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: **«Транскрибування аудіо в текст на основі сучасних технологій штучного інтелекту»**

Виконала: здобувачка вищої освіти
за освітньою програмою
Інформаційні управляючі системи
спеціальності 126 Інформаційні
системи та технології
ступеня вищої освіти бакалавр
групи 126ІСТ_бз_2022
Готра Євгенія Віталіївна
Керівник: Слюсар Вадим Іванович
Рецензент: Муравльов Володимир
В’ячеславович

Полтава – 2026 року

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

Освітня програма Інформаційні управляючі системи
Спеціальність 126 Інформаційні системи та технології
Рівень вищої освіти перший (бакалаврський)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Юрій УТКІН
«10» квітня 2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧКИ ВИЩОЇ ОСВІТИ

Готри Євгенії Віталіївни

1. Тема кваліфікаційної роботи:
«Транскрибування аудіо в текст на основі сучасних технологій штучного інтелекту»,
Керівник роботи: д. т. н., професор, професор кафедри інформаційних систем та технологій Слюсар Вадим Іванович.
Затверджено наказом закладу вищої освіти від «19» березня 2026 року № 282-ст

2. Строк подання здобувачем вищої освіти роботи «26» травня 2026 року

3. Вихідні дані до роботи: науково-технічна література, аналітичні джерела мережі Інтернет, що містять інформацію про транскрибування, моделі ASR, пакетну обробку аудіофайлів, Google Cloud Speech-to-Text, Amazon Transcribe, OpenAI Speech to Text API, Whisper, WhisperX, Faster Whisper, Pyannote.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Розділ 1 Аналіз особливостей застосування транскрибування аудіо у текст
Розділ 2. Розробка моделі локального транскрибування аудіо в текст
Розділ 3. Рекомендації щодо практичної реалізації локального транскрибування аудіо в текст

5. Перелік графічного матеріалу: схеми, рисунки, діаграми за темою та об’єктом дослідження.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання отримав
Розрахунок економічної ефективності результатів дослідження	Шкурупій О. В., д. е. н., професор, професор кафедри економіки та публічного управління	24.03.2026 р.	09.04.2026 р.

7. Дата видачі завдання «10» квітня 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Вибір і затвердження теми роботи	01.04.2025 р.	
2	Складання і затвердження розгорнутого плану та завдання на кваліфікаційну роботу	02.04.2025 р. – 10.04.2025 р.	
3	Опрацювання джерел інформації	03.06.2025 р. – 15.06.2025 р.	
4	Збір, вивчення і обробка інформації, необхідної для виконання роботи	16.06.2025 р. – 04.07.2025 р.	
5	Виконання теоретико-методологічного розділу роботи	08.09.2025 р. – 15.11.2025 р.	
6	Виконання дослідницько-аналітичного розділу роботи	16.11.2025 р. – 30.01.2026 р.	
7	Виконання проектно-рекомендаційного розділу роботи	01.02.2026 р. – 21.03.2026 р.	
8	Розрахунок економічної ефективності результатів дослідження	24.03.2026 р. – 09.04.2026 р.	
9	Оформлення тексту роботи	10.04.2026 р. – 25.05.2026 р.	
10	Попередній захист роботи на кафедрі	26.05.2026 р.	
11	Доопрацювання роботи з урахуванням зауважень і пропозицій	27.05.2026 р. - 28.05.2026 р.	
12	Нормоконтроль	29.05.2026 р.	
13	Захист кваліфікаційної роботи	03.06.2026 р.	

Здобувач вищої освіти **Євгенія ГОТРА**

Керівник роботи **Вадим СЛЮСАР**

ЗМІСТ

ВСТУП

Актуальність теми кваліфікаційної роботи підтверджується необхідністю використання транскрибування аудіо у текст там, де потрібно швидко перетворювати усні повідомлення, консультації, лекції, наради або звернення користувачів у зручну текстову форму. Системи транскрибування базуються на автоматичному розпізнаванні мовлення та використанні нейронних мереж. Важливою перевагою сучасних систем є підтримка як обробки готових аудіофайлів, так і потокового розпізнавання в реальному часі. Локальні рішення забезпечують можливість самостійно керувати даними, використовуваними моделями та етапами їх обробки, водночас зменшуючи залежність від зовнішніх сервісів і пов'язаних із ними ризиків. Однак якість транскрибування залежить від обраної моделі, якості аудіо та складності мовлення. Все це свідчить про актуальність теми роботи.

Метою кваліфікаційної роботи є підвищення ефективності транскрибування аудіо в текст шляхом застосування локального AI-агента для пакетної обробки аудіофайлів.

Завданнями кваліфікаційної роботи є:

– обґрунтувати вибору інструментарію для реалізації локального AI-агента;

– розробити модель локального транскрибування аудіо в текст;

– сформувати рекомендацій щодо практичної реалізації локального транскрибування аудіо в текст;

– виконати економічне обґрунтування прийнятих рішень.

Об'єктом дослідження є процес транскрибування аудіо в текст.

Предметом дослідження є моделі та програмні засоби побудови локального AI-агента для пакетного транскрибування аудіофайлів у текст.

Методами дослідження є аналітичний метод – для вивчення сучасних підходів до транскрибування аудіо в текст, аналізу інструментарію для

створення локального AI-агента та обґрунтування вибору програмних засобів; порівняльний аналіз – для зіставлення існуючих технологій автоматичного розпізнавання мовлення, моделей ASR та варіантів їх локального розгортання; метод системного аналізу – для визначення структури локального AI-агента, його основних компонентів і взаємозв'язків між ними; моделювання – для побудови логіки роботи локальної системи транскрибування аудіоданих.

Інформаційна база кваліфікаційної роботи сформована з Інтернет-ресурсів, що місять інформацію про транскрибування, моделі ASR, пакетну обробку аудіофайлів.

Практична значущість роботи полягає у розробці моделі локального транскрибування аудіо в текст, рекомендацій щодо практичної реалізації локального транскрибування аудіо в текст. Результати можуть бути використані для подальших досліджень за даною тематикою та при проектуванні локальних AI-агентів.

Апробація результатів відбувалася в рамках XXI щорічної студентської наукової конференції «Сучасні інформаційні технології та інноваційні методики в економіці, менеджменті та бізнесі» Полтавського державного аграрного університету (22 квітня 2026 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 64 сторінки формату А4. Вона містить 10 рисунків і 3 таблиці.

РОЗДІЛ 1
АНАЛІЗ ОСОБЛИВОСТЕЙ ЗАСТОСУВАННЯ ТРАНСКРИБУВАННЯ АУДІО У ТЕКСТ

1.1 Галузі застосування транскрибування аудіо у текст

Транскрибування аудіо у текст є трендом впровадження штучного інтелекту (Artificial Intelligence, AI) [1]. Це процес перетворення усного мовлення, записаного у вигляді аудіофайлу або голосового сигналу, у текстову форму за допомогою ПЗ або AI-технологій. Під час такого процесу система аналізує мовлення людини, розпізнає окремі слова, фрази та мовні конструкції, після чого формує текстовий запис сказаного. Тобто, транскрибування є одним з ключових етапів цифрової обробки мовленнєвої інформації, оскільки дає можливість перетворити неструктуроване голосове повідомлення у текст, придатний для подальшого аналізу, пошуку, зберігання або автоматичної обробки іншими ІС. На даний час, існує кілька галузей застосування цього процесу. Однією з перспективних галузей є обробка звернень користувачів у службах підтримки. Дзвінки, голосові повідомлення або записи консультацій можуть автоматично транскрибуватися, після чого локальний AI-агент аналізує зміст розмови. Це оптимізує роботу служб підтримки та дозволяє швидко відреагувати на типові проблеми користувачів. Це дозволяє: визначати основну проблему користувача; класифікувати звернення за тематикою; формувати короткий підсумок розмови; виявляти повторювані проблеми; оцінювати якість комунікації; готувати рекомендації для оператора або менеджера.

У сфері освіти транскрибування може використовуватися для автоматичного перетворення лекцій, семінарів, консультацій або усних відповідей студентів у текст. Локальний AI-агент може додатково формувати конспекти, виділяти ключові поняття, створювати короткі підсумки та допоміжні навчальні матеріали. Локальна обробка є важливою для

навчальних закладів, оскільки дозволяє не передавати записи занять стороннім сервісам. Застосування в освіті є доцільним, оскільки воно: спрощує підготовку конспектів; підвищує доступність навчальних матеріалів; допомагає студентам швидше знаходити потрібну інформацію; дає змогу аналізувати зміст лекцій або відповідей; може використовуватися для підтримки дистанційного навчання.

У медичній сфері транскрибування може застосовуватися для фіксації консультацій лікаря з пацієнтом, створення текстових нотаток, попереднього структурування скарг пацієнта або підготовки медичної документації. У цій галузі локальність є особливо важливою, оскільки медична інформація належить до чутливих персональних даних. Використання локального агента зменшує ризик витоку конфіденційної інформації. Локальний AI-агент може допомагати: перетворювати голосові записи консультацій у текст; виділяти скарги, симптоми та рекомендації; формувати короткі службові нотатки; структурувати інформацію для подальшого внесення до медичної системи.

У державних установах, органах місцевого самоврядування та ін. часто виникає потреба у фіксації усних обговорень, засідань, допитів, консультацій або службових нарад. Локальна обробка є доцільною через необхідність захисту службової, юридичної та персональної інформації. Система транскрибування з локальним AI-агентом може використовуватися для: створення текстових протоколів; пошуку ключових фрагментів у записах; формування підсумків засідань; структурування службової інформації; підготовки аналітичних довідок.

У бізнесі транскрибування може застосовуватися для аналізу нарад, переговорів, інтерв’ю, презентацій і внутрішніх обговорень. Локальний AI-агент може автоматично формувати підсумки зустрічей, виділяти прийняті рішення, відповідальних осіб і подальші завдання. Для бізнесу локальний підхід також важливий з погляду захисту комерційної таємниці. Це дозволяє: зменшити час на ручне ведення протоколів; покращити контроль виконання завдань; швидко знаходити важливу інформацію в записах; аналізувати

комунікацію всередині компанії; зберігати корпоративні дані всередині організації. Журналісти, редактори та медіаорганізації часто працюють з інтерв'ю, подкастами, репортажами й аудіозаписами. Автоматичне транскрибування дозволяє швидко отримати текстову версію матеріалу, а локальний AI-агент може допомогти зі скороченням тексту, виділенням цитат і підготовкою чернеток публікацій. Практична користь полягає у: швидкому розшифруванні інтерв'ю; пошуку важливих фрагментів; формуванні коротких підсумків; підготовці текстової основи для статей; збереженні контролю над неопублікованими матеріалами. Окремим перспективним напрямом застосування транскрибування у поєднанні з локальним AI-агентом є AI-аналітика звернень користувачів. Цей напрям передбачає автоматизовану обробку голосових звернень, телефонних розмов, аудіоповідомлень або записів консультацій з метою перетворення мовлення у текст та подальшого інтелектуального аналізу отриманої інформації. У межах такого підходу система спочатку виконує транскрибування аудіозапису, тобто переводить мовленнєву інформацію у текстову форму. Після цього локальний AI-агент аналізує зміст звернення, визначає його основну тему, виявляє проблему користувача, класифікує тип запиту та може формувати короткий підсумок розмови або рекомендації щодо подальших дій. AI-аналітика звернень є особливо актуальною для організацій, які регулярно взаємодіють із великою кількістю користувачів, клієнтів або громадян. Це можуть бути служби підтримки, контакт-центри, освітні установи, державні організації, медичні заклади, банки, телекомунікаційні компанії та інші установи, де важливо швидко обробляти запити й виявляти типові проблеми. Основними завданнями AI-аналітики звернень є [2]: автоматичне перетворення голосових звернень у текст; визначення суті звернення користувача; класифікація звернень за темами або категоріями; виявлення повторюваних проблем; формування короткого змісту розмови; визначення емоційного забарвлення або рівня незадоволеності користувача; підготовка рекомендацій для оператора, аналітика або адміністратора; накопичення структурованих даних

для подальшого аналізу. Використання саме локального AI-агента у цьому напрямі має важливі переваги. По-перше, обробка аудіозаписів і текстових розшифровок відбувається без передавання даних до зовнішніх хмарних сервісів. Це підвищує рівень конфіденційності та зменшує ризик витоку персональної або службової інформації. По-друге, локальний агент може працювати в межах внутрішньої інформаційної системи організації, що забезпечує більший контроль над даними та процесами їх обробки. Практична цінність AI-аналітики звернень полягає в тому, що вона дозволяє не лише зберігати текстову версію розмови, а й отримувати з неї корисні аналітичні висновки. Наприклад, система може автоматично визначати, які проблеми найчастіше виникають у користувачів, які теми потребують додаткової уваги, які звернення є терміновими, а також які процеси в організації потребують покращення. Таким чином, AI-аналітика звернень користувачів може розглядатися як окремий прикладний напрям використання транскрибування та локального AI-агента. Вона поєднує технології Automatic Speech Recognition (ASR), NLP та інтелектуального аналізу даних, забезпечуючи ефективну, конфіденційну та автоматизовану обробку мовленнєвої інформації.

1.2 Аналіз сучасних AI-технологій для транскрибування аудіо

Основою транскрибування аудіо у текст є технологія ASR [3]. Її сутність полягає в тому, що система отримує на вхід аудіосигнал з мовленням людини, аналізує його та формує текстовий результат. Такі системи використовуються у голосових помічниках, сервісах створення субтитрів, системах протоколювання зустрічей, контакт-центрах та інших задачах, де потрібно швидко перетворити усне мовлення у текст.

Сучасні AI-технології для транскрибування значно відрізняються від старіших підходів, які були побудовані переважно на правилах, словниках і

статистичних моделях. Сьогодні основну роль відіграють нейронні мережі, які здатні навчатися на великих наборах аудіоданих. Завдяки цьому вони краще розпізнають мовлення в різних умовах: при різній швидкості мовлення, акцентах, фоновому шумі або різній якості запису.

Однією з найбільш відомих сучасних моделей для транскрибування є Whisper. Це модель розпізнавання мовлення, побудована на архітектурі Transformer. Вона може виконувати кілька пов’язаних задач: розпізнавання мовлення різними мовами, визначення мови аудіо, транскрибування та переклад мовлення англійською мовою. Особливістю Whisper є те, що модель навчалася на великій кількості різноманітних аудіоданих, тому вона досить добре працює з реальними записами, які можуть містити шум або неідеальну вимову. Також важливим напрямом є використання моделей типу wav2vec 2.0. Їхня ідея полягає в тому, що модель спочатку навчається на великій кількості аудіо без текстової розмітки, а потім донавчається на даних, де вже є правильні текстові транскрипти. Такий підхід є корисним, тому що підготувати аудіозаписи простіше, ніж створити для них якісні ручні розшифровки. Це дозволяє будувати системи розпізнавання мовлення навіть тоді, коли для певної мови або предметної області є обмежена кількість розмічених даних.

Для практичної реалізації систем транскрибування часто використовують бібліотеки та платформи, які спрощують роботу з AI-моделями. Наприклад, бібліотека Hugging Face Transformers дає можливість використовувати готові моделі автоматичного розпізнавання мовлення через зручні програмні інтерфейси [4]. Це дозволяє не створювати модель з нуля, а взяти вже навчену модель і застосувати її для конкретної задачі, наприклад для розпізнавання аудіофайлів, створення субтитрів або обробки записів звернень користувачів. Окрему роль відіграють технології, які підтримують як пакетне, так і потокове розпізнавання мовлення. Пакетне розпізнавання використовується тоді, коли система обробляє вже готовий аудіофайл. Потокове розпізнавання застосовується у випадках, коли текст потрібно

отримувати майже в реальному часі, наприклад під час телефонної розмови або консультації on-line. Такі можливості важливі для контакт-центрів, служб підтримки та систем AI-аналітики звернень. Наприклад, NVIDIA Riva [5] описується як GPU-прискорений конвеєр для ASR, який підтримує offline/batch і streaming режими розпізнавання. Після розпізнавання текст може передаватися локальному AI-агенту, який виконує аналіз змісту. Наприклад, агент може визначати тему звернення, знаходити ключові слова, формувати короткий підсумок, класифікувати запит користувача або готувати відповідь на основі внутрішньої бази знань. Таким чином, транскрибування є першим етапом більш складної системи AI-аналітики. Разом з перевагами сучасні системи транскрибування мають і певні обмеження. Якість результату залежить від мови, якості запису, наявності шумів, кількості співрозмовників та правильності вимови. Крім того, AI-моделі можуть іноді помилятися або створювати неточні фрагменти тексту, тому в критично важливих сферах результати транскрибування бажано перевіряти людиною. Проблема можливих помилок і «вигаданих» фрагментів у транскриптах досить актуальна у чутливих галузях, таких як медицина. Отже, сучасні AI-технології для транскрибування аудіо базуються на нейронних мережах і дають змогу автоматично перетворювати мовлення в текст із досить високою якістю. Найбільш перспективними є рішення, які підтримують багатомовність, роботу з шумними записами, локальне розгортання та інтеграцію з іншими AI-модулями.

1.3 Обґрунтування доцільності локального розгортання системи транскрибування

Під час розроблення системи AI-аналітики звернень важливим питанням є вибір способу обробки даних. Оскільки система працює з аудіозаписами, голосовими повідомленнями та текстовими розшифровками

звернень, такі дані можуть містити персональну, службову або комерційну інформацію. Наприклад, у зверненнях користувачі можуть повідомляти своє ім’я, контактні дані, опис проблеми, фінансову інформацію, медичні відомості або інші конфіденційні дані. Тому передавання таких матеріалів до зовнішніх хмарних сервісів може створювати додаткові ризики для безпеки. Локальне розгортання AI-агента передбачає, що основні етапи обробки виконуються безпосередньо на обладнанні організації або користувача. До таких етапів належать транскрибування аудіо у текст, попередня обробка отриманого тексту, класифікація звернення, формування короткого змісту та підготовка аналітичних висновків. У цьому випадку аудіофайли та результати їх обробки не потрібно передавати на сторонні сервери, що підвищує рівень контролю над даними. З точки зору конфіденційності локальний підхід має декілька важливих переваг. По-перше, зменшується ризик витоку інформації під час передавання даних через мережу. По-друге, організація самостійно визначає, де зберігаються аудіозаписи, текстові транскрипти та результати AI-аналітики. По-третє, доступ до системи можна обмежити лише авторизованими користувачами, що дозволяє краще контролювати роботу з чутливою інформацією.

Особливо важливим локальне розгортання є для сфер, де обробляються персональні або службові дані. До таких сфер можна віднести медичні заклади, освітні установи, державні організації, юридичні компанії, банки, служби підтримки користувачів та внутрішні корпоративні системи. У цих випадках навіть звичайне звернення користувача може містити інформацію, яка не повинна потрапляти до сторонніх сервісів.

Крім конфіденційності, локальне розгортання є доцільним і з практичної точки зору. Така система може працювати незалежно від стабільності інтернет-з’єднання та доступності зовнішніх API. Це важливо для організацій, які потребують постійної роботи сервісу. Також локальний AI-агент може бути налаштований під конкретні потреби організації: власні категорії звернень, внутрішню базу знань, специфічну термінологію та

правила обробки даних. Ще однією перевагою є можливість зменшення довгострокових витрат. Використання хмарних сервісів часто передбачає оплату за кількість запитів, обсяг обробленого аудіо або кількість використаних токенів мовної моделі. У випадку локального розгортання основні витрати пов’язані з обладнанням і налаштуванням системи, після чого її можна використовувати без постійної залежності від зовнішнього постачальника послуг.

Водночас локальне розгортання має певні обмеження. Для роботи AI-агента потрібні достатні обчислювальні ресурси, а якість обробки залежить від обраних моделей транскрибування та аналізу тексту. Також необхідно забезпечити технічне обслуговування системи, оновлення моделей і захист локальної інфраструктури. Однак для багатьох практичних задач ці недоліки є прийнятними, особливо якщо пріоритетом є захист даних і незалежність від хмарних сервісів.

Таким чином, локальне розгортання AI-агента є доцільним рішенням для реалізації AI-аналітики звернень, оскільки воно забезпечує вищий рівень конфіденційності, контроль над даними, автономність роботи та можливість адаптації системи до конкретної предметної області. Такий підхід дозволяє безпечно обробляти мовленнєву інформацію користувачів і використовувати результати транскрибування для подальшого аналізу, класифікації та прийняття рішень.

1.4 Порівняння хмарних і локальних рішень для транскрибування

Для перетворення мовлення в текст існують методи: синхронний, асинхронний та потоковий режим. Кожен метод повертає результати тексту залежно від того, чи потрібна транскрипція під час постобробки, періодично або в реальному часі. Тобто вводять аудіодані і отримувати текст у відповідь. Для реалізації системи транскрибування аудіо в текст можуть

використовуватися два основні підходи: хмарні рішення та локальні рішення. Обидва варіанти мають недоліки та переваги. Вибір залежить від мети системи, вимог до конфіденційності, доступних обчислювальних ресурсів і умов використання.

Хмарні рішення для транскрибування працюють за принципом передавання аудіофайлу або голосового потоку на сервер постачальника послуги. Там аудіо обробляється за допомогою моделей автоматичного розпізнавання мовлення, після чого користувач отримує готовий текстовий результат. Прикладами такого підходу можуть бути онлайн-сервіси розпізнавання мовлення, API для транскрибування або вбудовані інструменти у платформах для відеоконференцій. Головною перевагою хмарних сервісів є простота використання. Користувачу не потрібно самостійно встановлювати моделі, налаштовувати середовище, підбирати обладнання або оптимізувати роботу системи. Достатньо завантажити аудіозапис або підключити сервіс через API. Також хмарні рішення часто мають високу якість розпізнавання, оскільки використовують потужні сервери та великі мовні моделі, які постійно оновлюються розробниками. Ще однією перевагою хмарного підходу є масштабованість. Якщо потрібно обробляти велику кількість аудіозаписів, хмарний сервіс може автоматично виділяти необхідні ресурси. Це зручно для великих компаній, контакт-центрів або медіаорганізацій, де щодня створюється багато аудіоконтенту. Крім того, деякі сервіси підтримують додаткові функції: визначення мови, розділення мовців, створення субтитрів, переклад, пошук по тексту та інтеграцію з іншими платформами. Однак хмарні рішення мають і суттєві недоліки. Найважливішим із них є питання конфіденційності. Для обробки аудіозапис потрібно передати на сторонній сервер, а це означає, що організація частково втрачає контроль над даними. Якщо в аудіо містяться персональні дані, комерційна інформація, службові відомості або інші чутливі матеріали, таке передавання може бути небажаним. Особливо це стосується звернень користувачів, медичних консультацій, внутрішніх нарад, юридичних розмов

або записів роботи державних установ. Крім того, хмарні сервіси залежать від стабільного інтернет-з’єднання. Якщо доступ до мережі відсутній або нестабільний, система транскрибування може працювати з перебоями або взагалі бути недоступною. Також слід враховувати вартість використання. Більшість хмарних сервісів працюють за моделлю оплати за обсяг обробленого аудіо, кількість запитів або тривалість записів. На початковому етапі це може бути зручно, але при постійному використанні витрати можуть значно зростати.

Серед популярних хмарних рішень для транскрибування можна виділити кілька. Їхньою спільною перевагою є простота інтеграції, відсутність потреби у власному потужному обладнанні та підтримка сучасних моделей розпізнавання мовлення. Розглянемо коротко їхні властивості.

Одним із відомих прикладів є Google Cloud Speech-to-Text [6]. Це сервіс Google Cloud, який дозволяє перетворювати мовлення в текст і вбудовувати розпізнавання мовлення у власні програмні застосунки через API. Таке рішення може використовуватися для створення субтитрів, обробки голосових команд, розшифрування дзвінків, аудіозаписів або відеоматеріалів. Його перевагою є готова хмарна інфраструктура та можливість швидкої інтеграції в інформаційні системи.

Ще одним поширеним прикладом є Amazon Transcribe [7]. Це хмарний сервіс автоматичного розпізнавання мовлення від Amazon Web Services. Він призначений для перетворення аудіо в текст і може використовуватися як самостійний сервіс або як частина більшої програмної системи. AWS зазначає, що Amazon Transcribe підтримує розпізнавання записаного та потокового мовлення, що є важливим для контакт-центрів, систем аналітики дзвінків і сервісів підтримки користувачів.

До хмарних рішень також належить Microsoft Azure AI Speech, зокрема функція Speech to text [8]. Вона дає змогу виконувати розпізнавання мовлення в реальному часі або обробляти аудіо пакетно. Це означає, що сервіс можна використовувати як для готових аудіофайлів, так і для сценаріїв, де текст

потрібно отримувати безпосередньо під час розмови. Такий підхід є корисним для систем онлайн-консультацій, протоколювання зустрічей, голосових інтерфейсів і служб підтримки.

Окремо можна виділити OpenAI Speech to Text API [9]. У документації OpenAI зазначено, що для перетворення аудіо в текст використовуються два основні напрями: транскрибування та переклад аудіо. Для транскрибування можуть застосовуватися ресурс Whisper AI [10] та моделі whisper-1, gpt-4o-transcribe, gpt-4o-mini-transcribe та gpt-4o-transcribe-diarize. Остання модель підтримує розділення мовців, що є корисним для аналізу діалогів, зустрічей або звернень користувачів. Whisper – це універсальна модель розпізнавання мовлення (рис. 1.1) [11]. Існує 6 розмірів моделей (рис. 1.2), 4 з яких мають виключно англійські версії, що пропонує компроміси між швидкістю та точністю. Продуктивність Whisper значно варіюється залежно від мови.

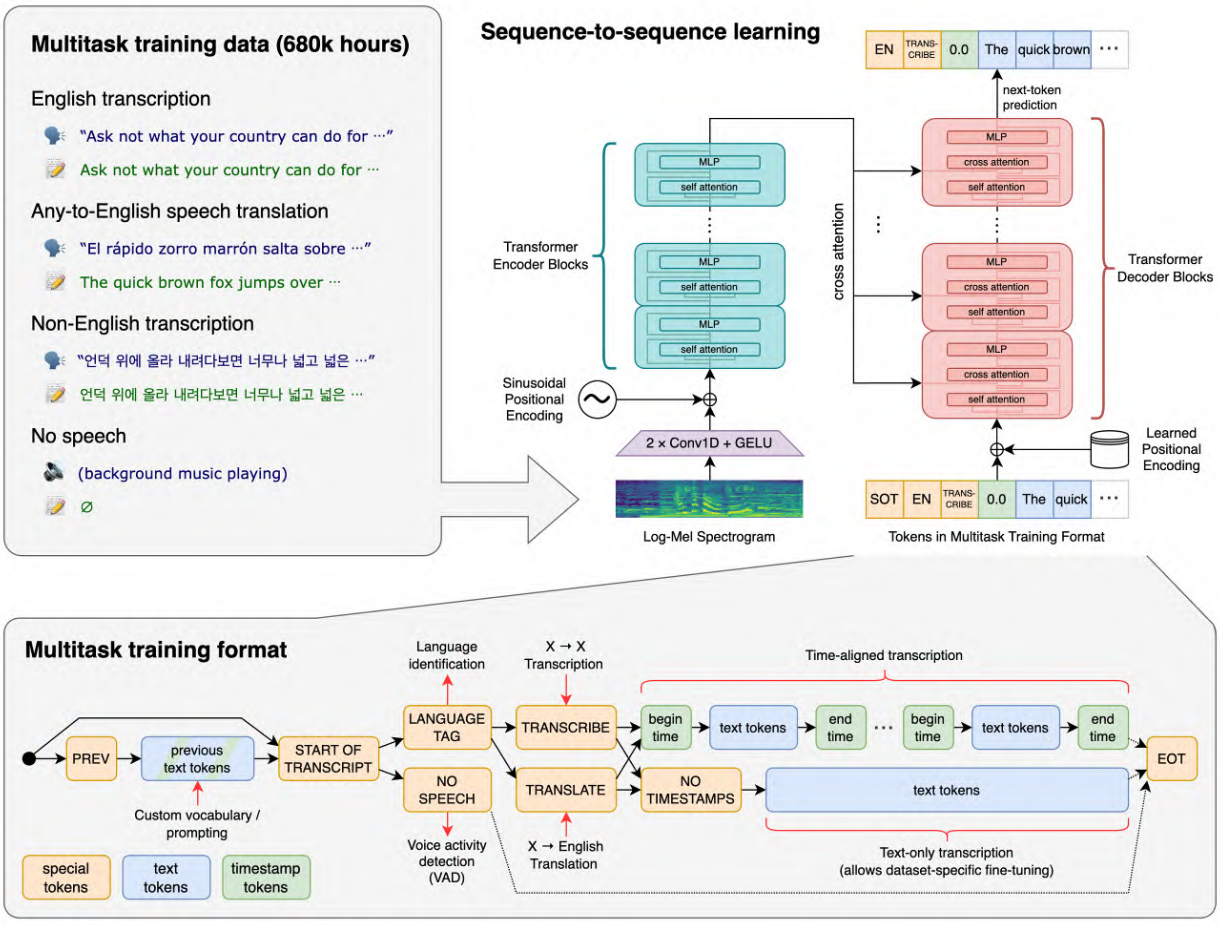

Рисунок 1.1 – Сутність роботи Whisper AI [11]

Тобто, розподіл продуктивності моделей large-v3 та large-v2 за мовами, використовуючи показники WER (коефіцієнти помилок слів) або CER (коефіцієнти помилок символів, виділені курсивом), оцінені на наборах даних Common Voice 15 та Fleurs.

Size	Parameters	English-only model	Multilingual model	Required VRAM	Relative speed
tiny	39 M	`tiny.en`	`tiny`	~1 GB	~10x
base	74 M	`base.en`	`base`	~1 GB	~7x
small	244 M	`small.en`	`small`	~2 GB	~4x
medium	769 M	`medium.en`	`medium`	~5 GB	~2x
large	1550 M	N/A	`large`	~10 GB	1x
turbo	809 M	N/A	`turbo`	~6 GB	~8x

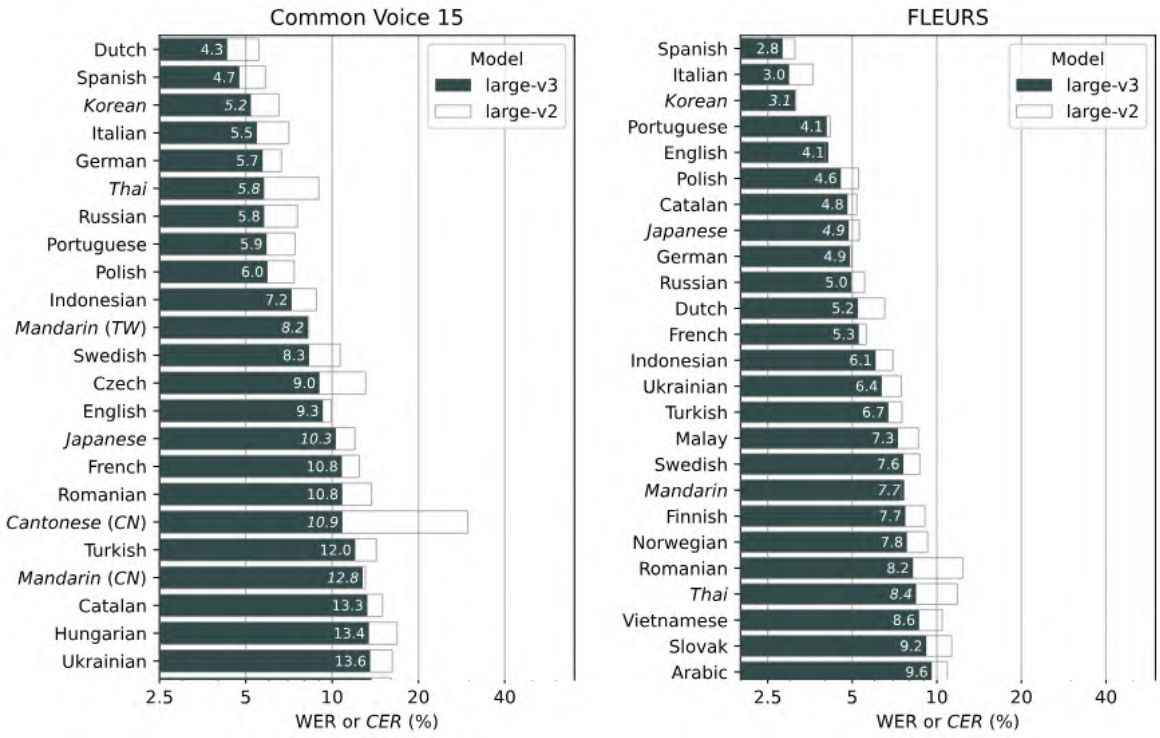

Рисунок 1.2 – Фрагмент порівняння версій Whisper [11]

Наведені відносні швидкості вимірюються транскрипцією англійської мови на A100, і реальна швидкість може суттєво відрізнятися залежно від багатьох факторів, включно з мовою, швидкістю мовлення та доступним обладнанням. Також прикладом спеціалізованого хмарного сервісу є AssemblyAI [12]. Це API для перетворення мовлення в текст, орієнтоване на

обробку реальних аудіозаписів, зокрема записів із шумами, акцентами або спеціальною термінологією. Такий сервіс може бути корисним для розшифрування інтерв'ю, подкастів, дзвінків, лекцій або звернень користувачів. Крім звичайного транскрибування, подібні платформи часто пропонують додаткові можливості для подальшого аналізу тексту. Ще одним прикладом є Deepgram Speech-to-Text API [13]. Deepgram пропонує хмарні API для розпізнавання мовлення, які можуть працювати як з потоковим аудіо в реальному часі, так і з попередньо записаними файлами. Це робить сервіс придатним для голосових агентів, контакт-центрів, онлайн-комунікацій і систем автоматизованої обробки звернень.

Локальні рішення для транскрибування працюють інакше. Модель ASR встановлюється безпосередньо на локальний комп'ютер (сервер). Аудіофайли не передаються до зовнішніх сервісів, а всі етапи обробки виконуються всередині локальної інфраструктури. Це дозволяє краще контролювати доступ до інформації, зменшує ризик витоку даних під час передачі через мережу та підвищує рівень конфіденційності. Такий підхід є особливо доцільним для систем, які працюють з чутливою інформацією. Система може працювати без постійного доступу до Інтернет, що є важливим для автономних робочих місць, внутрішніх корпоративних систем або організацій з підвищеними вимогами до безпеки. Крім того, локальне рішення можна гнучко налаштовувати під конкретні задачі: обирати модель, мову, формат вихідного тексту, спосіб збереження результатів і подальшу інтеграцію з локальним AI-агентом. Транскрибування розглядається не лише як окрема функція, а як частина системи AI-аналітики звернень. Після перетворення аудіо в текст локальний AI-агент може аналізувати зміст звернення, визначати тему, виділяти ключові слова, формувати короткий підсумок, класифікувати проблему та готувати рекомендації. Якщо всі ці етапи виконуються локально, організація зберігає повний контроль над даними. Разом із тим локальні рішення також мають певні недоліки. Деякі сучасні моделі ASR можуть вимагати продуктивного CPU, достатнього обсягу ОЗП або відеокарти. Якщо

обладнання слабке, транскрибування може виконуватися повільно. Також користувач повинен самостійно встановлювати ПЗ, налаштовувати моделі, оновлювати їх і контролювати стабільність роботи системи. Ще одним обмеженням локального підходу є те, що якість розпізнавання може залежати від обраної моделі та її налаштувань. Деякі локальні моделі можуть працювати швидко, але менш точно, тоді як більш точні моделі потребують більше ресурсів. Тому необхідно знайти компроміс між якістю транскрибування, швидкістю роботи та можливостями обладнання.

Порівнюючи обидва підходи, можна зазначити, що хмарні рішення краще підходять для випадків, коли важливі простота запуску, швидка інтеграція та мінімальні вимоги до локального обладнання. Водночас локальні рішення є більш доцільними тоді, коли пріоритетом є конфіденційність, автономність, контроль над даними та можливість адаптації системи під конкретну предметну область.

Хмарні сервіси є зручними та забезпечують високу якість розпізнавання, але створюють залежність від інтернету, зовнішнього постачальника та умов оплати. Локальні рішення складніші в налаштуванні, проте вони забезпечують більшу конфіденційність, автономність і контроль над процесом обробки даних. Саме тому для розроблення системи транскрибування та AI-аналітики звернень доцільно розглядати локальне розгортання як основний варіант реалізації.

РОЗДІЛ 2
РОЗРОБКА МОДЕЛІ ЛОКАЛЬНОГО ТРАНСКРИБУВАННЯ АУДІО В ТЕКСТ

2.1 Хмарне рішення корпоративного рівня для транскрибування на основі Google Cloud

Перетворення мовлення на текст (Speech-to-Text, STT) може використовувати Chirp 3 [14], базову модель Google Cloud для мовлення, навченого на мільйонах годин аудіоданих та мільярдах текстових речень. Це контрастує з традиційними методами розпізнавання мовлення, які зосереджені на великих обсягах контрольованих даних, специфічних для певної мови. Ці методи забезпечують користувачам кращу здатність розпізнавати та транскрибування [15] для більшої кількості розмовних мов та акцентів. Розпізнавання та транскрибування мовлення на основі AI використовує адаптацію моделей [16] для підвищення точності часто вживаних слів, розширення словникового запасу для транскрипції та покращення транскрипції з гучного аудіо.

1. Покращення точності слів і фраз, які часто зустрічаються у вашій Аудіодані. Наприклад, ви можете сповіщити модель розпізнавання на голос команди, які зазвичай вимовляють ваші користувачі.

2. Розширення словникового запасу слів, розпізнаних за допомогою Speech to Text. Мовлення в текст містить дуже великий словниковий запас. Однак, якщо ви Аудіодані часто містять слова, які рідко використовуються в загальній мові (наприклад, як власні імена або слова, специфічні для домену), можна додавати їх за допомогою моделі адаптація.

Адаптація моделі дозволяє користувачам налаштовувати мову в текст для частішого розпізнавання конкретних слів або фраз, ніж інші варіанти, які могли б бути запропоновані інакше. Адаптація моделі особливо корисна у двох ситуаціях. Підвищити ймовірність того, що модель розпізнає це слово

«погода» – коли він транскрибує ваші аудіодані, можна передати одне слово «погода» в об'єкті у ресурсі SpeechAdaptation [17]. Якщо надати багатословну фразу, то модель має більшу ймовірність розпізнання цих слів послідовно. Надання фрази також збільшує ймовірність розпізнавання частин фрази, включаючи окремі слова. Покращити розпізнавання можна також за допомогою класів. Вони представляють поширені поняття, що зустрічаються в природній мові, такі як грошові одиниці та календарні дати. Щоб використовувати клас у адаптації моделі, необхідно додати токен класу у полі ресурс. Можна використовувати класи як окремі предмети в масиві або вбудовувати один або більше класових жетонів у довших багатословних фразах. Наприклад, ви можете позначити номер адреси у більшій фразі шляхом включення токена класу у рядок: . Однак ця фраза не допоможе у випадках де аудіо містить схожу, але неідентичну фразу, наприклад «Я є на 123 Мейн-стріт». Щоб допомогти розпізнавати схожі фрази, важливо Додатково включайте токен класу окремо. Такий корпус домагає покращити точність транскрибування для великих груп слів, які відповідають спільній концепції. Також можна покращити точність транскрибування мовлення якщо аудіо містить шум або мовлення не дуже чітке.

За замовчуванням, STT не містить позначки пунктуації в результатах транскрибування. Однак користувач може запитати, щоб автоматично визначалась і вставлялась пунктуація у результатах транскрибування. Коли вмикається автоматична пунктуація, також автоматично пишеться перша літеру з великої літери після кожної крапки та знаку питання. API Cloud Speech-to-Text пропонує функції усної пунктуації та емодзі з усним спостереженням.

Транскрибувати можна короткі, довгі та потокові аудіодані. Потокове розпізнавання мовлення дозволяє транслювати аудіо в хмару STT та отримання потоку розпізнавання мовлення в реальному часі, оскільки аудіо оброблено. Синхронне розпізнавання мовлення повертає розпізнаний текст для короткого аудіо (менше 60 секунд). Відповідно, асинхронне

розпізнавання мовлення може передбачати пакетне розпізнавання мовлення. Це запускає тривалу операцію обробки аудіо. Тому, зазвичай, асинхронне розпізнавання мовлення використовується для транскрибування аудіо довше за 1 хв. Для коротшого аудіо синхронне розпізнавання мовлення [18] швидше та простіше. Верхня межа для асинхронного Розпізнавання мовлення – 480 хвилин (8 годин). Пакетне розпізнавання мовлення може транскрибувати лише аудіо, збережене в хмарному сховищі. Вихід після транскрибування може бути поданий у лінії відповідей (для однофайлних запитів на розпізнавання пакетів) або записаний у хмарне сховище. При цьому користувач може дізнатися, коли операція завершена та будуть доступні результати транскрибування [19].

При цьому необхідно враховувати обмеження контенту [20]. Контент для хмарного перетворення мовлення на текст надається у вигляді аудіоданих, або безпосередньо в полі вмісту запиту, або за посиланням у URI хмарного сховища в полі URI запиту. Аудіоконтент можна надсилати безпосередньо в хмарний формат STT з локального файлу, або Хмарне STT може обробляти аудіоконтент. Існує обмеження в 10 МБ для всіх окремих запитів, надісланих до API за допомогою локальних файлів. У випадку методів Recognize та LongRunningRecognize це обмеження застосовується до розміру надісланого запиту. У випадку методу StreamingRecognize обмеження в 10 МБ застосовується як до початкового запиту StreamingRecognize, так і до розміру кожного окремого повідомлення в потоці. Перевищення цього обмеження призведе до помилки. Немає обмеження розміру для запитів, надісланих за допомогою аудіоданих, що зберігаються в корзині хмарного сховища. API містить такі обмеження щодо розміру цього вмісту (і є з можливістю зміни): синхронні запити: ≈ 1 хвилина; асинхронні запити: ≈ 480 хвилин; запити на трансляцію: ≈ 5 хвилин.

Для запитів StreamingRecognize аудіо має надсилатися зі швидкістю, що наближається до реального часу. Спроба обробити контент, що перевищує ці обмеження, призведе до помилки. У будь-якому запиті також можна надати

ресурс PhraseSet, що містить список фраз, характерних для запиту. (Одне слово вважається фразою в цьому контексті.) До такого контексту застосовуються такі обмеження: фраз на запит – 5000; загальна кількість символів на запит – 100000; символів на фразу – 100.

Поточні ліміти використання API для хмарного перетворення мовлення на текст такі (і можуть змінюватися): запити на розпізнавання за 60 секунд – 900; запити на ресурси адаптації за 60 секунд – 10; обробка аудіо за день – 480 годин. В даному випадку, Кожен сеанс StreamingRecognize вважається окремим запитом, навіть якщо він містить кілька кадрів аудіо StreamingRecognizeRequest у потоці. Запити або спроби обробки аудіо, що перевищують ці обмеження, призведуть до помилки. Щоденні квоти оновлюються опівночі (0:00) за тихоокеанським стандартним часом (PST) або за літнім часом (PDT), залежно від пори року.

Для глобальної бази існує широка підтримка мов [21]. У Chirp 3 транскрибування було створене за допомогою самостійного навчання на мільйонах годин аудіо та 28 мільярдах речень тексту, що охоплюють понад 100 мов. Потокове розпізнавання (у реальному часі) передбачає, що API обробляє аудіовхід, що передається з мікрофона від ПЗ користувача або надсилається з заздалегідь записаного аудіофайлу. STT працює у багатоканальних ситуаціях (відеоконференція) і анотувати транскрибування для збереження порядку. Сервіс може обробляти шумний звук з багатьох середовищ без необхідності додаткового шумозаглушення.

Існують доменно-специфічні моделі для керування голосом, транскрибування телефонних дзвінків і відео, оптимізованих для специфічних вимог до якості, специфічних для домену [22]. Наприклад, модель для телефонного дзвінка налаштована на аудіо з частотою дискретизації 8 кГц. Існує фільтр нецензурної лексики, який допомагає виявляти недоречний або непрофесійний контент у аудіоданих і фільтрувати нецензурні слова у текстових результатах. У бета-версії є автоматична пунктуація, наприклад, вказуючи коми, знаки питання та крапки. Також є

функція діаризації, щоб дізнатись, хто що сказав, отримуючи автоматичні прогнози щодо того, хто з спікерів у розмові вимовив кожне слово. У Google Cloud пропонується два варіанти використання транскрибування: через API (дві модифікації: V1 і V2), та у Agent Studio. У першому випадку є легка інтеграція транскрипції в існуюче ПЗ, а також створення масштабованих, корпоративних додатків. Chirp 3 з API V2 підтримує транскрибування для понад 85 мов, діаризацію, адаптацію моделей, автоматичне розпізнавання мовлення та транскрибування STT, багатомовне виявлення та транскрибування. Хоча, згідно технічної документації, не всі функції можуть бути реалізовані для мало представлених мов у Chirp 3 (рис. 2.1).

Location	Name	BCP-47	Model	Automatic punctuation	Diarization	Model adaptation	Word-level confidence	Profanity filter
europe-west4	Ukrainian (Ukraine)	uk-UA	chirp	✓				
europe-west4	Ukrainian (Ukraine)	uk-UA	chirp_2	✓		✓	✓	✓

Рисунок 2.1 – Функціонал Chirp для української мови

У другому випадку забезпечується швидке тестування аудіофайлів і прототипування, створення аудіотранскрибування, завантаження аудіо або запису безпосередньо у веб-браузер. Також реалізований простий у використанні веб-орієнтований GUI без коду. Знову ж підтримується транскрибування для понад 85 мов, діаризація, адаптація моделей, автоматичне розпізнавання мовлення та транскрибування STT, багатомовне виявлення та транскрибування. Враховуючі те, що корпоративні клієнти прагнуть мати повний контроль над своєю інфраструктурою та захищеними мовними даними та одночасно застосовувати технології транскрибування, Google може дозволити їх застосовувати локально [23], прямо у власних корпоративних дата-центрах. Але такий підхід потребує додаткових перемовин з к. Google.

Більш простий варіант полягає у використанні API V2. Це надає корпоративним і бізнес-клієнтам реалізувати додаткові вимоги до безпеки та регуляторних вимог одразу. Резиденція даних [24] дозволяє використовувати моделі транскрибування через повністю регіональни сервіс, що використовує регіони Google Cloud, такі як Сінгапур і Бельгія. Логи для генерації ресурсів і транскрибування легко доступні в консолі Google Cloud. Крім цього API v2 пропонує корпоративне шифрування з керованими клієнтом ключами [25] шифрування для всіх ресурсів, а також пакетне транскрибування. За замовчуванням, шифрування з ключами шифрування, що належать Google, керуються Google, Усі дані, що зберігаються в Google Cloud, шифруються у стані спокою за допомогою тієї системи управління ключами, які Google Cloud використовує для власних зашифрованих даних. Ці системи управління ключами забезпечують суворий контроль доступу до ключів а також аудит і шифрування даних користувачів у стані спокою за допомогою шифрування AES-256 (стандартно). Google Cloud володіє та контролює ключі, які використовуються для шифрування даних клієнта. Тобто, він не може переглядати або керувати цими ключами, а також переглядати журнали використання ключів. Дані від кількох клієнтів можуть використовувати один і той самий ключ шифрування ключів (KEK). В такому випадку непотрібна підготовка конфігурація або управління.

Ключі шифрування, що керовані клієнтом, дозволяють мати більший контроль над ключами, що використовуються для шифрування даних, що зберігаються в підтримуваних сервісах Google Cloud. Це забезпечує криптографічну межу навколо даних клієнта. Керувати ключами CMEK можна безпосередньо у Cloud KMS або автоматизувати надання та призначення за допомогою автоматичного ключа Cloud KMS Autokey [26]. Сервіси підтримують CMEK. Інтеграція CMEK – це технологія шифрування на стороні сервера, яку можна використовувати замість шифрування Google Cloud за замовчуванням. Після налаштування CMEK операції з шифрування та розшифрування ресурсів обробляються агентом служби ресурсів. Оскільки

сервіси, інтегровані з CMEK, обробляють доступ до зашифрованого ресурсу, шифрування та розшифрування можуть відбуватися прозоро, без зусиль кінцевого користувача. Хмарний KMS Autokey спрощує створення та управління CMEK шляхом автоматизації постачання та призначення. З Autokey брелоки та ключі є генеруються за запитом у рамках створення ресурсів, а також агенти сервісу, які використовують ключі для операцій шифрування та дешифрування автоматично надаються необхідні ролі з управління ідентичністю та доступом (IAM). Використання автоключа простіше ніж самостійно набирати ключі, і це рекомендований вибір, якщо ключі створено Autokey і відповідає всім вимогам користувача. Використання ключів, згенерованих Autokey, допоможе послідовно узгоджуватися з вимогами стандартів і рекомендованих практик безпеки даних. Все ж таки варто пам’ятати, що для такого варіанту хмарного рішення транскрибування здійснюється передача даних у хмару. Це не завжди є прийнятним для деяких відомств чи корпорацій. Тому надалі доцільно розглянути варіанти реалізації локальних рішень для транскрибування.

2.2 Принцип роботи локального AI-агента для транскрибування

Локальний AI-агент для транскрибування аудіо у текст є програмною системою, яка виконує обробку мовленнєвої інформації без передавання даних до зовнішніх хмарних сервісів [27]. Основна ідея такого підходу полягає в тому, що всі етапи роботи з аудіозаписом відбуваються на локальному комп’ютері, сервері або іншому пристрої організації. Це дозволяє зберігати контроль над аудіофайлами, текстовими результатами та подальшими аналітичними висновками. Принцип роботи локального AI-агента можна подати як послідовність кількох етапів: аудіозапис → попередня обробка аудіо → транскрибування → аналіз тексту → формування результату (рис. 2.2). Спочатку користувач завантажує аудіофайл або передає

голосове повідомлення до системи. Це може бути запис телефонної розмови, консультації, звернення користувача, голосова нотатка або інший аудіоматеріал. Після цього система виконує попередню обробку аудіо: перевіряє формат файлу, за потреби перетворює його у потрібний формат, вирівнює гучність, видаляє зайві фрагменти або готує аудіо до розпізнавання.

Рисунок 2.2 – Принцип роботи локального AI-агента

Наступним етапом є транскрибування. Для цього використовується модель ASR, яка аналізує аудіосигнал і перетворює його у текст. Вона визначає мовні одиниці, зіставляє їх із відповідними словами та формує текстову розшифровку. У результаті користувач отримує транскрипт, тобто текстову версію аудіозапису. Після отримання тексту локальний AI-агент може виконувати додаткову інтелектуальну обробку. Якщо система використовується для AI-аналітики звернень, агент може також визначати тип проблеми, рівень терміновості, емоційне забарвлення повідомлення та можливі подальші дії. Важливою особливістю локального AI-агента є те, що він може працювати з внутрішньою базою знань організації (рис. 2.3).

У такій структурі транскрибування є базовим етапом, оскільки саме воно перетворює мовленнєву інформацію у текстову форму, з якою надалі може працювати AI-агент. Без якісної транскрипції подальший аналіз звернення буде менш точним, тому вибір моделі розпізнавання мовлення має важливе значення для всієї системи. Локальний режим роботи також впливає

на конфіденційність. Оскільки аудіофайли, транскрипти та результати аналізу залишаються в межах локальної інфраструктури, зменшується ризик несанкціонованого доступу до даних під час їх передавання через мережу. Це особливо важливо у випадках, коли в аудіозверненнях містяться персональні дані, службова інформація, комерційна таємниця або інші чутливі відомості.

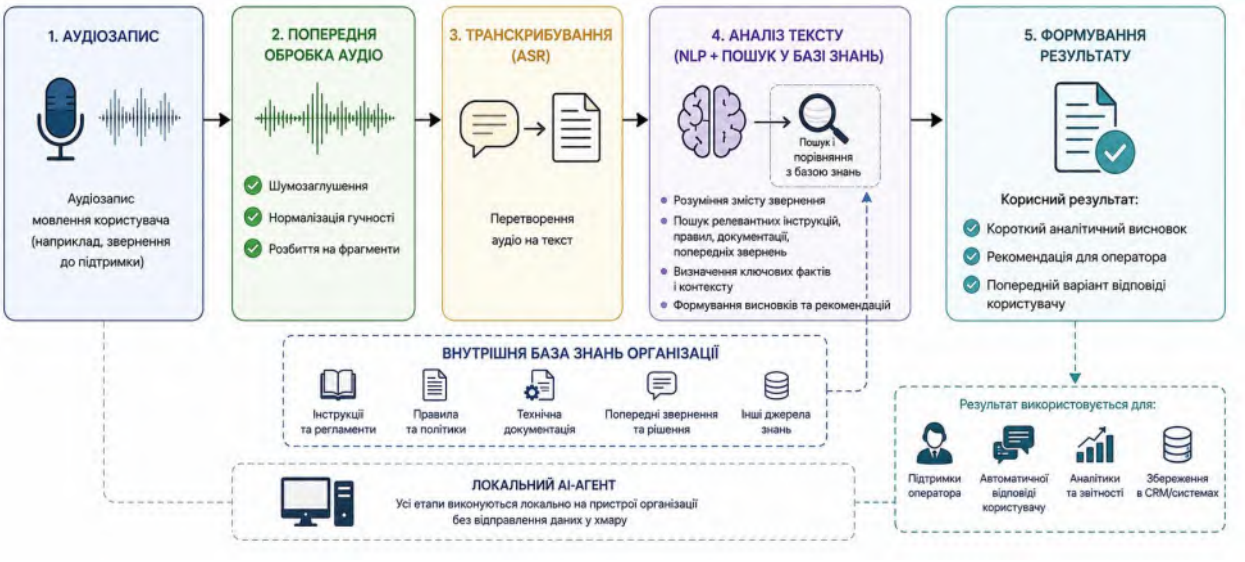

Рисунок 2.3 – Використання внутрішньої бази знань організації

Таким чином, локальний AI-агент для транскрибування виконує не лише технічну функцію перетворення аудіо у текст, а й виступає частиною більш широкої системи інтелектуальної обробки звернень. Його використання дозволяє автоматизувати роботу з мовленнєвою інформацією, підвищити швидкість аналізу аудіоданих і водночас забезпечити більший рівень конфіденційності порівняно з хмарними сервісами.

При цьому може використовуватись технологію пошуково-підкріпленої генерації (Retrieval-Augmented Generation, RAG) – рис. 2.4 [28]. Її основна ідея полягає у поєднанні можливостей мовної моделі з пошуком інформації у внутрішній базі знань організації. Це дозволяє AI-агенту не лише виконувати транскрибування аудіо у текст, а й аналізувати отриману інформацію з урахуванням наявної документації та накопичених даних. Після перетворення аудіозапису у текст система за допомогою RAG-модуля здійснює пошук

релевантної інформації у внутрішніх джерелах даних. Такими джерелами можуть бути технічна документація, інструкції, правила роботи, база типових помилок, попередні звернення користувачів або внутрішні регламенти організації. На основі знайденої інформації мовна модель формує більш точний та змістовний результат.

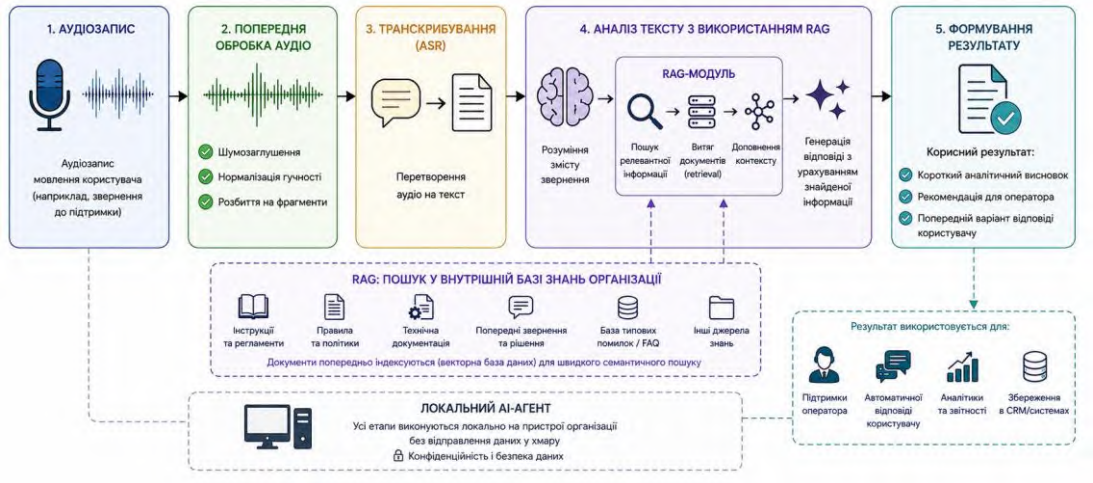

Рисунок 2.4 – Застосування технології RAG

Використання RAG у локальному AI-агенті дозволяє автоматично створювати короткі аналітичні висновки, рекомендації для оператора, попередні відповіді користувачам або визначати можливі причини проблеми. Крім цього, локальне розгортання такої системи забезпечує конфіденційність даних, оскільки обробка аудіо, тексту та внутрішньої документації виконується без передачі інформації у хмарні сервіси. На відміну від хмарних сервісів, локальний AI-агент забезпечує більшу автономність, конфіденційність і незалежність від зовнішньої інфраструктури. Це особливо важливо для сфер, де обробляються персональні, службові, медичні, освітні або комерційні дані.

Для реалізації локального AI-агента з використанням технології RAG важливе значення має апаратна частина системи, оскільки всі етапи обробки даних виконуються без використання хмарних сервісів безпосередньо на локальному пристрої або сервері організації. Тобто, система повинна мати

достатній рівень обчислювальних ресурсів для стабільної та швидкої роботи. Основною складовою апаратної платформи є CPU, який забезпечує виконання базових операцій системи, роботу ПЗ, обробку аудіосигналів та взаємодію між окремими модулями AI-агента. Для невеликих локальних систем можуть використовуватись сучасні багатоядерні процесори рівня Intel Core i5 / Ryzen 5 або енергоефективні одноплатні комп'ютери, наприклад Raspberry Pi 5 (рис. 2.5) [29]. Проте для роботи великих мовних моделей та RAG-систем бажано використовувати продуктивніші процесори з більшою кількістю ядер та підтримкою сучасних інструкцій обробки даних.

Рисунок 2.5 – SBC Raspberry Pi 5 (16GB)

Важливим компонентом системи є ОЗП, оскільки мовні моделі, ASR-системи та векторні бази даних можуть займати значний обсяг пам'яті. Для базових локальних AI-агентів мінімально рекомендованим є обсяг 16 ГБ ОЗП, тоді як для моделей середнього розміру та RAG-модуля доцільно використовувати 32 ГБ і більше. Недостатній обсяг ОЗП може призводити до затримок у роботі системи або неможливості запуску окремих моделей.

Для прискорення роботи AI-моделей доцільним є використання GPU. Саме він забезпечує швидке виконання операцій нейронних мереж під час транскрибування аудіо, генерації тексту та пошуку релевантної інформації у RAG-системі. Для локального розгортання можуть використовуватися відеокарти NVIDIA з підтримкою CUDA, наприклад серії RTX. Наявність

GPU значно скорочує час обробки запитів та дозволяє використовувати складніші мовні моделі. Однак у деяких випадках можливе функціонування системи лише на CPU, хоча швидкодія при цьому буде нижчою. Технологія RAG передбачає використання внутрішньої бази знань, яка може містити документацію, інструкції, історію звернень та інші текстові матеріали. Для швидкого доступу до таких даних рекомендується використання SSD-накопичувачів, оскільки вони забезпечують вищу швидкість читання та запису інформації порівняно зі звичайними жорсткими дисками. Крім цього, частина дискового простору використовується для зберігання мовних моделей, векторних баз даних та кешу системи. Апаратна частина локального AI-агента повинна забезпечувати достатню продуктивність для одночасної роботи ASR-модуля, мовної моделі та RAG-системи. Правильно підібрана конфігурація обладнання дозволяє забезпечити стабільність роботи, швидку обробку звернень користувачів та конфіденційність даних завдяки локальному виконанню всіх обчислень. Таким чином, поєднання технологій автоматичного розпізнавання мовлення та локального AI дозволяє створити ефективний інструмент для AI-аналітики мовленнєвої інформації, який може бути використаний у службах підтримки, освіті, медицині, державному секторі, бізнесі, медіа та інших галузях. Використання транскрибування у поєднанні з локальним AI-агентом є актуальним через зростання обсягів аудіоінформації та потребу в її швидкій автоматизованій обробці. Перетворення мовлення в текст створює основу для подальшого аналізу, пошуку, класифікації та узагальнення інформації.

2.3 Використання технології виявлення мовленнєвої активності

У системах транскрибування аудіо важливим етапом є попередня обробка звукового сигналу. Однією з технологій, яка може застосовуватися на цьому етапі, є виявлення мовленнєвої активності (Voice Activity Detection,

VAD) [30]. Її основне завдання полягає в тому, щоб визначити, у яких фрагментах аудіозапису присутнє мовлення людини, а які частини містять тишу, паузи, фоновий шум або інші звуки, що не несуть корисної мовленнєвої інформації.

У локальному AI-агенті для транскрибування VAD може виконувати роль допоміжного модуля, який готує аудіозапис до подальшого розпізнавання. Перед тим як передати файл до моделі автоматичного розпізнавання мовлення, система аналізує аудіосигнал і відокремлює ділянки, де є голос. Наприклад, якщо в аудіозаписі є довгі паузи, тиша на початку або в кінці файлу, шумові фрагменти без мовлення, то VAD дозволяє їх не обробляти або обробляти окремо. Завдяки цьому модель транскрибування працює тільки з тими частинами аудіо, які справді містять мовлення. Загальний принцип роботи такого підходу можна описати так: аудіозапис → VAD-аналіз → виділення мовленнєвих фрагментів → транскрибування → обробка тексту локальним AI-агентом (рис. 2.5).

Рисунок 2.5 – VAD-конвеєр у локальному AI-агенті

Використання VAD є особливо корисним у задачах обробки звернень користувачів, телефонних розмов, голосових повідомлень або записів консультацій. У таких аудіофайлах часто трапляються паузи, фонові звуки, очікування відповіді оператора, технічні шуми або фрагменти без корисної інформації. Якщо передавати весь запис на транскрибування без попереднього аналізу, система витрачатиме більше часу та ресурсів. VAD

допомагає зменшити обсяг аудіо, яке потрібно обробляти, і тим самим підвищує ефективність роботи локального AI-агента.

Ще однією перевагою VAD є покращення структурування аудіозапису. Система може поділити довгий запис на окремі мовленнєві фрагменти, які потім простіше транскрибувати, аналізувати та зберігати. Це важливо для локального AI-агента, оскільки він може працювати не з одним великим текстом, а з окремими логічними частинами розмови. Наприклад, після транскрибування агент може окремо аналізувати вступ, основну проблему користувача, уточнення деталей і завершення звернення. У контексті локального розгортання VAD також має практичне значення для економії ресурсів. Локальний AI-агент працює на комп'ютері або сервері організації, його продуктивність залежить від наявного обладнання. Якщо модель транскрибування обробляє тільки фрагменти з мовленням, зменшується навантаження на процесор, оперативну пам'ять або відеокарту. Це особливо важливо, коли система використовується при обмежених ресурсах або коли потрібно обробляти багато аудіозаписів. VAD може позитивно впливати і на якість транскрибування. Якщо модель розпізнавання мовлення отримує аудіо, у якому менше тиші та сторонніх звуків, результат може бути більш стабільним. Крім того, система може уникати ситуацій, коли модель намагається «розпізнати» шум або тишу як слова. Це зменшує кількість зайвих або помилкових фрагментів у текстовій розшифровці.

Для локального AI-агента VAD можна розглядати як один із елементів інтелектуального конвеєра обробки аудіо. Спочатку модуль VAD визначає, де в записі є мовлення. Далі ці фрагменти передаються до ASR-моделі, яка перетворює їх у текст. Після цього локальний AI-агент виконує аналіз тексту: визначає тему звернення, виділяє ключові слова, формує короткий зміст, класифікує проблему або готує рекомендації для оператора чи адміністратора.

Важливо, що використання VAD у локальній системі не потребує передавання аудіоданих до зовнішніх сервісів. Усі етапи – від виявлення мовлення до транскрибування та подальшої аналітики – можуть виконуватися

всередині локальної інфраструктури. Це відповідає вимогам конфіденційності, оскільки аудіозаписи, голосові повідомлення та текстові результати залишаються під контролем організації.

Разом з тим VAD має певні обмеження. Якщо аудіозапис має дуже сильний фоновий шум, низьку якість або кілька людей говорять одночасно, система може неточно визначати межі мовлення. У деяких випадках вона може обрізати початок або кінець фрази, або навпаки – залишати частини шуму як мовленнєві фрагменти. Тому під час налаштування локального AI-агента потрібно враховувати якість аудіо, тип записів і параметри чутливості VAD. Отже, використання VAD у локальному AI-агенті транскрибування є доцільним, оскільки ця технологія дозволяє попередньо відокремити мовлення від тиші та шумів, зменшити обсяг аудіо для обробки, підвищити швидкість роботи системи та покращити якість отриманого тексту. У поєднанні з моделлю автоматичного розпізнавання мовлення та модулем AI-аналітики VAD допомагає побудувати більш ефективну, автономну й конфіденційну систему обробки звернень користувачів.

2.4 Моделі для транскрибування на основі локального рішення

Для реалізації транскрибування на українській мові в роботі виконаний аналіз кількох сімейств. Моделі Vosk [31] мають два типи – великі та малі. Малі моделі ідеально підходять для виконання обмежених завдань у мобільних додатках. Вони можуть працювати на смартфонах, Raspberry Pi. Вони також рекомендуються для настільних додатків. Маленька модель, зазвичай, має розмір близько 50 МБ і вимагає близько 300 МБ пам'яті при виконанні. Великі моделі призначені для високоточного транскрибування на сервері. Їм потрібно до 16 ГБ пам'яті, оскільки вони застосовують передові алгоритми AI. Їх слід запускати на високопродуктивних серверах, таких як i7 або останній AMD Ryzen. На AWS можна поглянути на машини c5a та

подібні машини в інших хмарах. Більшість малих моделей дозволяють динамічну реконфігурацію словника. Великі моделі є статичними, словник не можна змінювати при роботі. Список моделей (сумісних з Vosk-API) для роботи з українською мовою наведений у табл. 2.1. Чим більше використовується модель, тим точніше, теоретично, вона перетворює мова в текст і тим повільніше вона працює, а так само більше займає ресурсів (місце на диску і місце в пам'яті) комп'ютера. Адаптація мовної моделі Vosk дозволяє додати слова. Традиційно моделі Vosk компілюють джерела даних для побудови графа розпізнавання: акустична модель (модель звуків мови) – мовна модель (модель послідовностей слів). Фонетичний словник – модель розкладання слів на звуки. Таке розділення аспектів має переваги та недоліки. Можна швидко замінити джерело знань, наприклад, ввести нове слово з нестандартною вимовою (можливо, технічний термін). Можна навчити свою модель на одній області та використовувати для іншої області, просто замінивши мовну модель. При цьому можна налаштувати акустичну модель окремо. Система може бути компактною та може навчатися на чистих текстах.

Таблиця 2.1 – Список моделей Vosk для роботи з українською мовою

Модель	Розмір, МБ	Примітка	Ліцензія
vosk-model-small-uk-v3-nano	73	Наномодель з розпізнавання мовлення для української мови	Apache 2.0
vosk-model-small-uk-v3-small	133	Невелика модель з розпізнавання мовлення для української мови	Apache 2.0
vosk-model-uk-v3	343	Більша модель з розпізнавання мовлення для української мови	Apache 2.0
vosk-model-uk-v3-lgraph	325	Велика динамічна модель від розпізнавання мовлення для української мови	Apache 2.0

Слова компілюються. Новіші системи використовують нейронну мережу, яка компілює як мовну модель, так і вимову, та акустичну модель в єдину монолітну модель. Система точніша на даних, на яких вона навчається, оскільки вона вивчає складніші залежності даних. Декодування ще швидше.

Не потрібно вказувати фонетичний словник. Міждоменне перенесення складніше, щоб ввести нове слово, потрібно застосовувати складні архітектурні трюки або навчатися на аудіоданих.

Моделі Whisper зараз є самими поширеними у завдання транскрибування і досить детально розглянуті у відкритих джерелах. І Vosk і Whisper виконують одну глобальну ціль – перетворення слова в текст (STT), для подальшої роботи з останнім. Всі вони можуть працювати на відеокарті, і в принципі всі ці перетворювачі мови в текст (або назад) намагаються запускати саме на відеокарті, так як це дає зовсім інші швидкості, ніж запуск на CPU. З Vosk іноді виникають деякі проблеми, щоб скомпілювати під використання на відеокарті. Це різні проекти, незважаючи на одну ціль, у них різні можливості «з коробки». Тому доцільно виконати їх порівняння. У Vosk, глобально, існують лише два види моделей (big і small), а у Whisper їх прямий широкий вибір (tiny, base, small, medium, large). Також варто відзначити, що у Whisper модель відразу визначає пунктуацію і великі літери, ніж у Vosk. У Vosk для пунктуації треба використовувати спеціальну модель, у цьому між моделями Vosk та Whisper принципова відмінність. Ще варто згадати і те, що моделі Whisper можуть бути під англійську мову, тоді в назві є «en» або можуть бути мультимовними. У Vosk у цьому плані кожна модель заточена під певну мову. Аудіофайли, що подаються на вхід, повинні відповідати вимогам. Для Vosk та Whisper це: wav формат аудіофайлу, моно канал, 16000 Гц частота дискретизації, 16 бітів динамічний діапазон, тривалість не більше 30 секунд. Щодо деяких пунктів, варто зробити невелике пояснення. Vosk може працювати і на більшій частоті дискретизації (суб'єктивно це дає краще розпізнавання), але у Whisper це 16000 Гц. Whisper може працювати на глибині динамічного діапазону 8 бітів та 32 біти, але у Vosk це лише 16 бітів. У Whisper аудіо файл не може бути тривалішим 30 секунд. Якщо перевищити це значення, то контекст почнеться заново (він як би переповниться) і буде розпізнавати звук (слова в аудіо файлі), що йде далі, без урахування попереднього. Це пов'язано з розміром контекстного вікна нейронної мережі

(у будь-якій моделі Whisper), яка лежить в основі. Тому тут варто згадати про проект Whisper Fast, що є перспективним розвитком Whisper. В цілому, початкове застосування Whisper – це потокове розпізнавання, і звідси такі обмеження та особливості. Проведені тести на звичайному розпізнаванні, а не стрімінговому показують, що швидкість розпізнавання у Vosk вища, причому приблизно на порядок, а іноді навіть більше, ніж у Whisper, при цьому якість/точність теж краща. Сильна різниця у часі завантаження моделей. Середня або велика модель Vosk завантажується близько 2-3 хвилин. Whisper у цьому плані значно швидше і навіть найбільша модель завантажується протягом декількох секунд. У Vosk будь-яка модель повністю вантажиться в ОЗП і займає в ній: розмір моделі на диску приблизно 30-50 % від цього розміру. Наприклад, модель з розміром 3,5 ГБ на диску, займатиме близько 5,5 ГБ ОЗП. Whisper у цьому плані в пам'яті тримає рівно той розмір, що модель займає на диску або навіть декілька менше. Наприклад, модель ggml large v3.bin, розміром 3 ГБ на диску, в пам'яті займає стільки ж або дещо менше. Whisper, у момент роботи/розпізнавання завантажує процесор приблизно на 16-20 %, Vosk завантажує на 5-8 %. ОЗП, що споживається, не підвищується ні у Vosk, ні у Whisper. Варто звернути увагу, що крім STT у Whisper має вбудовану функцію перекладу тексту іншою мовою, цього немає у Vosk. У Vosk є модель vosk model spk-0.4, яка є багатомовною.

2.5 Пакетна обробка при транскрибуванні

Пакетна обробка при транскрибуванні – це підхід, за якого система обробляє не один аудіофайл у ручному режимі, а цілу групу аудіозаписів. Тобто користувач або адміністратор може завантажити декілька файлів, після чого система автоматично виконує їх послідовне або паралельне транскрибування. Такий підхід є зручним у випадках, коли потрібно обробити велику кількість записів: телефонні розмови, голосові

повідомлення, інтерв'ю, лекції, наради або звернення користувачів. На відміну від транскрибування в реальному часі, пакетна обробка зазвичай застосовується для вже готових аудіофайлів. Принцип пакетної обробки можна подати як послідовність етапів (рис. 2.6): набір аудіофайлів → попередня обробка → транскрибування кожного файлу → збереження транскриптів → AI-аналіз результатів. На першому етапі система отримує список аудіозаписів, які потрібно обробити. Це можуть бути файли, наприклад WAV,. Далі виконується попередня підготовка аудіо: перевірка формату, перетворення у потрібний вигляд, нормалізація гучності, видалення зайвих пауз або застосування VAD для визначення мовленнєвих фрагментів. Після цього кожен файл потрапляє до модуля ASR.

Рисунок 2.6 – Сутність пакетної обробки

Важливою перевагою пакетної обробки є автоматизація рутинної роботи. Якщо аудіозаписів багато, їх ручне завантаження та обробка займають багато часу. Пакетний режим дозволяє один раз вказати папку з файлами або список записів, після чого система самостійно виконує транскрибування. Це особливо корисно для служб підтримки, освітніх установ, медіаорганізацій та підприємств, де регулярно накопичуються великі обсяги аудіоданих. У локальному AI-агенті пакетна обробка має ще перевагу – вона може виконуватися без передавання даних до хмарних сервісів. Усі аудіозаписи, текстові транскрипти та результати аналізу залишаються в межах локального комп'ютера або сервера організації. Це

підвищує рівень конфіденційності, оскільки аудіофайли можуть містити персональні дані, службову інформацію або комерційні відомості. Після завершення транскрибування локальний AI-агент може виконувати додаткову обробку отриманих текстів. Наприклад, для кожного звернення він може визначити тему, тип проблеми, ключові слова, рівень терміновості, короткий зміст і можливі рекомендації. Якщо обробляється велика кількість звернень, система може також формувати узагальнену статистику: які питання виникають найчастіше, які проблеми повторюються, які звернення потребують швидкого реагування. Пакетна обробка також дозволяє раціональніше використовувати обчислювальні ресурси. Такий підхід є вдалим для локальних рішень, де продуктивність залежить від наявного обладнання. Разом з перевагами пакетна обробка має певні обмеження. Якщо аудіофайлів дуже багато або вони мають велику тривалість, процес транскрибування може займати значний час. Також потрібно передбачити обробку можливих помилок: пошкоджених файлів, неправильних форматів, надто шумних записів або ситуацій, коли модель не може якісно розпізнати мовлення. Система повинна зберігати інформацію про статус обробки кожного файлу: успішно оброблено, виникла помилка, файл очікує в черзі або потребує повторного запуску. Для підвищення зручності система може формувати окремий результат для кожного аудіозапису. Після обробки для кожного файлу можна створювати текстовий документ із транскриптом, короткий підсумок, файл із ключовими словами або таблицю з результатами AI-аналітики. Пакетна обробка при транскрибуванні працює з великою кількістю аудіозаписів. Вона дозволяє автоматизувати процес перетворення мовлення в текст, зменшити ручну працю, ефективніше використовувати ресурси та забезпечити подальший аналіз отриманих транскриптів. У контексті локального AI-агента пакетна обробка є особливо доцільною, оскільки поєднує автоматизацію, конфіденційність і можливість виконання AI-аналітики звернень без використання зовнішніх хмарних сервісів.

РОЗДІЛ 3
РЕКОМЕНДАЦІЇ ЩОДО ПРАКТИЧНОЇ РЕАЛІЗАЦІЇ ЛОКАЛЬНОГО ТРАНСКРИБУВАННЯ АУДІО В ТЕКСТ

3.1 Метрики для оцінювання якості транскрибування

Для оцінювання якості транскрибування доцільно враховувати практичні критерії якості транскрибування, наприклад: правильність розпізнавання ключових слів; збереження змісту висловлювання; коректність розстановки розділових знаків; наявність часових міток; підтримку потрібної мови; стійкість до шумів; роботу з різними форматами аудіо; можливість пакетної обробки. Крім того існують класичні кількісні метрики точності системи розпізнавання мовлення. Це особливо важливо у випадку порівняння різних моделей ASR або під час перевірки роботи локального AI-агента для транскрибування. Основна ідея оцінювання полягає в тому, що результат роботи системи порівнюється з правильним текстом, який заздалегідь підготовлений людиною. Такий правильний текст називають еталонною транскрипцією. Після цього автоматично отриманий текст порівнюється з еталоном, і на основі кількості помилок визначається якість розпізнавання.

Найбільш поширеною метрикою для оцінювання якості транскрибування є коефіцієнт помилок на рівні слів (Word Error Rate, WER). Вона показує, яка частка слів була розпізнана неправильно. При розрахунку *WER* враховуються три основні типи помилок: заміна слова, пропуск слова та вставка зайвого слова:

$$WER = \frac{S + D + I}{N} \cdot 100\%, \tag{3.1}$$

де S – кількість замінених слів;

D – кількість пропущених слів;

I – кількість зайвих вставлених слів;

N – загальна кількість слів в еталонному тексті.

Наприклад, якщо в еталонному тексті було 100 слів, а система допустила 5 замін, 3 пропуски та 2 зайві вставки, тоді: $WER = 10\%$. Чим менше значення WER, тим кращою є якість транскрибування.

Однак WER не завжди повністю відображає якість результату, особливо для мов зі складною морфологією, до яких належить і українська мова. Наприклад, система може помилитися лише в закінченні слова, але на рівні WER це все одно буде вважатися повною помилкою. Тому додатково може використовуватися коефіцієнт помилок на рівні символів (Character Error Rate, CER). Метрика *CER* розраховується подібно до *WER*, але замість слів порівнюються окремі символи. Вона дозволяє точніше оцінити, наскільки сильно відрізняється автоматично отриманий текст від правильного. Наприклад, якщо система розпізнала слово майже правильно, але допустила одну-дві помилки в літерах, *CER* покаже меншу помилку, ніж *WER*. Формула CER має такий вигляд:

$$CER = \frac{S + D + I}{N} \cdot 100\%, \tag{3.2}$$

де S – кількість замінених символів;

D – кількість пропущених символів;

I – кількість зайвих вставлених символів;

N – загальна кількість символів в еталонному тексті.

Окрім точності розпізнавання, важливо оцінювати і швидкодію системи. Для цього може використовуватися показник Real Time Factor (RTF). Він показує, скільки часу система витрачає на обробку аудіозапису відносно його тривалості:

$$RTF = \frac{T_{обробки}}{T_{аудіо}}, \tag{3.3}$$

де $T_{обробки}$ – час, який система витратила на транскрибування;

$T_{аудіо}$ – тривалість аудіозапису.

Якщо *RTF* дорівнює 1, система працює приблизно в реальному часі. Якщо *RTF* більше 1, то обробка триває довше, ніж сам аудіозапис. Тобто,

швидкість роботи залежить від потужності комп’ютера, процесора, відеокарти та обраної моделі. Також для оцінювання системи можна враховувати стабільність роботи. Наприклад, під час пакетної обробки важливо перевірити, чи може система обробити кілька аудіофайлів без збоїв, чи зберігає результати для кожного файлу, чи правильно обробляє помилки у випадку пошкодженого або неякісного аудіо. Для системи транскрибування, яка використовується разом з локальним AI-агентом, важливо оцінювати не лише сам текст, а й придатність транскрипту для подальшого аналізу. Як наслідок, треба додатково оцінювати наступні властивості.

1. Точність розпізнавання ключових слів – саме ключові слова визначають проблеми користувача.
2. Якість розпізнавання в умовах шуму – реальні аудіозаписи можуть бути гіршої якості (фонові звуки, завади, низьку якість мікрофона або паузи.
3. Якість роботи з довгими записами – деякі моделі можуть добре працювати з короткими фрагментами, але мати труднощі з довгими аудіозаписами.
4. Зручність подальшого аналізу тексту. Транскрипт повинен бути достатньо зрозумілим для локального AI-агента, щоб той міг виконувати класифікацію, виділення основної проблеми, формування короткого змісту та рекомендацій.
5. Коректність розбиття на фрагменти. Якщо в системі використовується VAD, потрібно оцінити, чи правильно відокремлює ться мовлення від тиші, пауз і шумів. Помилки на цьому етапі можуть призвести до втрати частини мовлення або появи зайвих фрагментів у транскрипті.

Для проведення оцінювання можна сформувати тестовий набір аудіозаписів. До нього варто включити декілька типів аудіо: короткі та довгі записи, чисте мовлення, записи з фоновим шумом, голосові повідомлення або приклади звернень користувачів. Для кожного аудіофайлу потрібно підготувати еталонний текст, після чого порівняти його з результатом автоматичного транскрибування. Результати оцінювання можна подати у

вигляді таблиці, де для кожного аудіозапису вказати тривалість, умови запису, значення *WER*, *CER*, час обробки та *RTF*. Такий підхід дозволить не лише описати роботу системи, а й показати її ефективність на практиці.

3.2 Аналіз сучасних моделей на основі модифікацій Whisper

Модель Whisper [11] створена OpenAI. Вона перетворює аудіо в текст, підтримує багато мов, переклад у англійську та працює локально. Whisper став дуже популярним через хорошу якість навіть на шумних записах. На даний час є проблема звичайного Whisper: дає timestamps лише для фраз/сегментів; іноді помиляється з часом; не визначає спікерів; погано працює з довгими аудіо без додаткових інструментів [32]. Word timestamps – це часові мітки для кожного окремого слова в аудіо. Тобто система визначає: коли слово почалося, а коли слово закінчилося. WhisperX (рис. 3.1) – це надбудова над Whisper, яка додає: точні timestamps для кожного слова; diarization (визначення, хто говорить); швидшу пакетну обробку; VAD; forced alignment (вирівнювання тексту по фонемах) [33].

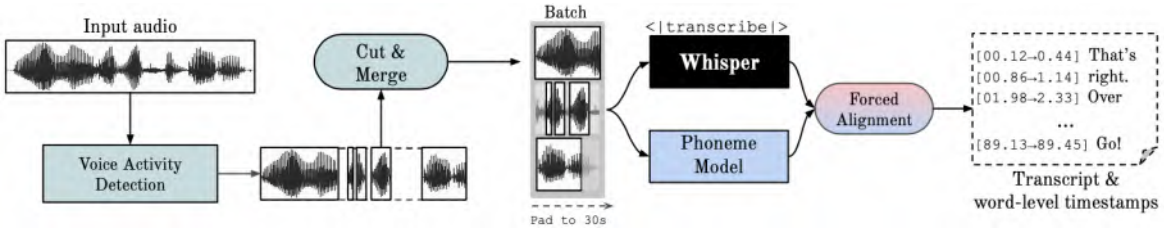

Рисунок 3.1 – Принцип роботи WhisperX «70x realtime with large-v2»

WhisperX комбінує кілька моделей: Whisper → робить базову транскрипцію; VAD → знаходить ділянки з мовленням; Wav2Vec2 або інша phoneme-модель → уточнює timestamps слів; Pyannote → визначає різних спікерів [34]. Таким чином, головні відмінності можна узагальнити – табл. 3.1. Faster Whisper [35] – це оптимізована реалізація Whisper на базі:

CTranslate2; C++; оптимізацій FP16 / INT8 та efficient inference. Оригінальна реалізація Whisper написана на PyTorch, проста для досліджень, споживає більше VRAM/RAM і є повільнішою. Faster-Whisper на відміну від оригіналу працює швидше (наприклад, згідно [36], до 4× швидше), менше використовує пам’ять, краще пристосований для використання CPU, добре підходить для локальних AI-агентів, а також підтримує quantization INT8.

Таблиця 3.1 – Головні відмінності Whisper і WhisperX

Функція	Whisper	WhisperX
Транскрипція	Так	Так
Word-level timestamps	Ні	Так
Speaker diarization	Ні	Так
VAD	Ні	Так
Точність часу	Середня	Висока
Робота з довгими аудіо	Обмежена	Краща
Batch inference	Ні	Так
Швидкість	Нормальна	Часто швидше

WhisperX і Faster-Whisper не є конкурентами. WhisperX часто сам використовує Faster-Whisper всередині [37]. Тому, якщо потрібне просте транскрибування – Whisper або Faster-Whisper, якщо потрібна швидкість – Faster-Whisper. Останній чинник досить актуальний для SBC Raspberry Pi, систем CPU-only, побудови локальних агентів, транскрибування в реальному часі. Якщо потрібні: timestamps для кожного слова, субтитри, визначення спікерів, AI-аналітика дзвінків (call-центрів) – WhisperX. Як наслідок, в роботі запропонований варіант архітектури локального AI-агента транскрибування на базі WhisperX, що наведений на рис. 3.2. Це дуже добре підходить під вимоги щодо конфіденційності, локальне розгортання, обробки звернень, AI-аналітику мовлення та ін. [27]. У WhisperX історично основним і найбільш протестованим варіантом був саме Whisper large-v2. Але зараз WhisperX може працювати також з моделлю large-v3. WhisperX використовує backend від Faster-Whisper, тому підтримує практично будь-які Whisper-моделі: large-v2, large-v3, large-v3-turbo, distil-large-v2, medium, small, tiny та інші моделі Faster-Whisper [38]. Варто мати на увазі, large-v3 не завжди

кращий варіант. Є цікава особливість: для деяких мов large-v3 точніший; для шумного аудіо large-v2 інколи стабільніший; large-v3 може більше hallucinate у певних сценаріях. У транскрибуванні hallucination – це не «галюцинація» в буквальному сенсі, а ситуація, коли ASR-модель додає у транскрипт слова, фрази або навіть цілі речення, яких людина в аудіо не вимовляла. Faster-Whisper і WhisperX реалізують пакетну обробку, але з певними відмінностями – табл. 3.2 [39].

Рисунок 3.2 – Архітектури локального AI-агента на базі WhisperX

Таблиця 3.2 – Відмінності Faster-Whisper і WhisperX

Характеристика	Faster-Whisper	WhisperX
Batch inference	Так	Так
Швидкість	Дуже висока	Висока
Word timestamps	Ні	Так
Diarization	Ні	Так
VAD	Частково	Так
AI-аналітика	Базова	Краща
Навантаження	Менше	Більше

Faster-Whisper підтримує batch inference. Це одна з головних переваг Faster-Whisper над оригінальним Whisper. При цьому, у WhisperX пакетна обробка в основному стосується: сегментів мовлення; chunk-обробки; VAD-фрагментів. Тобто WhisperX розбиває аудіо через VAD, формує пакет

сегментів, передає їх у Faster-Whisper, а потім виконує вирівнювання слів із таймкодами та діаризацію мовців. Для первинного отримання моделі діаризації може використовуватися сервіс Hugging Face з механізмом авторизації через токен доступу (застосовується pyannote/speaker-diarization-3.1). Ця модель розміщена на Hugging Face. Після завантаження модель розгортається локально. Однак можлива реалізація діаризації взагалі без Hugging Face. Для цього є альтернативи: локально встановлені pyannote-моделі, інші diarization-фреймворки, власна реалізація embeddings + clustering, локальні speaker-encoder моделі. Але WhisperX «з коробки» найчастіше орієнтований саме на pyannote. NVIDIA NeMo – це open-source фреймворк від NVIDIA для створення та використання AI-моделей для ASR, TTS, діаризації спікерів та генеративного AI [40]. Бібліотека побудована на основі PyTorch і містить готові моделі для локального запуску та донавчання. У задачах транскрибування NeMo часто застосовується для побудови комплексних конвеєрів: ASR → діаризація → аналітика мовлення. Добре масштабується на GPU та підтримує обробку великих обсягів аудіоданих. SpeechBrain – це open-source бібліотека для обробки мовлення та аудіо на основі PyTorch, орієнтована на швидке створення ASR, діаризації, ідентифікації спікерів, синтезу мовлення та аудіоаналізу [41]. Бібліотека надає готові моделі та модульну архітектуру для побудови локальних AI-рішень. У задачах діаризації SpeechBrain використовує підхід на основі speaker embeddings та алгоритмів кластеризації для автоматичного розділення мовлення за спікерами.

3.3 Модель пакетної обробки для локального транскрибування

Для практичної реалізації локального AI-агента у роботі запропонована модель пакетної обробки аудіо (Додаток А), що представлена у вигляді Google-блокноту. У ньому реалізовано повний цикл пакетної обробки

аудіофайлів. Програма автоматично знаходить аудіофайли у визначеній папці, завантажує модель Whisper, послідовно виконує транскрибування кожного запису, зберігає результати в окремі текстові та JSON-файли, а також формує зведену таблицю і загальний текстовий документ. Транскрибування аудіофайлів виконується за допомогою моделі Whisper. Такий підхід є зручним для практичних задач, коли потрібно швидко обробити велику кількість записів.

На початковому етапі в блокноті передбачено встановлення необхідних програмних компонентів. Для транскрибування використовується Whisper. Також застосовується утиліта FFmpeg для коректного зчитування аудіофайлів різних форматів (WAV, MP3, M4A, AAC, FLAC та OGG). Це робить реалізацію більш універсальною – е потрібно вручну перетворювати всі записи до одного формату перед обробкою.

В блокноті є можливість роботи з Google Drive. Це дозволяє використовувати аудіофайли, які зберігаються не безпосередньо у середовищі Google Colab, а на хмарному диску користувача. Це зручно для зберігання великих наборів аудіоданих, оскільки Google Colab має тимчасове середовище виконання, а файли на Google Drive можуть зберігатися постійно. Для запуску обробки достатньо вказати шлях до папки.

Важливим елементом реалізації є блок налаштувань, у якому визначається вхідна папка з аудіофайлами, перелік підтримуваних форматів, назва моделі Whisper та параметри мови. Саме ці налаштування дають змогу адаптувати блокнот до різних умов використання. Наприклад, користувач може змінити папку, з якої будуть зчитуватися аудіофайли, вибрати іншу модель Whisper залежно від доступних обчислювальних ресурсів або вказати конкретну мову мовлення. Якщо мову не задано вручну, модель може автоматично визначати її під час транскрибування. Це корисно під час пакетної обробки, коли в одній папці можуть міститися записи різними мовами. Перед початком транскрибування в коді створюється окрема папка для збереження результатів. Це дозволяє відокремити вхідні аудіофайли від

отриманих текстових даних. Така організація структури файлів є важливою, оскільки після обробки кожного аудіозапису формується декілька результатів: окремий текстовий файл, файл із розширеною інформацією у форматі JSON, а також зведені файли для всієї пакетної обробки. Завдяки цьому може легко знайти результати транскрибування та використовувати їх для подальшого аналізу. Пакетна обробка починається з автоматичного пошуку всіх аудіофайлів у вказаній папці. Програма переглядає каталог і відбирає лише ті файли, розширення яких відповідають заданим аудіоформатам. Усі знайдені файли додаються до загального списку, який потім використовується для послідовного транскрибування. Це є одним з моментів реалізації, оскільки користувачу не потрібно вручну вказувати кожен аудіозапис. Достатньо розмістити файли у відповідній папці, після чого програма самостійно сформує список для обробки.

Після формування списку аудіофайлів завантажується модель Whisper. Важливо, що модель завантажується лише один раз перед початком обробки всього набору файлів. Це підвищує ефективність роботи програми, оскільки повторне завантаження моделі для кожного окремого запису потребувало б значно більше часу та ресурсів. Після завантаження модель використовується для послідовного транскрибування всіх знайдених аудіофайлів.

Основна логіка пакетної обробки реалізована у вигляді циклу, який проходить по кожному файлу зі сформованого списку. Для кожного аудіозапису програма визначає його назву, формує шляхи для збереження результатів і запускає процес транскрибування. Якщо мову в налаштуваннях було задано вручну, вона передається моделі під час обробки. Якщо мову не задано, Whisper виконує автоматичне визначення мови. Після завершення розпізнавання з результату витягується основний текст транскрипції. Для кожного обробленого аудіофайлу створюється окремий текстовий файл. У ньому зберігається розпізнаний текст без додаткових службових даних. Такий файл зручно використовувати для читання, редагування або вставлення у текстові документи. Крім цього, повний результат транскрибування

зберігається у форматі JSON. У цьому файлі міститься не лише загальний текст, а й додаткова інформація, зокрема визначена мова, сегменти мовлення та інші службові дані для подальшого аналізу.

Особливістю реалізації є те, що після обробки кожного файлу інформація про результат додається до загального списку. До нього входять назва або шлях до аудіофайлу, визначена мова, кількість сегментів, тривалість запису та отриманий текст. На основі цього списку після завершення обробки формується зведена таблиця. Такий підхід дозволяє отримати не лише окремі транскрипції, а й загальну структуру результатів для всього набору аудіофайлів. У програмі також передбачено обробку можливих помилок. Якщо під час транскрибування певного аудіофайлу виникає помилка, наприклад через пошкоджений файл або непідтримуваний формат, виконання всієї програми не зупиняється. Натомість інформація про помилку зберігається у загальному списку результатів, а програма переходить до наступного файлу. Це є важливою перевагою для пакетної обробки, оскільки при роботі з великою кількістю аудіозаписів окремі проблемні файли не повинні переривати весь процес.

Після завершення циклу обробки результати перетворюються у таблицю. У цій таблиці кожен рядок відповідає окремому аудіофайлу, а стовпці містять основну інформацію про процес транскрибування та отриманий текст. Далі таблиця зберігається у CSV-файл. Такий формат є зручним для подальшої роботи, оскільки його можна відкрити в Excel, Google Sheets або використати в інших програмах для аналізу даних. Крім CSV-файлу, у блокноті формується також загальний текстовий файл, у якому об’єднуються всі отримані транскрипції. Перед текстом кожного аудіозапису додається службовий заголовок із назвою файлу, що дозволяє швидко орієнтуватися у великому обсязі тексту. Такий файл зручний для перегляду всіх результатів в одному документі без необхідності відкривати кожну транскрипцію окремо. Для використання наведеного програмного коду без хмарних сервісів його потрібно запускати не в Google Colab, а безпосередньо

на локальному комп’ютері або локальному сервері. У такому випадку всі аудіофайли, модель розпізнавання мовлення та результати транскрибування зберігаються на власному пристрої користувача, а обробка даних виконується локально. Перед запуском коду на локальному комп’ютері потрібно підготувати робоче середовище. Для цього встановлюється Python, бажано версії 3.9 або новішої, а також необхідні бібліотеки для роботи Whisper. Крім того, потрібно встановити FFmpeg. Якщо FFmpeg не встановлено, програма може не розпізнавати частину аудіофайлів або видавати помилки під час їх обробки. На відміну від Google Colab, де встановлення бібліотек виконується прямо в комірках блокнота, під час локального запуску ці дії виконуються один раз у терміналі або командному рядку ОС. Після встановлення всіх залежностей код можна запускати у Jupyter Notebook, PyCharm, Visual Studio Code, або як звичайний Python-файл. Якщо використовується Jupyter Notebook, структура роботи буде схожою на Google Colab, але всі файли зберігатимуться локально. Важливою зміною є відмова від підключення Google Drive. У локальному варіанті цей блок не потрібний, оскільки аудіофайли розміщуються безпосередньо в папці на комп’ютері. Наприклад, користувач може створити окрему директорію для вхідних аудіозаписів і вказати її як робочу папку програми. Тобто шлях типу /content, який використовується в Google Colab, потрібно замінити на локальний шлях до папки з аудіофайлами. Для Windows це може бути шлях на зразок папки на диску D:, а для Linux або macOS – шлях у окремому каталозі користувача. Після налаштування локального шляху програма автоматично переглядає вказану папку та формує список аудіофайлів, які відповідають заданим розширенням. Саме цей механізм забезпечує пакетну обробку. Користувачу достатньо помістити всі потрібні аудіозаписи в одну папку, після чого програма самостійно знайде їх, відсортує та передасть на обробку.

Під час першого запуску модель Whisper завантажується на локальний комп’ютер. Якщо потрібно повністю виключити залежність від Інтернет, модель можна заздалегідь завантажити та зберегти локально. Для локального

запуску можна використовувати як CPU, так і відеокарту. Якщо комп’ютер не має GPU, модель Whisper все одно може працювати на CPU, але швидкість транскрибування буде нижчою. Якщо доступна відеокарта NVIDIA з підтримкою CUDA, обробка буде значно швидшою, особливо для довгих аудіозаписів або великої кількості файлів. Модель завантажується лише один раз перед початком циклу пакетної обробки. Це важливо з погляду ефективності, оскільки повторне завантаження моделі для кожного файлу збільшило б час виконання програми. Для кожного аудіозапису створюється окремий текстовий файл із результатом транскрибування. Крім того, формується JSON-файл, у якому зберігається більш детальна інформація про результат розпізнавання, зокрема мова, сегменти мовлення та службові дані. Після завершення обробки всіх файлів програма створює зведену CSV-таблицю, де кожен рядок відповідає окремому аудіофайлу. Також формується загальний текстовий файл, у якому об’єднуються всі отримані транскрипції. У локальному варіанті всі ці результати зберігаються у спеціальній папці на комп’ютері, наприклад у підпапці transcripts, яка створюється всередині папки з аудіофайлами. Це дозволяє зручно відокремити початкові аудіозаписи від результатів їх обробки. Така структура є зручною для подальшої роботи: текстові файли можна редагувати, CSV-таблицю можна відкрити в Excel або Google Sheets, а JSON-файли можна використати для подальшої автоматизованої обробки.

3.4 Економічне обґрунтування результатів

Запропонований програмний підхід передбачає пакетну обробку аудіозаписів, тобто одночасне завантаження набору файлів із визначеної папки, їх послідовне транскрибування та збереження результатів у структурованому вигляді. З економічної точки зору це дозволяє зменшити витрати часу на ручне опрацювання аудіозаписів, скоротити потребу в

постійному залученні оператора та уникнути регулярної оплати хмарних сервісів розпізнавання мовлення. Для оцінювання економічного ефекту можна розглянути умовний приклад. Нехай потрібно обробити пакет із 50 аудіофайлів, середня тривалість кожного з яких становить 10 хвилин. Тоді загальна тривалість аудіоматеріалу визначається за формулою:

$$T_{audio} = \frac{N \cdot t_{avg}}{60} = \frac{50 \cdot 10}{60} \approx 8{,}33 \text{ год}, \tag{3.3}$$

де T_{audio} – загальна тривалість аудіо у годинах;

N – кількість аудіофайлів;

t_{avg} – середня тривалість одного файла у хвилинах.

У разі ручного транскрибування співробітник витрачає більше часу, ніж фактична тривалість аудіо, оскільки потрібно прослуховувати запис, зупиняти його, повертатися до складних фрагментів, вводити текст і редагувати помилки. Для розрахункового прикладу приймемо, що ручне транскрибування однієї години аудіо потребує приблизно 4 години роботи. Тоді загальні витрати часу на ручну обробку можна визначити за виразом:

$$T_{manual} = T_{audio} \cdot k_{manual}, \tag{3.4}$$

де T_{manual} – час ручного транскрибування;

k_{manual} – коефіцієнт трудомісткості ручного транскрибування.

Для ручного транскрибування пакета з 50-ти аудіофайлів потрібно приблизно 33,32 години роботи оператора. У випадку автоматизованого транскрибування за допомогою Whisper основний текст формується програмою. Людина витрачає час переважно на запуск обробки, перевірку результатів і виправлення неточностей. Для прикладу приймемо, що автоматизована обробка з урахуванням подальшої перевірки потребує 0,7 години роботи на одну годину аудіо. Тоді час роботи при автоматизованому підході становитиме:

$$T_{auto} = T_{audio} \cdot k_{auto} = 8{,}33 \cdot 0{,}7 \approx 5{,}83 \text{ год}, \tag{3.5}$$

де T_{auto} – час автоматизованої обробки з перевіркою;

k_{auto} – коефіцієнт трудомісткості автоматизованого транскрибування.

Отже, економія робочого часу визначається як різниця між часом ручної та автоматизованої обробки:

$$E_T = T_{manual} - T_{auto} = 33,32 - 5,83 = 27,49 \text{ год.} \quad (3.6)$$

Це означає, що автоматизоване рішення дозволяє значно зменшити трудові витрати на транскрибування. Для оцінювання грошової економії можна використати погодинну оплату праці співробітника. Нехай умовна вартість однієї години роботи оператора становить 150 грн/год. Тоді витрати на ручне транскрибування визначаються за формулою:

$$C_{manual} = T_{manual} \cdot S, \quad (3.7)$$

де C_{manual} – вартість ручної обробки;

S – погодинна ставка співробітника.

Для наведеного прикладу $C_{manual} = 4998$ грн. Витрати на автоматизовану обробку з урахуванням перевірки результатів відповідають виразу:

$$C_{auto} = T_{auto} \cdot S. \quad (3.8)$$

Тобто, вони становитимуть $C_{auto} = 874,5$ грн. Отже, орієнтовна економія коштів на одному пакеті аудіофайлів може бути визначена так:

$$E_C = C_{manual} - C_{auto} = 4998 - 874,5 = 4123,5 \text{ год.} \quad (3.9)$$

Окремо можна оцінити підвищення продуктивності праці. Для цього доцільно використати коефіцієнт прискорення обробки:

$$K = \frac{T_{manual}}{T_{auto}} = \frac{33,32}{5,83} \approx 5,7. \quad (3.10)$$

Тобто, автоматизована система у $K \approx 5,7$ рази швидше, ніж ручна обробка. Це важливий аргумент на користь використання пакетної обробки, особливо тоді, коли аудіофайли надходять регулярно. Також можна порівняти собівартість обробки однієї години аудіо. Для ручного підходу вона визначається за формулою:

$$C_{1h\_auto} = \frac{C_{auto}}{T_{audio}}. \quad (3.11)$$

Для 50-ти файлів це дорівнює $C_{1h\_auto} \approx 105$ грн/год аудіо. Отже, собівартість обробки однієї години аудіо зменшується приблизно з 600 грн до 105 грн. Це значне зниження витрат на транскрибування при використанні локального рішення. Додатковою економічною перевагою є відсутність постійної оплати хмарних сервісів (організація сплачує за кожну хвилину або годину обробленого аудіо).

У локальному варіанті основні витрати пов'язані з наявністю комп'ютера, встановленням ПЗ та витратами електроенергії. Якщо обладнання вже є в наявності, витрати на впровадження – мінімальні. Витрати електроенергії можна приблизно оцінити за формулою:

$$C_{el} = P \cdot t \cdot c, \tag{3.12}$$

де C_{el} – вартість електроенергії;

P – потужність комп'ютера в кВт;

t – час роботи в годинах;

c – вартість 1 кВт·год.

Наприклад, якщо комп'ютер під час обробки споживає 0,3 кВт, час роботи становить 5,83 години, а умовна вартість електроенергії – 5 грн/кВт·год, тоді $C_{el} = 8,75$ грн. Як видно з розрахунку, витрати на електроенергію є незначними порівняно з економією робочого часу. Навіть якщо врахувати ці витрати, загальна вартість автоматизованої обробки залишається значно нижчою за ручне транскрибування. Якщо для впровадження потрібні одноразові витрати, наприклад на налаштування середовища, встановлення Python, Whisper, FFmpeg та тестування роботи програми, їх також можна врахувати. Нехай умовні витрати на впровадження становлять 3000 грн. Тоді строк окупності можна оцінити за формулою:

$$T_{payback} = \frac{C_{setup}}{E_C}, \tag{3.13}$$

де $T_{payback}$ – кількість пакетів аудіофайлів, після яких рішення окупиться;

C_{setup} – витрати на впровадження;

E_C – економія коштів на одному пакеті.

Це означає, що за наведених умов система окупається менш ніж за один повний пакет обробки з 50 аудіофайлів:

$$T_{payback} = \frac{3000}{4123,5} = 0,73.$$

Якщо ж аудіозаписи обробляються регулярно, економічний ефект буде накопичуватися з кожним наступним пакетом. Отже, запропоноване рішення є економічно доцільним, оскільки забезпечує суттєве скорочення часу на транскрибування, зменшує вартість обробки однієї години аудіо, підвищує продуктивність праці та дозволяє уникнути постійних витрат на хмарні сервіси. Особливо важливою є реалізація пакетної обробки, оскільки саме вона дає змогу автоматизувати роботу з великою кількістю аудіофайлів. Крім економії коштів, локальний запуск системи підвищує рівень конфіденційності, оскільки аудіодані не передаються на зовнішні сервери, а залишаються в межах комп’ютера або локальної мережі організації.

ВИСНОВКИ

Транскрибування аудіо може бути корисним для освіти, медицини, бізнесу, державних установ, медіа та служб підтримки, а також AI-аналітики звернень користувачів, де транскрибування виступає першим етапом для класифікації проблем, формування підсумків і виявлення повторюваних запитів. Хмарні сервіси зручні для швидкого запуску, не потребують власного потужного обладнання та часто забезпечують високу якість розпізнавання. Проте вони залежать від Інтернет, зовнішнього постачальника послуг і передбачають передачу аудіоданих на сторонні сервери.

Локальне розгортання системи транскрибування дозволяє виконувати транскрибування та подальшу AI-аналітику без передачі інформації до зовнішніх хмарних сервісів і є доцільним у випадках, коли обробляються персональні, службові або комерційні дані. Такі рішення складніші у налаштуванні.

Робота локального AI-агента має кілька етапів: завантаження аудіо, попередню обробку, транскрибування, аналіз тексту та формування кінцевого результату. При цьому є можливість поєднання транскрибування з AI-аналітикою, внутрішньою базою знань і RAG. Це дозволяє сформувати короткий висновок, класифікувати звернення або підготувати рекомендацію для оператора. Використання VAD дозволяє відокремити фрагменти з мовленням, завдяки чому зменшується обсяг аудіо для ASR-моделі.

Для реалізації транскрибування на українській мові у роботі виконаний аналіз моделей кількох сімейств: Vosk і Whisper. У результаті зроблений висновок про доцільність застосування модифікацій Whisper, WhisperX і Faster-Whisper. Вибір конкретної модифікації Whisper залежить від вимог до швидкодії, точності, ресурсів комп’ютера та потреб подальшого аналізу транскрибованого тексту. Пакетна обробка є важливим підходом для систем, що працюють з великою кількістю аудіофайлів. Вона дозволяє автоматизувати процес обробки записів, зменшити ручну працю та

послідовно формувати транскрипти для кожного файлу. Пакетна обробка має додаткову перевагу, оскільки всі аудіозаписи, текстові результати та аналітичні висновки залишаються в межах локальної інфраструктури. Такий підхід також дає змогу ефективніше використовувати обчислювальні ресурси й запускати транскрибування у зручний час.

Для оцінки якості транскрибування аудіо в текст використовують метрики WER, CER та RTF. Запропонована модель пакетної обробки аудіофайлів демонструє можливість локального транскрибування. Модель може бути основою для локального AI-агента, орієнтованого на обробку великої кількості аудіозаписів.

Проведене економічне обґрунтування показує, що використання автоматизованого локального транскрибування є доцільним з погляду скорочення часу та фінансових витрат. Порівняно з ручною обробкою аудіозаписів, запропонований підхід значно зменшує трудомісткість роботи та підвищує продуктивність. Додатковою перевагою є відсутність постійної оплати хмарних сервісів, оскільки система може працювати на локальному комп’ютері або сервері. Витрати на електроенергію та початкове налаштування є незначними порівняно з економією робочого часу.

Таким чином, результатами роботи є модель локального транскрибування аудіо в текст, рекомендації щодо практичної реалізації локального транскрибування аудіо в текст. Вони можуть бути використані для подальших досліджень за даною тематикою та при проектуванні локальних AI-агентів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Slyusar V., Sliusar I. Leveraging pre-trained neural networks for image classification in audio signal analysis for mobile applications of home automation. *Research Tendencies and Prospect Domains for AI Development and Implementation*. River Publishers, 2024. P. 109-128.

2. Контролюйте якість кожної розмови з клієнтом. *ConnectiveOne*. URL: https://www.connectiveone.io/service/ai-insights-quality-assurance (дата звернення: 05.05.2026).

3. Що таке технологія перетворення мовлення в текст і як вона працює в автоматичному розпізнаванні мовлення. *Shaip*. URL: https://uk.shaip.com /blog/automatic-speech-recognitiona-asr (дата звернення: 05.05.2026).

4. Transformers. *Hugging Face*. URL: https://huggingface.co/docs/ transformers/index (дата звернення: 05.05.2026).

5. NVIDIA Riva Documentation. *NVIDIA*. URL: https://docs.nvidia.com/ riva/index.html (дата звернення: 05.05.2026).

6. Google Cloud Speech-to-Text. URL: https://cloud.google.com/speech-to-text (дата звернення: 05.05.2026).

7. Amazon Transcribe. URL: https://aws.amazon.com/ru/transcribe/ (дата звернення: 05.05.2026).

8. Microsoft Azure AI Speech. URL: https://learn.microsoft.com/en-us/azure/ai-services/speech-service/index-speech-to-text (дата звернення: 05.05.2026).

9. OpenAI Speech to Text API. URL: https://developers.openai.com/api/docs/guides/speech-to-text (дата звернення: 05.05.2026).

10. Whisper AI. URL: https://openai.com/uk-UA/index/whisper (дата звернення: 05.05.2026).

11. Whisper. URL: https://github.com/openai/whisper#available-models-and-languages (дата звернення: 05.05.2026).

12. AssemblyAI. URL: https://www.assemblyai.com/products/speech-to-text (дата звернення: 05.05.2026).

13. Deepgram Speech-to-Text API. URL: https://deepgram.com (дата звернення: 05.05.2026).

14. Chirp 3 Transcription: Enhanced multilingual accuracy. *Google-Cloud*. URL: https://docs.cloud.google.com/speech-to-text/docs/models/chirp-3 (дата звернення: 05.05.2026).

15. Compare transcription models. *Google-Cloud*. URL: https://docs.cloud.google.com/speech-to-text/docs/transcription-model (дата звернення: 05.05.2026).

16. Improve accuracy with model adaptation. *Google-Cloud*. URL: https://docs.cloud.google.com/speech-to-text/docs/models/chirp-3#improve_accuracy_with_model_adaptatio (дата звернення: 05.05.2026).

17. SpeechAdaptation. *Google-Cloud*. URL: https://docs.cloud.google.com/speech-to-text/docs/reference/rpc/google.cloud.speech.v2#speechadaptation (дата звернення: 05.05.2026).

18. Transcribe short audio files. *Google-Cloud*. URL: https://docs.cloud.google.com/speech-to-text/docs/sync-recognize (дата звернення: 05.05.2026).

19. Operations. *Google-Cloud*. URL: https://docs.cloud.google.com/speech-to-text/docs/operations (дата звернення: 05.05.2026).

20. Quotas and limits. *Google-Cloud*. URL: https://docs.cloud.google.com/speech-to-text/docs/v1/quotas (дата звернення: 05.05.2026).

21. Cloud Speech-to-Text V2 supported languages. *Google-Cloud*. URL: https://docs.cloud.google.com/speech-to-text/docs/speech-to-text-supported-languages (дата звернення: 05.05.2026).

22. Cloud Speech-to-Text recognition requests. *Google-Cloud*. URL: https://docs.cloud.google.com/speech-to-text/docs/overview#recognition-requests (дата звернення: 05.05.2026).

23. Cloud Speech-to-Text On-Prem documentation. *Google-Cloud*. URL: https://docs.cloud.google.com/speech-to-text/priv (дата звернення: 05.05.2026).

24. Regional availability. *Google-Cloud*. URL: https://docs.cloud.google.com/speech-to-text/docs/locations (дата звернення: 05.05.2026).

25. Customer-managed encryption keys (CMEK). *Google-Cloud*. URL: https://docs.cloud.google.com/kms/docs/cmek (дата звернення: 05.05.2026).

26. Cloud KMS Autokey. *Google-Cloud*. URL: https://docs.cloud.google.com/kms/docs/autokey-overview (дата звернення: 05.05.2026).

27. Готра Є. Реалізація AI-аналітики звернень за допомогою локального агента. *Збірник XXI щорічної студентської наукової конференції «Сучасні інформаційні технології та інноваційні методики в економіці, менеджменті та бізнесі» Полтавського державного аграрного університету* (квітень 2026 р., м. Полтава). Полтава: ПДАУ, 2026 р. С. 66, 67.

28. Enen M., Saad S., Nazmy T. A survey on retrieval-augmentation generation (RAG) models for healthcare applications. *Neural Computing and Applications*. 2025. 37(33). P. 28191-28267. DOI: 10.1007/s00521-025-11666-9.

29. Raspberry Pi 5. URL: https://www.raspberrypi.com/products/raspberry-pi-5/ (дата звернення: 05.05.2026).

30. Voice activity detection (VAD). *OpenAI*. URL: https://developers.openai.com/api/docs/guides/realtime-vad дата звернення: 05.05.2026).

31. Vosk. *Alphacephei*. URL: https://alphacephei.com/vosk/ дата звернення: 05.05.2026).

32. Bain M., Huh J., Han T., Zisserman A. WhisperX: Time-Accurate Speech Transcription of Long-Form Audio. URL: https://doi.org/10.48550/arXiv.2303.00747 (дата звернення: 05.05.2026).

33. WhisperX. URL: https://github.com/m-bain/whisperx (дата звернення: 05.05.2026).

34. Unlocking Audio Insights: Speaker Diarization with WhisperX for "Who Said What". *Data Curious*.URL: https://datacurious.hashnode.dev/unlocking-audio-insights-speaker-diarization-with-whisperx-for-who-said-what (дата звернення: 05.05.2026).

35. Faster-Whisper. URL: fhttps://github.com/SYSTRAN/faster-whisper (дата звернення: 05.05.2026).

36. Faster Whisper transcription with CTranslate2. URL: https://pypi.org/project/faster-whisper/0.3.0 (дата звернення: 05.05.2026).

37. WhisperX with Diarization. *Clore.AI*.URL: https://docs.clore.ai/guides/audio-and-voice/whisperx (дата звернення: 05.05.2026).

38. Whisper large V3 turbo support? #898. URL: https://github.com/m-bain/whisperX/issues/898?utm_source=chatgpt.com дата звернення: 05.05.2026).

39. Batched Transcription. URL: https://github.com/SYSTRAN/faster-whisper (дата звернення: 05.05.2026).

40. NeMo. URL: https://github.com/NVIDIA/NeMo (дата звернення: 05.05.2026).

41. Speechbrain. URL: https://github.com/speechbrain/speechbrain (дата звернення: 05.05.2026).