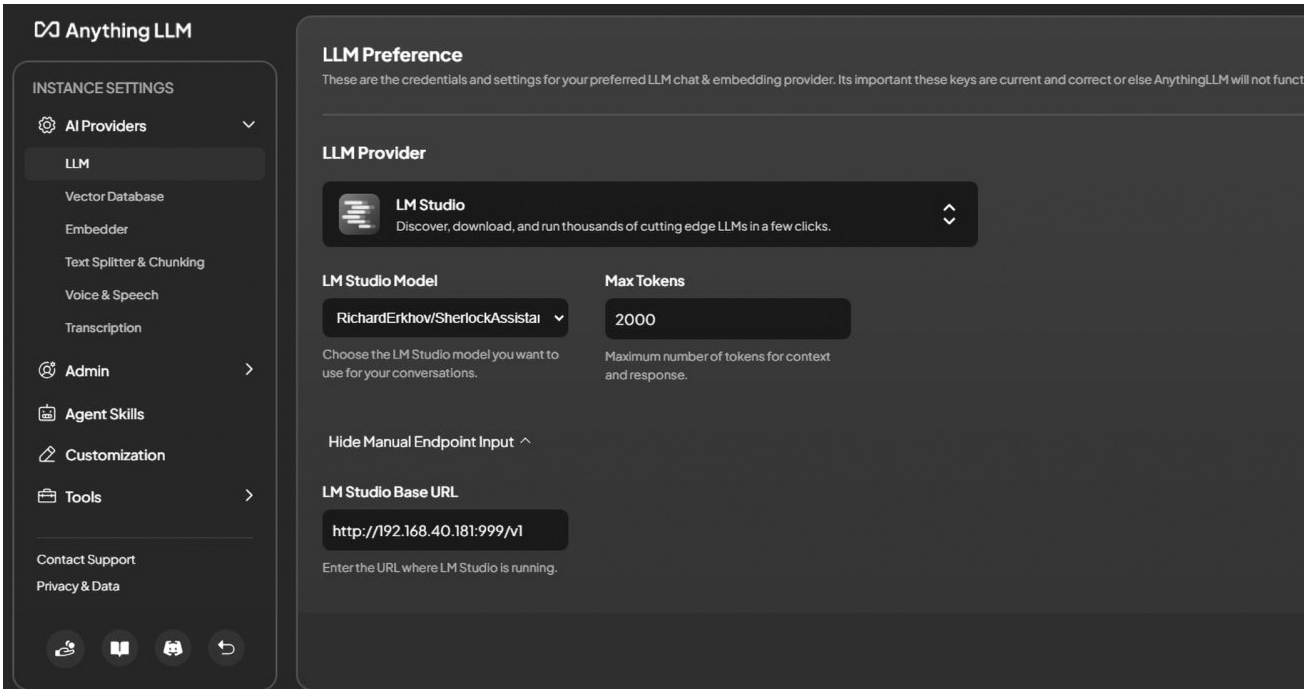
В якості висновку слід зазначити, що залежно від вирішуваного завдання на даному етапі розвитку технологій LLM кращим вибором серед розглянутих варіантів їх локального розгортання є застосування в якості серверних платформ фреймворків Ollama або Koboldcpp. У разі потреби у використанні додатково налаштованих за допомогою тюнинга мовних моделей Koboldcpp лишається єдиним серверним рішенням.



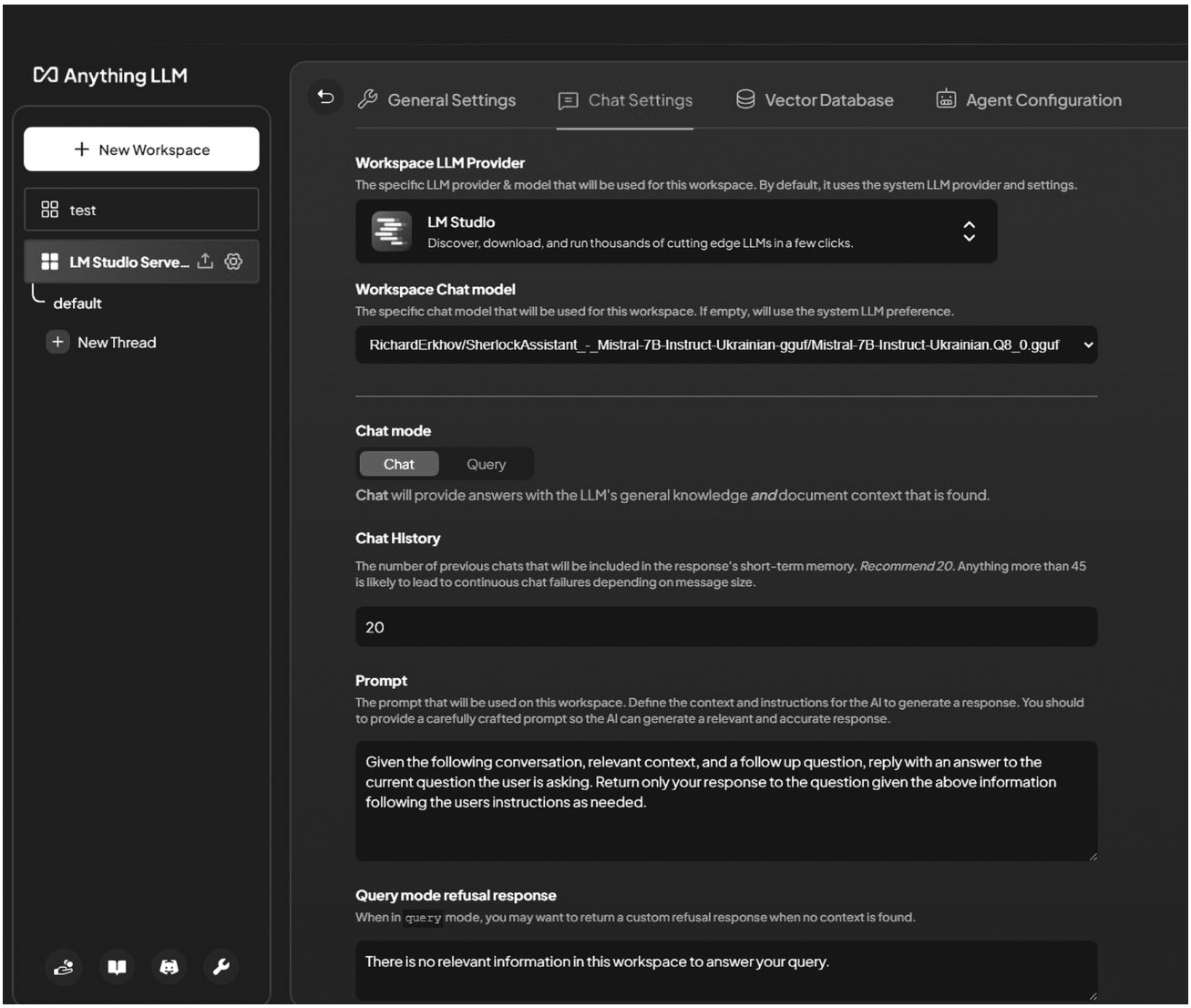
Р и с . 14. Запуск функції Сервер на платформі LM Studio



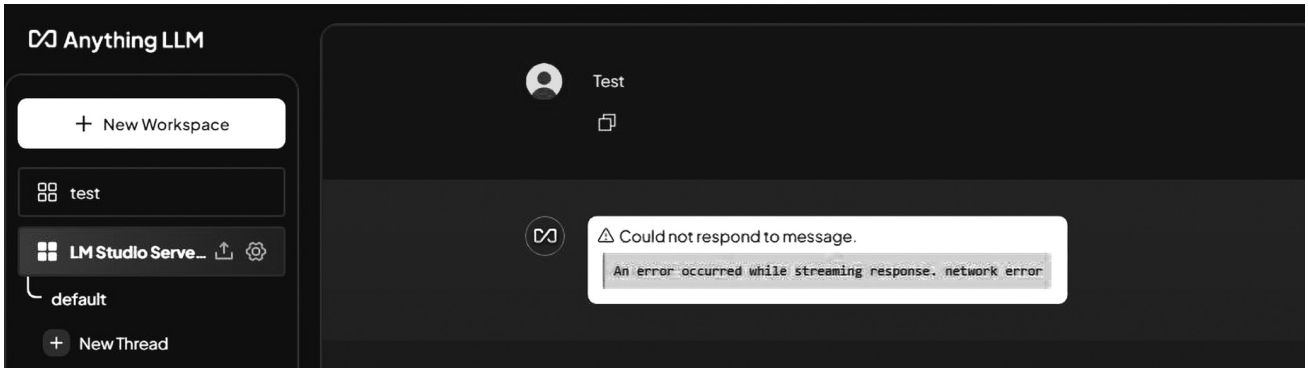
Р и с . 15. Результат перевірки доступу до серверної частини LM Studio



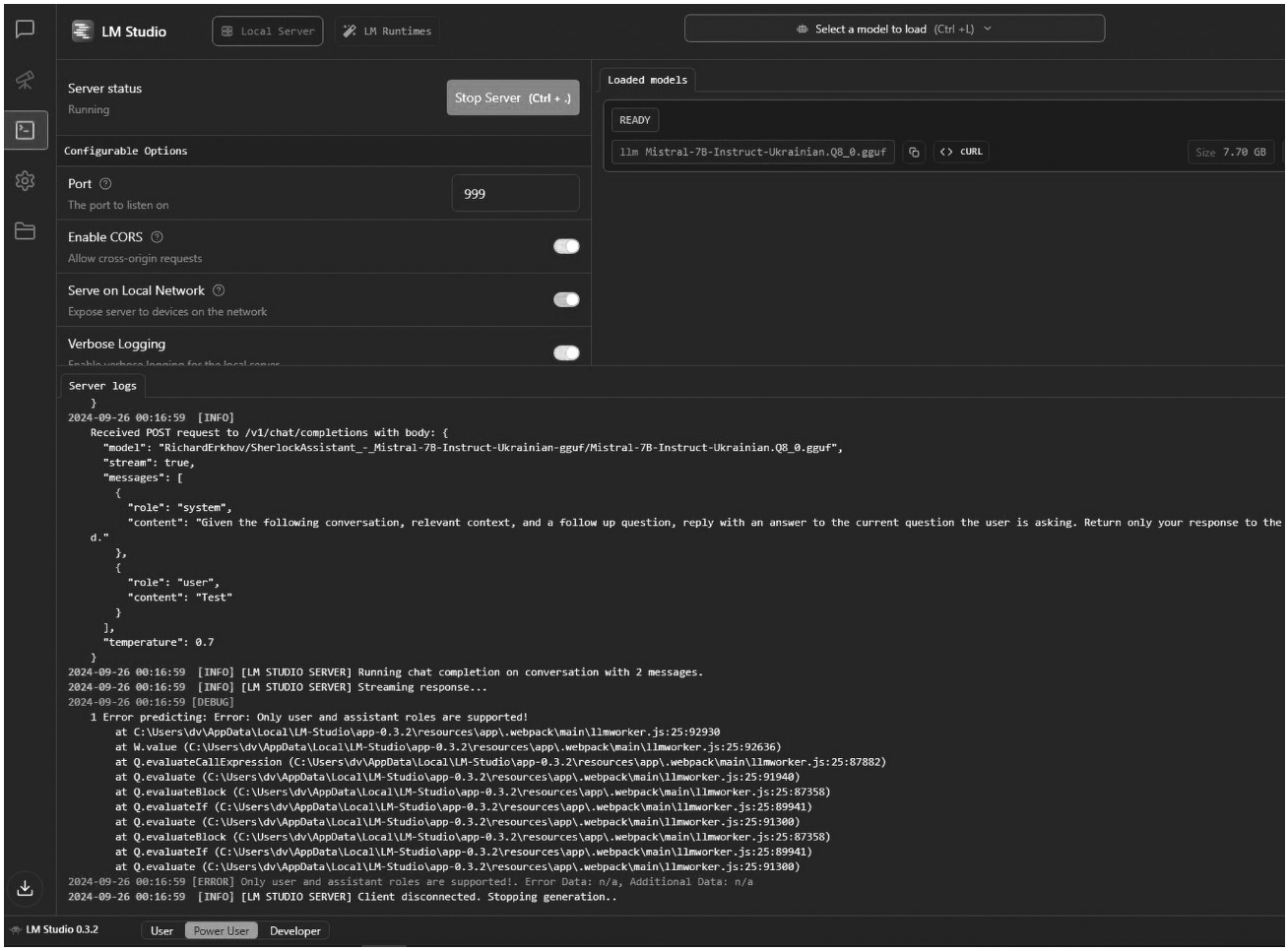
Р и с . 16. Налаштування клієнтської частини AnythingLLM



Р и с . 17. Налаштування робочого простору AnythingLLM



Р и с . 18. Результат виконання запиту «Test»



Р и с . 19. Лог виконання запиту «Test»

При цьому для підтримки локальних баз даних з конфіденційною інформацією на основі механізму RAG слід обирати комбінацію в якості клієнтського програмного застосунку Anything LLM, а сервера – Koboldcpp. Такий підхід дозволить зекономити кошти шляхом відмови від розробки власного або застосування пропрієтарного програмного забезпечення і створить передумови для впровадження технологій штучного інтелекту у повсякденну військову та наукову діяльність у короткі строки.

СПИСОК ПОСИЛАНЬ

1. Слюсар В.І. Локальні великі мовні моделі для обробки конфіденційної інформації. Озброєння та військова техніка. 2024. № 4(44). С. 79—91. https://doi.org/1034169/2414-0651.2024.4(44).79-91.
2. OpenAI. GPT-4 technical report. arXiv, 2024. [Online]. Available: https://arxiv.org/abs/2303.08774.
3. R. Marino. Fast Analysis of the OpenAI O1-Preview Model in Solving Random K-SAT Problem: Does the LLM Solve the Problem Itself or Call an External SAT Solver. arXiv preprint arXiv:2409.11232, 2024. [Online]. Available: https://arxiv.org/abs/2409.11232.
4. LM Studio. 2024. [Online]. Available: https://lmstudio.ai/.
5. Ollama. 2024. [Online]. Available: https://ollama.com/.
6. AnythingLLM. The all-in-one AI application, 2024. [Online]. Available: https://anythingllm.com/.

7. P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems. Vol. 33. Pp. 9459—9474. [Online]. Available: https://arxiv.org/abs/2005.11401.
8. A. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. Singh Chaplot, D. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. Renard Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, W. E. Sayed. Mistral 7B. 2023. 9 p. https://arxiv.org/pdf/2310.06825.pdf.
9. Parthasarathy, V.B., Zafar, A., Khan, A. & Shahid, A. (2024). The Ultimate Guide to Fine-Tuning LLMs from Basics to Breakthroughs: An Exhaustive Review of Technologies, Research, Best Practices, Applied Research Challenges and Opportunities. arXiv preprint arXiv:2408.13296. [Online]. Available: https://arxiv.org/abs/2408.13296.
10. Slyusar Vadym. Large language models (LLM) in the military area. XXIII Intern. Scientific and Technical Conf. «Artificial Intelligence and Intellegent Systems» (AIIS'2023). 10 − 11 October, 2023. https://doi.org/10.13140/RG.2.2.30196.94086.
11. Slyusar Vadym. Reducing the Cognitive Burden of a Soldier with the Help of Personal AI and LLM Assistant. The LCGDSS Human System Integration (HSI) symposium, 12 January 2024. https://doi.org/10.13140/RG.2.2.10264.57605/1.
12. Slyusar, V.I., Kondratenko, Y.P., Shevchenko, A.I. & Yeroshenko, T.V. (2024). Some Aspects of Artificial Intelligence Development Strategy for Mobile Technologies. J. of Mobile Multimedia. Vol. 20_3. Pp. 525—554. https://doi.org/10.13052/jmm1550-4646.2031.
13. Vakulenko, M. & Slyusar, V. (2024) Automatic Smart Subword Segmentation for the Reverse Ukrainian Physical Dictionary Task. Proc. of the Modern Data Science Technologies Workshop (MoDaST-2024). Lviv. Ukraine. May 31 – June 1. Pp. 59—73.
14. Слюсар В. І., Сотник В. В., Чепков І. Б. Методологія досліджень систем озброєння: теоретичні та практичні аспекти. Озброєння та військова техніка. 2024. № 3(43). С. 3—8. https://doi.org/1034169/2414-0651.2024.3(43).3-8.
15. Koboldcpp. 2024. [Online]. Available at: https://github.com/LostRuins/koboldcpp.
16. TroyDoesAI/Agent-Flow-Phone_Demo_3GB_RAM. 2024. [Online]. Available at: https://huggingface.co/TroyDoesAI/Agent-Flow-Phone_Demo_3GB_RAM.
17. YorkieDev. LM Studio Server Examples. GitHub. Jun. 2024. [Online]. Available at: https://github.com/YorkieDev/lmstudioservercodeexamples. [Accessed: Sep. 28, 2024].
18. Issa2k23. LMStudio integration. GitHub. Jun. 18. 2024. [Online]. Available: https://github.com/open-webui/open-webui/discussions/3280. [Accessed: Sep. 28, 2024].

REFERENCES

1. Slyusar, V. I. (2024). Local large language models for confidential information processing. Weapons and Military Equipment. № 4(44). Pp. 79—91. DOI: 1034169/2414-0651.2024.4(44).79—91.
2. OpenAI. GPT-4 technical report. arXiv, 2024. [Online]. Available: https://arxiv.org/abs/2303.08774.
3. R. Marino. Fast Analysis of the OpenAI O1-Preview Model in Solving Random K-SAT Problem: Does the LLM Solve the Problem Itself or Call an External SAT Solver. arXiv preprint arXiv:2409.11232, 2024. [Online]. Available: https://arxiv.org/abs/2409.11232.
4. LM Studio. 2024. [Online]. Available: https://lmstudio.ai/.
5. Ollama. 2024. [Online]. Available: https://ollama.com/.
6. AnythingLLM. The all-in-one AI application, 2024. [Online]. Available: https://anythingllm.com/.
7. P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems. Vol. 33. Pp. 9459—9474. [Online]. Available: https://arxiv.org/abs/2005.11401.
8. A. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. Singh Chaplot, D. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. Renard Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, W. E. Sayed. Mistral 7B. 2023. 9 p. https://arxiv.org/pdf/2310.06825.pdf.
9. Parthasarathy, V.B., Zafar, A., Khan, A. & Shahid, A. (2024). The Ultimate Guide to Fine-Tuning LLMs from Basics to Breakthroughs: An Exhaustive Review of Technologies, Research, Best Practices, Applied Research Challenges and Opportunities. arXiv preprint arXiv:2408.13296. [Online]. Available: https://arxiv.org/abs/2408.13296.
10. Slyusar Vadym. Large language models (LLM) in the military area. XXIII Intern. Scientific and Technical Conf. «Artificial Intelligence and Intellegent Systems» (AIIS'2023). 10 − 11 October, 2023. https://doi.org/10.13140/RG.2.2.30196.94086.
11. Slyusar Vadym. Reducing the Cognitive Burden of a Soldier with the Help of Personal AI and LLM Assistant. The LCGDSS Human System Integration (HSI) symposium, 12 January 2024. https://doi.org/10.13140/RG.2.2.10264.57605/1.
12. Slyusar, V.I., Kondratenko, Y.P., Shevchenko, A.I. & Yeroshenko, T.V. (2024). Some Aspects of Artificial Intelligence Development Strategy for Mobile Technologies. J. of Mobile Multimedia. Vol. 20_3. Pp. 525—554. https://doi.org/10.13052/jmm1550-4646.2031.
13. Vakulenko, M. & Slyusar, V. (2024) Automatic Smart Subword Segmentation for the Reverse Ukrainian Physical Dictionary Task. Proc. of the Modern Data Science Technologies Workshop (MoDaST-2024). Lviv. Ukraine. May 31 – June 1. Pp. 59—73.
14. Slyusar, V.I., Sotnyk, V.V. & Chepkov, I.B. (2024). Weapon systems research methodology: teoretical and practical aspects. Weapons and Military Equipment. № 3(43). Pp. 3—8. https://doi.org/1034169/2414-0651.2024.3(43).3-8.
15. Koboldcpp. 2024. [Online]. Available at: https://github.com/LostRuins/koboldcpp.
16. TroyDoesAI/Agent-Flow-Phone_Demo_3GB_RAM. 2024. [Online]. Available at: https://huggingface.co/TroyDoesAI/Agent-Flow-Phone_Demo_3GB_RAM.
17. YorkieDev. LM Studio Server Examples. GitHub. Jun. 2024. [Online]. Available at: https://github.com/YorkieDev/lmstudioservercodeexamples. [Accessed: Sep. 28, 2024].
18. Issa2k23. LMStudio integration. GitHub. Jun. 18. 2024. [Online]. Available at: https://github.com/open-webui/open-webui/discussions/3280. [Accessed: Sep. 28, 2024].

Slyusar V.I., Husakovskyi I.P.

PRACTICAL ASPECTS OF DEPLOYING LARGE LANGUAGE MODELS IN LOCAL NETWORKS

The article examines practical aspects of deploying large language models (LLMs) in local networks for processing confidential information. The authors analyze the capabilities of frameworks such as LM Studio, Ollama, Anything LLM and Koboldcpp within different client-server architecture scenarios. Particular attention is paid to functional limitations, technical nuances of configuration, as well as the efficiency and security of utilizing these systems to address tasks in military and scientific domains. Specifically, the article describes a combination of Anything LLM and Ollama frameworks, which ensures integration with knowledge bases through the Retrieval-Augmented Generation (RAG) mechanism, enabling secure data processing in a local environment.

The advantages of architecture based on the Koboldcpp framework, which supports model customization and integration with GPUs to accelerate computations, are detailed. Challenges associated with using LM Studio to create a server infrastructure are also explored, particularly due to its limited API functionality and the complexity of access configuration. The article emphasizes that the optimal choice for locally deploying LLMs in a client-server architecture is the combination of Anything LLM as a user interface and Koboldcpp as a server platform. This system provides flexibility, resource efficiency and high data security, which is particularly critical for applications in the field of national security.

Additionally, the article highlights the importance of testing and configuring the chosen frameworks to ensure system reliability, scalability and compatibility with various user scenarios. The potential use of such architectures for tasks such as real-time intelligence analysis, automation of operational commands and reporting processes in military contexts is also discussed. These solutions are presented as practical and cost-effective approaches to rapidly integrate artificial intelligence technologies into everyday military and scientific activities.

Keywords: *local large language models, confidential information processing, client-server architecture, LLM frameworks, local network, Retrieval-Augmented Generation, data security, server configuration.*

Відомості про авторів:

Слюсар Вадим Іванович
доктор технічних наук, професор
https://orcid.org/0000-0002-2912-3149

Гусаковський Іван Петрович
https://orcid.org/0009-0007-9421-9749

Information about the authors:

Vadym Slyusar
Doctor of Technical Sciences, Professor
https://orcid.org/0000-0002-2912-3149

Ivan Husakovskyi
https://orcid.org/0009-0007-9421-9749

и ls Ols ls w dkOн'?h dkOнts tgjнʕtsdjgґ' 14.11.2024.