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

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавр

на тему: **«Локалізація великих мовних моделей на основі KoboldCpp»**

Виконав: здобувач вищої освіти
за освітньою програмою
Інформаційні управляючі системи
спеціальності 126 Інформаційні
системи та технології
ступеня вищої освіти бакалавр
групи 126ІСТ_бд_2021
Повалій Владислав Володимирович
Керівник: Слюсар Вадим Іванович
Рецензент: Муравльов Володимир
Вячеславович

Полтава – 2025 року

ВСТУП

Актуальність теми кваліфікаційної роботи підтверджується необхідністю реалізації локальних агентів на основі великих мовних моделей (LLM). Локальні LLM особливо цінні для критичних і офлайн-сценаріїв. На даний час, спостерігається активне зростання конкуренції між LLM, які нарощують можливості та експериментують з мультимодальністю, локальним запуском і архітектурою MoE. Трендом стає прагнення до автономності моделей на пристроях користувачів і B2B-інтеграції.

Локальне застосування LLM має переваги: конфіденційність і безпека даних; автономність від зовнішніх сервісів; зменшення затримок; економія ресурсів на довгостроковій перспективі; повний контроль і адаптація; підвищення стійкості до зовнішніх ризиків; освітні та дослідницькі переваги.

Однак питання локалізації LLM потребує додаткових досліджень. Все це свідчить про актуальність теми роботи.

Метою кваліфікаційної роботи є підвищення ефективності обробки природної мови за рахунок використання локалізації open source LLM.

Завданнями кваліфікаційної роботи є:

– обґрунтування вибору інструментарію для реалізації локальних LLM;

– визначення особливостей KoboldCpp;

– формування рекомендацій щодо локалізації великих мовних моделей;

– техніко-економічне обґрунтування прийнятих рішень.

Об'єктом дослідження є процес обробки природної мови.

Предметом дослідження є програмні засоби локалізації великих мовних моделей.

Методами дослідження є в рамках визначення інструментарію для локалізації LLM і техніко-економічного обґрунтування прийнятих рішень використовувався аналітичний метод досліджень, а для локалізації LLM – моделювання.

Інформаційна база кваліфікаційної роботи сформована з Інтернет-ресурсів, що місять інформацію про NLP, LLM, нейронні мережі на основі архітектури трансформера, інструментарій для локалізації LLM.

Практична значущість роботи полягає у розробці рекомендацій щодо локалізації великих мовних моделей – можуть бути використані для подальших досліджень за даною тематикою та при проектуванні локальних сервісів NLP.

Апробація результатів відбувалася в рамках XX щорічної студентської наукової конференції «Сучасні інформаційні технології та інноваційні методики в економіці, менеджменті та бізнесі» Полтавського державного аграрного університету (16 квітня 2025 р., м. Полтава).

За результатами досліджень здійснено публікацію тез доповідей.

Структура кваліфікаційної роботи логічно пов'язана з завданнями досліджень і містить вступ, три розділи основної частини, висновки, список використаних джерел, додатки. Загальний обсяг пояснювальної записки кваліфікаційної роботи складає 53 сторінки формату А4. Вона містить 19 рисунків і 5 таблиць.

РОЗДІЛ 1
АНАЛІЗ ТЕНДЕНЦІЙ ВИКОРИСТАННЯ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ

1.1 Аналіз динаміки використання великих мовних моделей

Як відомо, широко поширення набули великі мовні моделі (Large Language Model, LLM) [1]. Проведений в рамках роботи аналіз популярності актуальних LLM за трафіком (кількістю візитів) дозволяє сформувати певний список. Для цього використовуються дані станом за квітень 2025 р.

1. ChatGPT [2] – 5,1 млрд. візитів на місяць, що є абсолютним переважанням. ChatGPT – це чат-бот на основі LLM, розроблений компанією OpenAI. Його перша версія, заснована на моделі GPT-3.5, була випущена 30 листопада 2022 р. Незважаючи на скромні очікування, ChatGPT швидко став вірусним, залучивши 1 мільйон користувачів за 5 днів і 100 мільйонів – за два місяці. Це перетворило OpenAI з нішевої дослідницької організації на глобального технологічного лідера. У лютому 2025 р. випущена версія GPT-4.5.

2. DeepSeek [3] – 480 млн. візитів на місяць. Хороша відповідь з новою архітектурою та відкритістю коду. Але, порівняно з минулими місяцями, трафік плавно йде на спад. Можливе швидке зростання за рахунок інтеграції в споживчу робототехніку, в яку активно кинулися більшість автовиробників Китаю. Остання версія – DeepSeek-V3-0324 доступна для використання через вебінтерфейс, мобільні додатки (iOS та Android) та API. З нею можна ознайомитися з моделлю на офіційному сайті DeepSeek або завантажити ваги моделі з платформи Hugging Face [4].

3. Gemini [5] від к. Google – 409 млн. візитів на місяць. Вперше представлена 6 грудня 2023 року, Gemini стала наступником моделей LaMDA та PaLM 2. Незважаючи на довгий контекст моделі, високу мультимодальність і ресурси компанії – поки що в 10 разів поступається

лідеру ринку з трафіку. Можливо NotebookLM та AgentSpace щось змінять. Найновішою версією є Gemini 2.5 Pro (експериментальна), випущена 25 березня 2025 р. Ця модель відзначається покращеними можливостями логічного мислення та кодування, а також здатністю до «моделювання мислення», що дозволяє їй розмірковувати перед наданням відповіді. Вона підтримує мультимодальність та має контекстне вікно до 1 мільйона токенів.

4. Grok [6] – 196 млн. візитів на місяць. Це серія LLM, розроблених к. xAI у 2023 р. Метою проєкту є створення конкурентоспроможної альтернативи OpenAI та ChatGPT з акцентом на відкритість, гумор і прямий доступ до платформи X. Ставка на інтеграцію із соцмережею X принесла швидке зростання, але глибина використання просідає. Grok-3 (лютий 2025 р.) – найновіша версія, навчена з використанням суперкомп'ютера Colossus, що містить близько 200000 графічних процесорів (GPU). Модель демонструє покращені можливості логічного мислення та підтримує режими «Think» і «Big Brain» для вирішення складних завдань.

5. Perplexity від к. Perplexity AI [7] – 113 млн. візитів на місяць. Стратегія на нішевий сервіс з урахуванням нової парадигми пошуку дала результат, але трафік мінус 10 млн. до минулого місяця. Запуск першої версії публічного продукту (Perplexity Ask) відбувся у січні 2023 р. Perplexity не розкриває точних назв чи номерів своїх моделей. Однак, остання версія LLM Perplexity з мультимодальною підтримкою (на травень 2025) розроблена у партнерстві з Mistral, Meta (LLaMA), OpenAI та Anthropic. Їхня власна модель використовується в режимі «Instant», який забезпечує надшвидкі відповіді без зовнішніх API.

6. Claude [8] від к. Anthropic – 95,6 млн. візитів на місяць. У березні 2023 р. Випущено Claude 1 – першу публічну версію. Стабільно, але без вибухів. З погляду експертів застосована найкраща мовна математика, але продукту трохи не вистачає яскравості. Той самий випадок, коли продукт є проекцією свого CEO. Актуальна версія (станом на травень 2025 р.). Claude 3 Opus – найпотужніша з поточно доступних, випущена в березні 2024 р. Вона

включає також Claude 3 Sonnet (збалансована) та Claude 3 Haiku (швидка та легка). В цілому, рівень міркування, наближений до GPT-4; підтримка зображень (multimodal); великий контекст (~200k токенів); пріоритет – безпека, контроль і рівень пояснення.

7. Copilot [9] від к. Microsoft – 86 млн. візитів на місяць. Не є чистою LLM, тобто, гібрид з GPT від к. OpenAI та власних моделей. Станом на травень 2025 р., Microsoft Copilot використовує комбінацію GPT-4o, Phi-4, OpenAI o1 (використовується у функції «Think Deeper» для глибшого аналізу запитів). Приріст: + 19 млн. відвідувачів за 2 місяці.

8. Qwen (також відома як Tongyi Qianwen) від к. Alibaba [10] – 46 млн. візитів на місяць. Публічний реліз першої версії Qwen був у вересні 2023 р. Як і DeepSeek, побудована на архітектурі Mixture-of-experts. Поки що залишається суто локальним гравцем. Qwen3 – це флагманська серія моделей, випущена 28 квітня 2025 року, яка включає як щільні (dense), так і розріджені (Mixture-of-Experts, MoE) архітектури, та має такі режими роботи: Thinking mode – для складного багатокрокового міркування; Non-thinking mode – для швидких відповідей.

MoE – це архітектурний підхід у ШІ, зокрема в глибокому навчанні, який дозволяє моделі динамічно вибирати частини мережі для обробки кожного запиту, замість того щоб активувати всю модель повністю. Модель складається з великої кількості "експертів" (окремих нейронних блоків або підмереж), але при кожному проході даних активується лише підмножина експертів, зазвичай, 1-4 із десятків або сотень.

9. Mistral [11] від французів – 8,1 млн. візитів на місяць. Європейська альтернатива від вихідців з DeepMind та Meta за місяць втратила 3 млн. відвідувачів, майже 30 %. У листопаді 2023 р. був випуск Mistral 7B – першої моделі з відкритим кодом, побудованої без використання LoRA чи MoE. Остання версія моделі Mistral Large (станом на травень 2025 р.) – Це комерційна модель, доступна через API (la plateforme Mistral), порівнянна за потужністю з GPT-4. Вона Працює повністю офлайн або через Hugging Face.

10. LLama [12] від Meta – 1,7 млн. візитів на місяць. Відкрита модель з гарним міркуванням поки не може розігнатися, незважаючи на застосування в багатьох сторонніх додатках. LLaMA 1 задекларована у лютому 2023 р, а остання (LLaMA 3) – 18 квітня 2024 р. та має формати: PyTorch, GGUF, сумісність із платформами як Hugging Face, Ollama, KoboldCpp.

11. Command R+ [13] від к. Cohere. Це гравець без масового публічного трафіку, зате з чіткою спеціалізацією. Модель Command R+ була вперше представлена у квітні 2024 р. як частина серії моделей Command від к. Cohere. Вона була розроблена для оптимізації Retrieval-Augmented Generation (RAG) і багатокрокового використання інструментів, що робить її особливо корисною для складних бізнес-завдань. У серпні 2024 р. було випущено оновлену версію Command R+ 08-2024, яка забезпечує приблизно на 50 % вищу пропускну здатність і на 25 % меншу затримку порівняно з попередньою версією, при цьому зберігаючи ті ж апаратні вимоги. Широту застосування оцінити важко, через те що платформа спрямована під B2B-інтеграцію у корпоративні системи та орієнтована на RAG. Трафік на сайті Cohere – 0,83 млн.

12. Цікавий ще ось який тренд. Стрімко зростає інтерес установки LLM на локальних пристроях. Ollama [14] – платформа, яка дозволяє за лічені хвилини підняти та розгорнути у себе на машині моделі типу LLaMA, DeepSeek, Qwen, Mistral та Gemma 3 збирає – 5,4 млн. візитів на місяць. Це незалежна платформа з відкритим кодом. Завдяки локальному запуску моделей, Ollama забезпечує підвищену конфіденційність, контроль над даними та зменшення залежності від хмарних рішень. У травні 2025 р. було випущено версію Ollama v0.7, яка додала підтримку провідних візуальних моделей, що дозволяє запускати мультимодальні моделі локально. Ollama активно розвивається та підтримується спільнотою розробників, пропонуючи гнучкий інструмент для роботи з LLM у різних сферах, від досліджень до бізнес-застосувань. Динаміка візитів трохи знижується, але не через падіння інтересу. Швидше навпаки: великі вендори почали покращувати UX власних

інсталяторів, і все більше користувачів, минаючи Ollama, встановлюють моделі безпосередньо з сайтів розробників.

1.2 Інтерфейси взаємодії в агентах на основі великих мовних моделей

Зараз, LLM навчилися як генерувати текст, так і виконувати реальні завдання, використовуючи команди природною мовою. Це відкрило нову еру в автоматизації, породивши LLM-агентів. Дослідження [15] розбирає ключові підходи до створення таких агентів. Тому доцільно дослідити, у чому їхня суть, відмінності та перспективи. Сьогодні існують два основні типи LLM-агентів (рис. 1.1).

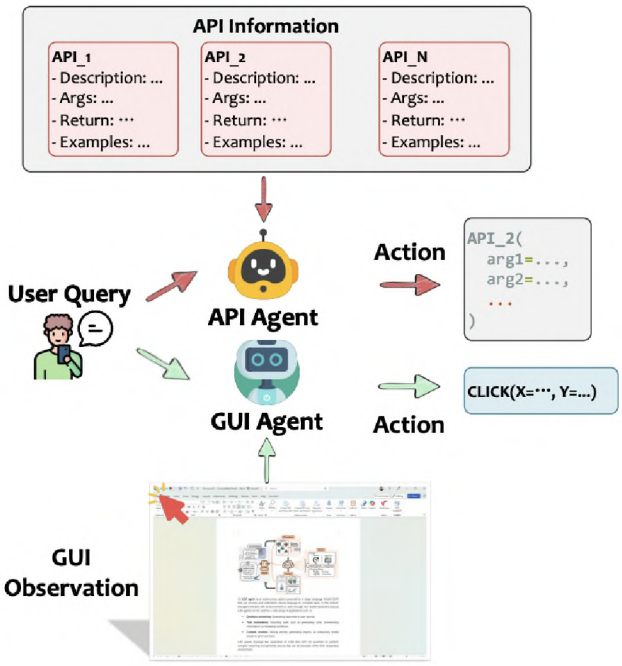


Рисунок 1.1 – Різниця між API-агентом та GUI-агентом щодо вводу/виводу

API-агенти взаємодіють із програмами через спеціально створені програмні інтерфейси (API). Цей підхід популярний завдяки своїй надійності, ефективності та відносній простоті інтеграції в існуючі системи.

GUI-агенти працюють інакше. Вони взаємодіють з програмами через їх графічний інтерфейс (GUI), імітуючи дії людини: кліки мишкою, введення тексту в поля і так далі. Їхня поява стала можливою завдяки розвитку мультимодальних LLM, які можуть «бачити» екран і розуміти, що на ньому зображено (рис. 1.2).

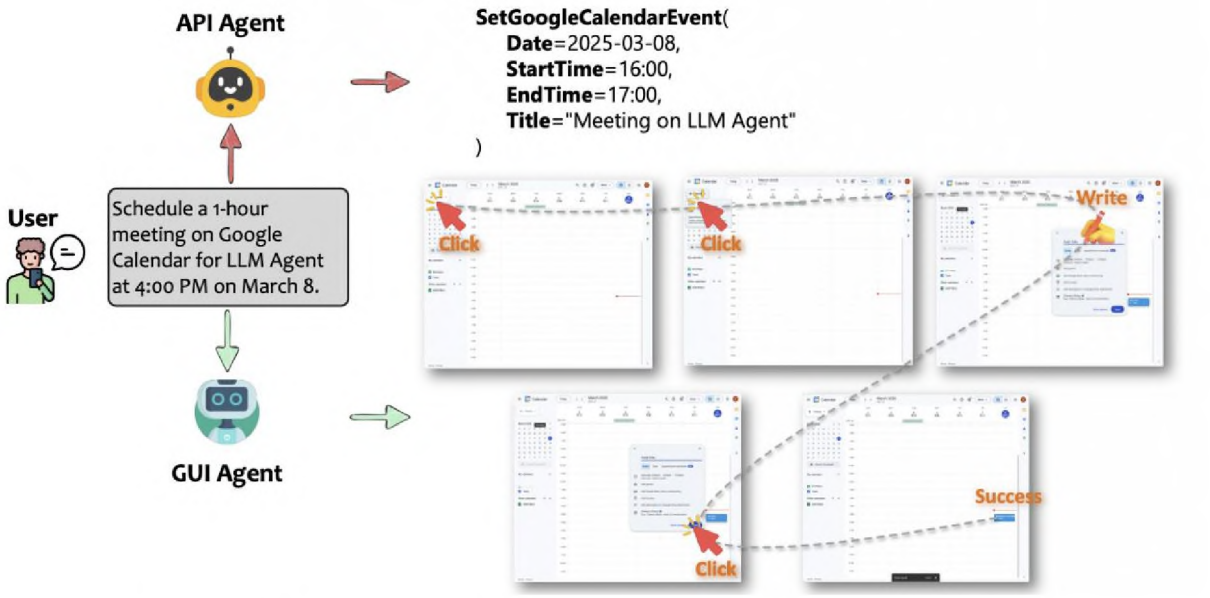


Рисунок 1.2 – Різниця між API-агентом та GUI-агентом при виконанні певного завдання

Хоча мета в обох типів агентів одна – автоматизувати завдання – вони кардинально різняться за своєю архітектурою, принципами роботи, надійністю та сферами застосування. Досі не було чіткого розуміння їхніх порівняльних плюсів та мінусів, а головне – як їх можна використати разом. Згідно [15], проведено комплексне порівняння API-агентів та GUI-агентів. Автори порівняли два підходи за 9 важливими параметрами: модальність введення/виводу, надійність, ефективність, доступність (наскільки легко застосувати до різних програм), гнучкість, безпека, підтримка ПЗ, прозорість роботи та людиноподібність взаємодії. Для кожного аспекту вони навели аргументи, що наголошують на відмінностях (табл. 1.1). Дослідники вивчили існуючі та потенційні способи комбінування API- та GUI-взаємодій:

– створення API-обгорток над GUI-інтерфейсами (коли API немає, але можна імітувати його через GUI) – рис. 1.3;

– використання інструментів оркестрації, які можуть вирішувати, коли використовувати API (якщо доступний і ефективний), а коли – GUI (якщо API немає або він не підходить) – рис. 1.4;

– застосування Low-code/No-code платформ, які приховують від користувача деталі реалізації агенту (API або GUI) – рис. 1.5.

Таблиця 1.1 – Відмінності агентів

Параметр	Агенти API	Агенти GUI
Модальність	Спираються на текстові дзвінки API	Залежить від знімків екрана чи структур доступності
Надійність	Як правило, вище за наявності чітко визначених кінцевих точок	Нижче через візуальний аналіз та зміни в макеті
Ефективність	Виконують складні завдання за один виклик	Вимагають кількох дій, схожих на дії користувача
Доступність	Обмежені опублікованими або визначеними API	Можуть працювати з будь-яким видимим елементом інтерфейсу
Гнучкість	Обмежені існуючими API	Високо адаптовані до нових або нерозкритих функцій
Безпека	Управляються рахунок детального контролю кінцевих точок	Більш ризиковані через широкий доступ до елементів інтерфейсу
Підтримка ПЗ	Стабільні, якщо API залишаються залежні від номеру версії	Схильні до збоїв при редизайні інтерфейсу
Прозорість	Часто приховані, керовані серверною частиною	Покроково, візуально відстежуються
Людино-подібна взаємодія	Чисто програмні	Імітують дії користувача на екрані

Як наслідок, був виконаний детальний аналіз сценаріїв застосування API-, GUI-агентів та гібридного підходу (табл. 1.2).

Для вибору оптимального підходу (API, GUI чи гібрид) залежно від завдання та вимог можна використовувати на ступні положення. API-агенти мають такі переваги: висока ефективність (одна команда API замінює багато кліків), надійність (API змінюються рідше та більш передбачувано, ніж GUI), безпека (чітко визначені права доступу через API), підтримка ПЗ (версій API).

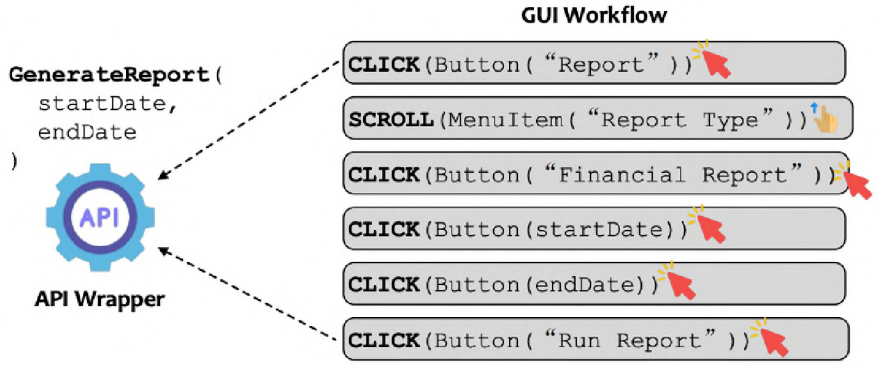


Рисунок 1.3 – Приклад API-обгортки над робочим процесом GUI

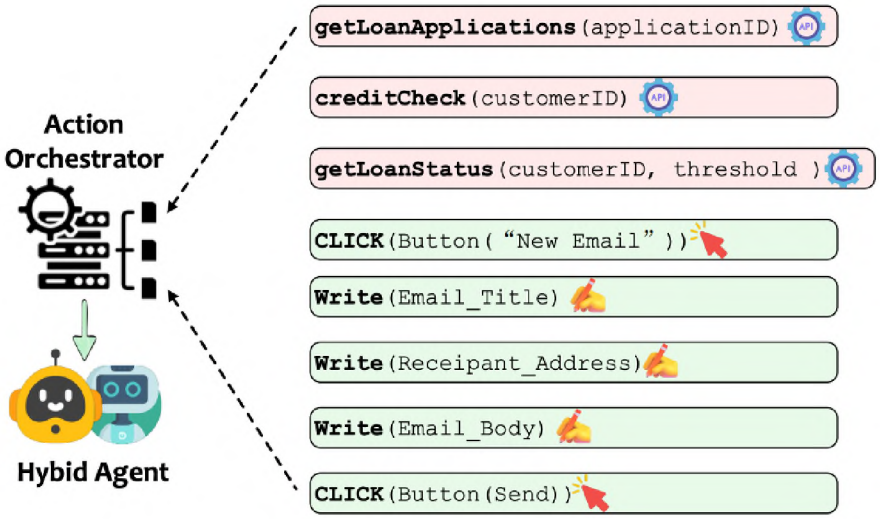


Рисунок 1.4 – Приклад єдиного оркестратора для керування діями API і GUI

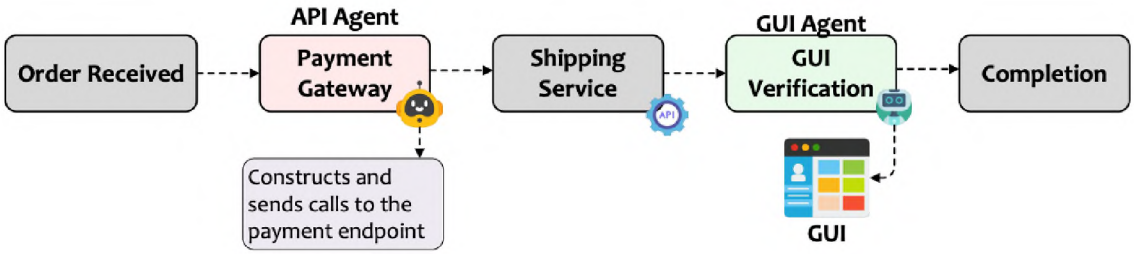


Рисунок 1.5 – Приклад платформи без коду для створення робочих процесів, що інтегрують як виклики API, так і GUI-агентів

До недоліків варто віднести: доступність та гнучкість обмежені наявністю та можливостями існуючих API; робота агенту часто є непрозорою для кінцевого користувача.

GUI-агенти мають такі переваги: висока доступність (можуть працювати майже з будь-яким ПЗ, у якого є GUI), гнучкість (можуть взаємодіяти з будь-якими елементами на екрані), прозорість (користувач бачить, що робить агент), імітація людиноподібної взаємодії.

Таблиця 1.2 – Аналіз сценаріїв застосування API-, GUI-агентів

Сценарій	Рекомендо-ваний підхід	Обґрунтування
Стабільні, добре документовані API	API-агенти	Використовують надійні кінцеві точки для швидкодії та надійності.
Критичні за продуктивністю операції	API-агенти	Знижують затримки та накладні витрати завдяки прямим викликам функцій
Контрольований доступ до додатків	API-агенти	Забезпечують безпеку та захист
Застаріле або пропрієтарне ПЗ	GUI-агенти	Автоматизують завдання без необхідності інтеграції з бекендом
Візуальна перевірка або тестування UI	GUI-агенти	Перевіряють текст або елементи безпосередньо на екрані
Інтерактивне чи графічне управління	GUI-агенти	Імітують дії користувача для роботи з візуальними елементами
Часткове покриття API	Гібридний підхід	Комбінують UI-операції, де API відсутні, з прямими викликами для ресурсомістких завдань
Адаптація до майбутніх змін	Гібридний підхід	Забезпечують перехід від GUI до API у міру розвитку кінцевих точок.

При цьому недоліками вважаються: низька ефективність (багато кроків для простих дій), низька надійність (дуже чутливі до найменших змін в інтерфейсі – редизайн ламає агента), проблеми з безпекою (агент отримує широкий доступ до інтерфейсу), складність підтримки.

В цілому, межа між підходами не така вже й жорстка. Наприклад, оркестратор може використовувати API для швидких і надійних операцій, а якщо потрібного API немає – перейти на GUI-агента для виконання завдання через інтерфейс. API-агенти – найкращий вибір, якщо є стабільні API, важлива висока продуктивність та безпека. GUI-агенти – підходять для роботи із застарілими системами без API, коли потрібна візуальна перевірка дій або точна імітація роботи користувача.

В якості прикладу GUI для роботи з локальними LLM можна вказати платформу LM Studio (рис. 1.6) [16].

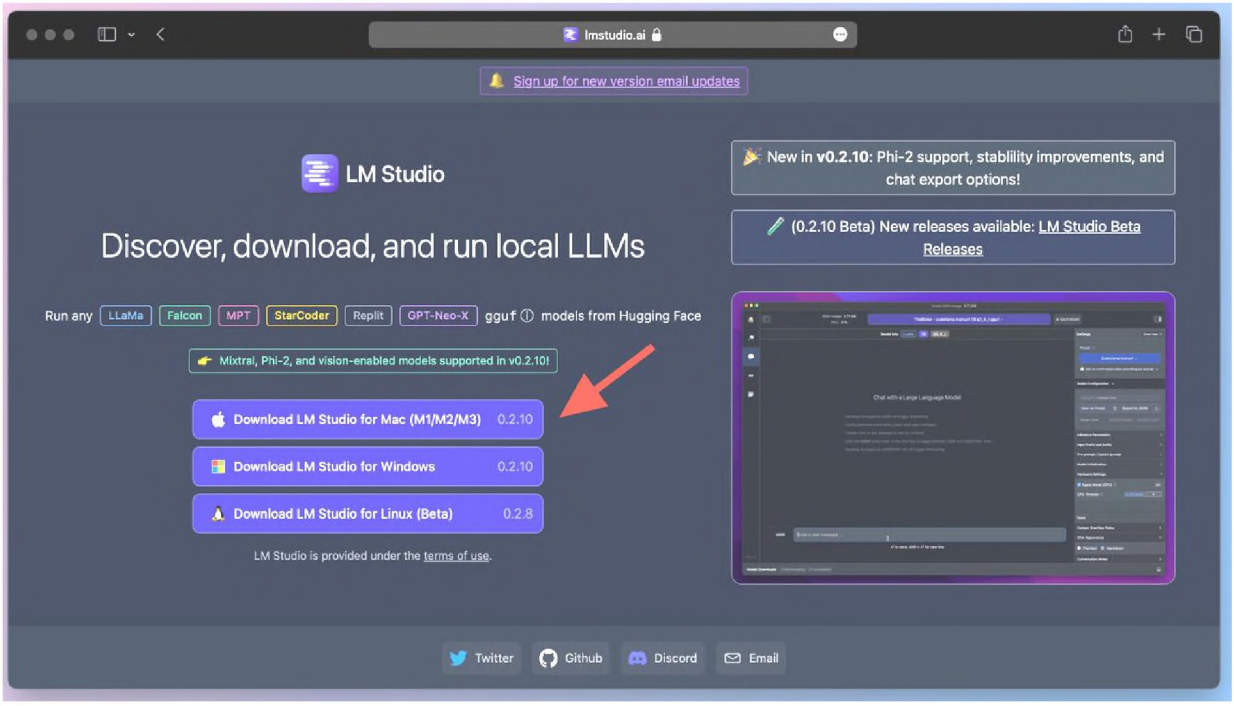


Рисунок 1.6 – LM Studio

Гібридний підхід – рекомендується, якщо API покривають лише частину потрібної функціональності або якщо потрібна гнучкість на майбутнє (наприклад, при роботі з ПЗ, що постійно змінюється).

1.3 Обґрунтування вибору варіанту застосування LLM

API- та GUI-агенти – це не так конкуренти, як взаємодоповнюючі інструменти. Розуміння їх сильних і слабких сторін є дуже важливим для вибору правильного рішення. Гібридні підходи – найперспективніший напрямок. Вони дозволяють створювати більш універсальні та адаптивні системи автоматизації, здатні працювати у складних та різноманітних програмних середовищах. А розвиток мультимодальних LLM і надалі прискорюватиме прогрес GUI-агентів, роблячи їх надійнішими та

розумнішими. Поява уніфікованих платформ і low-code/no-code рішень спростить створення складних робочих процесів з використанням обох типів агентів, роблячи просунуту автоматизацію доступнішою.

Але надійність GUI-агентів залишається серйозними проблемами. Зміни в інтерфейсі програм легко ламають їх, вимагаючи постійного доопрацювання. Також і безпека GUI-агентів потребує особливої уваги – вони потенційно можуть виконати небажані дії через інтерфейс користувача.

Локальне застосування LLM дозволяє отримати низку переваг, особливо в контексті автономності, безпеки та ефективності. При цьому всі вхідні дані обробляються безпосередньо на пристрої користувача або в закритій мережі. Це критично важливо для галузей, де діють суворі вимоги до збереження приватності:

– уникнення витоку персональних або комерційно важливих даних;

– відповідність вимогам GDPR та інших норм захисту даних.

Локальна модель працює незалежно від Інтернет чи хмарних сервісів. Відповідно, робота в офлайн-режимі важлива для завдань відомчого призначення, віддалених об'єктів, безпілотників та ін. Також присутнє зменшення залежності від сторонніх компаній та API-з'єднань.

Обробка запитів локально значно скорочує час відповіді. Реакція відбувається практично миттєво, без потреби в мережевих викликах. Це важливо для застосунків реального часу (керування роботами, доповнена реальність, аналіз відео). Попри первинні витрати на запуск локальної LLM, немає абонплати за використання API сторонніх сервісів (наприклад, OpenAI, Anthropic). Також є контроль над обчислювальними витратами та масштабуванням. Локальні LLM можна оптимізувати під конкретне обладнання (наприклад, Raspberry Pi 5 або GPU сервери), доучувати на внутрішніх даних, створюючи кастомізовану модель, а також обирати тип моделі: від швидкої (наприклад, GGUF/int4) до точної (fp16/bfloat16).

Локальна LLM не залежить від політичних чи комерційних обмежень (API може бути недоступне через санкції, геоблокування, зміни політик). Має

вищу надійність у критичних сценаріях (наприклад, в умовах надзвичайного стану чи кіберзагроз).

Розгортання локальних моделей дозволяє інженерам і дослідникам експериментувати з архітектурою, токенізацією, квантованими варіантами без обмежень хмари. Також сприяє розвитку відкритих рішень у сфері ШІ.

Таким чином, в якості пріоритетного рішення розглядається варіант локального використання LLM з можливістю підключення по API.

РОЗДІЛ 2
ЛОКАЛЬНЕ СЕРЕДОВИЩЕ ДЛЯ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ НА ОСНОВІ KOBOLDCPP

2.1 Інструментарій для локального запуску великих мовних моделей

У зв'язку з активним розвитком LLM зросла потреба у їхньому локальному розгортанні на ПК, одноплатних мікрокомп'ютерах та ін. Це дозволяє забезпечити автономність, підвищену безпеку даних, зниження затримок та зменшення витрат на хмарні обчислення. Для реалізації таких можливостей існує низка фреймворків та рушіїв, які оптимізовані для роботи LLM у локальному середовищі.

По-перше розглянемо llama.cpp. Це високоефективна реалізація моделей GPT-подібної архітектури (особливо серій LLaMA від Meta) з використанням C++ і апаратного прискорення через SIMD-інструкції. llama.cpp підтримує кросплатформну роботу (Linux, Windows та macOS) і адаптований для запуску на CPU без необхідності у GPU. На даний час, реалізовано інтерфейси quantization – int4/int8), що дозволяє запускати LLM навіть на Raspberry Pi.

KoboldCpp – оптимізований для GGUF-моделей, працює на C++ з інтеграцією GUI. Він створений на базі llama.cpp, оптимізований для інтерактивного текстового покоління. Також має інтеграцію з інструментами roleplay/novel-writing, підтримує chat-інтерфейси, API-сумісність з OpenAI, а також багатопотоковість. На даний час, KoboldCpp має кілька варіантів: KoboldAI, KoboldCpp і Kobold Lite, які розглядаються вже як окремі проєкти. Назва KoboldCpp відсилає до проєкту KoboldAI – вебінтерфейсу для генерації історій. KoboldCpp є бінарним рушієм, який сумісний із цим інтерфейсом, замінюючи попередні рушії на базі Transformers або GPT-j на значно ефективніший варіант.

GPT4all – це open-source екосистема для локального запуску, розроблена компанією Nomic AI. Проєкт орієнтований на забезпечення доступного,

приватного та ефективного використання LLM на локальних машинах, без необхідності підключення до хмари або великих обчислювальних кластерів. Основними компонентами GPT4all є GPT4all Backend (gpt4all.cpp), GPT4all Chat UI, модельний хаб GPT4all. Gpt4all.cpp – це рушій на C++ (форк llama.cpp), який дозволяє запускати LLM з підтримкою квантування. Він оптимізований для запуску моделей без GPU – на звичайних CPU, включаючи середовища Windows, Linux та macOS. GPT4All Chat UI – графічний інтерфейс, який дозволяє користувачеві легко взаємодіяти з моделлю. Програму можна встановити як десктоп-додаток (Electron-based), що працює офлайн. Має зручний інтерфейс для ведення розмов, історії чатів та перемикання між моделями. Модельний хаб GPT4all [17] містить галерею підтримуваних моделей, серед яких:

– GPT-J (класичний, базовий варіант);
– MPT (MosaicML) – з відкритим кодом;
– LLaMA, LLaMA2 – з квантуванням (Q4, Q5);
– Mistral – новіші моделі з більш ефективною архітектурою;
– Replit-code-v1 – для генерації коду.

В свою чергу, є також GPT4all Python API. Це простий SDK для інтеграції GPT4All-моделей у свої додатки.

Ollama – один з найзручніших фреймворків для локального використання LLM. Завдяки інтегрованому API, автоматизованому менеджменту моделей, підтримці кастомізації та сучасним моделям він підходить як для ентузіастів, так і для професійних розробників, які хочуть використовувати LLM у своїх продуктах без залежності від хмари.

Ollama – це високорівневий фреймворк та CLI-рушій, створений для максимально простого, стандартизованого та ефективного запуску великих мовних моделей (LLM) локально на звичайному комп'ютері. Він орієнтований на розробників, дослідників та інтеграторів, яким потрібна швидка інтеграція локального ШІ в програми або агентні системи. Ollama використовує унікальний підхід: LLM-модель – це своєрідний контейнер (як у Docker), який

можна завантажити, інсталювати й запускати однією командою. Це усуває складну інсталяцію моделей вручну (немає потреби шукати GGUF, завантажувати tokenizer тощо). Усі залежності інтегровані в один образ. Ollama надає простий CLI (термінал), з якого можна запускати модель у чат-режимі, а також повноцінний REST API для підключення до програм. Це дає змогу інтегрувати Ollama у ваші застосунки, боти, скрипти, LangChain, LlamaIndex, AutoGen та інші агенти. Ollama дозволяє створювати свої власні кастомні моделі на основі існуючих через Modelfile — подібно до Dockerfile. В цілому, перевагами слід вважати REST API для production, підтримка LangChain, LLM-agent tools.

LM Studio – це кросплатформний десктопний застосунок для Windows, macOS та Linux, який дозволяє запускати великі мовні моделі локально на комп'ютері без потреби у хмарних API або серверному розгортанні. Його основна мета – надати простий та інтуїтивно зрозумілий графічний інтерфейс (GUI) для взаємодії з LLM, з можливістю використовувати найсучасніші моделі прямо з HuggingFace. Базується на llama.cpp і підтримує GGUF-моделі. При цьому, є підтримка кількох моделей одночасно; можливість завантажити GGUF-файл вручну; автоматичне оновлення моделей; кешування відповідей, що прискорює повторне використання.

Text Generation WebUI – універсальний рушій, що підтримує запуск різних моделей (LLaMA, Mistral, Falcon, GPT-J) з допомогою Transformers або llama.cpp. Має веб-інтерфейс, графічні налаштування параметрів температури, довжини відповіді, prompt engineering.

GGML (Generative Graphical Machine Learning) – це бібліотека, що лежить в основі llama.cpp і сумісних рушіїв. Новий формат GGUF (Unified Format) спрощує обмін моделями між фреймворками та зберігає метадані, токенізатори, параметри квантування тощо.

MLC LLM – фреймворк від Apache TVM, що забезпечує компіляцію LLM у власні виконувані файли, оптимізовані під конкретну архітектуру (наприклад, CUDA, Metal, Vulkan).

MLC LLM дозволяє створювати повністю автономні застосунки без зовнішніх залежностей, зокрема на Android.

Локальні фреймворки підтримують різні рівні квантування (Q2_K, Q4_0, Q5_K, тощо), що значно знижує об'єм пам'яті моделей (до 4-6 ГБ замість 30-60 ГБ), дозволяючи використовувати їх на пристроях з обмеженими ресурсами. Також забезпечується інтеграція з зовнішніми бібліотеками на Python (LangChain, Transformers), що відкриває широкі можливості для розробки агентних систем, чат-ботів, локальних.

2.2 Порівняльна оцінка KoboldCpp та GPT4all

KoboldCpp і GPT4ALL охоплюють різну аудиторію, але мають однакові цілі – зробити роботу з мовними моделями зручніше для звичайних користувачів, додати більше контролю над генерацією та забезпечити стабільну роботу текстових нейромереж. KoboldCpp – програма з вебінтерфейсом (Web UI), яка запускається у новій вкладці браузера. GPT4all – окрема програма з графічним інтерфейсом (GUI), яка не потребує браузера для роботи. Крім принципу запуску існує різниця в позиціонуванні ПЗ. Спільнота KoboldAI з'явилася задовго до поширення мовних моделей із сімейства LLaMa, став притулком авторів, які займаються створенням новел, пригодницьких історій, фанфіків та рольових ігор. Тоді KoboldAI був альтернативою не ChatGPT, а Novel AI – платного генератора текстових історій. До моменту виникнення загальнодоступних LLM у KoboldAI сформувалася ціла плеяда власних моделей, заточених на створення цікавих оповідань. На момент появи загальнодоступних LLM у KoboldAI сформувалася ціла плеяда своїх моделей [18], заточених створення цікавих оповідань. Серед них зустрічалися як ресурсомісткі нейромережі, які не запускалися без топових відеокарт, так і легковагі текстові генератори, які міг потягнути навіть старий ноутбук (рис. 2.1). Переписувати існуючий

дистрибутив з нуля - довго і накладно, тому з'явилася окрема версія програмного забезпечення, що поєднує в собі найкращі можливості llama.cpp та розширену функціональність API Kobold – KoboldCpp (рис. 2.2).

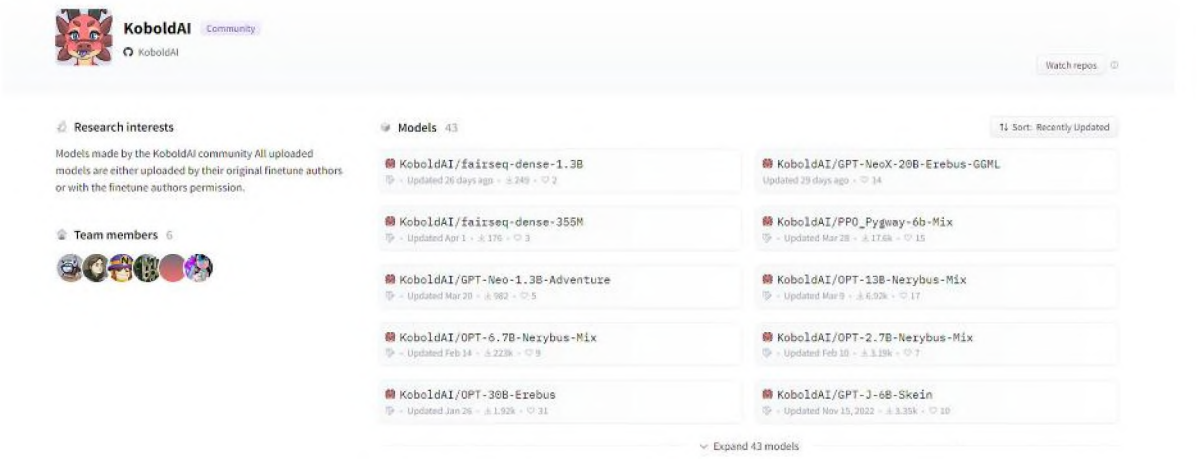


Рисунок 2.1 – Моделі Kobold AI

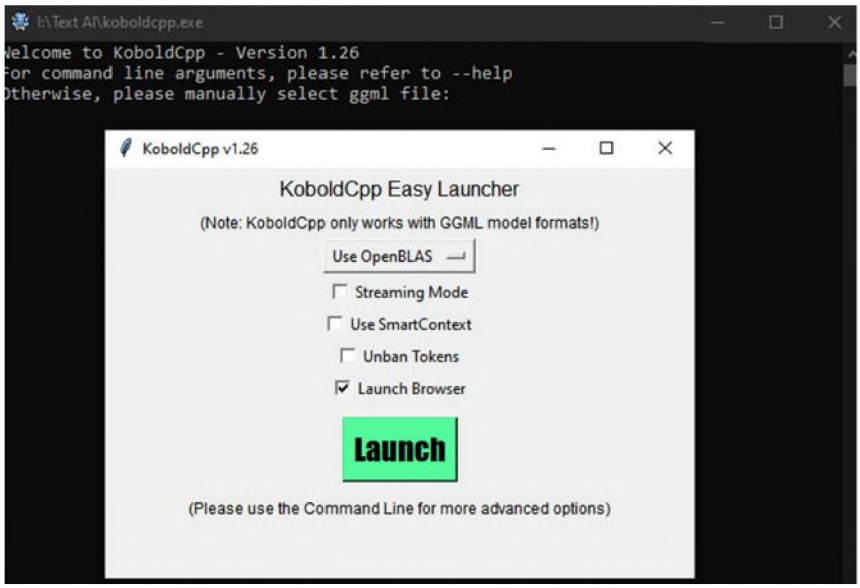


Рисунок 2.2 – Інтерфейс для запуску KoboldCpp

Програма відрізняється крихітним розміром (приблизно 10 МБ, не рахуючи ваги нейромереж). Можна сказати, що KoboldCpp запускає будь-які існуючі формати GGML, якщо вони мають попит. Всі інші файли не працюватимуть у ПЗ. Наприклад, така доля спіткала bloomz.cpp. Головні плюсами KoboldCpp слід вважати такі положення: підтримка більшості

мовних моделей, у тому числі тих, що використовують новий метод переквантизації; високошвидкісна генерація тексту; може зберігати та завантажувати історії у форматі .JSON; варіанти сценаріїв, які підходять на будь-який випадок життя; великий перелік пресетів для швидкого старту; розширене меню налаштувань; легка генерація історій, діалогів, гайдів та інструкцій; можна вибрати іншу версію, якщо нова вам чомусь не підійшла. Інтерфейс при налаштуванні відповідає рис. 2.3.

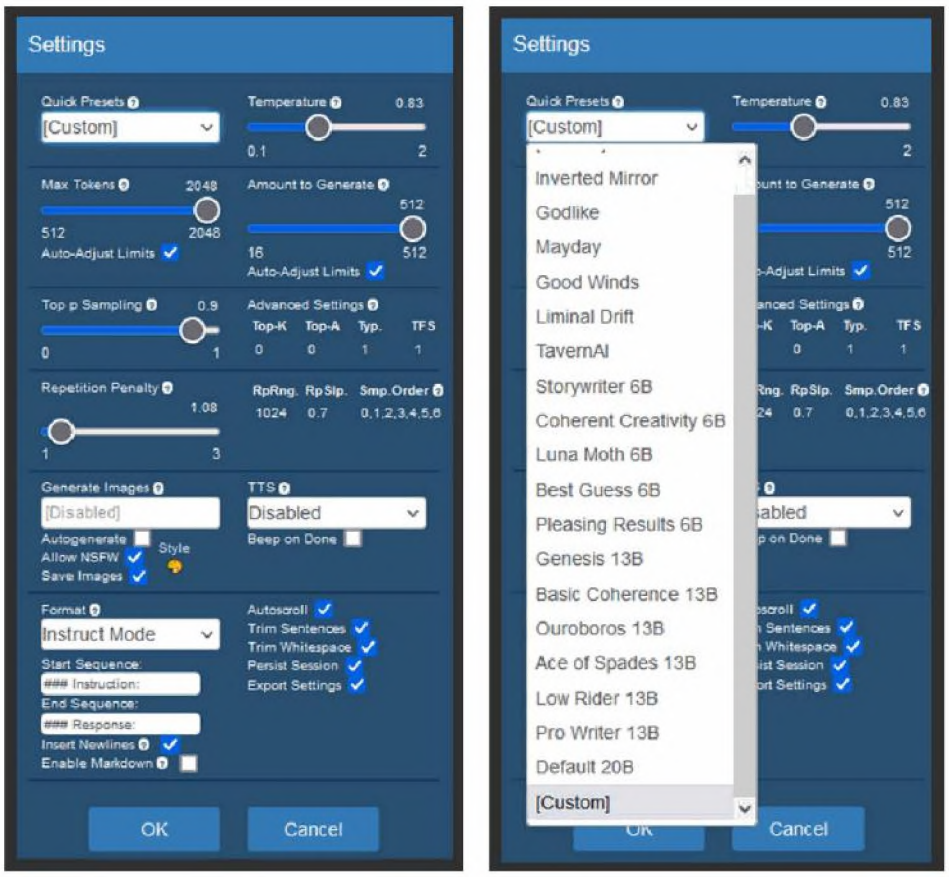


Рисунок 2.3 – Інтерфейс при налаштуванні KoboldCpp

Разом з тим, на даний час, відсутня підтримка документів (не можна додати в якості джерелі даних файли PDF або TXT), не підтримує тривале запам'ятовування контексту (максимальна кількість токенів – 2048); може довго запускатись; працює в окремій вкладці браузера; немає автоматичного оновлення; більше підходить для створення оповідань, ніж для довгих

діалогів; щоб запустити нову LLM, потрібно окремо її відкривати за допомогою провідника Windows.

GPT4all найбільше нагадує безкоштовний варіант ChatGPT. Головне вікно програми – сторінка з чатом (рис. 2.4). Вгорі – панель вибору доступних моделей, у бічній вкладці зліва – можливість створення нового чату або збереження наявного, панель для завантаження моделей та перевірки оновлень. Вгорі праворуч знаходяться базові налаштування, а також панель для керування ними.

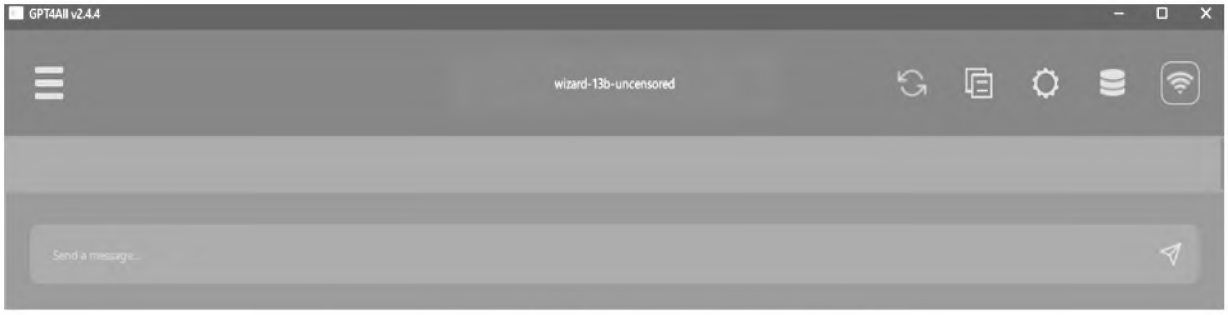


Рисунок 2.4 – Головне вікно GPT4all

Тобто здійснюється оптимальна реалізація для локального чат-бота або програми для виконання інструкцій користувача. Про це говорять і розробники програми, які прагнуть надати найкращий софт для звичайних людей, які бажають отримати просунутого нейромережевого асистента та використовувати його для творчості, роботи чи експериментів. На основі досліджень можна вказати такі переваги GPT4all:

– зручний інтерфейс (рис. 2.5);

– автономна робота, яка не потребує запуску браузера;

– підтримує багатопотокові процесори, а також дозволяє вручну призначати кількість потоків, що використовуються для прискорення генерації тексту;

– може зчитувати дані з документів на комп'ютері за допомогою плагіну LocalDocs;

– максимальна довжина відповіді не обмежена 2048 токена. Додаток дозволяє призначити більший ліміт – 4096 або 131072 токенів;

– підтримує збереження чатів на диску;

– крута робота з чатом;

– можна вручну настроїти зразок запиту для кращої взаємодії з LLM (Prompt Template);

– швидкий запуск скачаних LLM;

– зручне перемикання між доступними нейромережами;

– стабільні оновлення;

– програма сповіщає про вихід патчів, після підтвердження автоматично завантажує та встановлює їх, показує, що сталося в оновленнях;

– розробники постійно додають нові функції.

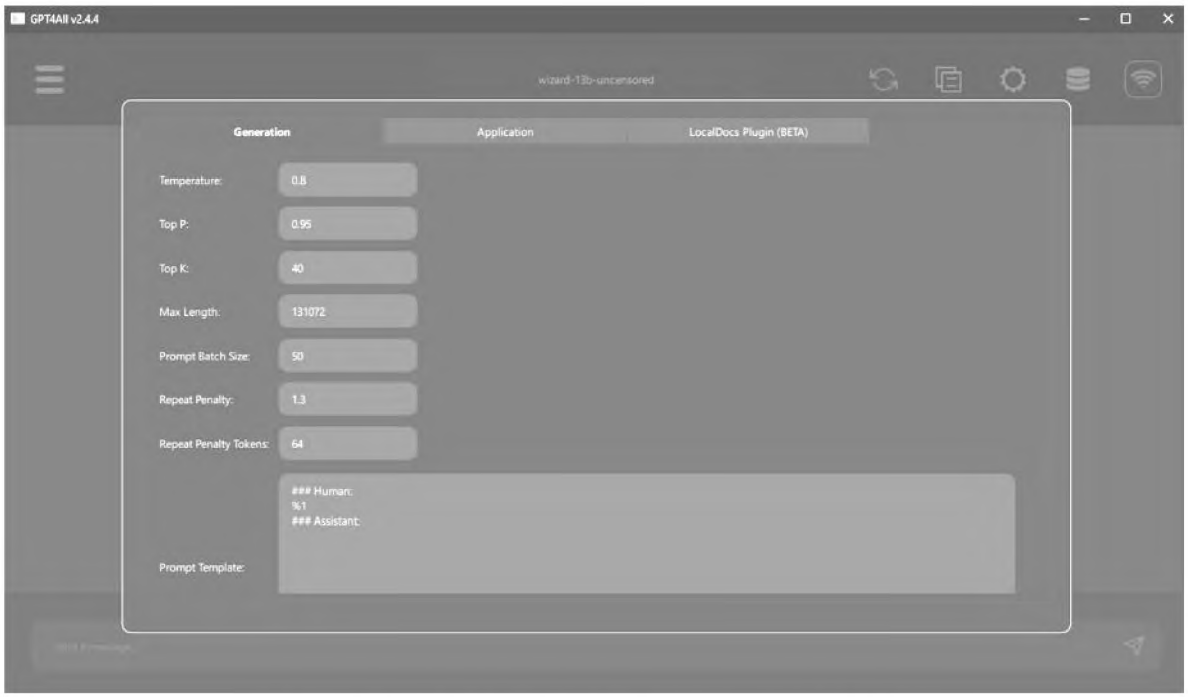


Рисунок 2.5 – Вікно налаштувань GPT4all

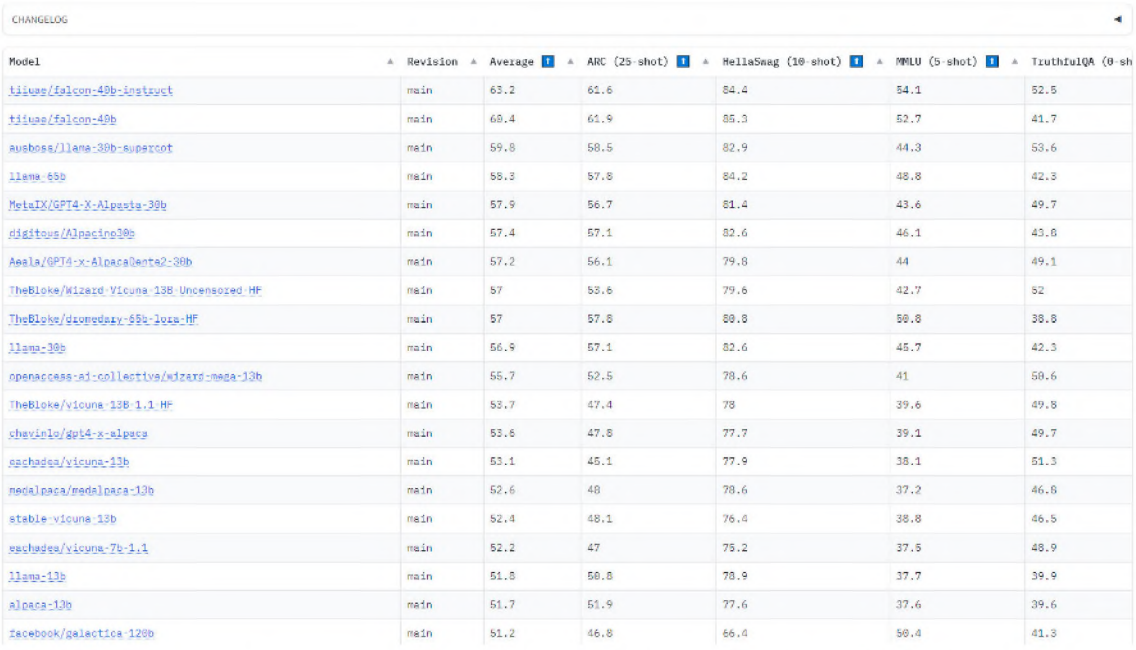
До недоліків GPT4all слід віднести такі положення. Можливість збільшити максимальну довжину відповіді не означає, що чат підтримуватиме довгі діалоги. Реальна пам'ять контексту повністю залежить від обраної моделі та ваших запитів. Погано підходить для генерації новел, історій та оповідань.

Не підтримує нові моделі зі зміненим методом переквантизації, що порушує сумісність програми зі старими LLM. Поки ця проблема не буде вирішена або не розроблять рішення, що дозволяє використовувати моделі різних варіантів квантизації одночасно, користувачі обмежені нейромережами зі списку GPT4all. Іноді зациклюється на довгих діалогах, а також при натисканні Enter без введення нових подробиць. Це трапляється набагато частіше, ніж у KoboldCpp. Немає налаштування точної довжини тексту, що генерується. З одного боку, завжди можна вказати тривалість словесно, з іншого – налаштування мінімальної кількості генерованих токенів істотно полегшить життя користувачам. Хоча в установках є функція видалення компонентів, що включає свіжі оновлення, її не завжди вдається використовувати без видалення програми (наприклад, якщо оновлення видалено з кешу). Це досить незручно, якщо виникає помилка.

Таким чином, на основі проведеного дослідження можливо зробити наступні висновки.

Застосування KoboldCpp чи GPT4all залежить від мети використання ПЗ. Якщо експериментувати з новими моделями, які регулярно з'являються на Reddit, найкраще використовувати KoboldCpp. Програма підтримує більше форматів LLM, включаючи їх «нецензурні» версії, а також нейромережі, які спочатку були призначені для створення творчих творів. Якщо потрібний зручний та швидкий чат-бот, який підтримує роботу з документами, що зберігаються на ПК, більше підійде GPT4all. Програма від Nomic AI краще справляється з роллю помічника, швидше запускається, стабільно працює і рідко блокується. У KoboldCpp не така гарний підтримка, але він має велику перевагу – доступ до моделей будь-якого розміру. Наприклад, можна завантажити LLM з 33 або 65 мільярдами параметрів і запустити нейромережу в KoboldCpp. У GPT4all обмежено моделями із 7-13 мільярдами параметрів. Оскільки кількість параметрів безпосередньо впливає на «словниковий запас» LLM, точніше, можливість оперувати датасетом і краще будувати фрази, це ультимативна перевага, про яку варто пам'ятати. Проте LLM понад 13

мільярдів параметрів займають багато місця та потребують великого обсягу RAM. Наприклад, від 20-32 ГБ ОЗП. Далеко не всі звичайні користувачі можуть дозволити собі навіть 16 ГБ RAM, не кажучи вже про 32, 64 і більше. Навіть якщо вистачає оперативної пам'яті, масштабні GGML нейромережі генерують текст у рази повільніше, ніж компактні моделі. Низька швидкість роботи нівелює переваги, які дають LLM із 30 або 65 мільярдами параметрів, якщо не використовуєте машини з високопродуктивною начинкою. Припустимо, LLM генеруватиме середній за якістю текст, але доведеться чекати 1000-2000 символів приблизно 5, 10, 20 хв. Багато параметрів не завжди означає кращу якість. Наприклад, у переліку лідерів (рис. 2.6) серед LLM на Hugging Face є Wizard-Vicuna-13B-Uncensored-HF, що обганяє LLaMa-30B, OPT-66B та Galactica-120B, а на першому місці закріпилася модель з 40 мільярдами параметрів - Falcon-40B-Instruct.



CHANGELOG

Model	Revision	Average	ARC (25-shot)	HellaSwag (10-shot)	MMLU (5-shot)	TruthfulQA (0-sh
tiiuae/falcon-40b-instruct	main	63.2	61.6	84.4	54.1	52.5
tiiuae/falcon-40b	main	60.4	61.9	85.3	52.7	41.7
ausboss/llama-30b-supercot	main	59.8	58.5	82.9	44.3	53.6
llama-65b	main	58.3	57.8	84.2	48.8	42.3
MetaIX/GPT4-X-Alpasta-30b	main	57.9	56.7	81.4	43.6	49.7
digitous/Alpacino30b	main	57.4	57.1	82.6	46.1	43.6
Aeala/GPT4-x-AlpacaDente2-30b	main	57.2	56.1	79.8	44	49.1
TheBloke/Wizard-Vicuna-13B-Uncensored-HF	main	57	53.6	79.6	42.7	52
TheBloke/dromedary-65b-lora-HF	main	57	57.8	80.8	50.8	38.8
llama-30b	main	56.9	57.1	82.6	45.7	42.3
openaccess-ai-collective/wizard-mega-13b	main	55.7	52.5	78.6	41	50.6
TheBloke/vicuna-13B-1.1-HF	main	53.7	47.4	78	39.6	49.8
chavinlo/gpt4-x-alpaca	main	53.6	47.8	77.7	39.1	49.7
eachadea/vicuna-13b	main	53.1	45.1	77.9	38.1	51.3
medalpaca/medalpaca-13b	main	52.6	48	78.6	37.2	46.8
stable-vicuna-13b	main	52.4	48.1	76.4	38.8	46.5
eachadea/vicuna-7b-1.1	main	52.2	47	75.2	37.5	48.9
llama-13b	main	51.8	50.8	78.9	37.7	39.9
alpaca-13b	main	51.7	51.9	77.6	37.6	39.6
facebook/galactica-120b	main	51.2	46.8	66.4	50.4	41.3

Рисунок 2.6 – Перелік LLM на платформі Hugging Face

Список лідерів постійно змінюється. Falcon у будь-який момент можуть витіснити з п'єдесталу, виявив світові нового короля серед LLM. Проте свою роль LLM вже зіграла, показавши всьому світу, що нейромережа з меншою кількістю параметрів може обігнати навіть просунутих конкурентів.

Кількість параметрів важлива лише для однотипних моделей. Наприклад, між WizardLM 7B, 13B і 30B є суттєва різниця, що виявляється як у різноманітності відповідей, так і в можливості тонко сприймати запити. З творчими завданнями 30B справляється набагато цікавіше, ніж 13B і 7B. Зате короткі згенеровані пости для соцмереж у WizardLM 30B будуть лише на 15-25 % якіснішими, ніж у WizardLM 13B, тому відмінності між ними не настільки сильно впадають у вічі. Тобто, якщо є потужний ПК, має сенс використовувати все найкраще, що може запропонувати ринок. При цьому рекомендується порівняти моделі різного розміру і вирішити, що важливіше – технічна якість генерації, навіть якщо вона вимагає значного очікування, або швидкість виводу, що дозволяє перебрати більше варіантів і швидко переписати найоптимальніший з них, виправляючи всі знайдені недоліки.

ChatGPT, GPT-4, Lex.page або інші безкоштовні некомерційні проекти з онлайн-доступом, наприклад, Open Assistant, є корисними. Вони забезпечують швидкодію, яка і не досяжна більшості ПК рівня користувача, постійно навчаються, пропонують нові функції або плагіни, що розширюють можливості нейромереж. Однак у них є суттєві мінуси.

ChatGPT та інші LLM, що працюють онлайн, можуть збирати та пересилати дані користувача розробнику. Навмисно чи ненавмисно – не так вже й важливо. Локальні LLM не зливатимуть інформацію, включаючи результати генерації, якщо сам користувач не погодиться передавати відомості розробникам. Чим популярніший проект, тим більше обмежень. Цензура не лише захищає користувачів від небажаного контенту, а й значно обмежує креативність нейромереж. Якщо користувач генерує NSFW-контент або бажаєте отримати більше творчих відповідей, краще використовувати локальні LLM без цензури. Безкоштовні онлайн нейромережі можуть відключити доступ до них, платні – ввести регіональні обмеження або підвищити ціни без попередження. Онлайн-проекти можуть обмежувати кількість генерацій, щоб зменшити навантаження на сервери, або додавати користувачів до списку очікування. Розробники не завжди діляться з

користувачами інформацією про те, з чого будується архітектура мережі. Локальні LLM розбирають мало не по гвинтику та розвивають за допомогою спільноти, тому інформації про моделі набагато більше. Нейромережі загального призначення найчастіше генерують спрощені відповіді. Користувач може завантажити модель, навчену на вузько-спрямованих даних, або самостійно навчити її, щоб отримувати найкращі результати. Локальні LLM з локалізацією мови можуть як страждати від серйозних обмежень, а й видавати примітивні результати. Встановивши LLM від 30 мільярдів параметрів та переклавши текст буде набагато простіше отримати корисні відомості. Вибір моделей у комерційних проектах обмежений. Найчастіше користувачі отримують доступ до однієї моделі, рідше – до 2-3. Користувач можете завантажити та встановити будь-які LLM, що відповідають наявним обчислювальним потужностям хоч щодня. Не можна сказати, що некомерційні текстові нейромережі в рази кращі за платні проекти. Вони можуть поступатися за якістю та швидкістю генерації, точності відповідей, актуальністю даних, кількістю додаткових інструментів. Однак нові варіанти LLM з'являються чи не щодня, поступово усуваючи розрив між open-source та закритими рішеннями, тому їх варто впроваджувати в робочі процеси.

2.3 Визначення основних властивостей KoboldCpp

В сімействі ПЗ Kobold є 3 проєкти: KoboldAI, KoboldCpp, Kobold Lite, що призначені для локального запуску LLM з акцентом на взаємодію через текстовий інтерфейс (найчастіше – для інтерактивної генерації, таких як storytelling, RP та ін.). В табл. 2.1 наведено їх основні характеристики. Розглянемо їх більш детально. KoboldAI – це вебзастосунок з відкритим кодом, створений спеціально для генерації текстів у жанрі рольових ігор, інтерактивних історій та чат-ботів. Його головна відмінність – підтримка локального запуску LLM разом із зручним браузерним інтерфейсом – Web UI

(на Flask), оптимізованим під наративну взаємодію. Моделі, що підтримуються: GPT-J, GPT-NeoX, Pygmalion, Fairseq, OpenAI (через API), HuggingFace.

Таблиця 2.1 – Характеристики ПЗ Kobold

Характеристика	KoboldAI	KoboldCpp	Kobold Lite
Мова реалізації	Python	C++ (llama.cpp)	Python
Інтерфейс	Web UI	Web UI / CLI	Web UI
Моделі	GPT-J, GPT-NeoX та ін.	LLaMA, Mistral (GGUF)	HuggingFace Transformers
Продуктивність	Середня-висока	Висока (особливо CPU)	Середня
Потребує Python	Так	Ні	Так
Розширення/модулі	Так	Ні	Ні
Встановлення	Помірне складне	Просте	Просте
Підходить для слабких ПК	Не завжди	Так	Частково

Таким чином, можна виділити такі особливості: найстаріший та найбільший із 3-ох проєктів, підтримка розширень, інтеграція з Colab, підтримує локальні та хмарні моделі, є різні «м двигуни»: KoboldAI GPT, HuggingFace, TextGen, AI Horde та ін. Разом з тим, вимагає Python, установки залежностей, може бути важким для слабких машин.

KoboldCpp – це окремий виконуваний файл, який запускає LLM у вигляді локального API-сервера та не потребує Python чи додаткових залежностей. KoboldCpp це надійний, легкий і надзвичайно ефективний рушій для запуску LLM локально на CPU. Завдяки підтримці GGUF, багатопотоковості, сумісності з інтерфейсами як KoboldAI або LangChain, і простоті запуску – KoboldCpp оптимально підходить для ентузіастів, дослідників та розробників, які хочуть використовувати LLM без GPU та без хмари. ПЗ підтримує quantized-моделі (Q4_K_M, Q5, Q6 та ін.) – економія ОЗП. По суті, цей сервер може обслуговувати запити як самостійно, так і через KoboldAI чи будь-який інший OpenAI-сумісний інтерфейс.

Вбудований API-сервер надає локальний REST API, який можна підключити до: KoboldAI, LM Studio, Text Generation WebUI, LangChain, LlamaIndex, або будь-якої програми з підтримкою OpenAI-style API.

Наприклад, для інтеграції з KoboldAI треба запустити KoboldCpp як API-сервер: /koboldcpp − −model mistral. gguf − −port 5001 − −api. У KoboldAI вибереться рушій – Connect to external KoboldCpp. Усі запити з інтерфейсу KoboldAI будуть оброблятися KoboldCpp

Він побудований на основі llama.cpp з підтримкою досить нового формату GGUF (всі метадані + tokenizer в одному файлі). Він має повну сумісність з популярними моделями: LLaMA 1/2/3, Mistral, Mixtral, OpenHermes, Nous Hermes, Mythalion, Dolphin, а також кастомні fine-tuned GGUF-моделі.

Багатопотокова генерація забезпечує повну підтримку OpenMP, POSIX Threads (pthreads) або Microsoft Threads для багатоядерної генерації. Параметр --threads визначає кількість потоків, що використовуються при генерації тексту. Це дозволяє запускати навіть 7B/13B моделі з комфортною швидкістю на звичайних ПК.

KoboldCpp також працює в інтерактивному режимі безпосередньо з командного рядка без запуску сервера, дозволяючи використовувати його без графічного інтерфейсу, так само, як і llama.cpp. Для цього використовується прапорець --cli, щоб увійти в режим терміналу.

У новій версії KoboldCppp передбачено підтримку створення локальних зображень, завдяки stable-diffusion.cpp. Він надає кінцеву точку txt2img, сумісну з ComfyUI та A1111, яку користувач може використовувати у вбудованому Kobold Lite або в багатьох інших сумісних інтерфейсах, таких як SillyTavern. Для цього вибирається сумісна модель SD3, Flux, SD1.5 або SDXL для завантаження через панель запуску графічного інтерфейсу або за допомогою методу .safetensors --sdmodel. В даному варіанті запуску VAE та LoRA повинні бути вбудовані всередині самої моделі. Рекомендується використовувати fp16. --sdvae --sdvaeauto --sdlora. Актуальна версія підтримує прапорці, що наведені на рис. 2.7.

Kobold Lite – це мінімалістичний, швидкий і портативний вебінтерфейс для локального використання LLM, розроблений як спрощена альтернатива

KoboldAI. Основна мета – надати зручний браузерний інтерфейс без складної інсталяції, з можливістю запуску моделей на CPU через KoboldCpp або llama.cpp. Kobold Lite сам не виконує моделі, а підключається до API-серверів, таких як: KoboldCpp, сервер llama.cpp, Ollama (обмежено), GPT4All з REST API. Потрібно просто вказати URL локального рушія (наприклад, http://localhost:5001/api).

```
--sdmodel      Specify a stable diffusion model to enable image generation.
--sdthreads    Use a different number of threads for image generation if specified.
--sdquant      If specified, loads the model quantized to save memory.
--sdclamped    If specified, limit generation steps and resolution settings for shared
use. Optionally allows setting max size.
--sdlora       If specified, tries to load a Stable Diffusion LoRA model
--sdloramult   Set the LoRA multiplier
--sdvae        Set a custom VAE
--sdvaeauto    Use built in fallback VAE (TAESD)
--sdnotile     Disable VAE tiling
--sdt5xxl      For Flux and SD3, you need to load a T5-XXL language model as well. Some
files bundle it with the main model.
--sdclipl      For Flux and SD3, you need to load a Clip-L model as well. Some files
bundle it with the main model.
--sdclipg      For SD3, you need to load a Clip-L model as well. Some files bundle it
with the main model.
```

Рисунок 2.7 – Варіанти запуску моделі в KoboldCpp

В цілому, інтерфейс має декілька функцій:

- поле чату – для інтерактивного спілкування з LLM;
- історія – зберігає всі попередні відповіді;
- системні інструкції – можна вказати поведінку моделі (як «system prompt»);
- налаштування генерації: Temperature, top-k, top-p, repetition penalty, максимальна довжина відповіді;
- підтримка Persona / Memory (частково).

Kobold Lite підтримує моделі HuggingFace (через Transformers).

Таким чином, KoboldAI – найбільш гнучкий і потужний, але важкий; KoboldCpp – найлегший і найшвидший, особливо для CPU та слабких машин; Kobold Lite – застаріла полегшена версія, зараз майже не використовується. Тобто, якщо використовується Raspberry Pi або ПК без GPU – KoboldCpp краще. Якщо є потужний ПК а робоча станція чи сервер і потрібна гнучкість –

KoboldAI. Надалі доцільно визначити особливості актуальних форматів файлів сучасних LLM.

2.4 Особливості форматів GGML і GGUF

GGML і GGUF (GPT Generated Unified Format) позначають однакову концепцію, причому GGUF – це новіша версія, що включає додаткові дані про модель. Це покращення дозволяє краще підтримувати різні архітектури та включає шаблони запитів. GGUF можна запускати лише на CPU або частково/повністю переносити на GPU. Використовуючи k-квантування, GGUF може змінюватись від 2 до 8 бітів. В даному випадку, раніше GPTQ служив методом квантування, оптимізованим лише для GPU. Однак його перевершив AWQ, який приблизно вдвічі швидше. Останнє досягнення в цій галузі – EXL2, який забезпечує ще кращу продуктивність. Зазвичай, ці методи квантування реалізуються з використанням 4 бітів.

GGUF більш сучасний, ніж GGML, є більш зрозумілим, розширюваним, універсальним та здатним включати нову інформацію без шкоди для сумісності з попередніми моделями. Розглянемо основні типи квантизації GGUF, за якими з назви LLM можна легко визначити, чи підходить вона чи ні:

– Q2_K, Q3_K_S/M/L – найменший розмір, висока швидкість, значна втрата якості;

– Q4_K_S/M – хороший баланс розміру та якості;

– Q5_K_S/M – покращена якість порівняно з Q4, незначне зниження швидкості;

– Q6_K – висока якість, помірна швидкість;

– Q8_0 – найвища якість, найнижча швидкість.

На даний час, існують такі рекомендації щодо вибору GGUF моделей:

– 4-6 ГБ VRAM – моделі Q3_K_M або Q4_K_S;

– 8 ГБ VRAM – моделі Q4_K_M або Q5_K_S;

– 12-16 ГБ VRAM – моделі Q5_K_M або Q6_K;

– 24 ГБ VRAM – моделі Q6_K або Q8_0.

Підсумовуючи, можна сказати, що перевагами GGUF перед GGML є такі положення.

Велика гнучкість – GGUF забезпечує більшу гнучкість у зберіганні даних моделі, що спрощує додавання нових функцій та адаптацію до різних архітектур моделі.

Найкраща сумісність – GGUF розроблено з урахуванням більшої сумісності з ширшим спектром інструментів та бібліотек, що спрощує робочий процес для розробників.

Підвищена ефективність – GGUF може забезпечити більшу ефективність з точки зору розміру файлу та швидкості завантаження, що може призвести до підвищення продуктивності LLM.

Стандартизація – GGUF прагне стати єдиним форматом, що сприяє взаємодії між різними інструментами та моделями.

РОЗДІЛ 2
ЛОКАЛЬНЕ СЕРЕДОВИЩЕ ДЛЯ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ НА ОСНОВІ KOBOLDCPP

2.1 Інструментарій для локального запуску великих мовних моделей

У зв'язку з активним розвитком LLM зросла потреба у їхньому локальному розгортанні на ПК, одноплатних мікрокомп'ютерах та ін. Це дозволяє забезпечити автономність, підвищену безпеку даних, зниження затримок та зменшення витрат на хмарні обчислення. Для реалізації таких можливостей існує низка фреймворків та рушіїв, які оптимізовані для роботи LLM у локальному середовищі.

По-перше розглянемо llama.cpp. Це високоефективна реалізація моделей GPT-подібної архітектури (особливо серій LLaMA від Meta) з використанням C++ і апаратного прискорення через SIMD-інструкції. llama.cpp підтримує кросплатформну роботу (Linux, Windows та macOS) і адаптований для запуску на CPU без необхідності у GPU. На даний час, реалізовано інтерфейси quantization – int4/int8), що дозволяє запускати LLM навіть на Raspberry Pi.

KoboldCpp – оптимізований для GGUF-моделей, працює на C++ з інтеграцією GUI. Він створений на базі llama.cpp, оптимізований для інтерактивного текстового покоління. Також має інтеграцію з інструментами roleplay/novel-writing, підтримує chat-інтерфейси, API-сумісність з OpenAI, а також багатопотоковість. На даний час, KoboldCpp має кілька варіантів: KoboldAI, KoboldCpp і Kobold Lite, які розглядаються вже як окремі проєкти. Назва KoboldCpp відсилає до проєкту KoboldAI – вебінтерфейсу для генерації історій. KoboldCpp є бінарним рушієм, який сумісний із цим інтерфейсом, замінюючи попередні рушії на базі Transformers або GPT-j на значно ефективніший варіант.

GPT4all – це open-source екосистема для локального запуску, розроблена компанією Nomic AI. Проєкт орієнтований на забезпечення доступного,

приватного та ефективного використання LLM на локальних машинах, без необхідності підключення до хмари або великих обчислювальних кластерів. Основними компонентами GPT4all є GPT4all Backend (gpt4all.cpp), GPT4all Chat UI, модельний хаб GPT4all. Gpt4all.cpp – це рушій на C++ (форк llama.cpp), який дозволяє запускати LLM з підтримкою квантування. Він оптимізований для запуску моделей без GPU – на звичайних CPU, включаючи середовища Windows, Linux та macOS. GPT4All Chat UI – графічний інтерфейс, який дозволяє користувачеві легко взаємодіяти з моделлю. Програму можна встановити як десктоп-додаток (Electron-based), що працює офлайн. Має зручний інтерфейс для ведення розмов, історії чатів та перемикання між моделями. Модельний хаб GPT4all [17] містить галерею підтримуваних моделей, серед яких:

– GPT-J (класичний, базовий варіант);
– MPT (MosaicML) – з відкритим кодом;
– LLaMA, LLaMA2 – з квантуванням (Q4, Q5);
– Mistral – новіші моделі з більш ефективною архітектурою;
– Replit-code-v1 – для генерації коду.

В свою чергу, є також GPT4all Python API. Це простий SDK для інтеграції GPT4All-моделей у свої додатки.

Ollama – один з найзручніших фреймворків для локального використання LLM. Завдяки інтегрованому API, автоматизованому менеджменту моделей, підтримці кастомізації та сучасним моделям він підходить як для ентузіастів, так і для професійних розробників, які хочуть використовувати LLM у своїх продуктах без залежності від хмари.

Ollama – це високорівневий фреймворк та CLI-рушій, створений для максимально простого, стандартизованого та ефективного запуску великих мовних моделей (LLM) локально на звичайному комп'ютері. Він орієнтований на розробників, дослідників та інтеграторів, яким потрібна швидка інтеграція локального ШІ в програми або агентні системи. Ollama використовує унікальний підхід: LLM-модель – це своєрідний контейнер (як у Docker), який

можна завантажити, інсталювати й запускати однією командою. Це усуває складну інсталяцію моделей вручну (немає потреби шукати GGUF, завантажувати tokenizer тощо). Усі залежності інтегровані в один образ. Ollama надає простий CLI (термінал), з якого можна запускати модель у чат-режимі, а також повноцінний REST API для підключення до програм. Це дає змогу інтегрувати Ollama у ваші застосунки, боти, скрипти, LangChain, LlamaIndex, AutoGen та інші агенти. Ollama дозволяє створювати свої власні кастомні моделі на основі існуючих через Modelfile — подібно до Dockerfile. В цілому, перевагами слід вважати REST API для production, підтримка LangChain, LLM-agent tools.

LM Studio – це кросплатформний десктопний застосунок для Windows, macOS та Linux, який дозволяє запускати великі мовні моделі локально на комп'ютері без потреби у хмарних API або серверному розгортанні. Його основна мета – надати простий та інтуїтивно зрозумілий графічний інтерфейс (GUI) для взаємодії з LLM, з можливістю використовувати найсучасніші моделі прямо з HuggingFace. Базується на llama.cpp і підтримує GGUF-моделі. При цьому, є підтримка кількох моделей одночасно; можливість завантажити GGUF-файл вручну; автоматичне оновлення моделей; кешування відповідей, що прискорює повторне використання.

Text Generation WebUI – універсальний рушій, що підтримує запуск різних моделей (LLaMA, Mistral, Falcon, GPT-J) з допомогою Transformers або llama.cpp. Має веб-інтерфейс, графічні налаштування параметрів температури, довжини відповіді, prompt engineering.

GGML (Generative Graphical Machine Learning) – це бібліотека, що лежить в основі llama.cpp і сумісних рушіїв. Новий формат GGUF (Unified Format) спрощує обмін моделями між фреймворками та зберігає метадані, токенізатори, параметри квантування тощо.

MLC LLM – фреймворк від Apache TVM, що забезпечує компіляцію LLM у власні виконувані файли, оптимізовані під конкретну архітектуру (наприклад, CUDA, Metal, Vulkan).

MLC LLM дозволяє створювати повністю автономні застосунки без зовнішніх залежностей, зокрема на Android.

Локальні фреймворки підтримують різні рівні квантування (Q2_K, Q4_0, Q5_K, тощо), що значно знижує об'єм пам'яті моделей (до 4-6 ГБ замість 30-60 ГБ), дозволяючи використовувати їх на пристроях з обмеженими ресурсами. Також забезпечується інтеграція з зовнішніми бібліотеками на Python (LangChain, Transformers), що відкриває широкі можливості для розробки агентних систем, чат-ботів, локальних.

2.2 Порівняльна оцінка KoboldCpp та GPT4all

KoboldCpp і GPT4ALL охоплюють різну аудиторію, але мають однакові цілі – зробити роботу з мовними моделями зручніше для звичайних користувачів, додати більше контролю над генерацією та забезпечити стабільну роботу текстових нейромереж. KoboldCpp – програма з вебінтерфейсом (Web UI), яка запускається у новій вкладці браузера. GPT4all – окрема програма з графічним інтерфейсом (GUI), яка не потребує браузера для роботи. Крім принципу запуску існує різниця в позиціонуванні ПЗ. Спільнота KoboldAI з'явилася задовго до поширення мовних моделей із сімейства LLaMa, став притулком авторів, які займаються створенням новел, пригодницьких історій, фанфіків та рольових ігор. Тоді KoboldAI був альтернативою не ChatGPT, а Novel AI – платного генератора текстових історій. До моменту виникнення загальнодоступних LLM у KoboldAI сформувалася ціла плеяда власних моделей, заточених на створення цікавих оповідань. На момент появи загальнодоступних LLM у KoboldAI сформувалася ціла плеяда своїх моделей [18], заточених створення цікавих оповідань. Серед них зустрічалися як ресурсомісткі нейромережі, які не запускалися без топових відеокарт, так і легковагі текстові генератори, які міг потягнути навіть старий ноутбук (рис. 2.1). Переписувати існуючий

дистрибутив з нуля - довго і накладно, тому з'явилася окрема версія програмного забезпечення, що поєднує в собі найкращі можливості llama.cpp та розширену функціональність API Kobold – KoboldCpp (рис. 2.2).

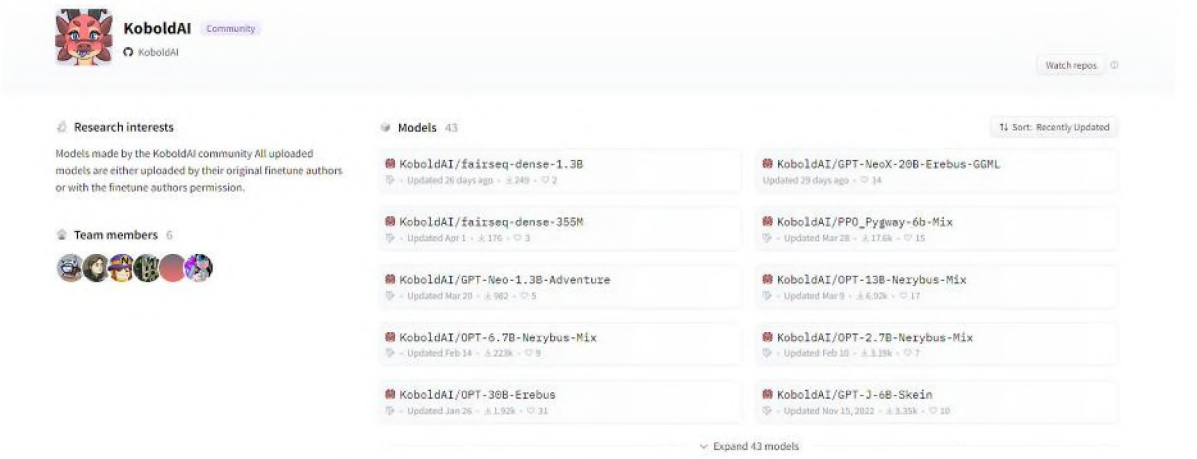


Рисунок 2.1 – Моделі Kobold AI

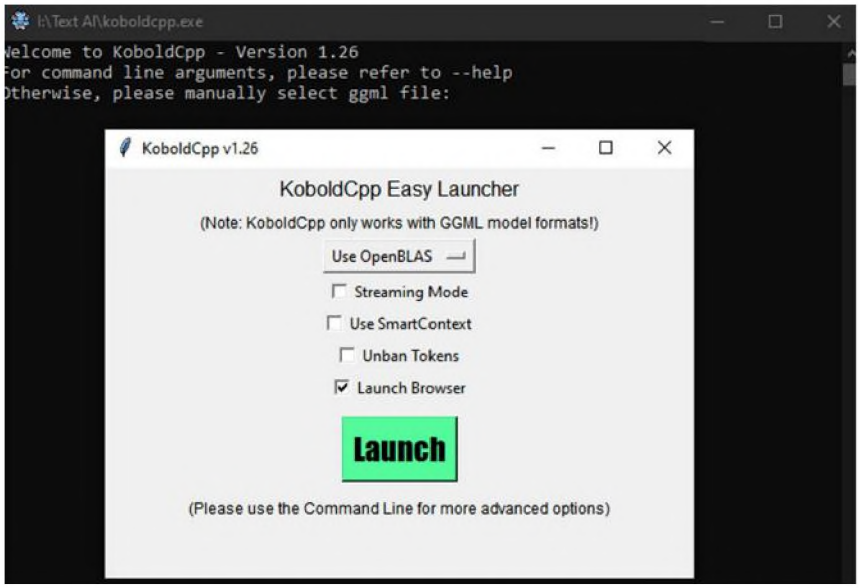


Рисунок 2.1 – Інтерфейс для запуску KoboldCpp

Програма відрізняється крихітним розміром (приблизно 10 МБ, не рахуючи ваги нейромереж). Можна сказати, що KoboldCpp запускає будь-які існуючі формати GGML, якщо вони мають попит. Всі інші файли не працюватимуть у ПЗ. Наприклад, така доля спіткала bloomz.cpp. Головні плюсами KoboldCpp слід вважати такі положення: підтримка більшості

мовних моделей, у тому числі тих, що використовують новий метод переквантизації; високошвидкісна генерація тексту; може зберігати та завантажувати історії у форматі .JSON; варіанти сценаріїв, які підходять на будь-який випадок життя; великий перелік пресетів для швидкого старту; розширене меню налаштувань; легка генерація історій, діалогів, гайдів та інструкцій; можна вибрати іншу версію, якщо нова вам чомусь не підійшла. Інтерфейс при налаштуванні відповідає рис. 2.3.

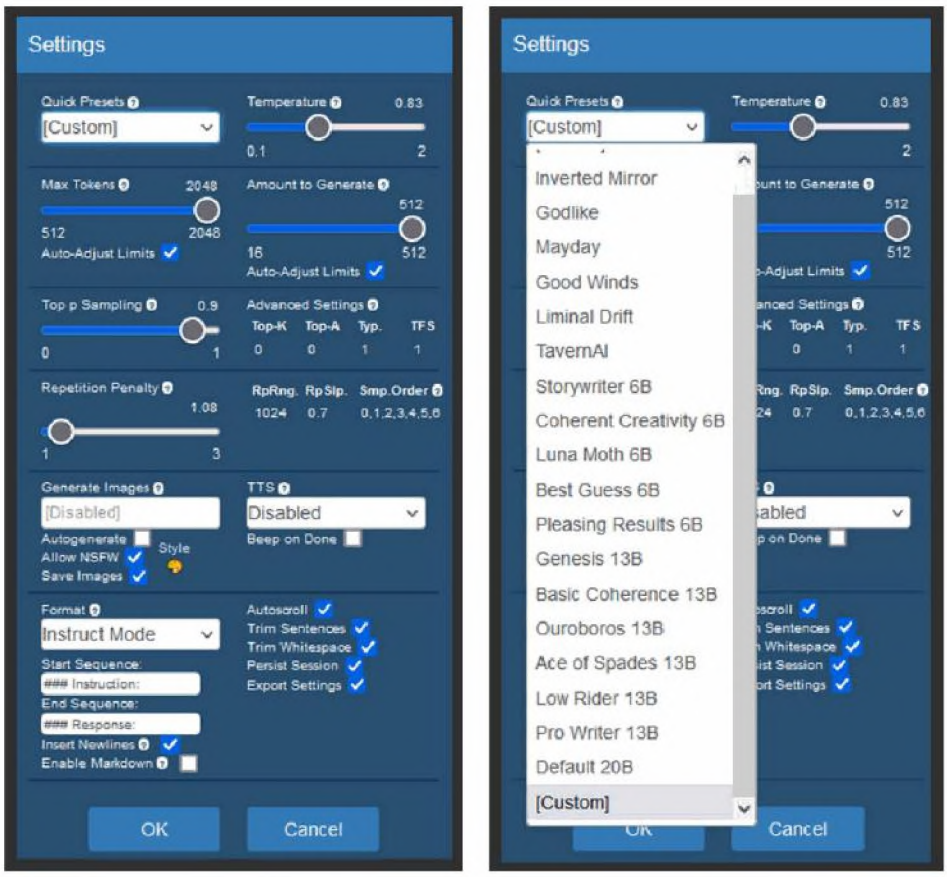


Рисунок 2.3 – Інтерфейс при налаштуванні KoboldCpp

Разом з тим, на даний час, відсутня підтримка документів (не можна додати в якості джерелі даних файли PDF або TXT), не підтримує тривале запам'ятовування контексту (максимальна кількість токенів – 2048); може довго запускатись; працює в окремій вкладці браузера; немає автоматичного оновлення; більше підходить для створення оповідань, ніж для довгих

діалогів; щоб запустити нову LLM, потрібно окремо її відкривати за допомогою провідника Windows.

GPT4all найбільше нагадує безкоштовний варіант ChatGPT. Головне вікно програми – сторінка з чатом (рис. 2.4). Вгорі – панель вибору доступних моделей, у бічній вкладці зліва – можливість створення нового чату або збереження наявного, панель для завантаження моделей та перевірки оновлень. Вгорі праворуч знаходяться базові налаштування, а також панель для керування ними.

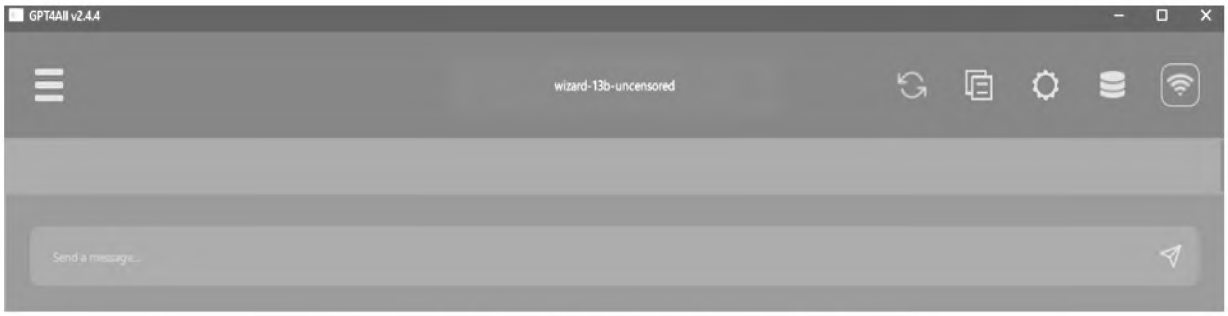


Рисунок 2.4 – Головне вікно GPT4all

Тобто здійснюється оптимальна реалізація для локального чат-бота або програми для виконання інструкцій користувача. Про це говорять і розробники програми, які прагнуть надати найкращий софт для звичайних людей, які бажають отримати просунутого нейромережевого асистента та використовувати його для творчості, роботи чи експериментів. На основі досліджень можна вказати такі переваги GPT4all:

– зручний інтерфейс (рис. 2.5);

– автономна робота, яка не потребує запуску браузера;

– підтримує багатопотокові процесори, а також дозволяє вручну призначати кількість потоків, що використовуються для прискорення генерації тексту;

– може зчитувати дані з документів на комп'ютері за допомогою плагіну LocalDocs;

– максимальна довжина відповіді не обмежена 2048 токена. Додаток дозволяє призначити більший ліміт – 4096 або 131072 токенів;

– підтримує збереження чатів на диску;

– крута робота з чатом;

– можна вручну настроїти зразок запиту для кращої взаємодії з LLM (Prompt Template);

– швидкий запуск скачаних LLM;

– зручне перемикання між доступними нейромережами;

– стабільні оновлення;

– програма сповіщає про вихід патчів, після підтвердження автоматично завантажує та встановлює їх, показує, що сталося в оновленнях;

– розробники постійно додають нові функції.

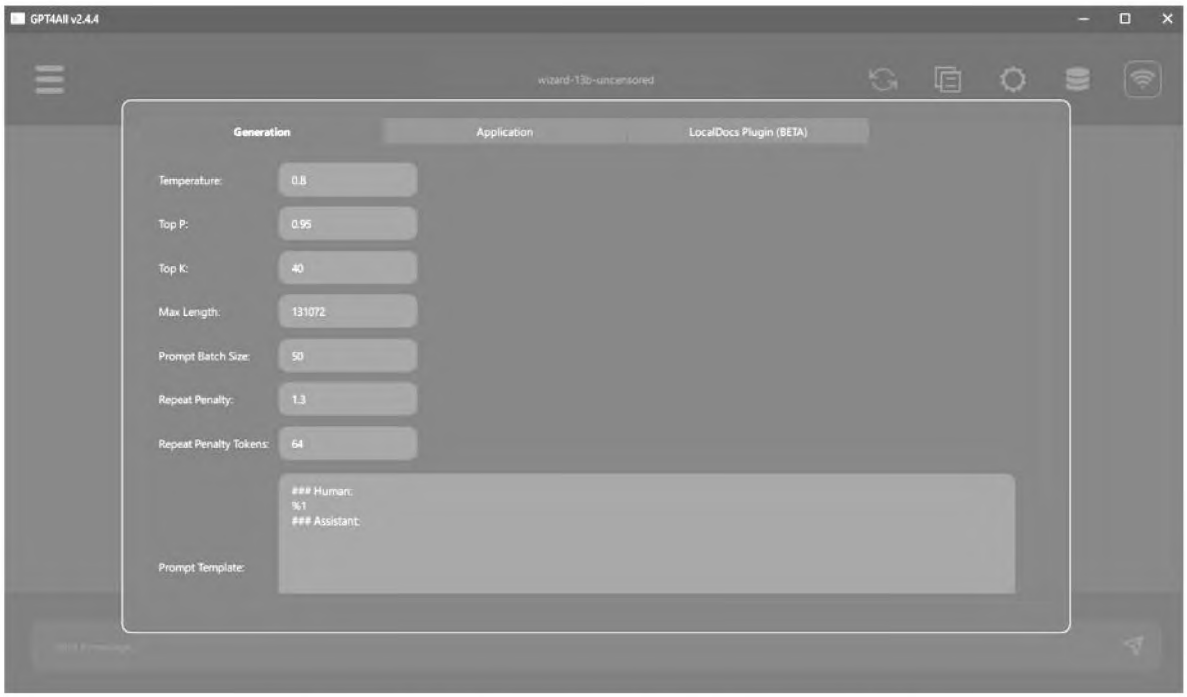


Рисунок 2.5 – Вікно налаштувань GPT4all

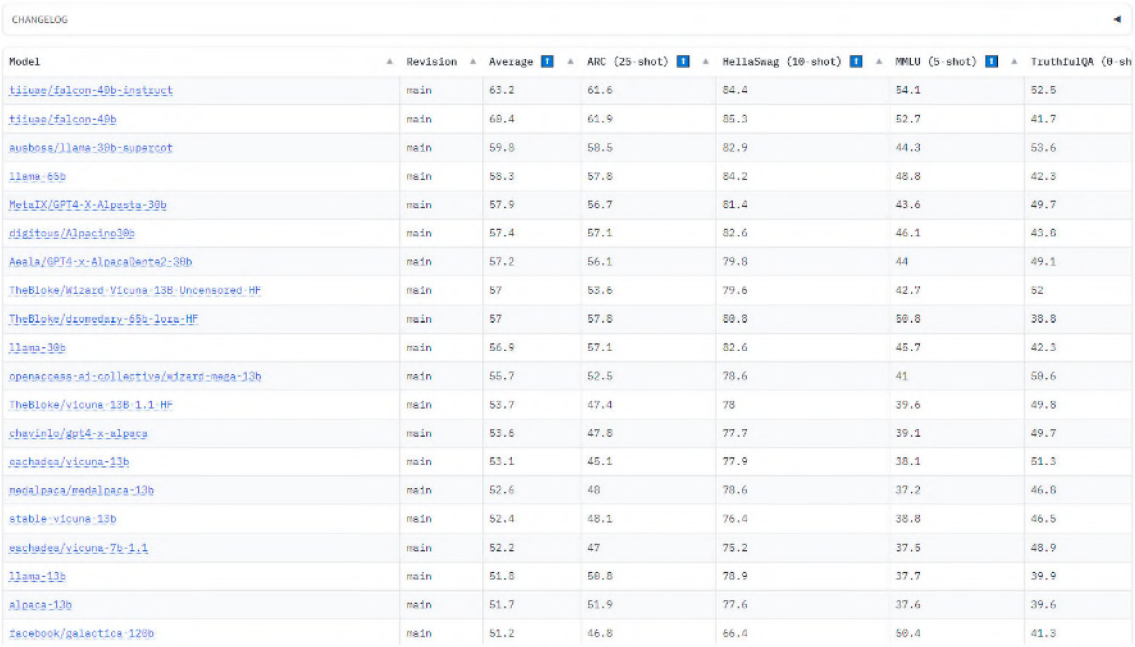
До недоліків GPT4all слід віднести такі положення. Можливість збільшити максимальну довжину відповіді не означає, що чат підтримуватиме довгі діалоги. Реальна пам'ять контексту повністю залежить від обраної моделі та ваших запитів. Погано підходить для генерації новел, історій та оповідань.

Не підтримує нові моделі зі зміненим методом переквантизації, що порушує сумісність програми зі старими LLM. Поки ця проблема не буде вирішена або не розроблять рішення, що дозволяє використовувати моделі різних варіантів квантизації одночасно, користувачі обмежені нейромережами зі списку GPT4all. Іноді зациклюється на довгих діалогах, а також при натисканні Enter без введення нових подробиць. Це трапляється набагато частіше, ніж у KoboldCpp. Немає налаштування точної довжини тексту, що генерується. З одного боку, завжди можна вказати тривалість словесно, з іншого – налаштування мінімальної кількості генерованих токенів істотно полегшить життя користувачам. Хоча в установках є функція видалення компонентів, що включає свіжі оновлення, її не завжди вдається використовувати без видалення програми (наприклад, якщо оновлення видалено з кешу). Це досить незручно, якщо виникає помилка.

Таким чином, на основі проведеного дослідження можливо зробити наступні висновки.

Застосування KoboldCpp чи GPT4all залежить від мети використання ПЗ. Якщо експериментувати з новими моделями, які регулярно з'являються на Reddit, найкраще використовувати KoboldCpp. Програма підтримує більше форматів LLM, включаючи їх «нецензурні» версії, а також нейромережі, які спочатку були призначені для створення творчих творів. Якщо потрібний зручний та швидкий чат-бот, який підтримує роботу з документами, що зберігаються на ПК, більше підійде GPT4all. Програма від Nomic AI краще справляється з роллю помічника, швидше запускається, стабільно працює і рідко блокується. У KoboldCpp не така гарний підтримка, але він має велику перевагу – доступ до моделей будь-якого розміру. Наприклад, можна завантажити LLM з 33 або 65 мільярдами параметрів і запустити нейромережу в KoboldCpp. У GPT4all обмежено моделями із 7-13 мільярдами параметрів. Оскільки кількість параметрів безпосередньо впливає на «словниковий запас» LLM, точніше, можливість оперувати датасетом і краще будувати фрази, це ультимативна перевага, про яку варто пам'ятати. Проте LLM понад 13

мільярдів параметрів займають багато місця та потребують великого обсягу RAM. Наприклад, від 20-32 ГБ ОЗП. Далеко не всі звичайні користувачі можуть дозволити собі навіть 16 ГБ RAM, не кажучи вже про 32, 64 і більше. Навіть якщо вистачає оперативної пам'яті, масштабні GGML нейромережі генерують текст у рази повільніше, ніж компактні моделі. Низька швидкість роботи нівелює переваги, які дають LLM із 30 або 65 мільярдами параметрів, якщо не використовуєте машини з високопродуктивною начинкою. Припустимо, LLM генеруватиме середній за якістю текст, але доведеться чекати 1000-2000 символів приблизно 5, 10, 20 хв. Багато параметрів не завжди означає кращу якість. Наприклад, у переліку лідерів (рис. 2.6) серед LLM на Hugging Face є Wizard-Vicuna-13B-Uncensored-HF, що обганяє LLaMa-30B, OPT-66B та Galactica-120B, а на першому місці закріпилася модель з 40 мільярдами параметрів - Falcon-40B-Instruct.



CHANGELOG

Model	Revision	Average	ARC (25-shot)	HellaSwag (10-shot)	MMLU (5-shot)	TruthfulQA (0-sh
tiiuae/falcon-40b-instruct	main	63.2	61.6	84.4	54.1	52.5
tiiuae/falcon-40b	main	60.4	61.9	85.3	52.7	41.7
ausboss/llama-30b-supercot	main	59.8	58.5	82.9	44.3	53.6
llama-65b	main	58.3	57.8	84.2	48.8	42.3
MetaIX/GPT4-X-Alpasta-30b	main	57.9	56.7	81.4	43.6	49.7
digitous/Alpacino30b	main	57.4	57.1	82.6	46.1	43.6
Aeala/GPT4-x-AlpacaDente2-30b	main	57.2	56.1	79.8	44	49.1
TheBloke/Wizard-Vicuna-13B-Uncensored-HF	main	57	53.6	79.6	42.7	52
TheBloke/dromedary-65b-lora-HF	main	57	57.8	80.8	50.8	38.8
llama-30b	main	56.9	57.1	82.6	45.7	42.3
openaccess-ai-collective/wizard-mega-13b	main	55.7	52.5	78.6	41	50.6
TheBloke/vicuna-13B-1.1-HF	main	53.7	47.4	78	39.6	49.8
chavinlo/gpt4-x-alpaca	main	53.6	47.8	77.7	39.1	49.7
eachadea/vicuna-13b	main	53.1	45.1	77.9	38.1	51.3
medalpaca/medalpaca-13b	main	52.6	48	78.6	37.2	46.8
stable-vicuna-13b	main	52.4	48.1	76.4	38.8	46.5
eachadea/vicuna-7b-1.1	main	52.2	47	75.2	37.5	48.9
llama-13b	main	51.8	50.8	78.9	37.7	39.9
alpaca-13b	main	51.7	51.9	77.6	37.6	39.6
facebook/galactica-120b	main	51.2	46.8	66.4	50.4	41.8

Рисунок 2.6 – Переліку LLM на платформі Hugging Face

Список лідерів постійно змінюється. Falcon у будь-який момент можуть витіснити з п'єдесталу, виявив світові нового короля серед LLM. Проте свою роль LLM вже зіграла, показавши всьому світу, що нейромережа з меншою кількістю параметрів може обігнати навіть просунутих конкурентів.

Кількість параметрів важлива лише для однотипних моделей. Наприклад, між WizardLM 7B, 13B і 30B є суттєва різниця, що виявляється як у різноманітності відповідей, так і в можливості тонко сприймати запити. З творчими завданнями 30B справляється набагато цікавіше, ніж 13B і 7B. Зате короткі згенеровані пости для соцмереж у WizardLM 30B будуть лише на 15-25 % якіснішими, ніж у WizardLM 13B, тому відмінності між ними не настільки сильно впадають у вічі. Тобто, якщо є потужний ПК, має сенс використовувати все найкраще, що може запропонувати ринок. При цьому рекомендується порівняти моделі різного розміру і вирішити, що важливіше – технічна якість генерації, навіть якщо вона вимагає значного очікування, або швидкість виводу, що дозволяє перебрати більше варіантів і швидко переписати найоптимальніший з них, виправляючи всі знайдені недоліки.

ChatGPT, GPT-4, Lex.page або інші безкоштовні некомерційні проекти з онлайн-доступом, наприклад, Open Assistant, є корисними. Вони забезпечують швидкодію, яка і не досяжна більшості ПК рівня користувача, постійно навчаються, пропонують нові функції або плагіни, що розширюють можливості нейромереж. Однак у них є суттєві мінуси.

ChatGPT та інші LLM, що працюють онлайн, можуть збирати та пересилати дані користувача розробнику. Навмисно чи ненавмисно – не так вже й важливо. Локальні LLM не зливатимуть інформацію, включаючи результати генерації, якщо сам користувач не погодиться передавати відомості розробникам. Чим популярніший проект, тим більше обмежень. Цензура не лише захищає користувачів від небажаного контенту, а й значно обмежує креативність нейромереж. Якщо користувач генерує NSFW-контент або бажаєте отримати більше творчих відповідей, краще використовувати локальні LLM без цензури. Безкоштовні онлайн нейромережі можуть відключити доступ до них, платні – ввести регіональні обмеження або підвищити ціни без попередження. Онлайн-проекти можуть обмежувати кількість генерацій, щоб зменшити навантаження на сервери, або додавати користувачів до списку очікування. Розробники не завжди діляться з

користувачами інформацією про те, з чого будується архітектура мережі. Локальні LLM розбирають мало не по гвинтику та розвивають за допомогою спільноти, тому інформації про моделі набагато більше. Нейромережі загального призначення найчастіше генерують спрощені відповіді. Користувач може завантажити модель, навчену на вузько-спрямованих даних, або самостійно навчити її, щоб отримувати найкращі результати. Локальні LLM з локалізацією мови можуть як страждати від серйозних обмежень, а й видавати примітивні результати. Встановивши LLM від 30 мільярдів параметрів та переклавши текст буде набагато простіше отримати корисні відомості. Вибір моделей у комерційних проектах обмежений. Найчастіше користувачі отримують доступ до однієї моделі, рідше – до 2-3. Користувач можете завантажити та встановити будь-які LLM, що відповідають наявним обчислювальним потужностям хоч щодня. Не можна сказати, що некомерційні текстові нейромережі в рази кращі за платні проекти. Вони можуть поступатися за якістю та швидкістю генерації, точності відповідей, актуальністю даних, кількістю додаткових інструментів. Однак нові варіанти LLM з'являються чи не щодня, поступово усуваючи розрив між open-source та закритими рішеннями, тому їх варто впроваджувати в робочі процеси.

2.3 Визначення основних властивостей KoboldCpp

В сімействі ПЗ Kobold є 3 проєкти: KoboldAI, KoboldCpp, Kobold Lite, що призначені для локального запуску LLM з акцентом на взаємодію через текстовий інтерфейс (найчастіше – для інтерактивної генерації, таких як storytelling, RP та ін.). В табл. 2.1 наведено їх основні характеристики. Розглянемо їх більш детально. KoboldAI – це вебзастосунок з відкритим кодом, створений спеціально для генерації текстів у жанрі рольових ігор, інтерактивних історій та чат-ботів. Його головна відмінність – підтримка локального запуску LLM разом із зручним браузерним інтерфейсом – Web UI

(на Flask), оптимізованим під наративну взаємодію. Моделі, що підтримуються: GPT-J, GPT-NeoX, Pygmalion, Fairseq, OpenAI (через API), HuggingFace.

Таблиця 2.1 – Характеристики ПЗ Kobold

Характеристика	KoboldAI	KoboldCpp	Kobold Lite
Мова реалізації	Python	C++ (llama.cpp)	Python
Інтерфейс	Web UI	Web UI / CLI	Web UI
Моделі	GPT-J, GPT-NeoX та ін.	LLaMA, Mistral (GGUF)	HuggingFace Transformers
Продуктивність	Середня-висока	Висока (особливо CPU)	Середня
Потребує Python	Так	Ні	Так
Розширення/модулі	Так	Ні	Ні
Встановлення	Помірне складне	Просте	Просте
Підходить для слабких ПК	Не завжди	Так	Частково

Таким чином, можна виділити такі особливості: найстаріший та найбільший із 3-ох проєктів, підтримка розширень, інтеграція з Colab, підтримує локальні та хмарні моделі, є різні «м двигуни»: KoboldAI GPT, HuggingFace, TextGen, AI Horde та ін. Разом з тим, вимагає Python, установки залежностей, може бути важким для слабких машин.

KoboldCpp – це окремий виконуваний файл, який запускає LLM у вигляді локального API-сервера та не потребує Python чи додаткових залежностей. KoboldCpp це надійний, легкий і надзвичайно ефективний рушій для запуску LLM локально на CPU. Завдяки підтримці GGUF, багатопотоковості, сумісності з інтерфейсами як KoboldAI або LangChain, і простоті запуску – KoboldCpp оптимально підходить для ентузіастів, дослідників та розробників, які хочуть використовувати LLM без GPU та без хмари. ПЗ підтримує quantized-моделі (Q4_K_M, Q5, Q6 та ін.) – економія ОЗП. По суті, цей сервер може обслуговувати запити як самостійно, так і через KoboldAI чи будь-який інший OpenAI-сумісний інтерфейс.

Вбудований API-сервер надає локальний REST API, який можна підключити до: KoboldAI, LM Studio, Text Generation WebUI, LangChain, LlamaIndex, або будь-якої програми з підтримкою OpenAI-style API.

Наприклад, для інтеграції з KoboldAI треба запустити KoboldCpp як API-сервер: /koboldcpp – –model mistral.gguf – –port 5001 – –api. У KoboldAI вибереться рушій – Connect to external KoboldCpp. Усі запити з інтерфейсу KoboldAI будуть оброблятися KoboldCpp

Він побудований на основі llama.cpp з підтримкою досить нового формату GGUF (всі метадані + tokenizer в одному файлі). Він має повну сумісність з популярними моделями: LLaMA 1/2/3, Mistral, Mixtral, OpenHermes, Nous Hermes, Mythalion, Dolphin, а також кастомні fine-tuned GGUF-моделі.

Багатопотокова генерація забезпечує повну підтримку OpenMP, POSIX Threads (pthreads) або Microsoft Threads для багатоядерної генерації. Параметр --threads визначає кількість потоків, що використовуються при генерації тексту. Це дозволяє запускати навіть 7B/13B моделі з комфортною швидкістю на звичайних ПК.

KoboldCpp також працює в інтерактивному режимі безпосередньо з командного рядка без запуску сервера, дозволяючи використовувати його без графічного інтерфейсу, так само, як і llama.cpp. Для цього використовується прапорець --cli, щоб увійти в режим терміналу.

У новій версії KoboldCppp передбачено підтримку створення локальних зображень, завдяки stable-diffusion.cpp. Він надає кінцеву точку txt2img, сумісну з ComfyUI та A1111, яку користувач може використовувати у вбудованому Kobold Lite або в багатьох інших сумісних інтерфейсах, таких як SillyTavern. Для цього вибирається сумісна модель SD3, Flux, SD1.5 або SDXL для завантаження через панель запуску графічного інтерфейсу або за допомогою методу .safetensors --sdmodel. В даному варіанті запуску VAE та LoRA повинні бути вбудовані всередині самої моделі. Рекомендується використовувати fp16. --sdvae --sdvaeauto --sdlora. Актуальна версія підтримує прапорці, що наведені на рис. 2.7.

Kobold Lite – це мінімалістичний, швидкий і портативний вебінтерфейс для локального використання LLM, розроблений як спрощена альтернатива

KoboldAI. Основна мета – надати зручний браузерний інтерфейс без складної інсталяції, з можливістю запуску моделей на CPU через KoboldCpp або llama.cpp. Kobold Lite сам не виконує моделі, а підключається до API-серверів, таких як: KoboldCpp, сервер llama.cpp, Ollama (обмежено), GPT4All з REST API. Потрібно просто вказати URL локального рушія (наприклад, http://localhost: 5001/api).

```
--sdmodel      Specify a stable diffusion model to enable image generation.
--sdthreads    Use a different number of threads for image generation if specified.
--sdquant      If specified, loads the model quantized to save memory.
--sdclamped    If specified, limit generation steps and resolution settings for shared
use. Optionally allows setting max size.
--sdlora       If specified, tries to load a Stable Diffusion LoRA model
--sdloramult   Set the LoRA multiplier
--sdvae        Set a custom VAE
--sdvaeauto    Use built in fallback VAE (TAESD)
--sdnotile     Disable VAE tiling
--sdt5xxl      For Flux and SD3, you need to load a T5-XXL language model as well. Some
files bundle it with the main model.
--sdclipl      For Flux and SD3, you need to load a Clip-L model as well. Some files
bundle it with the main model.
--sdclipg      For SD3, you need to load a Clip-L model as well. Some files bundle it
with the main model.
```

Рисунок 2.7 – Варіанти запуску моделі в KoboldCpp

В цілому, інтерфейс має декілька функцій:

- поле чату – для інтерактивного спілкування з LLM;
- історія – зберігає всі попередні відповіді;
- системні інструкції – можна вказати поведінку моделі (як «system prompt»);
- налаштування генерації: Temperature, top-k, top-p, repetition penalty, максимальна довжина відповіді;
- підтримка Persona / Memory (частково).

Kobold Lite підтримує моделі HuggingFace (через Transformers).

Таким чином, KoboldAI – найбільш гнучкий і потужний, але важкий; KoboldCpp – найлегший і найшвидший, особливо для CPU та слабких машин; Kobold Lite – застаріла полегшена версія, зараз майже не використовується. Тобто, якщо використовується Raspberry Pi або ПК без GPU – KoboldCpp краще. Якщо є потужний ПК а робоча станція чи сервер і потрібна гнучкість –

KoboldAI. Надалі доцільно визначити особливості актуальних форматів файлів сучасних LLM.

2.4 Особливості форматів GGML і GGUF

GGML і GGUF (GPT Generated Unified Format) позначають однакову концепцію, причому GGUF – це новіша версія, що включає додаткові дані про модель. Це покращення дозволяє краще підтримувати різні архітектури та включає шаблони запитів. GGUF можна запускати лише на CPU або частково/повністю переносити на GPU. Використовуючи k-квантування, GGUF може змінюватись від 2 до 8 бітів. В даному випадку, раніше GPTQ служив методом квантування, оптимізованим лише для GPU. Однак його перевершив AWQ, який приблизно вдвічі швидше. Останнє досягнення в цій галузі – EXL2, який забезпечує ще кращу продуктивність. Зазвичай, ці методи квантування реалізуються з використанням 4 бітів.

GGUF більш сучасний, ніж GGML, є більш зрозумілим, розширюваним, універсальним та здатним включати нову інформацію без шкоди для сумісності з попередніми моделями. Розглянемо основні типи квантизації GGUF, за якими з назви LLM можна легко визначити, чи підходить вона чи ні:

– Q2_K, Q3_K_S/M/L – найменший розмір, висока швидкість, значна втрата якості;

– Q4_K_S/M – хороший баланс розміру та якості;

– Q5_K_S/M – покращена якість порівняно з Q4, незначне зниження швидкості;

– Q6_K – висока якість, помірна швидкість;

– Q8_0 – найвища якість, найнижча швидкість.

На даний час, існують такі рекомендації щодо вибору GGUF моделей:

– 4-6 ГБ VRAM – моделі Q3_K_M або Q4_K_S;

– 8 ГБ VRAM – моделі Q4_K_M або Q5_K_S;

– 12-16 ГБ VRAM – моделі Q5_K_M або Q6_K;

– 24 ГБ VRAM – моделі Q6_K або Q8_0.

Підсумовуючи, можна сказати, що перевагами GGUF перед GGML є такі положення.

Велика гнучкість – GGUF забезпечує більшу гнучкість у зберіганні даних моделі, що спрощує додавання нових функцій та адаптацію до різних архітектур моделі.

Найкраща сумісність – GGUF розроблено з урахуванням більшої сумісності з ширшим спектром інструментів та бібліотек, що спрощує робочий процес для розробників.

Підвищена ефективність – GGUF може забезпечити більшу ефективність з точки зору розміру файлу та швидкості завантаження, що може призвести до підвищення продуктивності LLM.

Стандартизація – GGUF прагне стати єдиним форматом, що сприяє взаємодії між різними інструментами та моделями.

РОЗДІЛ 3
РЕКОМЕНДАЦІЇ ЩОДО ЛОКАЛІЗАЦІЇ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ

3.1 Особливості роботи з моделями GGUF у KoboldCpp на прикладі Mixtral 8×7B

У KoboldCpp підтримуються всі сучасні моделі GGUF. ПЗ також включає зворотну сумісність зі старими версіями (застарілими моделями GGML), хоча деякі нові можливості можуть бути недоступними. Загалом, якщо це модель GGUF, KoboldCpp має працювати з: Llama, Llama2, Llama3, Alpaca, GPT4All, Vicuna, Koala, Pygmalion, Metharme, WizardLM, Mistral, Mixtral, Miqu, Qwen, Qwen2, Yi, Gemma, Gemma2, GPT-2, Cerebras, Phi-2, Phi-3, GPT-NeoX, Pythia, StableLM, Dolly, RedPajama, GPT-J, RWKV4, MPT, Falcon, Starcoder, Deepseek та є багато сотень інших моделей GGUF. Найкраще місце для отримання текстових моделей GGUF – це Huggingface [19]. Вони повинні бути файлом у форматі .ggml, .gguf або .bin. Великий вибір високоякісних моделей також можна знайти у репозиторії Huggingface від TheBloke [20]. Нарешті, також можна перетворити моделі самостійно, використовуючи відповідні інструменти квантування та перетворення [21]. Для іміджевих моделей у CivitAI є непоганий вибір, які допоможуть почати роботу з KoboldCpp:

– розпізнавання мови – моделі Whisper для перетворення мовлення в текст [22];

– перетворення тексту в мову – моделі TTS для дикторського супроводу [23];

– іншими хорошими моделями генерації тексту, які варто спробувати, є L3-8B-Stheno-v3.2 [24] і Fimbulvetr-11B-v2 [25];

– генерація зображень: будь-що v3 або Deliberate V2 [26], або Dreamshaper SDXL [27];

– Image Recognition MMproj – Виберіть правильний варіант для архітектури вашої моделі у [28];

– швидка та проста текстова модель для початку – це Airoboros Mistral 7B [29] (менша і слабша) або Tiefighter 13B [30] (більша модель), або Beepo 22B [31] (найбільша і найпотужніша);

Надалі розглянемо модель Mixtral-8x7. Вона розроблена французькою к. Mistral AI. Вона поєднує в собі ефективність та високу продуктивність завдяки використанню архітектури Sparse Mixture of Experts (SMoE) – розрідженої суміші експертів. У традиційних LLM кожен токен обробляється всіма параметрами моделі, а підхід SMoE – для кожного токена маршрутизатор вибирає лише двох з 8-ми експертів, які обробляють токен, а їхні результати об'єднуються. Цей підхід дозволяє:

– зменшити обчислювальні витрати та споживання пам'яті;

– зберегти або навіть покращити якість результатів завдяки спеціалізації експертів;

– забезпечити масштабованість моделі без збільшення ресурсів.

Тобто, завдяки розрідженій архітектурі, Mixtral 8×7B забезпечує продуктивність, порівнянну з моделями, які мають значно більше активних параметрів, але з меншими обчислювальними витратами. Загальна кількість параметрів моделі складає 46,7 мільярдів, з яких активні параметри на токен – лише 12,9 мільярдів, завдяки активації лише двох з 8-ми експертів для кожного токена. Контекстне вікно – до 32768 токенів, що дозволяє працювати з довгими текстами. Модель пропонується лише з декодером, без енкодера. Модель важить близько 30 ГБ, залежно від квантування. У чистому вигляді від такої моделі користі небагато. Можна поговорити на різні пристойні теми, попрактикувати англійську. Ще модель «читала» вікіпедію та документацію до популярних бібліотек, і непогано відповідає на новачкові питання щодо них. Може видати з пам'яті приклад коду, що реалізує ту чи іншу задачу. Або анекдот на задану тему загалом все. Але головне - використовувана програма має REST API, що повністю повторює таку від OpenAI, а значить, її можна

використовувати замість облікового запису OpenAI з будь-якою іншою програмою, яка дозволить ввести свою адресу до API. Ось тут уже починається варіант можливостей: там і аналіз документів, і інтеграція з IDE та багато всього іншого.

До вказаного обсягу треба додати 10 ГБ на технічні потреби. І скільки там ще потрібно для ОС та браузера. Цей сумарний розмір потрібно розкидати між усіма відеокартами Nvidia та оперативною пам'яттю комп'ютера. Можна просто засунути все в 64 ГБ оперативної пам'яті. Це буде повільно, але все одно цілком працездатне. Краще призначити частину завдань відеокарти Nvidia. Тоді оперативної пам'яті вистачить 32 ГБ. Ще краще виконувати всі обчислення в ній, але для цього потрібно 1-2 карти з сумарною пам'яттю в 48 ГБ (відеокарти лише від к. Nvidia). Для карт AMD є плагін сумісності, але віна працює тільки в Linux.

Mixtral-8×7B-Instruct-v0.1-GGUF модель розміщена на ресурсі [32], на сторінці якого є список варіантів квантизації. На даний час, вибирати варто лише між Q-4-M або Q-5-M. Друга важить трохи більше, думає трохи довше і дає результати трохи краще. А решта варіантів або зламані, або неоптимальні. Для інших моделей Q-3-M буде гарним вибором.

Якщо для KoboldCpp потрібна підтримка CUDA (підтримка відеокарт Nvidia) – треб вибрати необхідний варіант версії. Нарешті, файли KoboldCpp і Mixtral-8x7B складаються в одну папку. Якщо є можливість скористатися відеокартою, то необхідно виконати таку послідовність. Відкрити панель налаштування Nvidia → Manage 3D Settings → Program Settings (рис. 3.1). Натиснути кнопку «Add» та вибрати файл Koboldcpp.exe. Коли він вибраний у списку 1, серед налаштувань внизу знайти CUDA – Sysmem Fallback Policy і вибрати «Prefer No System Fallback», натисніть «Застосувати». Якщо цього не зробити, при неправильних налаштуваннях відеопам'яті KoboldCpp може обрушити ОС або заблокувати відеопам'ять до перезавантаження. Глобальне встановлення цієї опції виставляти не треба.

KoboldCpp не вимагає інсталяції на ОС Windows – це один файл EXE. На жаль, він не запам'ятовує деякі налаштування між запусками. Як вихід, його простіше запускати з командного рядка (термінал запам'ятає команди користувача). Тому відкривають у провіднику теку, куди все завантажили, клацають за адресою в заголовку та вводять замість нього pwsh (у разі помилки – powershell). Відкриється вікно командного рядка з обраною текою як поточної.

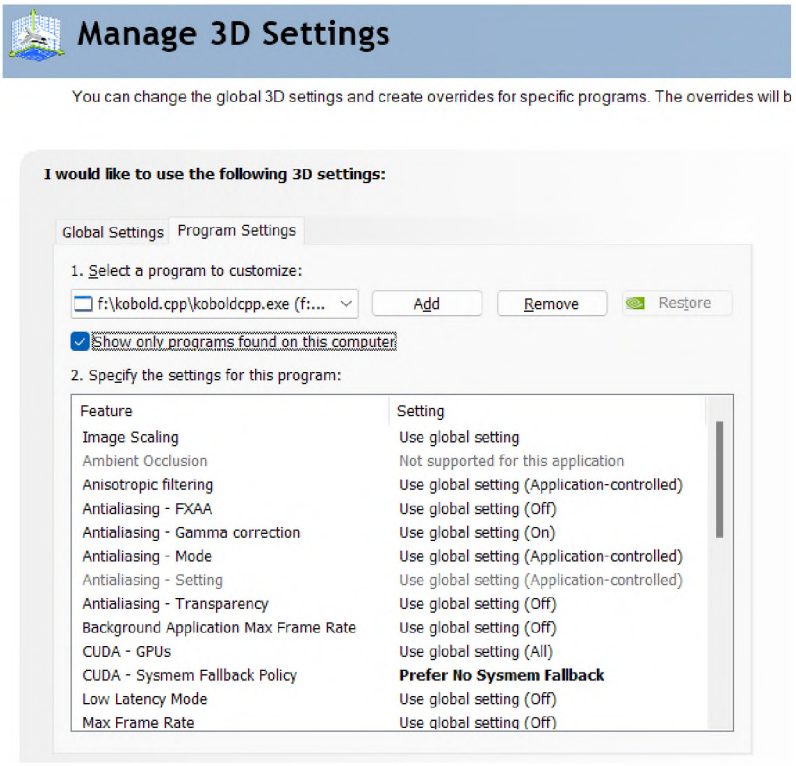


Рисунок 3.1 – Налаштування відеокарти Nvidia

Далі вводять команду:

.\koboldcpp.exe – –model .\mixtral – 8x7b – instruct – v0.1. Q4_K_M. gguf

Це запустить KoboldCpp без використання GPU. Щоб скористатися ними, до команди додають параметри: – – usecublas – –gpulayers 23. В даному випадку, 23 – це кількість шарів моделі, які потрібно завантажити на відеокарту. У Mixtral всього 33 шари, вони різного розміру, але так співпало, що гарним стартовим значенням буде об'єм відеопам'яті в ГБ мінус 1 (наприклад, відеокарта має обсяг 24 ГБ). Якщо вказати занадто багато, то буде

виникати CUDA Error або просто зависання ПЗ. Також має сенс додати параметр – – contextsize 4096. Розмір контексту – це, грубо кажучи, скільки складів розмови модель пам'ятає при відповіді. Число має бути ступенем двійки (до 32768 включно). Надалі запускають команду на перезевантаження. Воно має досить довго тривати, а потім вивести в консоль багато рядків тексту, з яких важливими є наступні:

– llm_load_tensors: using CUDA for GPU acceleration – визначає, що відеокарта успішно використовується;

– llm_load_tensors: VRAM used: 19564.84 MB – цей рядок виводиться перед тим, як програма почне закачувати модель у відеопам'ять.

Якщо написаний розмір більше, або трохи менше, ніж є відеопам'яті, треба негайно закрити консоль, інакше це виводить з ладу. По суті, що на відеокарту перенесено надто багато шарів. Відразу після цього рядка все може «замислитись» на пару хвилин (це нормально). Наступний рядок:

llama_new_context_with_model: total VRAM used:
20293.85 MiB (model: 19564.84 MiB, context: 729.00 MiB)

вказує, скільки пам'яті розмічено під модель і контекст.

Останній рядок:

Please connect to custom endpoint at http://localhost: 5001

означає, що все запрацювало. Затиснувши Ctrl, натискають на адресу. Повинний з'явитись інтерфейс KoboldUI (рис. 3.2).

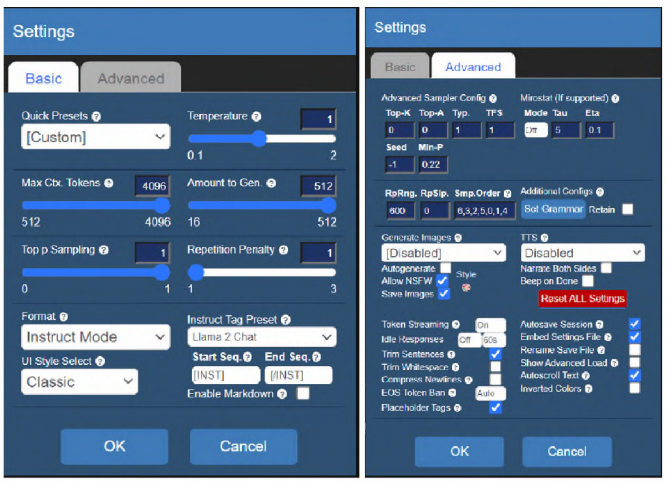


Рисунок 3.2 – Інтерфейс KoboldUI для налаштування ПЗ

Тепер потрібно ввести налаштування моделі Mixtral. Розмір контексту повинен дорівнювати тому, що був вказаний у командному рядку. Температурою можна спробувати варіювати в діапазоні від 1 до 2.

Решту міняти не варто. Справа в тому, що у випадку Mixtral усі спроби для скорочення числа повторів лише дають більше повторів, тому їх треба вимкнути. Ну і нарешті можна скористатися самою моделлю. Натисніть нагорі Scenario (рис. 3.3) → New Instruct і вводять свій запит.

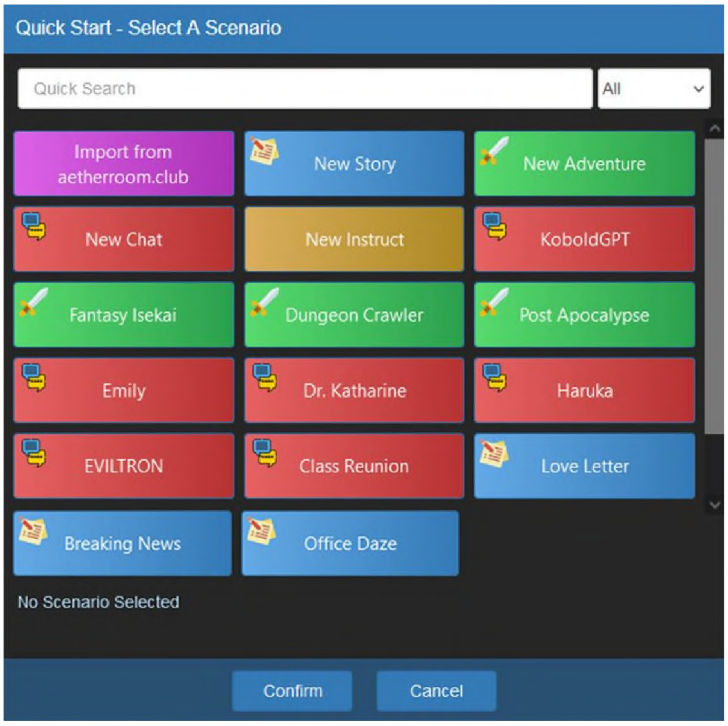


Рисунок 3.3 – Інтерфейс KoboldCpp для вводу запиту

Як відомо параметр Temperature – керує ступенем випадковості в тексті, що генерується ШІ:

– temperature = 0.0 – дуже передбачувано, максимально «логічно;

– temperature = 1.0 – баланс між передбачуваністю та креативністю;

– temperature > 1.0 – більше креативу, але більше хаосу.

Тобто, якщо потрібно щось точне та технічне, температуру краще не чіпати. Якщо ж хочеться більше творчості або «людського стилю», можна її трохи підвищити.

3.2 Використання KoboldCpp в проєктах Edge AI

На даний час, широко поширення набули рішення Edge AI, одним з яких є створення локального LLM-асистента. Зазвичай, при цьому орієнтуються на використання одноплатних мікрокомп'ютерів, наприклад, Raspberry Pi 5 (рис. 3.4) [33].

Рисунок 3.4 – Мікрокомп'ютер Raspberry Pi 5

Він має кілька версій, що відрізняються обсягом ОЗП. Для рішень Edge AI з LLM доцільно орієнтуватись на значення 8 або 16 ГБ ОЗП. При цьому використовуються квантизовані моделі, що можна класифікувати. Для Raspberry Pi 5 з 8 ГБ можна запустити модель Mistral-7B Q4 [34], досягши швидкості виводу 2,4 токенів за секунду. Якщо потрібна вища швидкість генерації, використовують моделі меншого розміру, таких як Phi-3.5 (3.8B) [35] – швидкість виводу близько 3,42 токенів за секунду. При цьому рекомендується активне охолодження (наприклад, вентилятор або радіатор) для підтримки стабільної температури процесора та використання SSD через USB 3.0 або NVMe через PCIe (рис. 3.5) для швидшого доступу до даних [36].

Для деяких проєктів рівень вказаних моделей є недостатнім. Щоб використовувати більші моделі потрібно збільшувати обсяг ОЗП. Але для Raspberry Pi 5 – межа 16 ГБ ОЗП [37]. Зняти це обмеження можна за рахунок SWAP. Це область на накопичувачі (SSD або SD-карті), яка використовується як резервна пам'ять, коли ОЗП пристрою закінчується. Існує кілька варіантів

організації swap, кожен з яких має свої плюси, мінуси та застосування для Raspberry Pi 5 з ОС на основі Linux (табл. 3.1).

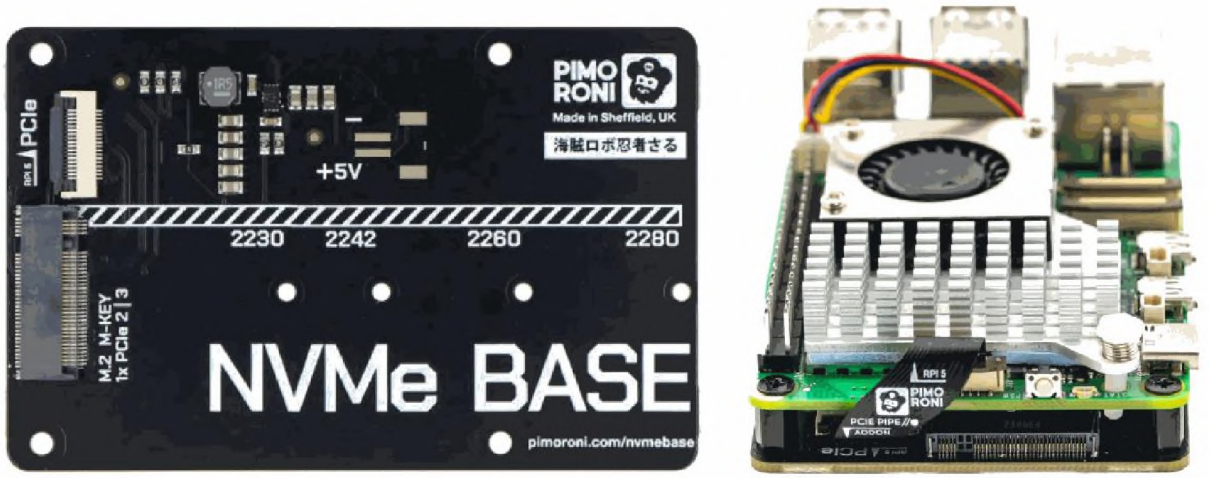


Рисунок 3.5 – Плата NVMe для підключення SSD через PCIe

Таблиця 3.1 – Варіанти SWAP для LLM на ОС Linux (Raspberry Pi)

Варіант	Опис	Швидкість	Надійність	Ресурс диску
Файл swap	Файл заданого розміру в файловій системі	Середня	Висока	Високий знос SD
Розділ swap	Окремий розділ диску	Середня	Висока	Високий знос SD
Zram (стиснутий у RAM)	Стискає дані та зберігає у RAM	Висока	Середня	Без зносу
4. zswap	Компресія у RAM + запис у swap-файл	Добра	Висока	Середній знос
SSD swap (USB 3.0/PCIe)	Зовнішній SSD як swap	Висока	Висока	Менший знос, ніж SD
Tmpfs (тільки RAM)	Псевдо-swap у RAM	Дуже висока	Лише при надлишку RAM	Без зносу

В даному випадку, можна використовувати варіант SSD swap. В свою чергу, при застосуванні KoboldCpp, його продуктивність, гнучкість і локальна автономність роблять його надійним компонентом для інтеграції ШІ без залежності від хмари. При його використанні в комбінації з Raspberry Pi має певні переваги, наприклад, ПЗ написане на C++ (немає потреби в Python), підтримує квантизовані GGUF-моделі, які займають меншій обсяг ОЗП, може

працювати прямо на CPU (без потреби у GPU чи зовнішніх прискорювачів), має легкий вебінтерфейс, подібний до KoboldAI.

Як наслідок, в роботі запропонована конфігурація для запуску KoboldCpp + Mixtral на Raspberry Pi 5 (16 GB RAM) для проєкту локального LLM-асистента (Додаток А). При цьому, для підвищення його функціоналу KoboldCpp автоматично стартує при запуску Raspberry Pi і працює у фоновому режимі як справжній LLM-сервер. Також розглянута можливість голосового вводу (тобто, функці Speech-to-Text) у локального LLM-асистента.

3.3 Техніко-економічне обґрунтування прийнятих рішень

Генеративні нейромережі для створення зображень кардинально змінили ринок ПЗ. Тепер їх можна зустріти будь-де – на сайтах, у месенджерах, стоках та графічних редакторах. Завдяки інтересу користувачів та конкуренції між корпораціями прогрес стрімко пішов уперед, а в ПЗ стало з'являтися все більше корисних функцій, що полегшують життя, роботу та творчість. Наприклад, у версії Adobe Photoshop додали генеративну заливку з урахуванням вмісту, яка замінює, розширює або доповнює новими об'єктами будь-яку частину редагованих фотографій, новий інструмент для видалення об'єктів, покращили виділення фону та людей. Все це стало можливим через суперництво компаній, бажання інвесторів вливати кошти в будь-які перспективні розробки, пов'язані з ШІ, а також масового використання локальних нейромереж, що буквально змусили Adobe почати гонку за користувачів. І це далеко не поодинокий приклад. Подібне відбувається з LLM Поява ChatGPT і GPT-4 спонукала корпорації на створення аналогів, а поширення локальних LLM зробило їх доступними для звичайних користувачів. Текстові нейромережі починають впливати на наше життя, з'являючись у браузерах, програмах, ботах в месенджерах. Більшість із них – з закритою інфраструктурою, що обмежує можливості передплатників. Нові локальні LLM з'являються майже кожен

день, кидаючи виклик гігантам індустрії, відкриваючи перед користувачами нові горизонти, допомагаючи автоматизувати рутинні завдання, вивільняючи час для творчості, покращуючи та переробляючи спірні шматочки тексту. Перспективи використання LLM – масштабні, наприклад: вести соціальні мережі; відповідати на запити та запитання передплатників; створювати новели, оповідання, формувати шаблони листів, рекламних постів, розсилок; перекладати текст з однієї мови на іншу; шукати помилки у публікаціях; узагальнювати довгі статті; вигадувати описи для персонажів, зображень, карток товарів; запускати чат-ботів у месенджерах; генерувати діалоги для комп'ютерних ігор та сценарії для рольових ігор живої дії; завершувати пропозиції, якщо ви вперлися у стіну і не знаєте, як продовжити розповідь; спрощувати складні фрази без втрати смислу; поліпшувати інші мовні моделі; коригувати портфоліо чи резюме; генерувати нові гіпотези та ідеї; створювати простий програмний код; оптимізувати статті для seo; розробляти бізнес-плани та модифікувати торгові пропозиції; шукати нові сфери діяльності чи незвичайні варіанти просування своїх послуг; генерувати кістяк тексту для лонгрідів та коротких постів для соцмереж; аналізувати фінансові звіти; шукати та заповнювати проблемні місця у складних публікаціях; використовувати LLM як віртуального помічника за допомогою KoboldCpp або GPT4all; виконувати домашні завдання; писати звіти, реферати, дипломні роботи та книги, а також сценарії для Youtube-каналів; розробляти онлайн-курси, гайди, уроки; аналізувати поведінку користувачів на сайті, у месенджерах чи соцмережах; робити прогнози на основі наявних даних; створювати розважальний контент будь-якої спрямованості. Це лише невелика частина прикладів, що показує, як можна використовувати LLM для вирішення практичних завдань. Користувач може вигадати абсолютно інші способи використання LLM, щоб поліпшити свою діяльність, обмежитися створенням розважального або рекламного контенту, або зачекати на подальший розвиток нейромереж. Перспективи розвитку технологій різноманітні, а граничні можливості генерації тексту в нейромережах – невідомі. В цілому, щоб

реалізувати запропонованого в п. 3.2 локального LLM-асистента з можливістю голосового вводу-виводу знадобиться обладнання, що наведене в табл. 3.2. Для оптимізації витрат, вибраний корпус з вбудованим NVMe (рис. 3.6) [38].

Таблиця 3.2 – Вартість обладнання для локального LLM-асистента на базі KoboldCpp + Mixtral на Raspberry Pi 5

№ з/п	Назва обладнання	Вартість, грн
1	Raspberry Pi 5 16 ГБ	6999
2	Блок живлення Waveshare для Raspberry Pi 27Вт USB-C	498
3	Корпус з активним охолодженням Argon NEO 5 M.2 NVME	1890
4	SSD M.2 256 ГБ	1950
5	Звукова карта ORICO SC2 3xAUX	425
6	Мікрофон 3.5 мм	199
7	Динамік серії QIV QBR-A 1000	166
	Разом:	12127

Рисунок 3.6 – Корпус з NVME

Через те, що KoboldCpp є open-source, а програмний код для конфігурації локального LLM-асистента не перевищує 100 рядків, витрати на ПЗ не враховуються. Таким чином, загальні витрати на реалізацію локального LLM-асистента на основі моделі Mixtral без необхідності доступу в Інтернет та можливістю голосового вводу/виводу становлять ≈ 12200 грн.

ВИСНОВКИ

На основі LLM-агентів будуть проходити великомасштабні зсуви у самих технологіях, способах мислення, взаємодії з реальністю та перепідпорядкування галузей застосування ШІ.

LLM-агенти мають інтерфейси основі поділяються на API і GUI, кожен з яких має певні переваги та недоліки. API-агенти ефективніші, надійніші та безпечніші, але менш гнучкі. GUI-агенти – більш універсальні та людиноподібні, однак чутливі до змін інтерфейсу. Оптимальним варіантом стає гібридний підхід з локальним розгортанням LLM. Він забезпечує автономність, безпеку, відповідність нормам захисту даних і мінімізацію залежності від сторонніх API. Хоча GUI-агенти мають певні обмеження, їх ефективність зростає завдяки мультимодальним моделям.

Інструменти для локального запуску LLM забезпечують автономність, безпеку та ефективність. Вони підтримують квантування, працюють без GPU і сумісні з різними ОС. Завдяки REST API, GUI та інтеграції з LangChain, ці рішення дозволяють гнучко застосовувати LLM у персональних і професійних проєктах без залежності від хмари.

KoboldCpp і GPT4All мають різні цільові аудиторії, але обидва забезпечують зручну локальну роботу з LLM. KoboldCpp – оптимальний для творчих завдань, підтримує широке коло моделей і сценаріїв. У сімействі Kobold найефективнішим для слабких ПК і CPU визнано KoboldCpp – легкий, автономний рушій із підтримкою GGUF, багатопотоковості та інтеграції з популярними інтерфейсами. Він не потребує Python, забезпечує запуск моделей без хмари, має REST API та CLI-режим. KoboldAI пропонує гнучкість і розширення, але складніший у налаштуванні. Kobold Lite – спрощена оболонка, що залежить від зовнішніх рушіїв і рідко використовується в сучасних сценаріях.

Формат GGUF є еволюцією GGML і пропонує розширену підтримку архітектур, кращу сумісність з інструментами та зручність у зберіганні

метаданих. Завдяки різним типам квантування (від Q2_K до Q8_0) він дозволяє ефективно балансувати між якістю, швидкістю та розміром моделі. GGUF забезпечує гнучкість, продуктивність і стандартизацію, що робить його оптимальним вибором для локального запуску LLM.

KoboldCpp підтримує широку гаму GGUF-моделей, зокрема Mixtral 8×7B, яка поєднує високу якість генерації з ефективністю завдяки архітектурі Sparse Mixture of Experts. Завдяки REST API та підтримці CUDA, KoboldCpp дозволяє інтеграцію з іншими системами та прискорення на відеокартах. Правильне налаштування параметрів запуску забезпечує стабільну роботу навіть на середніх ПК, роблячи KoboldCpp зручним інструментом для локального використання LLM.

Використання KoboldCpp на Raspberry Pi 5 у проєктах Edge AI забезпечує ефективний локальний запуск LLM без залежності від хмари. Завдяки підтримці GGUF-моделей, роботі без GPU та можливості запуску в середовищах із 8-16 ГБ ОЗП, ця конфігурація є придатною для створення компактного LLM-асистента. Застосування SWAP, SSD і оптимізація охолодження дозволяють розширити можливості системи навіть для більших моделей. Розвиток генеративних нейромереж і поява доступних локальних LLM, зокрема на базі KoboldCpp і Mixtral, відкрили широкі можливості для автоматизації, творчості та особистого використання. Реалізація локального LLM-асистента на Raspberry Pi 5 з голосовим інтерфейсом коштує приблизно 12200 грн, забезпечуючи автономність, гнучкість і функціональність без залежності від хмарних сервісів і обмежень.

Таким чином, результатами роботи є рекомендації щодо локалізації великих мовних моделей. Вони можуть бути використані для подальших досліджень за даною тематикою та при проектуванні локальних сервісів NLP.