УДК 004.58
DOI https://doi.org/10.32689/maup.it.2025.1.10

Владислав ІВАНОВ
студент кафедри інженерії програмного забезпечення,
Івано-Франківський національний технічний університет нафти і газу,
vlad.ivanov.if@gmail.com
***ORCID:** 0009-0000-3788-9526*

Лідія ГОБИР
асистент кафедри інженерії програмного забезпечення,
Івано-Франківський національний технічний університет нафти і газу,
lidagobyr@gmail.com
***ORCID:** 0009-0007-3176-2314*

Тетяна ВАВРИК
асистент кафедри інженерії програмного забезпечення,
Івано-Франківський національний технічний університет нафти і газу,
vavruk1060@gmail.com
***ORCID:** 0000-0002-0612-0084*

ПЕРСПЕКТИВИ РЕАЛІЗАЦІЇ АВТОНОМНОГО УКРАЇНОМОВНОГО ГОЛОСОВОГО АСИСТЕНТА НА ОСНОВІ МОВНОЇ МОДЕЛІ ТА ТЕХНОЛОГІЙ РОЗПІЗНАВАННЯ Й СИНТЕЗУ МОВЛЕННЯ

***Анотація. Метою статті** є аналіз сучасних підходів та технічних можливостей реалізації повноцінного україномовного голосового асистента для задоволення потреби в автономності, конфіденційності та можливостях його персоналізації для забезпечення гнучкості налаштування під конкретні вимоги цільових споживачів чи бізнесів. У роботі розглянуто проблеми наявних рішень на прикладі відомих хмарних платформ та підкреслено необхідність у розробці незалежних та автономних аналогів. Проведено аналіз найвідоміших доступних відкритих технологій розпізнавання мовлення, опрацювання тексту з отриманням відповіді та синтезу людино-подібного мовлення для виділення таких, що забезпечують високу якість оброблення за відносно низьких затрат ресурсів, здатні працювати з різними мовами, у тому числі українською, а також можуть бути використані для реалізації демонстраційного застосунку. Проаналізовано методики та техніки зменшення вимог до апаратного забезпечення кінцевої системи для забезпечення ефективної роботи системи в середовищах з лімітованими ресурсами. Окрема увага відводилася оптимізації виконання математичних операцій інференції моделей завдяки використанню можливостей апаратного прискорення окремих обчислювальних платформ. Окрім того, розглянуто особливості інтеграції кожного компонента в єдину систему на базі мікросервісної архітектури з можливостями адаптації цих інструментів під специфічні потреби користувачів.*

***Наукова новизна** полягає в систематизації відомостей про технології та засоби, придатних для створення подібного роду асистентів без використання хмарних сервісів та з поєднанням різних оптимізаційних технік для розгортання подібних систем на пристроях побутового рівня. У статті обґрунтовано, що за умов поєднання компонентів з мінімальними затримками взаємодії, покращення продуктивності та якості опрацювання без значного збільшення вимог до ресурсів, а також грамотної реалізації системи, що об'єднуватиме всі компоненти, можна отримати конкурентоспроможного автономного голосового асистента, придатного до інтеграції в будь-які системи завдяки можливостям відкритої платформи.*

***Ключові слова:** велика мовна модель, синтез мовлення, автоматичне розпізнавання мовлення, голосовий асистент.*

Vladyslav IVANOV, Lidiia HOBYR, Tetiana VAVRYK. DISCUSSING PERSPECTIVES OF DEVELOPMENT OF AN OFFLINE UKRAINIAN-SPEAKING LLM-BASED ASSISTANT INTEGRATED WITH SPEECH SYNTHESIS & RECOGNITION TECHNOLOGIES

***Abstract. The purpose** of the article is to analyze modern approaches and technical possibilities for implementing a full-fledged Ukrainian-speaking voice assistant to meet the need for autonomy, confidentiality, personalization and to ensure the flexibility of customization to the specific requirements of target consumers. The paper considers the problems of existing solutions on examples of well-known cloud platforms and emphasizes the need to develop independent and autonomous analogs. An analysis of known open-source projects for speech recognition, text-to-speech and human-like speech synthesis was conducted to identify those that provide high quality processing at relatively low resource costs, are capable of working with different languages, including Ukrainian, and can be used to implement a demo application. Methods and techniques for reducing the hardware requirements of the end system to ensure efficient operation of the system in resource-limited environments are analyzed. Particular attention was paid to optimizing the performance of mathematical operations of model inference by using the hardware acceleration capabilities of individual computing platforms. In addition, the peculiarities of integrating each component into a single system based on a microservice architecture with the ability to adapt these tools to the specific needs of users are considered.*

The scientific novelty *lies in the systematization of information on technologies and tools suitable for creating such assistants without using cloud services and with a combination of various optimization techniques for deploying such systems on consumer-level devices. The article proves that if the components are combined with minimal interaction delays, performance and processing quality are improved without a significant increase in resource requirements, and the system is properly implemented to combine all components, a competitive autonomous voice assistant capable of being integrated into any system thanks to an open platform can be implemented.*

Key words: *large language model, speech synthesis, automatic speech recognition, voice assistant.*

Вступ та постановка проблеми. Досягнення у сфері технологій штучного інтелекту в останні роки відкрили нові можливості для інтеграції різного роду асистентів у повсякденне життя. Чат-боти на основі великих мовних моделей (Large Language Model) дозволили аналізувати та опрацьовувати великі об'єми інформації, генеруючи людино-подібний текст. Незважаючи на бурхливий розвиток та повсюдне впровадження таких моделей, їх інтеграція із системами розпізнавання та синтезу мовлення досі перебуває на ранньому етапі розвитку.

Існуючі голосові асистенти переважно розроблені як клієнт-серверне програмне забезпечення із закритим вихідним кодом і функціонують на комерційних засадах. Вони мають доступ до чутливої інформації споживачів, яка постійно передається на сервери компаній та часто використовується в цілях, що не стосуються роботи самого асистента. Окрім того, вони неспроможні опрацювати запити, складніші за прості команди, й обмежені в персоналізації. Виникає необхідність створення платформ з широкими можливостями налаштування та здатними працювати автономно.

Метою статті є огляд сучасних рішень для створення голосових асистентів, порівняння їх ефективності та дослідження їх взаємної інтеграції на прикладі створення автономного голосового асистента на їх базі.

Завданням статті є отримання відповідей на наступні питання:

1. Способи пристосування моделей до ресурсолімітованих середовищ.
2. Які є відкриті технології для опрацювання мовлення наживо?
3. Особливості взаємної інтеграції компонентів та їх персоналізація.

Огляд існуючих рішень у сфері опрацювання мовлення. Для досягнення локального розпізнавання, опрацювання та синтезу потрібні легкі моделі й оптимізовані програмні бібліотеки, щоб зменшити затримки через виконання математичних обчислень. Моделі повинні підтримувати потокову взаємодію, щоб передавати проміжні опрацьовані дані наступній моделі в ланцюгу. У цьому розділі виконується огляд наявних рішень та порівняння ефективності використання в обмежених середовищах.

Технологія перетворення звукових даних природньої мови в текст (Automatic Speech Recognition, Speech-to-Text) широко використовується у відомих голосових асистентах для створення нотаток, автоматичних субтитрів, транскрипції розмов тощо. Найвідомішим прикладом є Google Асистент, а його проблемою – необхідність доступу до інтернету через залежність від хмарної платформи Google. Серед автономних засобів розпізнавання за гнучкість у впровадженні та високу якість результатів виділяють DeepSpeech, Streaming Citrinet та ContextNet, Data2Vec та W2v-BERT 2.0, а також Whisper [10].

Streaming Citrinet – це модель для розпізнавання мовлення в реальному часі від NVIDIA, яка демонструє чудову точність та швидкість завдяки оптимізаціям для роботи на GPU з підтримкою CUDA. ContextNet також використовує CUDA та орієнтована на зменшення затримок. Її основною перевагою є ефективність у великих масштабах завдяки передбаченню контексту під час розпізнавання [8]. Ці та інші моделі вимагають відповідного програмного забезпечення, а також залежні від апаратного забезпечення NVIDIA. Mozilla DeepSpeech базується на архітектурі Baidu Deep Speech 2 і орієнтована на офлайн-розпізнавання та простоту інтеграції від Raspberry Pi 4 до потужних графічних серверів. Вона підтримує навчання на власних наборах даних, що дозволяє адаптувати модель до нових мов або акцентів. З недоліків виділяють низьку продуктивність у реальних акустичних умовах і значно вищі затримки порівняно з іншими рішеннями [3]. Meta Data2Vec – це універсальна модель, яка використовується для обробки тексту, мовлення та зображень. Для розпізнавання мовлення потребує тренування, але показує хороші результати. Недоліком є складність інтеграції для роботи наживо [14]. Схожа модель W2v-BERT 2.0 поєднує властивості моделей w2v і BERT, чим досягає високої точності в шумних умовах [15], однак є вимогливою до ресурсів. OpenAI Whisper – сучасна модель, яка вирізняється високою точністю розпізнавання мовлення різними мовами та в умовах шуму, об'єднує виявлення мовлення, транскрипцію та переклад і є гнучкою у використанні [12]. Для роботи потрібен GPU з підтримкою CUDA.

Описані технології розпізнавання мовлення здатні працювати автономно й можуть бути адаптовані під конкретні акустичні умови. З-поміж них можна виділити Streaming Citrinet та ContextNet за високу точність розпізнавання та Whisper за ефективність у співвідношенні вимог до ресурсів до якості роботи.

Великі мовні моделі в останні роки здійснили революцію у сфері контекстуальної обробки тексту. Відомі приклади – OpenAI GPT-4, Google Gemini, Anthropic Claude. Більшість платформ є закритими та не

дозволяють використання без підключення до мережі, збирають та аналізують користувацькі дані для подальшого покращення цих сервісів, а також мають обмеження на використання. Серед відкритих же платформ популярними є серії моделей загального призначення, а саме: Llama, Mistral та Gemma.

LLaMA (Large Language Model Meta AI) – серія моделей для обробки тексту з високою точністю і гнучкістю, які базуються на архітектурі трансформерів, компактні та оптимізовані для ефективного локального виконання [17]. Моделі підтримують широкий спектр мов і відомі точністю згенерованого тексту та якістю контекстного аналізу. Mistral – серія відкритих моделей, які досягають максимальної продуктивності на одиницю параметрів, а тому є менш вимогливими. Моделі є досить привабливими для дослідників, але відзначається чутливість до якості даних для тренування [7]. Gemma – модель, оптимізована для роботи на пристроях із малими обчислювальними ресурсами, базується на архітектурі трансформерів і показує високу продуктивність. Моделі серії Gemma 3 забезпечують конкурентну точність, послідовність думки в згенерованому тексті, але потребують адаптації для конкретних задач [5]. Кожен інструмент має свої переваги та недоліки, а вибір моделі залежить від вимог конкретного проєкту до точності, швидкості та ресурсоємності.

Синтез мовлення – перетворення тексту на мовний сигнал. Відомі рішення включають Google TTS та Microsoft Azure TTS, причому не всі можуть забезпечити роботу офлайн та похвалитися якістю синтезу. Проблеми цих платформ ті ж що й в аналогах для розпізнавання. Інструменти й моделі для використання офлайн, що демонструють високу якість синтезу, включають P-Flow TTS та RAD-TTS, eSpeak NG, Coqui TTS, і StyleTTS 2 [10].

NVIDIA P-Flow TTS – це модель для синтезу мовлення, яка використовує підхід потокової генерації для досягнення реалістичності синтезованого мовлення. Серед переваг висока якість звуку, підтримка багатомовності та можливість налаштування стилю [6]. RAD-TTS – інструмент, який дозволяє генерувати високоінтонаційне мовлення. Відзначається гнучкістю у стилізації мовлення та стабільністю синтезу навіть за складних буквосполучень. Як і P-Flow TTS є дуже вимогливою до апаратних ресурсів [9]. Обидві ці моделі оптимізовані для використання на графічних процесорах NVIDIA і добре інтегруються в екосистему CUDA. eSpeak NG – відкрита система синтезу, яка підтримує велику кількість мов, включаючи українську і не вимоглива до ресурсів. Якість синтезованого звуку значно поступається нейромережевим моделям, але її функціональні можливості часто утилізують інші інструменти як один із компонентів [4]. Coqui TTS – це відомий просунутий фреймворк для синтезу, який надає численні натреновані моделі та можливості створення власних. Відкритий вихідний код та підтримка спільноти, а також доволі висока якість синтезу за порівняно невеликих ресурсних затрат роблять його незамінним при використанні для вбудованих платформ [2]. Недавно представлена модель StyleTTS 2 поєднує текстову та стильову інформацію для створення реалістичного й емоційно забарвленого мовлення. Виділяється високою якістю синтезу та можливістю налаштування емоційного тону та швидкості мовлення. Вона оптимізована для роботи на GPU, що значно прискорює генерацію відповідей, але висока складність інтеграції лімітує легкість впровадження, хоча відзначають вражаючі результати синтезу [16].

Кожна з-поміж згаданих моделей має особливості налаштування й забезпечує певний рівень якості синтезу. Найкращих результатів сьогодні досягає StyleTTS 2. Вибір моделі для використання в конкретному випадку залежить від платформи розгортання й вимог до апаратного забезпечення.

Методики пристосування моделей до середовищ з лімітованими ресурсами. Для опрацювання мовлення необхідні великі обчислювальні ресурси. За допомогою різних засобів можна зменшити вимоги до системи або прискорити виконання обчислень. Одним із способів адаптації великих моделей до середовищ із обмеженими ресурсами є зменшення їх розмірів, задля чого використовуються методики обрізання, квантизація й дистиляція знань.

Обрізання моделі (pruning) передбачає видалення малозначущих шарів моделі. Метод широко застосовується у мовних моделях і моделях для обробки аудіо, наприклад, моделі Whisper доступні в різних варіантах з різною кількістю параметрів для досягнення певного компромісу між продуктивністю та розміром. Варто пам’ятати, що надто агресивне обрізання впливає на точність. Квантизація (quantization) зменшує розмір моделі, представляючи її ваги меншою розрядністю, наприклад, 8-бітною замість 32-бітної, чим значно знижує використання оперативної пам'яті, сховища та прискорює обчислення. Відомі моделі представлені в кількох варіантах з різними рівнями квантизації, а фреймворки PyTorch і TensorFlow підтримують пост-навчальну квантизацію, щоб зберегти точність. Дистиляція знань (knowledge distillation) передбачає навчання меншої «моделі-студента» на базі висновків великої «моделі-вчителя». Така техніка дозволяє передати ключові шаблони поведінки від більш складної моделі до простої, що знижує її вимоги до пам'яті та швидкості обчислень. До прикладу, моделі Gemma 3 1B та 4B були створені використовуючи попередньо натреновану модель розміром 27B (B – мільярд параметрів) [5]. Такі методики значно зменшують розміри моделей, прискорюють обчислення,

але погіршують якість. Відповіді мовної моделі, квантизованої до 2-бітної розрядності, страждають від порушень логічності викладення думок. Для виконання простих команд оптимальною є модель 4- та 8-бітної розрядності ваг, бо забезпечуються баланс між ресурсоємністю та якістю. Подальше покращення продуктивності можливе за оптимізації математичних обчислень під час інференції.

Інференція моделей у машинному навчанні – це процес використання готової моделі для здійснення прогнозів або прийняття рішень на нових даних. На цьому етапі слід розглядати інші методи прискорення обчислень, такі як спеціалізовані обчислювальні платформи, які забезпечують оптимізовану обробку математичних операцій: OpenBLAS, CUDA, ROCm та oneMKL.

OpenBLAS – відкрита бібліотека, оптимізована для лінійної алгебри, яка працює за рахунок багатоядерного центрального процесора (CPU). Вона підходить для проєктів, де GPU недоступний для зменшення енергоспоживання. Бібліотека ефективно виконує множення великих матриць, але у швидкості обробки великих масивів даних програє CUDA, яка є фреймворком для паралельних обчислень на графічних процесорах. CUDA-ядра сьогодні присутні в більшості сучасних відеокарт NVIDIA, чим забезпечується висока доступність платформи. AMD ROCm (Radeon Open Compute) – це платформа для високопродуктивних обчислень на GPU, яка підтримує паралельну обробку даних і оптимізована для роботи з архітектурами RDNA та CDNA. Платформа несумісна з операційною системою Windows, але підтримується більшістю сучасних фреймворків машинного навчання, що дозволяє використовувати її там, де вже використовується CUDA. Менш відома oneMKL оптимізована для роботи на процесорах Intel і забезпечує високу продуктивність у задачах лінійної алгебри. Вона також підтримує багатопотоковість, але її продуктивність на процесорах інших виробників значно гірша. Ідеально підходить для CPU-орієнтованих систем, однак значно програє CUDA.

Порівняння продуктивності платформ часто є недоречним, бо не всі вони є взаємозамінними. CUDA найбільш популярна завдяки широкій підтримці інструментів і високій ефективності на GPU NVIDIA. У свою чергу, ROCm надає можливість розробки на GPU AMD, забезпечує відкритий код і конкурентоспроможність, хоча менш поширений. OpenBLAS і oneMKL більше підходять для процесороцентричних завдань. У питанні продуктивності для машинного навчання лідирують CUDA та ROCm, але залежать від специфіки архітектури, а кінцевий вибір платформи визначається проєктом.

Побудова архітектури демонстраційної системи. Прототипування – це процес створення спрощеної, але функціональної моделі продукту, щоб швидко перевірити працездатність ідеї, провести тестування. У цьому розділі йтиметься про особливості реалізації демонстраційного проєкту.

Вибір технологій для створення голосового асистента значною мірою визначається особливостями взаємної інтеграції компонентів та можливостями їх налаштування під особливі варіанти використання. Основні модулі повинні працювати безперебійно, забезпечувати мінімальні затримки та високу якість взаємодії. Для цього необхідно враховувати сумісність моделей, оптимізацію обчислювальних процесів та масштабованість системи згідно вимог проєкту.

Для реалізації експериментальної частини мовою програмування обрано Python, яка є стандартом у сфері штучного інтелекту завдяки великій кількості інструментів та бібліотек. Раніше згадані Whisper, Gemma 3 і StyleTTS 2 обрано основними модулями системи, адже вони є передовими рішеннями у своїх сегментах, мають хорошу документацію та підтримку спільноти. Їх використання вимагає встановлення відповідних бібліотек згідно рекомендацій залежно від цільової платформи. Для запуску Gemma 3 знадобиться llama-cpp-python (версії не нижче 0.3.8) – бібліотека, яка забезпечує інференцію Llama-моделей. Для розпізнавання мовлення за допомогою Whisper потрібен PyTorch – інструмент для створення нейронних мереж та машинного навчання, який забезпечує автоматичне диференціювання, оптимізацію, роботу з тензорами тощо. Запуск Whisper-моделей також потребує бібліотеки whisper, для прискорення якої використовуватиметься faster_whisper, а також whisper_streaming, яка відкриває доступ до потокових функцій.

Перед встановленням llama-cpp-python, PyTorch чи faster_whisper слід виконати налаштування обраної обчислювальної платформи, щоб покращити продуктивність. У цій статті налаштування розглядається під платформу CUDA. Спочатку потрібно отримати Microsoft Visual Studio, а також завантажити інструменти збирання MSVC та Windows SDK останніх версій через Visual Studio Installer. Опісля відбувається встановлення CUDA з інтеграцією для Visual Studio для успішного збирання бібліотек. Окрім того, faster_whisper використовує реалізацію моделі-трансформера Whisper на CTranslate2, чим забезпечує майже чотирикратне прискорення обчислень. Для її роботи потрібна бібліотека примітивів для нейронних мереж NVIDIA cuDNN, яка інсталюється окремо від CUDA. Для моделі StyleTTS 2 важливим моментом є встановлення бібліотеки Phonemizer для перетворення тексту в фонеми: наприклад, фраза «Привіт, як справи?» перетворюється в «privit jak sprɑvi» й буде передана моделі на синтез [1]. Такий підхід дозволяє покращити інтонацію й враховувати фонетичні особливості різних мов. Для цього

Phonemizer вимагає наявності eSpeak NG для цільової платформи, щоб виконувати кінцевий синтез мовлення [4]. Систему слід будувати на мікросервісній архітектурі, яка зменшує складність дроблячи конструкцію на окремі незалежні модулі. У цьому проєкті кожна модель буде запущена як окремий сервіс за допомогою FastAPI.

Створення ефективного голосового асистента вимагає інтеграції модулів опрацювання мовлення так, щоб затримки передавання та опрацювання були мінімальні. Для їх зменшення слід утилізувати можливості стримінгу (streaming), за якого потік даних розбивається на сегменти і вони поступають у чергу. Після опрацювання відбувається транслювання результату, а тому можна отримати перші результати ще до кінця введення вхідних даних. Хоч більшість моделей підтримують стримінг, він не завжди доцільний – наприклад, коли мовна модель приймає цілісний текстовий запит. Можна додатково впроваджувати обробку тексту та аудіо за допомогою регулярних виразів, шумопоглиначів, обробників команд, засобів доповнення контексту тощо. Загальну схему взаємодії компонентів представлено за допомогою UML-діаграми на рис. 1.

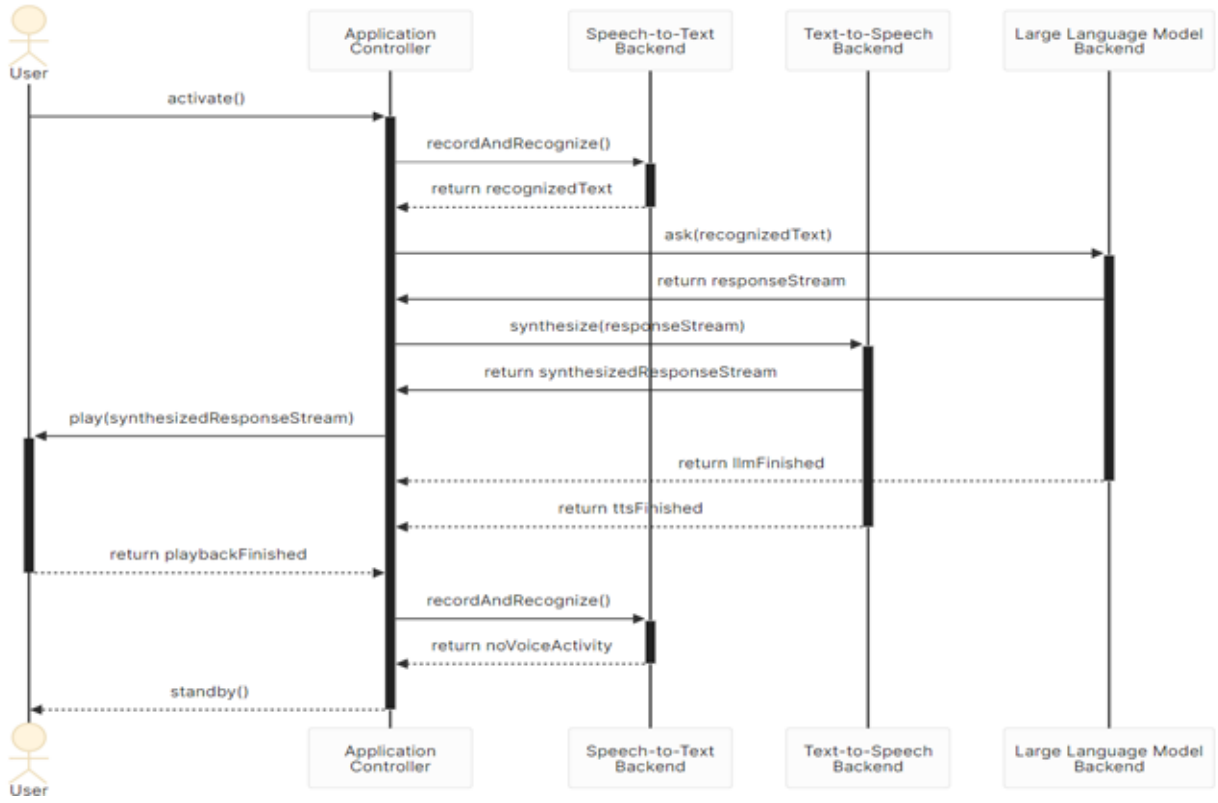

Рис. 1. UML-діаграма послідовності взаємодії компонентів та користувача

Для реалізації модулів проєкту обрано й протестовано низку україномовних моделей для синтезу та розпізнавання, які представлені в репозиторії проєкту Speech Recognition & Synthesis for Ukrainian [10] та з інших джерел, а також деякі мовні моделі для опрацювання й генерування тексту.

У якості моделі для розпізнавання мовлення було обрано whisper-small-uk-v2 [13] за високу точність розпізнавання українського голосу. Модель було перетворено з CTranslate2 для сумісності з faster_whisper. Для синтезу мовлення обрано StyleTTS2 Ukrainian [11] за відсутність сторонніх шумів у синтезованому аудіо, низьку кількість помилок наголошування та виразність. Як LLM обрано Gemma 3 4B Instruct Q4_K_M GGUF за швидкість роботи та можливість виведення у форматі JSON, що полегшує інтеграцію моделі.

Можливості налаштування асистента під потреби користувачів включають персоналізацію контексту, форматів відповідей та додавання спеціалізованих функцій для використання сторонніх інструментів. Додавання контексту в розмову за допомогою контекстних підказок (prompts) чи інструментів RAG сприятиме кращій взаємодії. Конкретний формат відповіді дозволяє зберігати історію спілкування моделі, розділяти співрозмовників, перехоплювати повідомлення, наприклад, для виконання команд. Для мовної моделі проста JSON-схема формату спілкування може виглядати наступним чином:

```
{ "type": "json_object",
  "schema": {
    "type": "object",
    "properties": {
      "actor_from": {"type": "string"},
      "actor_to": {"type": "string"},
      "message": {"type": "string"},
      "context": {"type": "array", "items": {"type": "string"} }
    },
    "required": [ "actor_from", "actor_to", "message" ] } }
```

Для отримання аудіопотоку використовуватиметься PyAudio. Аудіо ділиться на шматки по 1-4 секунди частотою дискретизації 16 кГц й відправляється моделі для розпізнавання. Для відтворення достатньо використовувати python-sounddevice.

Реалізація та тестування демонстраційного проєкту. Ціллю проєкту є перевірка життєздатності ідеї реалізації асистента балансуючи вимоги до апаратного забезпечення, якість і швидкість розпізнавання, опрацювання та синтезу й здатність працювати в середовищі з наявністю незначних шумів. Як уже відзначалося, модель Whisper вирізняється точністю розпізнавання в умовах шуму. Моделі Gemma 3 адаптовані для слабких пристроїв. StyleTTS 2 показує високу якість синтезу, а помилки наголошення можуть бути частково виправлені попередньою обробкою тексту. Описана раніше архітектура дозволяє розділити компоненти й об'єднати їх через мережу. Балансуючи параметри цих окремих систем можна звести вартість реалізації до мінімуму навіть підвищуючи продуктивність та якість комунікації.

Для тестування часу опрацювання мовлення прототип запущено в синхронному режимі, щоб оцінити продуктивність кожної моделі без впливу на неї іншої. «Прогрів» компонентів триває близько 150 секунд уперше й близько 45 секунд кожного наступного запуску. Споживання оперативної пам'яті досягає 24 ГіБ, відеопам'яті NVIDIA GeForce GTX 1660 – всі 6 ГіБ. Діалог складався з речень без запитів на пояснення чи довгих описів – усього 10 ітерацій циклу (20 коротких реплік). Витрати часу на кожну фазу опрацювання мовлення ітерації циклу діалогу, контрольний час запису та програвання аудіо, а також середні значення (СЗ) наведено в таблиці 1 та на рисунку 2.

Таблиця 1

Витрачений час на кожну фазу ітерації циклу діалогу короткими реченнями

Фаза діалогу	Тривалість фази ітерації, с										СЗ
Запис аудіо	12.0	12.0	10.0	14.0	10.0	10.0	10.0	8.0	8.0	6.0	10.0
Розпізнавання	11.1	18.5	12.9	19.2	13.7	10.0	10.0	14.3	8.4	6.4	12.4
Генерування відповіді	2.2	2.3	2.1	3.5	2.1	2.9	4.0	3.0	2.3	1.8	2.6
Синтез мовлення	1.0	1.0	1.0	1.7	0.9	1.4	2.1	1.4	1.1	0.9	1.3
Відтворення	10.2	12.4	11.3	17.5	7.9	15.2	20.5	11.2	7.8	6.4	12.0

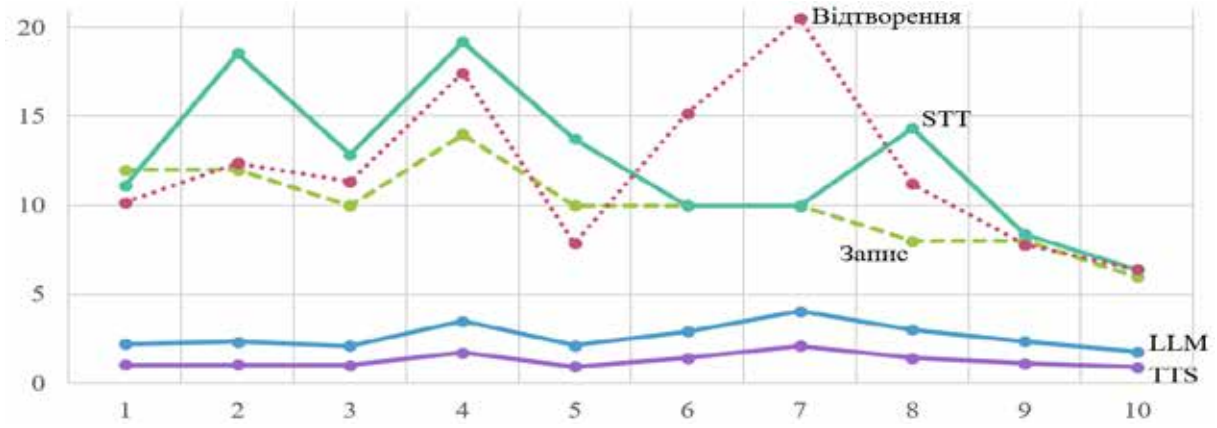

Рис. 2. Візуалізація часу, витраченого на кожну фазу циклу

Моделі опрацювання та синтезу справляються із завданнями добре, у той час як Whisper є найвимогливішою до часу та ресурсів. У синхронному режимі ітерація діалогу триває близько 40 секунд, але час на опрацювання запиту моделями від кінця запису до початку відтворення складає близько 16 секунд, що є ідеальною тривалістю повної ітерації циклу за реалізації асинхронного режиму не враховуючи змін до архітектури прототипу й набору моделей.

Висновки. У цій статті було розглянуто ключові аспекти створення автономного україномовного голосового асистента на основі відкритих технологій розпізнавання, опрацювання та синтезу мовлення. Основна увага приділялася аналізу сучасних моделей, інструментів та методів для досягнення високої якості, низької затримки, гнучкості налаштування, щоб підкреслили здатності сучасних технологій та наявність усіх можливостей розробляти автономні системи без залежності від сторонніх сервісів чи хмарних платформ.

Дослідження засобів опрацювання мовлення підкреслило перспективні інструменти для реалізації системи залежно від вимог кінцевого проєкту. Було проаналізовано й протестовано способи зменшення розмірів моделей, а також технології для прискорення їх інференції, щоб сформувати систему, здатну забезпечити продуктивність у реальному часі.

На основі отриманих результатів тестування можна стверджувати, що розвиток відкритих технологій відкриває нові горизонти для створення персоналізованих і доступних голосових асистентів. Необхідність незалежних голосових систем очевидна, коли приватність й автономність стають пріоритетними, а відкриті технології створюють можливість для розробників розвивати локальні рішення, що не залежать від сторонніх платформ. Майбутні дослідження можуть зосередитися на вдосконаленні інтерактивності, зменшенні затримок та покращенні продуктивності. Окремим напрямком можна відзначати розроблення моделей типу voice-to-voice (голос-до-голосу) для прямої взаємодії користувача та мовної моделі в реальному часі.

Список використаних джерел:

1. Bernard M., Titeux H. Phonemizer: Text to Phones Transcription for Multiple Languages in Python. *Journal of Open Source Software*. Vol. 6, Issue 68. P. 3958. DOI:10.21105/joss.03958.

2. Coqui TTS – deep learning toolkit for Text-to-Speech, battle-tested in research and production. *GitHub*. URL: https://github.com/coqui-ai/TTS.

3. DeepSpeech is an open source embedded (offline, on-device) speech-to-text engine which can run in real time on devices ranging from a Raspberry Pi 4 to high power GPU servers. *GitHub*. URL: https://github.com/mozilla/DeepSpeech.

4. eSpeak NG is an open source speech synthesizer that supports more than hundred languages and accents. *GitHub*. URL: https://github.com/espeak-ng/espeak-ng.

5. Gemma Team, Google Deepmind. Gemma 3 Technical Report. URL: https://storage.googleapis.com/deepmind-media/gemma/Gemma3Report.pdf.

6. Kim S., Shih K. J., Badlani R. et al. P-Flow: A Fast and Data-Efficient Zero-Shot TTS through Speech Prompting. *Thirty-seventh Conference on Neural Information Processing Systems.* (2023). URL: https://openreview.net/forum?id=zNA7u7wtIN.

7. Mistral AI Models Overview. *Mistral AI Documentation*. URL: https://docs.mistral.ai/getting-started/models/models_overview.

8. NVIDIA NeMo Framework ASR Models. *NeMo Framework User Guide*. URL: https://docs.nvidia.com/nemo-framework/user-guide/latest/nemotoolkit/asr/models.html.

9. Shih K. J., Valle R., Badlani R. et al. RAD-TTS: Parallel Flow-Based TTS with Robust Alignment Learning and Diverse Synthesis. *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*(2021). URL: https://openreview.net/forum?id=0NQwnnwAORi.

10. Speech Recognition & Synthesis for Ukrainian. *GitHub*. URL: https://github.com/egorsmkv/speech-recognition-uk.

11. StyleTTS2 Ukrainian Demo. *Hugging Face*. URL: https://huggingface.co/spaces/patriotyk/styletts2-ukrainian.

12. Whisper is an automatic speech recognition (ASR) system trained on 680,000 hours of multilingual and multitask supervised data collected from the web. *OpenAI*. URL: https://openai.com/index/whisper.

13. whisper-small-uk-v2. *Hugging Face*. URL: https://huggingface.co/nikes64/whisper-small-uk-v2.

14. Baevski A., Hsu W.-N., Xu Q. et al. Data2vec: A General Framework for Self-supervised Learning in Speech, Vision and Language. URL: https://ai.meta.com/research/data2vec-a-general-framework-for-self-supervised-learning-in-speech-vision-and-language.

15. Chung Y.-A., Zhang Y., Han W. et al. W2v-BERT: Combining Contrastive Learning and Masked Language Modeling for Self-Supervised Speech Pre-Training. arXiv, 2021. DOI:10.48550/ARXIV.2108.06209.

16. Li Y. A., Han C., Raghavan V. S. et al. StyleTTS 2: Towards Human-Level Text-to-Speech through Style Diffusion and Adversarial Training with Large Speech Language Models. arXiv, 2023. DOI:10.48550/ARXIV.2306.07691.

17. Touvron H., Lavril T., Izacard G. et al. LLaMA: Open and Efficient Foundation Language Models. arXiv, 2023. DOI:10.48550/ARXIV.2302.13971.