Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

**МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА**

на тему:

# «РОЗРОБКА МЕТОДУ РОЗПІЗНАВАННЯ УКРАЇНСЬКОГО МОВЛЕННЯ МЕДИЧНОГО СПРЯМУВАННЯ З ПЕРЕТВОРЕННЯМ АУДІОЗАПИСІВ У ТЕКСТ»

Виконав: студент 2 курсу, групи 3АКІТ-22м
спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології
(шифр і назва напряму підготовки, спеціальності)
Петро ПЕТРУК
(прізвище та ініціали)

Керівник: д.т.н., проф., зав. каф. АІІТ
Олег БІСІКАЛО
(прізвище та ініціали)
«4» грудня 2023 р.

Опонент: к.т.н., доцент каф. КСУ
Марія ЮХИМЧУК
(прізвище та ініціали)
«7» грудня 2023 р.

**Допущено до захисту**
Зав. кафедри АІІТ
Олег БІСІКАЛО
«11» грудня 2023 р.

Вінниця ВНТУ – 2023 рік

**Вінницький національний технічний університет**

Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій
Рівень вищої освіти 2-й (магістерський)
Галузь знань 15 – «Автоматизація та приладобудування»
Спеціальність 151– «Автоматизація та комп'ютерно-інтегровані технології»
Освітня програма Інформаційні системи і Інтернет речей

**ЗАТВЕРДЖУЮ**
**Завідувач кафедри АІІТ**
д.т.н., проф. Олег БІСІКАЛО
«10» вересня 2023р.

**ЗАВДАННЯ**
**НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ**

Петруку Петру Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка методу розпізнавання українського мовлення медичного спрямування з перетворенням аудіозаписів у текст»

Керівник роботи д.т.н., проф. каф. АІІТ Бісікало О. В.

Затверджені наказом вищого навчального закладу від "18" вересня 2023 року №____

2. Строк подання роботи студентом: до «12» грудня 2023 року.

3. Вихідні дані до роботи: аудіофайл у форматі WAV, мова розпізнавання – українська, доступ до месенджера Telegram.

4. Зміст текстової частини: Вступ. Аналіз та обгрунтування розробки методу. Розробка архітектури програмного засобу. Реалізація системи для здійснення анотацій. Розробка моделі розпізнавання українського мовлення. Висновки. Список використаних джерел. Додатки.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): UML-діаграма об'єктів UML-діаграма розгортання Схема архітектури Структура даних Збережені папки з аудіозаписами та мета інформацією Вигляд розроблених ендпоінтів в FastAPI Приклад анотації в телеграм боті Графіки статистичної взаємодії із телеграм ботом.

6. Консультанти розділів роботи

| Розділ | Посада консультанта, ім'я, прізвище | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | виконано прийняв |
| 1-4 | д.т.н., проф., зав. кафедри АІІТ Олег БІСІКАЛО | | |

7. Дата видачі завдання 16.09.2023 р.

## КАЛЕНДАРНИЙ ПЛАН

| № | Назва та зміст етапу | Термін виконання | | Примітки |
|---|---|---|---|---|
| | | початок | закінчення | |
| 1. | Вибір, узгодження та затвердження теми МКР | 25.09.2023 | 05.10.2023 | виконано |
| 2. | Аналіз та обгрунтування розробки програмного продукту. Попередня розробка основних розділів | 06.10.2023 | 09.10.2023 | виконано |
| 3. | Розробка технічного завдання (ТЗ) | 09.10.2023 | 10.11.2023 | виконано |
| 4. | Аналіз та обгрунтування розробки методу | 10.11.2023 | 17.11.2023 | виконано |
| 5. | Розробка архітектури програмного засобу | 17.11.2023 | 24.11.2023 | виконано |
| 6. | Реалізація системи для здійснення анотацій | 24.11.2023 | 04.12.2023 | виконано |
| 7. | Розробка моделі розпізнавання українського мовлення | 04.12.2023 | 08.12.2023 | виконано |
| 8. | Аналіз результатів роботи | 08.12.2023 | 10.12.2023 | виконано |
| 9. | Оформлення індивідуального завдання та графічної частини | 10.12.2023 | 11.02.2023 | виконано |

Студент _____ (підпис) Петро ПЕТРУК

Керівник роботи _____ (підпис) Олег БІСІКАЛО

# АНОТАЦІЯ

УДК 004.512.2

Петрук П. О. Розробка методу розпізнавання українського мовлення медичного спрямування з перетворенням аудіозаписів у текст. Магістерська кваліфікаційна робота зі спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології, освітня програма – Інформаційні системи і Інтернет речей. Вінниця: ВНТУ, 2023. 108 с.

На укр. мові. Бібліогр.: 45 назв; рис.: 15; табл. 1.

У даній магістерській кваліфікаційній роботі проведено глибокий аналіз існуючих рішень у сфері розпізнавання українського мовлення та перетворення отриманих аудіозаписів у текст, з акцентом на медичне мовлення. Були визначені ключові поняття та особливості, що впливають на процес розпізнавання мовлення, сформовано технічне завдання для розробки нового програмного засобу.

Основою роботи стала розробка архітектури програмного забезпечення, включаючи графічний інтерфейс та модулі обробки мовлення, з використанням Python та інших сучасних технологій. Було створено діаграми процесів та UML-діаграми активності для формалізації структури програмного забезпечення.

У роботі представлений розроблений метод автоматизації збору датасету, що включає 2235 унікальних аудіозаписів українського медичного мовлення. Реалізована програмна система з використанням моделі Whisper продемонструвала високу точність розпізнавання медичних текстів. Окремо відзначається використання техніки LoRA (Low-Rank Adaptation) у тренуванні моделі, що сприяло створенню адаптивної та масштабованої системи.

Ключові слова: розпізнавання українського медичного мовлення, перетворення аудіозапису у текст, обробка природної мови, розробка програмного забезпечення, автоматизація збору датасету, LoRA, Whisper, WER.

# ABSTRACT

UDC 004.512.2

Petruk P. O. Development of a method for recognizing Ukrainian medical speech with the transformation of audio recordings into text. Master's qualification work in the specialty 151 – Automation and Computer-Integrated Technologies, educational program – Information Systems and Internet of Things. Vinnytsia: VNTU, 2023. 108 p. In Ukrainian. Bibliography: 45 titles; figures: 15; table: 1.

This master's qualification work conducts a thorough analysis of existing solutions in the field of Ukrainian speech recognition and the transformation of obtained audio recordings into text, with a focus on medical speech. Key concepts and features influencing the speech recognition process were identified, and a technical task was formulated for the development of a new software tool.

The foundation of the work is the development of the software architecture, including a graphical interface and speech processing modules, using Python and other modern technologies. Process diagrams and UML activity diagrams were created to formalize the structure of the software.

The work presents the developed method for automating dataset collection, which includes 2235 unique audio recordings of Ukrainian medical speech. The implemented software system, using the Whisper model, demonstrated high accuracy in recognizing medical texts. Notably, the use of Low-Rank Adaptation (LoRA) technique in model training contributed to the creation of an adaptive and scalable system.

Keywords: Ukrainian medical speech recognition, audio-to-text transformation, natural language processing, software development, dataset collection automation, LoRA, Whisper, WER.

# ЗМІСТ

# ВСТУП

У сучасному світі, де швидкість і точність обробки інформації є ключовими для успіху в багатьох сферах, особливо актуальним стає розвиток та застосування інноваційних технологій. Медична сфера, що потребує високої точності та оперативності у обробці даних, не є винятком. В цьому контексті, розробка методу розпізнавання українського мовлення медичного спрямування з перетворенням аудіозаписів у текст набуває особливої актуальності.

**Актуальність дослідження** полягає у необхідності підвищення ефективності медичного обслуговування через автоматизацію процесів документування та аналізу мовленнєвої інформації, зокрема в українському медичному контексті.

**Метою роботи** є підвищення ефективності перетворення аудіозаписів у текст шляхом розробки та апробації методу автоматизованого розпізнавання українського медичного мовлення.

**Об'єктом дослідження** є процеси розпізнавання медичного мовлення та їх перетворення в текст.

**Предметом дослідження** є методи автоматизованого розпізнавання мовлення, що враховують специфіку українського медичного мовлення.

Для досягнення поставленої мети необхідно розв'язати наступні задачі:

1. Аналіз існуючих технологій розпізнавання мовлення та їх застосування у медичній галузі.

2. Визначення специфічних особливостей українського медичного мовлення.

3. Розробка алгоритмів та методів для автоматизованого розпізнавання медичного мовлення та перетворення його у текстовий формат.

4. Програмна реалізація розробленого методу.

5. Експериментальна апробація та оцінка ефективності методу.

**Методи дослідження** включають аналітичний огляд існуючих рішень, методи машинного навчання, технології побудови програмних модулів, методи

критеріальної оцінки та планування експерименту, організації тестування на реальних медичних даних.

**Наукова новизна одержаних результатів.** Набув подальшого розвитку метод автоматизованого розпізнавання українського мовлення медичного спрямування з перетворенням аудіозаписів у текст, який, на відміну від існуючих, базується на зборі оригінального датасету, використанні моделі Whisper і техніки LoRA, що забезпечує високу точність та оперативність перетворення мовлення у текст.

**Практичною цінністю** є можливість впровадження розробленого методу у медичну практику для підвищення ефективності документування та оброблення мовленнєвої медичної інформації, що сприятиме покращенню якості медичного обслуговування за рахунок економії часу лікаря на створення та заповнення численних текстових документів.

**Апробація результатів дослідження:** На один із модулей розробленого програмного забезпечення в рамках даної магістерської кваліфікаційної роботи було отримане авторське право (наведено в додатку Б), згідно чинного законодавства України про інтелектуальну власність [1]. Також результати аналізу предметної області, а саме “Огляд аналогів програм та технологій для розпізнавання українського мовлення з перетворенням у текст” було представлено на конференції Вінницького медичного університету «Всеукраїнська науково-практична конференція з міжнародною участю: актуальні задачі медичної, біологічної фізики та інформатики (м. Вінниця 2022)» [2].

Також, окремі ідеї та засоби, що були використані у роботі, знайшли відображення у публікаціях [3, 4]:

О.В. Бісікало, І.В. Богач, П.О. Петрук. «Побудова термінологічного словника української мови» в Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2021)», Вінниця, 2021. [Електронний ресурс]. Режим доступу:

https://conferences.vntu.edu.ua/index.php/mn/mn2021/paper/view/13243. Дата звернення: Жовтень. 2023. – 2 с.

О.В. Бісікало, П.О. Петрук. «Розробка методу розпізнавання українського мовлення медичного спрямування з перетворенням аудіозаписів у текст» в Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2024)», Вінниця, 2023. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/19684. Дата звернення: Грудень. 2023. – 2 с.

Також результати даної кваліфікаційної роботи було успішно впроваджено у практичну діяльність приватної клініки, довідка про впровадження наведено в додатку Д .

# 1 АНАЛІЗ ТА ОБҐРУНТУВАННЯ РОЗРОБКИ МЕТОДУ

## 1.1 Актуальність розпізнавання української мови

Актуальність розпізнавання української мови в медичному контексті визначається сукупністю факторів, серед яких ключовими є зростаючий обсяг медичної інформації у формі аудіозаписів та необхідність їх ефективного оброблення. У сучасних умовах, коли обсяг доступної медичної інформації стрімко збільшується, стає важливим розробити ефективні інструменти для розпізнавання мовлення. Це не лише спрощує передачу важливої інформації, але й відіграє критичну роль у медичних процедурах, де час є вирішальним.

Наголос на відсутності відповідних систем розпізнавання української мови в медичній сфері підкреслює важливість дослідження та розробки в цій області. Це особливо актуально, враховуючи, що правильне розпізнавання мовлення може значно впливати на прийняття важливих медичних рішень.

Особливу увагу слід звернути на специфічні виклики, які створюються унікальними фонетичними, морфологічними та синтаксичними особливостями української мови. Це вимагає не лише детального дослідження цих особливостей, але й інноваційного підходу до розробки алгоритмів розпізнавання мовлення, які б були адаптовані до специфіки медичної сфери.

Враховуючи глобальну важливість підтримки різноманітності мов, включення української мови у розробку технологій розпізнавання мовлення не лише сприяє культурній ідентичності, але й відкриває нові можливості для ефективної комунікації та обміну інформацією у медичній сфері. Розробка цих технологій може зміцнити позиції української мови на міжнародній арені та сприяти розвитку інноваційних медичних рішень.

## 1.2 Аналіз предметної області дослідження

Головною метою даної магістерської роботи є розробка методу розпізнавання на основі програмного забезпечення, яке б імплементувало у собі функціонал обробки природної мови (NLP).

Обробка природної мови (Natural Language Processing – NLP) – міждисциплінарна галузь, що перетинається з комп'ютерними науками, штучним інтелектом та обчислювальною лінгвістикою [5]. Основою NLP є забезпечення взаємодії між людськими мовами та комп'ютером. Як зрозуміло з визначення, дана міждисциплінарна галузь оперує текстовою інформацією, як предметом дослідження. Інформація може бути не лише в текстовому представлені, але й у звуковому (рис. 1.1).

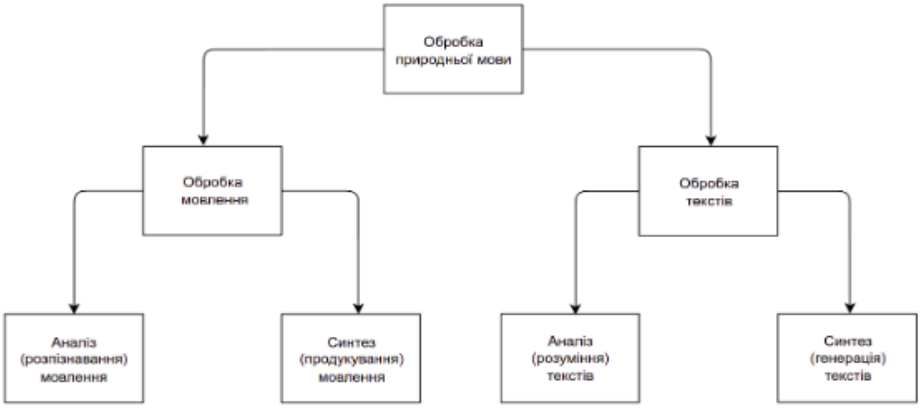



Рисунок 1.1 – Завдання обробки природної мови

Розпізнавання мовлення – це технологія, яка дозволяє комп'ютеру ідентифікувати і інтерпретувати слова та фрази в розмовній мові, а також перетворювати їх у тексти. Слід зазначити, що дане поняття не еквівалентне «розпізнавання мови», адже задача останнього визначити притаманність мови, до якої належить фрагмент. Як зрозуміло з назви операндом в даному випадку виступає звуковий сигнал, який саме і передає текстову інформацію, яку слід вилучити. Приклад розпізнаванням мовлення – рис. 1.2.

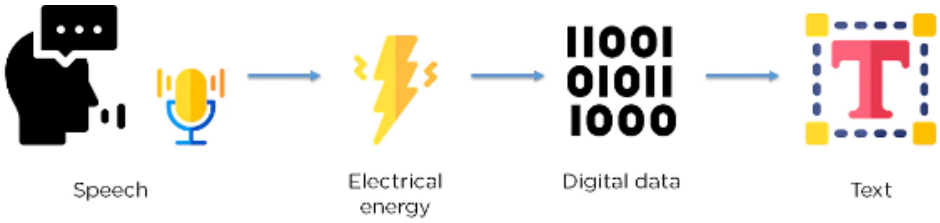



Рисунок 1.2 – Робота з розпізнаванням мовлення

Більш доступна інтерпретація даного терміну була представлена у книзі, що описує штучний інтелект [6]. Розпізнавання мовлення – це процес ідентифікації голосу на основі вимовленого слова шляхом виконання перетворення сигналу, який фіксується аудіопристроєм (пристроєм голосового введення). Розпізнавання мовлення також є системою, яка використовується для розпізнавання команд слів людського голосу, а потім їх перекладу на дані, якими може оперувати комп’ютер.

Звуковий сигнал – один з видів передавання інформації, що здійснюється за допомогою тону, частоти або періодичності. Збережені записи звукових сигналів називають аудіозаписом. Формат зберігання таких даних варіюється в залежності від міри стиснення. Одним з найпопулярніших форматів для зберігання без стиснення є Waveform audio format (WAV). Саме на нього буде орієнтовано завантаження та зберігання даних в розробленому програмному забезпеченні.

Звук – це те, що можна почути, і те, що має певні сигнальні характеристики, тоді як мова - це звуки, що складається з вимовлених слів.

Розпізнавання голосу або мовлення є одним із зусиль, необхідних для того, щоб зробити звук впізнаваним або ідентифікованим, щоб його можна було використовувати. Сама ідея розпізнавання мовлення, далеко не нова і має досить тривалу історію еволюції (рис. 1.3).

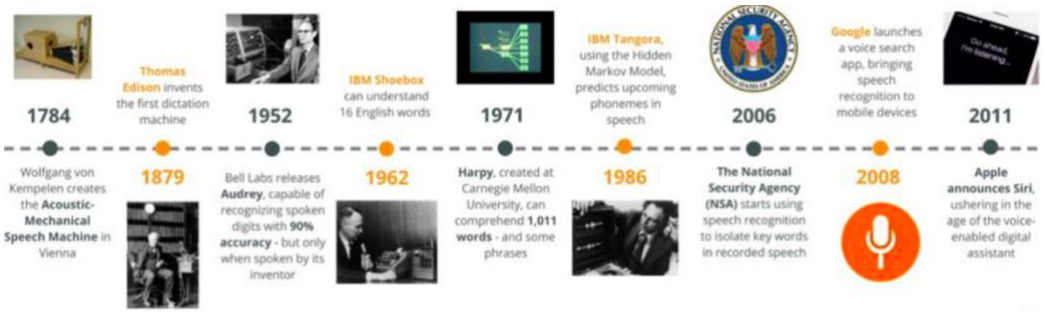



Рисунок 1.3 - Еволюція систем розпізнавання мовлення протягом десятиліть

Переглядаючи еволюцію даної технології видно, що перші системи були обмежені одним мовцем і мали малий лексикон, але в міру технологічного прогресу з'явилася сучасна система розпізнавання мовлення. Вона має словниковий запас для багатьох мов і підтримує розпізнавання кількох мовців одночасно.

Розпізнавання голосу в цілому можна розділити на три підходи, а саме акустично-фонетичний зустрічний підхід, зустрічний підхід зі штучним інтелектом і підхід розпізнавання образів. Підхід до розпізнавання образів для розпізнавання мовлення можна пояснити за допомогою блок-схеми, яка зображена на рис. 1.3 [7].

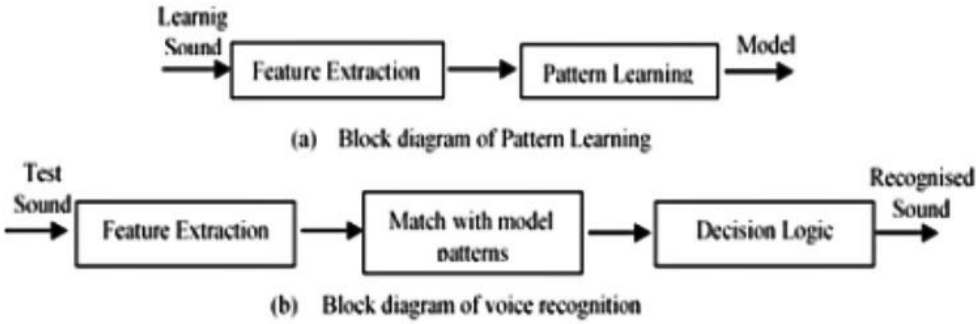



Рисунок 1.4 – Speech Recognition

На рис. 1.4 показана архітектура системи автоматичного розпізнавання мовлення (ASR). Він використовувався в багатьох програмах [8, 9], та має чотири компоненти: обробка сигналів і виділення ознак, акустична модель (AM), мовна

модель (LM) і гіпотетичний пошук. Компоненти обробки та вилучення функцій беруть аудіосигнал за вхід, покращують мовлення, усуваючи шум і спотворення каналу, перетворюють сигнали з часової області в частотну область та витягують векторні характеристики, які виділяються [10].

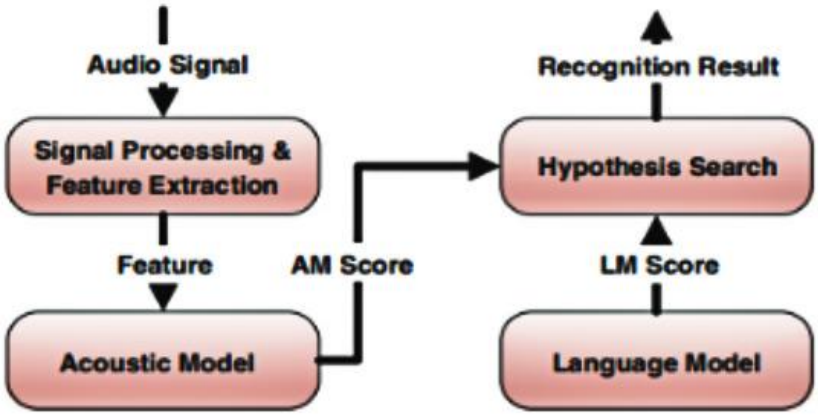



Рисунок 1.5 – Архітектура ASR системи

Отже, отримавши з аудіозапису текст, ми можемо проводити стандартні маніпуляції з ним на предмет оцінки якості виокремлення цього тексту з аудіопотоку. Для повноти розуміння, ще треба дати визначення терміну текст.

Текст – певна, з функціонально-смислового погляду упорядкована, група речень або їх аналогів, які являють собою, завдяки семантичним і функціональним взаємовідношенням елементів, завершену смислову єдність. Дискретною одиницею тексту є слово – це основна структурно-семантична одиниця мови, яка служить для найменування предметів та їх властивостей, явищ, відношень дійсності, і має сукупність семантичних, фонетичних та граматичних ознак, специфічних для кожної мови. Для зручності обробки текстів їх треба розділити на менші складові частини. Для цього використовується токенізація.

Токенізація – це особливий вид сегментації документів. Виділяють токенізацію як речень, так і слів, тобто розбиття тексту на речення, розбиття речення на слова відповідно [11]. Дана маніпуляція з текстами є базовою операцію, яка реалізована в багатьох варіаціях для різних мов програмування в

різних бібліотеках. Наприклад, результатом токенізації слів речення “Це був чудовий ранок” будуть наступні маркери (рис. 1.6).

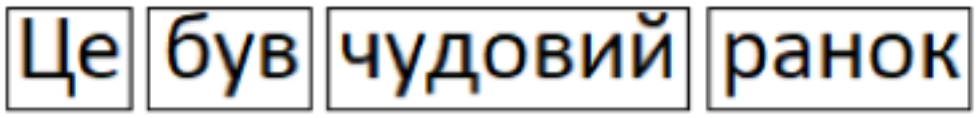



Рисунок 1.6 – Результат роботи простого токенізатора

Не всі слова в обробці природної мови несуть позитивний вплив на результат експериментів, навіть навпаки, якщо з первинних текстів не вилучити стоп-слова, це негативно вплине на хід досліджень.

Стоп-слова – це найпоширеніші слова на будь-якій мові, які зустрічаються дуже часто, але несуть в собі набагато менше змістовної інформації про сенс описаного. В українській мові налічується близько 1648 таких слів, прикладом яких є наступні слова: “якщо”, “це”, “був”, “аби де”, “вам”, “вас”. Такі слова треба видаляти з текстів при первинній обробці (рис. 1.7).

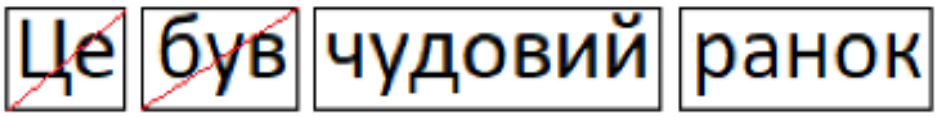



Рисунок 1.7 – Результат видалення стоп-слів з токенів

Як бачимо, видалення стоп-слів не змінили смислове навантаження даного речення. Не дивлячись на те, що стерлись часові рамки сказаного, для більшості задач обробки природної мови це не є критичним.

Лематизація – це процес зіставлення кожного слова в тексті до їх базової форми. Дієслова перетворюються в свою початкову форму, відновлюється в однині, а прислівники або прикметники передбачають їх позитивний формат. Цей метод заснований на морфологічному аналізі і часто використовує словник,

наприклад WordNet, де можна знайти лему для кожного зміненого слова. Цей етап попередньої обробки скорочує векторний простір шляхом зіставлення різних форм слів з їхнім спільним представленням.

Оскільки лематизація підтримується словниковими значенням, вона може зіставити «best» з його лемою «good» (рис. 1.8).

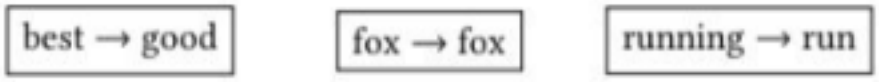



Рисунок 1.8 – Приклад роботи лематизації

Розглядаючи лематизацію як важливий етап у процесі обробки природної мови, особливо акцентуючи на її здатності скорочувати векторний простір та покращувати аналіз тексту, ми можемо перейти до розгляду сучасних технологій у сфері автоматичного розпізнавання мовлення (ASR).

Однією з передових систем у цій галузі є Whisper [12], яка забезпечує значні переваги у розпізнаванні мовлення, зокрема у багатомовному контексті. Whisper, навчена на 680 000 годин багатомовних даних, демонструє вражаючу гнучкість і точність у розпізнаванні мовлення, включаючи стійкість до акцентів, фонового шуму та технічної термінології. Це робить її ідеальною для специфічних потреб, таких як розпізнавання медичної української мови.

Основа Whisper - це кодер-декодер з архітектурою Transformer. Система працює, перетворюючи звук на 30-секундні фрагменти та спектрограми log-Mel, які подаються до кодера. Декодер навчений передбачати текстовий підпис зі спеціальними маркерами, що забезпечує високу точність транскрипції та перекладу.

Whisper, система розпізнавання мови розроблена OpenAI, відрізняється від традиційних підходів своєю особливою схемою навчання [13]. Зазвичай, коли мова йде про системи розпізнавання мови, вони навчаються на відносно невеликих наборах даних. Ці набори даних часто містять записи людей, які

говорять на певних мовах, і ці записи використовуються для тренування системи розпізнавати мову.

Відмінність Whisper полягає в тому, що вона використовує значно більші та різноманітніші набори даних. Ці дані включають в себе не тільки стандартні мовні записи, а й більш складні та різноманітні аудіоматеріали, такі як подкасти, відео з YouTube, аудіокниги та інші. Такий підхід дозволяє системі Whisper краще розпізнавати мову в різних умовах і контекстах, що робить її більш універсальною та ефективною.

Крім того, Whisper навчається не тільки розпізнавати слова, але й розуміти контекст, у якому ці слова використовуються. Це означає, що система може краще справлятися з різними акцентами, шумами на тлі та іншими факторами, які можуть ускладнювати розпізнавання мови.

Whisper, система розпізнавання мови від OpenAI, використовує інноваційну архітектуру трансформера, яка є важливою частиною її здатності точно перекладати та розуміти мовлення. В основі її навчання лежить перетворення аудіо в спектрограму, яка є двовимірним візуальним представленням звукових частот. Спектрограма забезпечує детальне зображення звуку, що дозволяє системі вловлювати нюанси мовлення, такі як інтонацію та акцент (рис. 1.8).

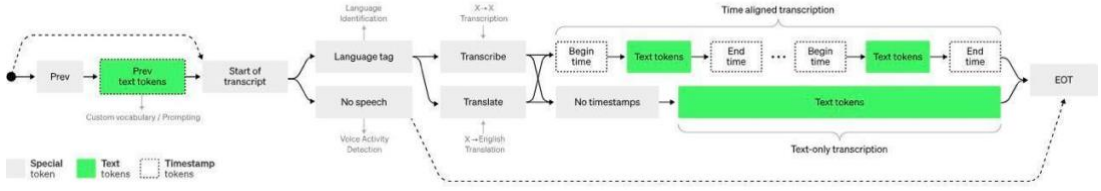



Рисунок 1.9 – Процес трансформації мовлення в текст в системі Whisper

Процес тренування Whisper починається з обробки спектрограми за допомогою згорткових нейронних мереж, які виявляють і виділяють характеристики звукового сигналу. Ці шари діють як фільтри, які вилучають важливі особливості з великої кількості даних, передаючи їх далі в енкодери.

Позиційне кодування, яке використовується на цьому етапі, дає системі зрозуміти послідовність звуків, важливу для розпізнавання слів та речень (рис. 1.9).

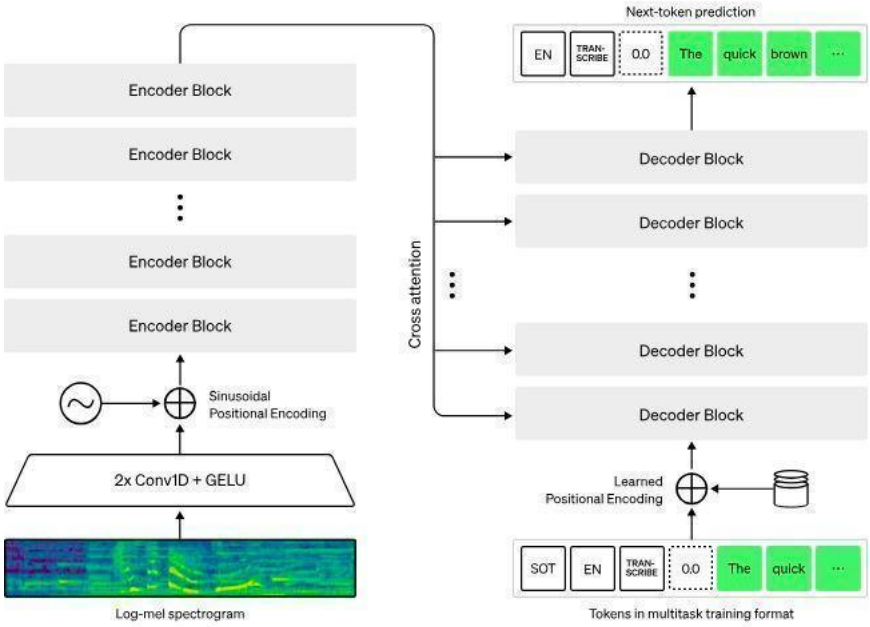



Рисунок 1.10 – Архітектура моделі трансформера

Енкодери трансформера подальше аналізують інформацію, використовуючи механізми самоуваги, які дозволяють моделі зосереджуватися на важливих аспектах спектрограми, ігноруючи менш значущі (рис. 1.10). Таке зосередження на контексті підвищує ефективність розпізнавання мовлення, дозволяючи Whisper точніше ідентифікувати мовленнєві команди та запити.

Декодери в архітектурі трансформера працюють паралельно з енкодерами, приймаючи закодовану інформацію та перетворюючи її на текст. Використовуючи крос-увагу, декодери аналізують вихід з енкодерів, вибираючи відповідні частини для генерації тексту. Цей процес включає передбачення наступного слова на основі попередніх, що поступово формує кінцевий текстовий вивід.

Основним елементом тренування Whisper є підгонка моделі до великої кількості аудіоданих і відповідних транскрипцій. Модель навчається на

прикладах, коригуючи свої параметри у відповідь на помилки для зменшення розбіжностей між її передбаченнями та фактичним текстом. Цей процес оптимізації є ітеративним і триває до тих пір, поки точність моделі не стане задовільною.

В результаті, Whisper може точно розпізнавати мовлення в різноманітних умовах і перекладати його на багато мов, навіть у присутності шумів або розмовних акцентів. Завдяки багатозадачності та гнучкості архітектури трансформера, Whisper здатна адаптуватися до нових мовленнєвих умов та продовжувати вдосконалюватися з плином часу.

Використання великого та різноманітного набору даних призводить до покращення універсальності та надійності системи, особливо у різноманітних умовах, що є критичним для медичного контексту.

Завершуючи, важливо відзначити високу точність та простоту використання Whisper, що відкриває нові можливості для розробки голосових інтерфейсів, а також стимулює подальші дослідження у галузі ASR.

## 1.3 Аналіз існуючих продуктів та сервісів у сфері розпізнавання мовлення в медицині

Проведення аналізу існуючих методів розпізнавання мовлення в медичному контексті. Виділення основних тенденцій та досягнень у галузі.

### 1.3.1 Огляд Amazon Transcribe Medical

У рамках прогресивного розвитку інформаційних технологій у медицині, Amazon Transcribe Medical виступає як втілення передових досягнень у сфері автоматичного розпізнавання мовлення (ASR).

Цей сервіс, розроблений корпорацією Amazon, спеціально адаптований для задоволення потреб медичної сфери, вражаючи своєю точністю та гнучкістю інтеграції з різними медичними системами [14]. Його особливістю є здатність точно розпізнавати медичну термінологію, що робить його важливим інструментом для перетворення аудіозаписів лікарських прийомів, операційних записів та інших медичних діалогів у текстовий формат. Таке застосування може оптимізувати робочі процеси медичних працівників, економлячи час і зусилля, які раніше витрачалися на ручне документування (рис. 1.11).

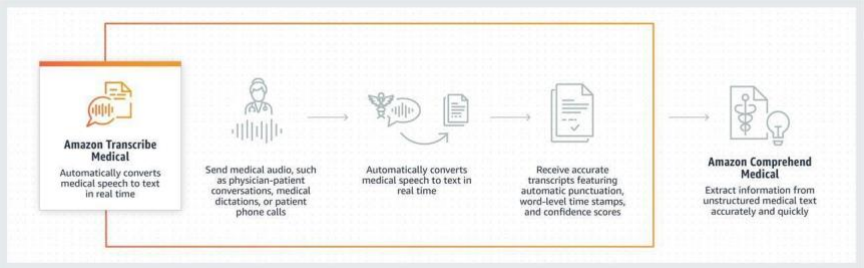



Рисунок 1.11 – Amazon Transcribe Medical

Проте, цей сервіс також стикається з певними обмеженнями та викликами. Особливо актуальними є питання конфіденційності та безпеки пацієнтських даних, особливо в контексті політики конфіденційності Amazon та стандартів захисту медичної інформації.

Крім того, доступність та сумісність цього сервісу в різних регіонах та серед різних медичних установ може варіюватися, що вимагає додаткових розглядів.

І останнім недоліком є те, що даний сервіс розроблено для інтеграції з іншими програмними додатками через Amazon Speech API. Іншими словами, цей сервіс призначений для використання в рамках інших додатків або для інтеграції в нові додатки, які розробляються сторонніми розробниками для конкретних випадків використання. Це робить його надмірно складним для звичайного лікаря або користувача. Таким чином, незважаючи на відмінну якість і високий рівень розробки AWS Transcribe Medical, на практиці він не є зручним для медичного застосування

Загалом, Amazon Transcribe Medical представляє собою значущий крок уперед у цифровізації медичних процесів, пропонуючи ефективні рішення для перетворення мовлення на текст. Його застосування обіцяє суттєво покращити якість та швидкість медичного обслуговування, однак потребує ретельного врахування важливих питань конфіденційності та інтеграції.

### 1.3.2 Огляд Google Speech Recognition

У сучасному світі цифрових технологій, одним із провідних гравців на ринку сервісів розпізнавання голосу є Google з їхнім продуктом Google Speech Recognition [15].

Цей сервіс, розроблений корпорацією Google, підтримує розпізнавання голосу для десятків різних мов, включаючи й найпопулярніші світові мови. Застосування цього сервісу надзвичайно ефективне в мобільних телефонах або через веб-браузер для створення загальних нотаток та коментарів на веб-сторінках, таких як Facebook.

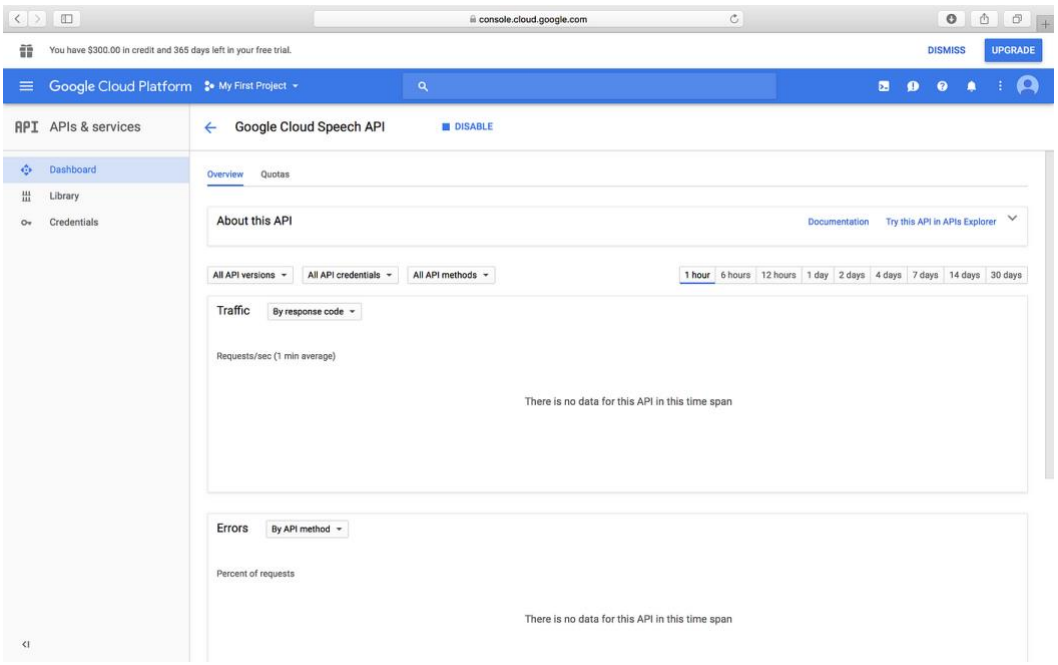



Рисунок 1.12 – Google Speech Recognition

Проте, не дивлячись на високу ефективність та універсальність, Google Speech Recognition має свої обмеження. Найбільш помітним з них є відсутність спеціалізованого медичного словника, що робить цей сервіс недостатньо адекватним для термінології, яка використовується в медичній галузі. Таким чином, хоча Google Speech Recognition і є відмінною системою розпізнавання голосу, вона все ж залишається засобом загальнопризначеного комунікаційного перетворення голосу в текст, а не професійною системою диктування. Цей факт вказує на існування значної ніші для розробки більш спеціалізованих систем, зокрема для української мови, яка б враховувала специфіку медичного дискурсу та термінологію.

### 1.3.3 Огляд Nuance Medical Speech Recognition Solutions

У контексті розвитку технологій розпізнавання мовлення в медичній галузі, компанія Nuance пропонує одне з передових рішень - Medical Speech Recognition Solutions. Ця система розроблена спеціально для потреб медицини, що відрізняє її від більшості універсальних інструментів розпізнавання мовлення. Nuance Medical Speech Recognition Solutions характеризується глибокою інтеграцією з медичними додатками та системами, що дозволяє лікарям здійснювати швидке та точне документування медичних записів за допомогою голосу [16].

Ця система особливо цінна завдяки своїй здатності точно розпізнавати специфічну медичну термінологію, що є ключовим фактором для ефективності медичної практики.

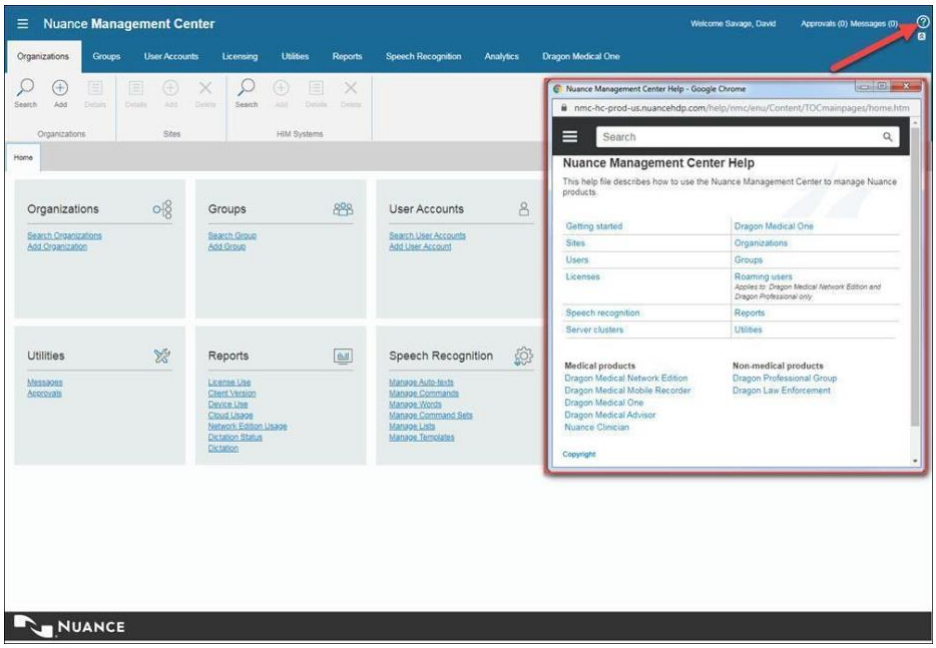



Рисунок 1.13 – Nuance Medical Speech Recognition Solutions

Однак, не дивлячись на її високу функціональність, існують питання, пов’язані з доступністю та адаптацією такої системи в різних країнах, особливо з огляду на мовні та регіональні особливості.

Також важливим є питання конфіденційності пацієнтських даних, яке є актуальним для всіх цифрових медичних технологій. Nuance Medical Speech Recognition Solutions, таким чином, є важливим кроком у напрямку автоматизації медичного документування, забезпечуючи лікарям потужний інструмент для ефективного та точного ведення медичних записів. Це рішення, без сумніву, відіграє значущу роль у покращенні якості медичного обслуговування, хоча й потребує врахування ряду викликів та обмежень.

### 1.3.4 Огляд SpeechText.AI Medical Speech Recognition Softwar

SpeechText.AI представляє свій продукт у галузі медичного розпізнавання мовлення - Medical Speech Recognition Software, який є відносно новим, але обнадійливим рішенням на ринку. Цей програмний продукт спрямований на

вирішення конкретних завдань медичної сфери, зокрема на перетворення голосових заміток та записів лікарів у текстовий формат [17].

Головною перевагою Medical Speech Recognition Software від SpeechText.AI є його здатність адаптуватися до специфіки медичної термінології та контексту використання. Це робить його цінним інструментом для лікарів, які шукають швидкі та точні способи документування медичної інформації.

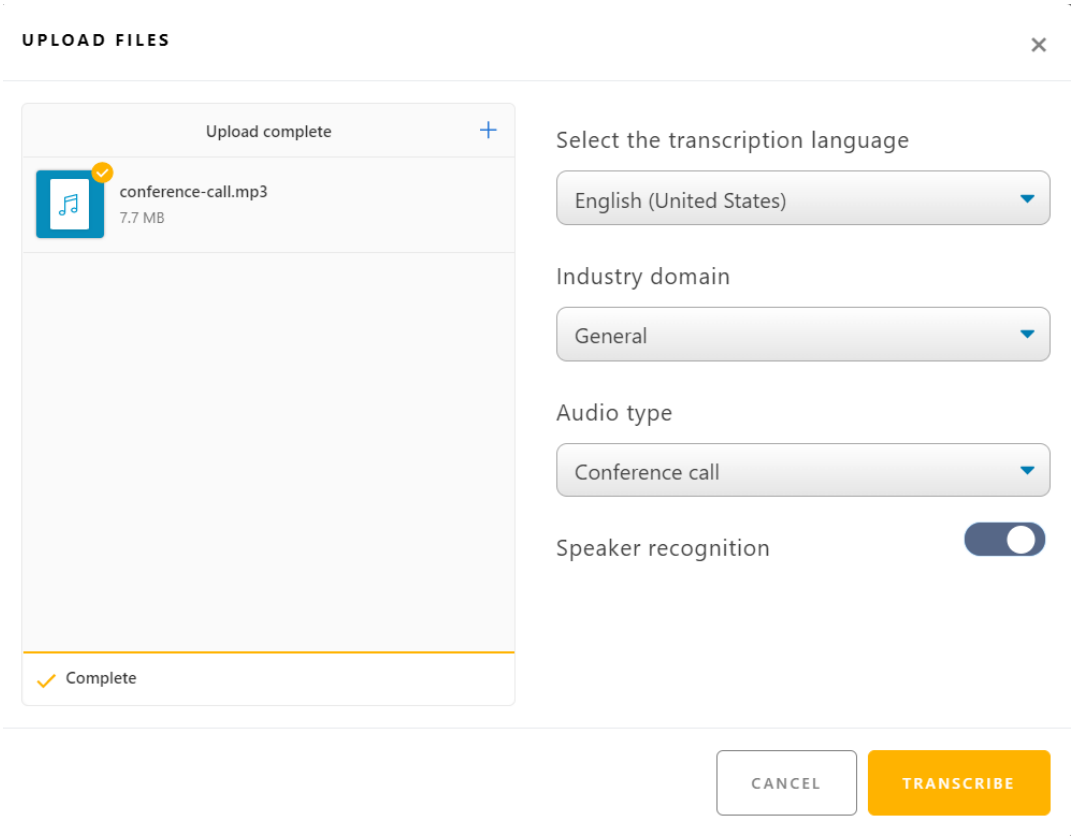



Рисунок 1.14 – SpeechText.AI Medical Speech Recognition Softwar

Проте, необхідно враховувати деякі потенційні обмеження цього сервісу. Важливим аспектом є його інтеграція з іншими медичними системами та програмами. Оскільки SpeechText.AI є відносно новим гравцем на ринку, можливості його сумісності та інтеграції з різноманітними медичними додатками можуть бути обмеженими. Крім того, питання конфіденційності та безпеки даних завжди залишаються актуальними, особливо в медичній сфері.

Загалом, Medical Speech Recognition Software від SpeechText.AI відкриває нові горизонти в області медичного розпізнавання мовлення, пропонуючи

інноваційні рішення для перетворення голосу в текст. Його потенціал у поліпшенні медичного документування є значним, хоча і потребує подальшого розвитку та адаптації до вимог медичних працівників.

### 1.3.5 Аналіз InVox MedicalEnglish Speech Recognition

InVox Medical представляє собою продукт у сфері англомовного медичного розпізнавання мовлення, що розроблений для задоволення потреб медичних фахівців, особливо в англомовних регіонах [18]. Цей сервіс фокусується на точному перетворенні медичного мовлення в текст, дозволяючи лікарям ефективно та швидко документувати медичні записи.

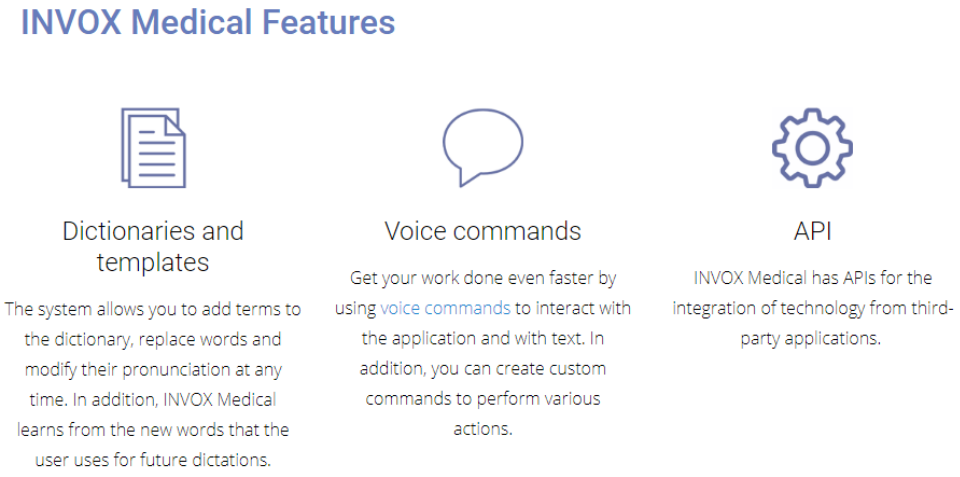



Рисунок 1.15 – InVox MedicalEnglish Speech Recognition

Основною перевагою InVox Medical є його висока точність у розпізнаванні англійської медичної термінології, що робить його інструментом вибору для лікарів, що працюють в англомовному середовищі. Система оптимізована для розпізнавання різних акцентів та діалектів, забезпечуючи широку сумісність у різних англомовних регіонах.

Проте, як і багато інших спеціалізованих рішень, InVox Medical має свої обмеження, зокрема щодо інтеграції з іншими медичними системами. Також важливим є питання доступності цього продукту у неангломовних регіонах та його адаптації до місцевих медичних умов та потреб.

Загалом, InVox Medical являє собою значний внесок у поле англомовного медичного розпізнавання мовлення, пропонуючи високоякісне рішення для медичного документування. Це рішення може значно полегшити роботу медичних фахівців у англомовних країнах, хоча й вимагає розгляду питань інтеграції та міжнародної доступності.

### 1.3.6 Порівняння систем розпізнавання мовлення

Порівняльний аналіз найбільш важливих характеристик існуючих технологічних рішень представлено в табл.1.1.

Таблиця 1.1 – Порівняльна характеристика відомих технологічних рішень (систем / програм) для розпізнавання україномовного аудіосигналу з перетворенням у текст.

| Критерій | Amazon | Google | Nuance | SpeechText.AI | InVox |
|---|---|---|---|---|---|
| Підтримка української мови | Ні | Так | Ні | Ні | Ні |
| Доступність через API | Так | Так | Так | Так | Так |

| | | | | | |
|---|---|---|---|---|---|
| Зручність використання для лікарів | Обмежена зручність | Висока зручність | Висока зручність | Середня зручність | Висока зручність |
| Цінова доступність | Висока | Середня | Висока | Середня | Висока |
| Інші важливі критерії | Хороша інтеграція з медичними системами; проблеми з конфіденційністю | Не спеціалізується на медичній термінології | Сильна спеціалізація на медичній термінології; проблеми з доступністю у різних регіонах | Новий на ринку; потенційні проблеми з інтеграцією | Спеціалізовано на англомовних регіонах; обмежена міжнародна доступність |

На основі проведеного аналізу та порівняння п'яти ключових систем розпізнавання мовлення — Amazon Transcribe Medical, Google Speech Recognition, Nuance Medical Speech Recognition Solutions, SpeechText.AI Medical Speech Recognition Software та InVox Medical - English Speech Recognition — можна зробити наступні висновки:

- Жодна з розглянутих систем не підтримує українську мову, окрім Google Speech Recognition, що вказує на значний пробіл у цьому сегменті ринку та можливість для розвитку спеціалізованих рішень.
- Усі аналізовані системи надають доступ через API, що є важливим для інтеграції з іншими медичними додатками та платформами.
- Google Speech Recognition, Nuance Medical Speech Recognition Solutions та InVox Medical забезпечують високу зручність використання для лікарів, тоді як Amazon Transcribe Medical та SpeechText.AI мають деякі обмеження з точки зору зручності.

- Amazon Transcribe Medical, Nuance Medical Speech Recognition Solutions та InVox Medical пропонують високу цінову доступність, в той час як Google Speech Recognition та SpeechText.AI мають середній рівень цінової доступності.
- Важливо відзначити спеціалізацію Nuance на медичній термінології, а також потенційні проблеми з інтеграцією та доступністю, які мають місце у деяких системах.

Таким чином, вибір системи розпізнавання мовлення значною мірою залежить від конкретних потреб користувачів, зокрема медичних працівників. Важливо враховувати не лише технічні характеристики, але й цінову доступність, зручність використання, а також можливість інтеграції з існуючими медичними системами. Наразі існує прогалина у підтримці української мови, що відкриває шлях для розробки спеціалізованих продуктів, здатних задовольнити цей попит.

## 1.4 Впровадження розпізнавання мовлення в медичній практиці

У сучасний час значна увага приділяється ефективності медичних процесів, де кожна хвилина має вирішальне значення. В цьому контексті технологія розпізнавання мовлення набуває особливої актуальності, відкриваючи нові можливості для оптимізації роботи медичних установ.

Застосування технології розпізнавання мовлення у медицині можна простежити в декількох ключових сферах. Найбільш поширеним є використання для документації пацієнтів, де лікарі використовують голосові команди для створення та оновлення медичних записів. Це значно спрощує процес документування, забезпечуючи одночасно точність та швидкість внесення інформації. Також ця технологія активно використовується для транскрипції медичних записів та у спілкуванні з пацієнтами, зокрема у кол-центрах.

Вивчення відгуків користувачів та результатів клінічних досліджень є важливим для оцінки ефективності використання цієї технології. Багато лікарів та медичних працівників позитивно відзначають впровадження розпізнавання

мовлення, оскільки воно дозволяє їм зосередитися на безпосередньому наданні медичної допомоги, замість витрачання часу на ручне ведення записів. Водночас, існують виклики, пов'язані з точністю розпізнавання медичної термінології та з питаннями конфіденційності інформації, що вимагають подальшої уваги та вдосконалення систем.

Також згідно з аналізом статті "An Analysis of the Implementation and Impact of Speech-Recognition Technology in the Healthcare Sector", можна визначити декілька ключових аспектів використання цієї технології в медицині [19]:

- Підвищення продуктивності медичних працівників - використання розпізнавання мовлення дозволяє лікарям та медичному персоналу ефективніше управляти своїм часом, забезпечуючи швидше та точніше документування медичних записів. Це сприяє зниженню адміністративного навантаження та дозволяє більше часу присвятити безпосередньому лікуванню пацієнтів.
- Вплив на якість обслуговування пацієнтів - технологія не лише покращує адміністративні аспекти медичного обслуговування, але й сприяє підвищенню загальної якості догляду за пацієнтами. Завдяки точному та оперативному веденню медичних записів, можна уникнути помилок та недоліків, пов'язаних з ручним введенням даних.
- Виклики та обмеження – слід враховувати виклики, з якими зіштовхуються системи розпізнавання мовлення, зокрема, що стосується точності розпізнавання медичної термінології та забезпечення конфіденційності даних пацієнтів. Ці аспекти вимагають постійного дослідження та вдосконалення.

Розглядаючи використання розпізнавання мовлення в медичній сфері, можна відзначити його значний потенціал у покращенні ефективності та точності медичного документування. Однак, для оптимального використання цієї технології необхідно враховувати існуючі виклики, зокрема пов'язані з точністю та конфіденційністю даних, що залишаються важливими аспектами для подальших досліджень та розвитку.

## 1.5 Висновки до розділу

На основі проведеного аналізу за темою магістерської кваліфікаційної роботи з'ясовано, що актуальність розпізнавання української мови у медичній сфері визначається зростаючою потребою в автоматизації процесів документації та комунікації між медичними працівниками та пацієнтами. Ця потреба підкреслює важливість розробки нових технологічних рішень, здатних ефективно обробляти медичну термінологію в українському мовному контексті.

Детальний аналіз предметної області підтверджує, що існуючі методи розпізнавання мовлення не повністю задовольняють специфічним вимогам медичної галузі. Це відкриває простір для розробки нових методів, які можуть врахувати унікальні аспекти української мови та медичного спілкування.

Огляд існуючих продуктів та сервісів показав, що багато з них мають обмежену функціональність, коли йдеться про роботу з українською мовою в контексті медичних термінів та діалогів. Це демонструє необхідність створення більш спеціалізованих рішень, що забезпечать точність та ефективність розпізнавання мовлення.

Вивчення потенціалу впровадження розпізнавання мовлення в медичній практиці підтвердило його значний вплив на якість медичних послуг. Автоматизація процесів запису медичної інформації може суттєво спростити рутинну роботу медиків та забезпечити більшу точність медичних записів.

Отже, необхідність розвитку та впровадження методів розпізнавання українського мовлення медичного спрямування є очевидною та обґрунтованою. Результати аналізу створюють тверду основу для подальшого дослідження та розробки в цій важливій області.

# 2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМНОГО ЗАСОБУ

## 2.1 Визначення основного функціоналу

При розробці методу розпізнавання українського мовлення медичного спрямування, важливим аспектом є створення програмного продукту, який відповідає визначеним вимогам.

Однією з основних вимог до цього продукту є забезпечення можливості анотації аудіозаписів. Користувачі мають можливість записувати та надсилати аудіозаписи через телеграм-бота, доповнюючи їх анотованим текстом. Для забезпечення високої якості анотацій, програма повинна надавати опцію перевірки цих анотацій супервайзером.

Ще одним критичним аспектом є інтеграція програми з базою даних. Це необхідно для ефективного зберігання аудіозаписів та анотованого тексту. Інтеграція з базою даних також дозволить користувачам отримувати доступ до розмічених даних за визначений період, що сприяє кращому управлінню та аналізу інформації.

Розпізнавання мовлення з аудіозаписів є ще однією важливою вимогою. У цьому контексті програма повинна використовувати Google Speech Recognition як базовий підхід, доповнюючи його власною моделлю, натренованою на зібраних даних. Таке поєднання підходів забезпечить не тільки високу точність розпізнавання, але й адаптацію до особливостей медичного мовлення.

Нарешті, ефективна взаємодія з користувачами через телеграм-бота є важливою для забезпечення зручності користування програмою. Інтерфейс повинен бути інтуїтивно зрозумілим та зручним, надаючи користувачам можливість легко отримувати інформацію про анотації, переглядати та аналізувати дані. Також важливо, щоб інтерфейс давав змогу користувачам налаштовувати параметри програми згідно з їхніми потребами.

У сукупності, ці вимоги визначають основний функціонал програми, забезпечуючи її ефективність і точність у роботі з медичними аудіоданими, а

також відповідають потребам медичних фахівців у сфері розпізнавання та анотації мовлення.

## 2.2 Вибір технологічних засобів

У даному підрозділі буде розглянуто вибір мови програмування та будуть проаналізовані основні інструменти для реалізації поставленої задачі.

### 2.2.1 Вибір мови програмування

Як відомо мова програмування - це лише інструмент для втілення в життя задуманого проекту. Якщо ж розглядати вибір мови програмування для проектів з обробкою природної мови, все зводиться до того, яка мова вам найбільше комфортна та на яку мову постачається максимальна кількість інструментів, які допоможуть у виконанні завдань, пов’язаних з NLP [21, 22].

В даному випаду будуть розглянуто три мови програмування, які відповідають вище описаним критеріям. Це мови Python, Java та R [23, 24].

Python вважається швейцарським армійським ножем програмування через його універсальність. Це також одна з найбільш зручних для початківців мов, яка відображає людськи та послідовний синтаксис. Python також постачається з безліччю пакетів, які допоможу в реалізації задач широкого спектру. Його семантика та синтаксис прозорі, що робить його чудовим вибором для обробки природної мови [25] . Крім того, це просто і забезпечує дивовижну підтримку інтеграції з іншими інструментами та мовами.

Безсумнівно, важко розробити програмне забезпечення, яке здатне обробляти природну мову. Але розширений набір інструментів Python дозволяє розробникам створювати чудові проекти.

Java – ще одна часто вживана мова програмування в галузі обробки природної мови . За допомогою цієї мови ви можете вивчити, як організувати текст за допомогою повнотекстового пошуку, вилучення інформації, кластеризації та позначення тегами [25, 26]. Оскільки Java є незалежною від платформи мовою, це полегшує обробку інформації. Він поставляється з OpenNLP, LingPipe та Stanford CoreNLP. Крім того, Lucene, яка є повнотекстовою бібліотекою пошуку, також може допомогти у забезпеченні токенізації та всебічного аналізу тексту.

І остання мова, яка буде розглянута – це R. Хоча вона популярна для використання в статистичному навчанні, він широко використовується для обробки природної мови . У контексті NLP мова відіграє вирішальну роль у дослідженні великих даних, а також стає корисною для обчислювальної аналітики навчання [27].

Незважаючи на переваги кожної з цих мов, для реалізації поставленої задачі буде використовуватися саме Python. Вирішальним критерієм вибору саме цієї мови стало те, що на ній легко можна розробити графічний інтерфейс користувача. Також треба зазначити, що саме на цій мові є переважний досвід програмування.

### 2.2.2 Вибір інструментальних засобів для розробки

У даному підрозділі досліджуються інструментальні засоби, які можуть бути застосовані для розробки методу розпізнавання українського мовлення медичного спрямування. Основна увага приділяється програмним бібліотекам, що використовуються в Python для реалізації поставленої задачі.

Першою бібліотекою є Pandas – це ефективний інструмент для маніпулювання даними та їх аналізу. Ця бібліотека включає структури даних для роботи з чисельними таблицями та часовими рядами, і буде корисною для

збереження результатів тестування та оцінки якості програмного забезпечення за допомогою різних метрик.

PyAudio, бібліотека для запису аудіо у Python. Вона здатна записувати аудіо потоки у форматі bytes, що дозволяє зберігати дані у форматі WAV за допомогою бібліотек sciру або wave. PyAudio є важливою для збору аудіо даних, зокрема, з мікрофона [28].

Loguru - бібліотека для логування в Python, яка дозволяє керувати налаштуваннями логування через один об'єкт logger. Це полегшує відстеження стану системи та швидке виявлення помилок або непередбачуваної поведінки у коді.

pyTelegramBotAPI – бібліотека для створення ботів у Telegram. Вона дозволяє легко інтегрувати розроблені алгоритми з інтерфейсом Telegram, забезпечуючи зручний доступ до функціоналу розпізнавання мовлення для кінцевих користувачів. Це особливо корисно для медичних фахівців, які можуть використовувати бот для швидкого перекладу аудіозаписів на текст.

Pymongo – інструмент для роботи з MongoDB, невід'ємною частиною проекту. pymongo дає можливість ефективно зберігати, запитувати та обробляти великі обсяги даних, що є критично важливим для зберігання та аналізу медичних записів і метаданих, пов'язаних з розпізнаванням мовлення.

FastAPI, сучасний і швидкий веб-фреймворк для Python, ідеально підходить для створення API для розпізнавання мовлення. Цей фреймворк забезпечує високу продуктивність та легкість у створенні маршрутів, що дозволяє ефективно інтегрувати різні компоненти системи [29, 30].

Transformers від Hugging Face є ще однією критично важливою бібліотекою, яка надає потужні засоби для роботи з передовими алгоритмами машинного навчання, зокрема тими, що базуються на моделях трансформерів. Ці моделі демонструють високу ефективність у задачах обробки природної мови, що робить їх ідеальними для розпізнавання та аналізу медичного мовлення.

PEFT-LoRA (Parallel Execution Framework with Tensor Decomposition and Low-Rank Approximation) представляє собою сучасний підхід в області

оптимізації обчислень, зокрема для завдань, що вимагають інтенсивних обчислень. Цей інструмент особливо корисний у сфері обробки природної мови та машинного навчання, де обсяги даних та складність моделей постійно зростають

Нарешті, бібліотека SpeechRecognition, яка підтримує різні механізми та API для розпізнавання мовлення, як у онлайн, так і в офлайн режимах. Ця бібліотека є безкоштовною для використання, хоча обмежується тривалістю звукового запису, проте це не становить перешкоди для даної роботи.

Вибір цих інструментів формує фундамент для розробки ефективного і надійного методу розпізнавання мовлення.

### 2.2.3 Вибір інтегрованого середовища розробки

У даному підрозділі буде вибрано IDE, для вибраної мови програмування, за допомогою якого в подальшому буде реалізована поставлена задача. Вибір буде здійснюватися з топ-3 найпопулярніших IDE для мови програмування Python [31, 32].

IDE (або інтегроване середовище розробки) – це програма, призначена для розробки програмного забезпечення. Як випливає з назви, IDE об'єднує кілька інструментів, спеціально призначених для розробки. Ці інструменти зазвичай включають редактор, призначений для роботи з кодом, інструменти збірки, виконання та налагодження, а також певну форму системи управління версіями. Статистичні характеристики застосування відомих IDE для мови програмування Python представлені на рис. 2.1.

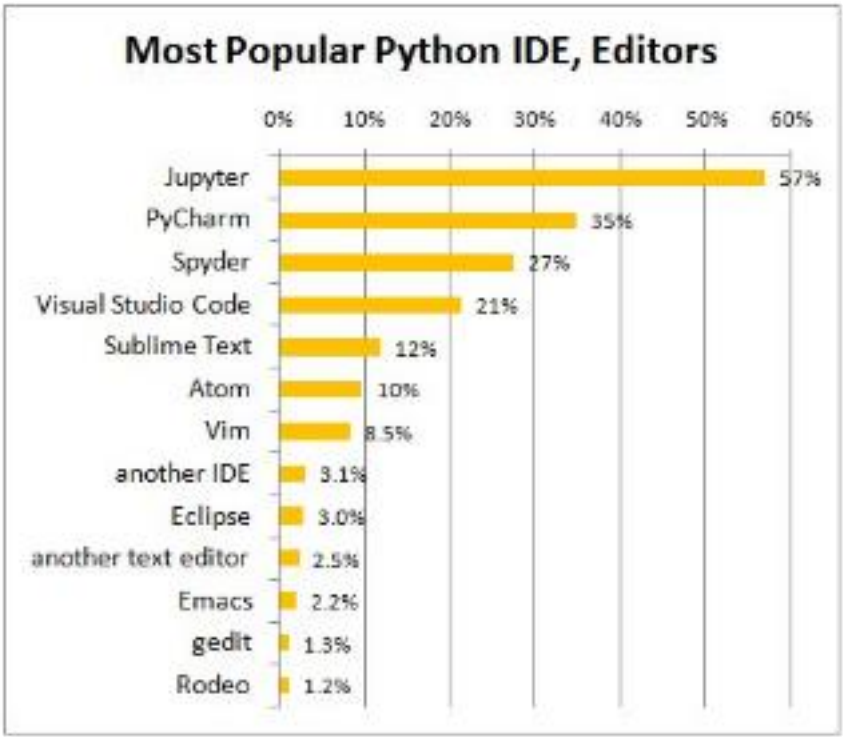



Рисунок 2.1 – Найпопулярніші IDE для мови програмування Python

Jupyter – це структура сервера-клієнта на основі веб-додатків, яка проста у використанні та дозволяє створювати, аналізувати та обробляти документи у вигляді блокнотів. Оскільки це веб-інтерфейс, він може інтегрувати багато існуючих веб-бібліотек для візуалізації даних, включаючи plotly.js .

PyCharm – інтегроване середовище розробки для мови програмування Python [34]. Надає засоби для аналізу коду, графічний відладник, інструмент для запуску юніт-тестів і підтримує веб-розробку на Django, Flask. PyCharm розроблена чеською компанією JetBrains на основі IntelliJ IDEA.

Spyder – це легкий IDE з відкритим кодом, який постачається з попередньо встановленим дистрибутивом Anaconda і був розроблений в основному для спеціалістів з обробки даних. Він може не мати привабливого інтерфейсу користувача, як PyCharm але його слід спробувати, враховуючи кількість функціональних можливостей, які він може запропонувати.

У даній роботі буде використовуватися саме PyCharm. Вибір зумовлений приємним інтерфейсом, широким спектром функціоналу та підтримкою веб-

розробки. Але для відладки деяких програмних функцій використовувався Jupyter.

## 2.3 Об'єктно-орієнтоване проектування та моделювання

У цьому розділі магістерської роботи особлива увага приділяється ролі використання діаграм для моделювання бізнес-процесів та об'єктно-орієнтованого проектування, зокрема у рамках розробки програмного забезпечення для анотації аудіозаписів з автоматичним анотуванням за допомогою методів штучного інтелекту.

Діаграми є незамінним інструментом у візуалізації та аналізі взаємодії об'єктів і послідовностей дій у програмній системі. Вони також відіграють ключову роль у моделюванні бізнес-процесів, що є надзвичайно важливим для ефективної реалізації проекту. Вступ до цього розділу визначає основні цілі та обсяг дослідження, а також підкреслює значимість діаграм у контексті магістерської роботи. Це дозволяє глибше зрозуміти як структуру розроблюваного програмного забезпечення, так і логіку його роботи, забезпечуючи тим самим міцну основу для подальшої реалізації проекту.

### 2.3.1 UML – object diagram

Діаграма об'єктів – в UML, діаграма, що відображає об'єкти та їх зв'язки в певний момент часу. Діаграма об'єктів може розглядатись як окремий випадок діаграми класів, на якій можуть бути представлені як класи, так і екземпляри (об'єкти) класів. На рисунку 2.2 зображено діаграму об’єктів системи магістерської кваліфікаційної роботи. Продемонстровано взаємодію об’єктів у певний момент часу. Таким чином всього можна виділити 9 об’єктів взаємодії.

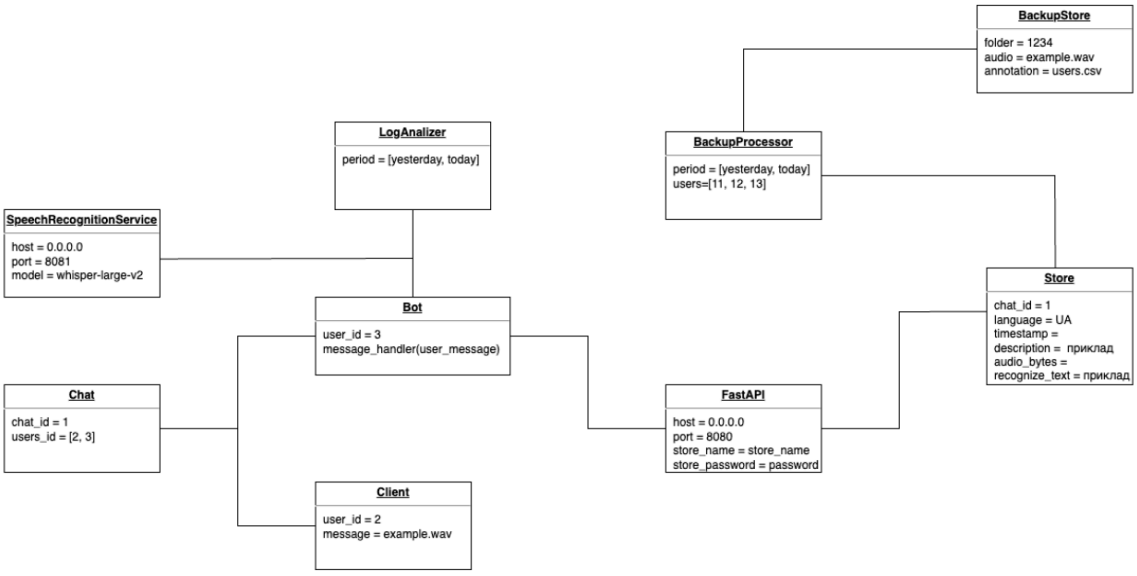



Рисунок 2.2 – UML-діаграма об’єктів

Клієнт взаємодії із ботом за допомогою чату. Кожен процес, який викликається ботом логується та аналізується аналізатором логів. Для здійснення розпізнавання мовлення бот звертається до об’єкту SpeechRecognitionService, який приймає аудіозапис та вертає анотацію.

Для зберігання всіх аудіозаписів бот взаємодіє з об’єктом Store за допомогою REST-API інтерфейсу в ролі якого виступає об’єкт FastAPI. Він надає змогу зберігати та отримувати аудіозаписи з бази даних, по відповідним запитам. Усі аудіозаписи з бази даних переносяться у об’єкт BackupStore під час виклику за тригером об’єкту BackupProcessor, який приймає як параметри визначений період та унікальні ідентифікатори користувачів.

### 2.3.2 UML - deployment diagram

Діаграма розгортання (deployment diagram) – UML-діаграма, що демонструє вузли системи та їх взаєморозміщення [35]. Її призначення це представлення загальної структури та топології системи.

Також діаграма розгортання показує наявність фізичних з'єднань – маршрутів передачі інформації між апаратними засобами, задіяними в реалізації системи. На ній графічно зображуються процесори, пристрої, процеси та зв'язки між ними. Діаграма розгортання, на відміну від логічних, є єдиною для системи в цілому, оскільки відображає всі особливості її реалізації (рис. 2.3).

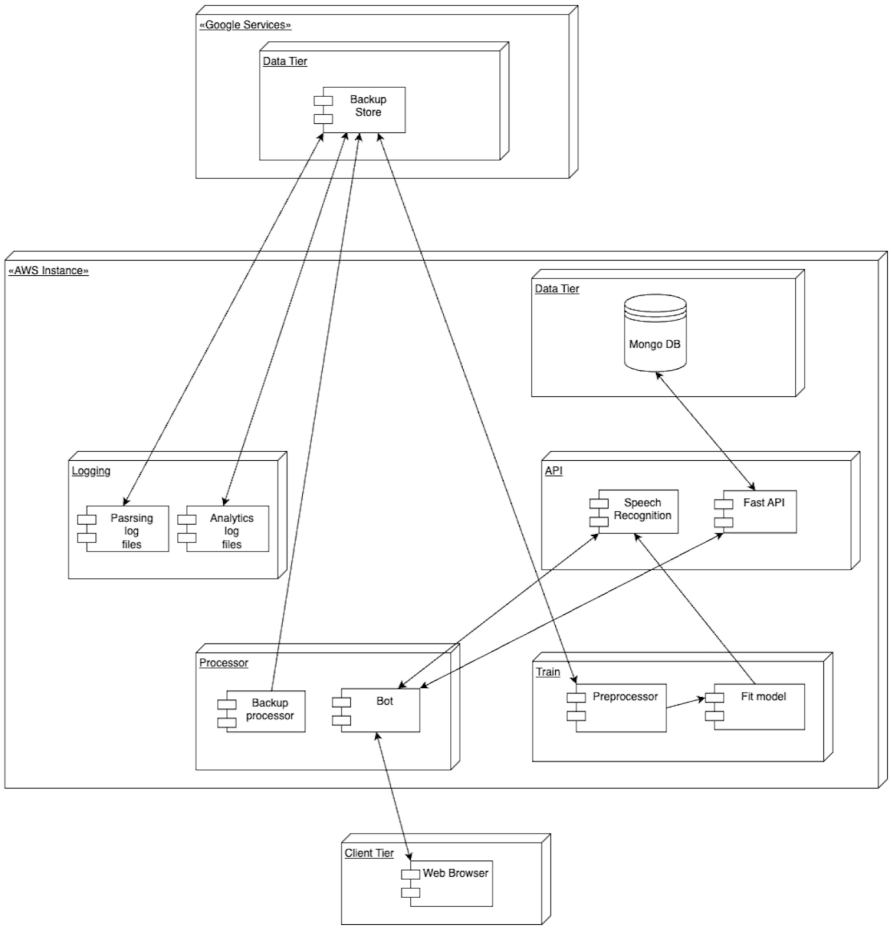



Рисунок 2.3 – UML-діаграма розгортання

Як видно з рисунку 2.3 діаграма розгортання містить декілька блоків, які містять відповідні підмодулі:

- Data Tier – містить базу даних MongoDB та гугл драйв для backup бази даних.
- API – містить сервіс для отримання анотації за допомогою SpeechRecognition сервісу та FastAPI для взаємодії між базою даних.

- Train – містить модуль для препроцесінгу даних та модуль для натренування моделі.
- Logging – містить модуль для структуризації логів та їхнього аналізу.
- Processor – містить підмодуль для здійснення управлінням телеграм боту та підмодуль для створення резервного копіювання бази даних.
- Client Tier- містить спосіб взаємодії бота з користувачем.

### 2.3.3 UML - use case diagram

Діаграма варіантів використання (use case diagram) – вихідна концептуальна модель процесів проектування та розробки системи [33]. На ній позначаються відношення між зовнішніми агентами і варіантами використання. Ця діаграма описує загальні функції модельованої системи без розгляду її внутрішньої структури.

Дана UML-діаграма являється графом, що представляє систему у вигляді певної кількості акторів чи сутностей, а взаємодія яких із системою відбувається за допомогою «варіантів використання». Останні використовуються для опису послуг (деякого набору дій), що система надає актору.

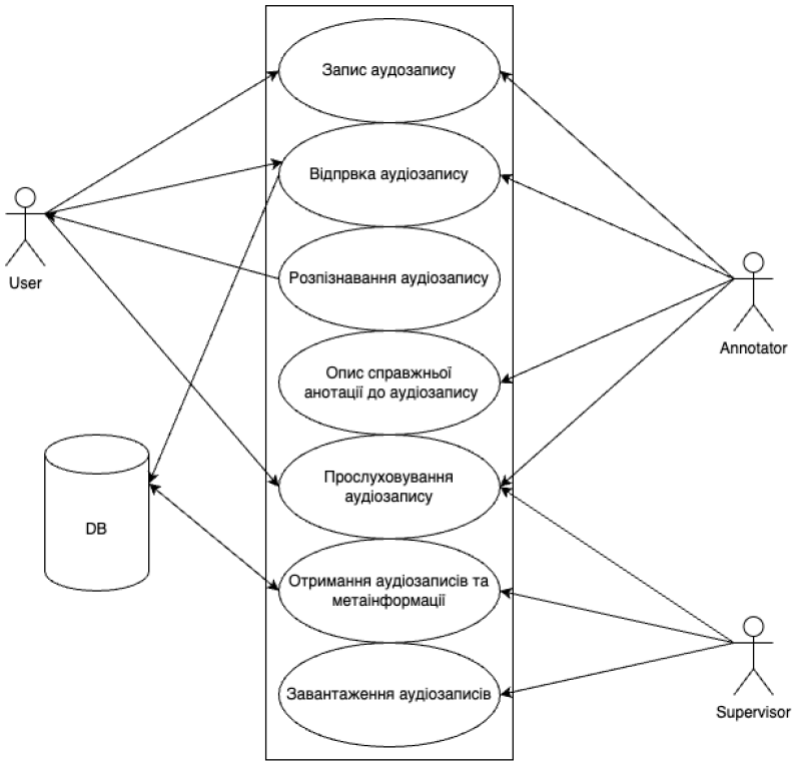



Рисунок 2.4 – UML- Use Case diagram діаграма

UML-діаграму варіантів використання, яка намальована на рисунку 2.4, і яка ілюструє різні способи використання розробленого телеграм-бота в залежності від типу користувача. Вираз "варіанти використання" вказує на різні сценарії або можливі дії, які можуть бути виконані за допомогою цього бота.

Існує 3 сценаричні ролі використання, а саме - користувач, анотатор і супервайзер.

User (користувач) – це звичайний користувач, який хоче використовувати бота. Головними діями для цього користувача є завантаження аудіозапису і отримання до нього анотації.

Annotator (анотатор) – це другий тип користувача, який має спеціальні права. Він також може завантажити аудіозапис та отримати анотацію, але, додатково, він має можливість створити валідаційний опис для аудіозапису. Валідаційний опис, імовірно, використовується для перевірки анотацій на правильність.

Supervisor (супервайзер) – це третій тип користувача, який також має особливі права та функції. Супервайзер може отримувати записані аудіозаписи з бази даних і перевіряти валідність анотацій. Його головною роллю може бути контроль якості анотацій та аудіозаписів, що зберігаються у системі.

Отже, дана UML-діаграма варіантів використання демонструє, як різні типи користувачів можуть взаємодіяти з телеграм-ботом та використовувати його функціональність у відповідності до їхніх потреб та прав доступу.

## 2.4 Висновки до розділу

На основі проведеного дослідження в рамках розділу "Розробка архітектури програмного засобу" можна зробити наступні висновки:

Розробка архітектури програмного засобу для розпізнавання українського мовлення медичного спрямування вимагає детального визначення основного функціоналу, який відповідає потребам кінцевих користувачів. Визначення функціоналу є фундаментом для створення ефективного та надійного програмного продукту. Важливо, щоб функціональні можливості програмного засобу охоплювали весь спектр медичних задач, від простого розпізнавання мовлення до комплексної обробки і аналізу медичної інформації.

Вибір технологічних засобів є критичним кроком, який впливає на якість та ефективність кінцевого продукту. В рамках розробки важливо вибрати ті засоби і технології, які найкраще відповідають вимогам проекту, забезпечують високу продуктивність, масштабованість та легкість в обслуговуванні. Такий підхід дозволяє забезпечити максимальну адаптивність та легкість інтеграції програмного засобу в медичні інформаційні системи.

Об'єктно-орієнтоване проектування та моделювання дозволяють створити гнучку та масштабовану архітектуру, яка може бути легко адаптована під змінювані вимоги користувачів та ринкові тенденції. Моделювання допомагає

чітко структурувати дані та процеси в рамках програмного продукту, забезпечуючи таким чином зручність подальшої розробки та підтримки.

У цілому, розробка архітектури програмного засобу була виконана з урахуванням найкращих практик та сучасних підходів до проектування програмного забезпечення, що повинно забезпечити високу ефективність та надійність майбутньої системи розпізнавання мовлення.

# 3 РЕАЛІЗАЦІЯ СИСТЕМИ ДЛЯ ЗДІЙСНЕННЯ АНОТАЦІЙ

У даному розділі основна увага буде зосереджена на формальних та практичних аспектах реалізації розробленого методу розпізнавання українського мовлення медичного спрямування. Цей розділ описує ключові кроки розробки, починаючи від підготовчих робіт, таких як збір та підготовка даних, до етапу тренування та тестування моделі.

Першим етапом у розробці є збір і аналіз даних, що використовуються для тренування моделі. Оскільки якість та репрезентативність даних безпосередньо впливають на ефективність розпізнавання мовлення, цей етап має вирішальне значення. Він включає збір аудіозаписів, анотацію, та підготовку датасету, забезпечуючи, щоб він був достатньо об'ємним і різноманітним для ефективного тренування.

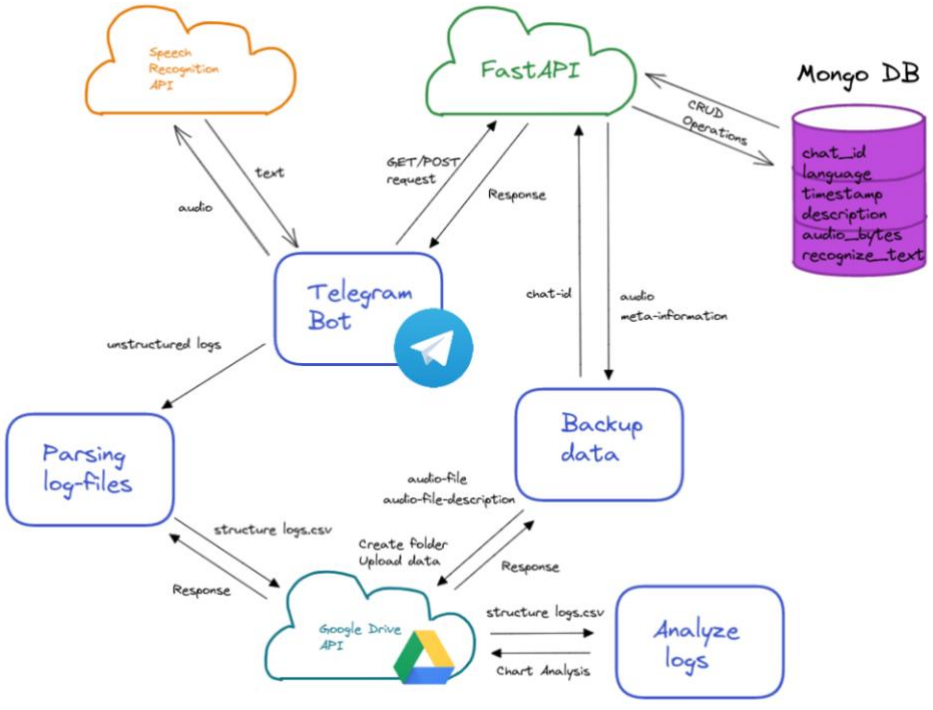



Рисунок 3.1 – Схема архітектури

В результаті цього розділу магістерської роботи буде створено повноцінний прототип програмного забезпечення, який демонструє можливості

та ефективність розробленого методу розпізнавання мовлення. Також будуть обговорені виклики, з якими зіткнулась розробка, та перспективи подальшого розвитку проекту.

### 3.1 Вибір графічного інтерфейсу

Визначення оптимального графічного інтерфейсу для поточної мети було ключовим етапом. Враховуючи досвід попередньої бакалаврської роботи, де було розроблено десктопне програмне забезпечення для найпоширеніших ОС (Windows, macOS, Linux), було зроблено висновок про необхідність зміни підходу для покращення масштабованості.

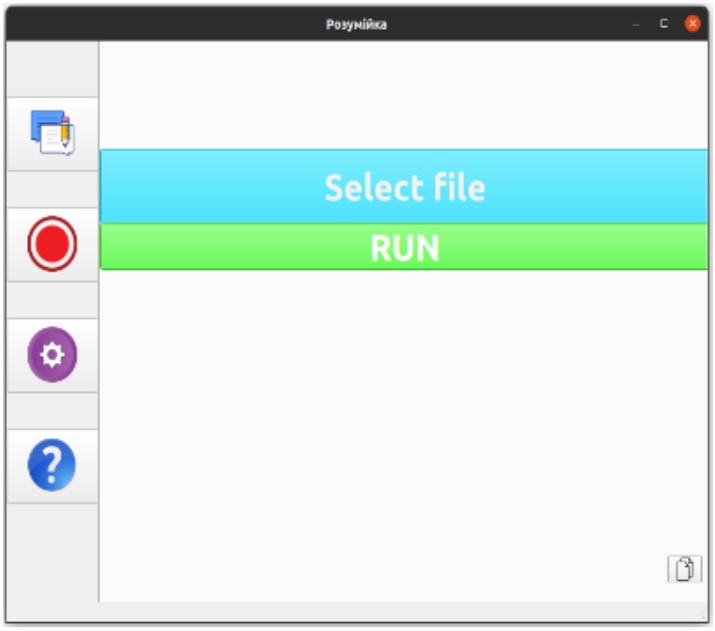



Рисунок 3.2 - Головне меню програми

Основним недоліком раніше обраного підходу до розробки десктопного програмного забезпечення була його обмежена масштабованість та відсутність гнучкості у використанні (рис. 3.1). Така статична архітектура виявилася не найефективнішою для впровадження в широкому діапазоні медичних установ та серед різних категорій користувачів.

Вирішенням цієї проблеми стало переорієнтування на кросплатформність, яка значно розширює можливості використання та інтеграції програмного продукту.

У результаті аналізу різних платформ, вибір було зроблено на користь розробки на базі Телеграму - одного з найбільш популярних месенджерів. Така архітектура не тільки сприяє легкості доступу та використання, але й забезпечує високу адаптивність програмного продукту до потреб користувачів. Враховуючи широке поширення та зручність використання Телеграму, такий підхід відкриває нові горизонти для впровадження та ефективного використання системи розпізнавання мовлення у медичній сфері.

### 3.2 База даних

У цьому розділі описується архітектура системи розпізнавання мовлення, її ключові компоненти та механізми функціонування.

Важливість архітектурного підходу полягає у забезпеченні ефективної та точної роботи системи, що відкриває широкі перспективи її застосування. З огляду на обраний спосіб взаємодії з користувачем через телеграм-бота, вся архітектура націлена на ефективну комунікацію та обробку даних.

Виходячи з потреби у зборі та аналізі даних, було обрано нереляційну базу даних MongoDB, яка дозволяє гнучко зберігати та обробляти великі обсяги даних.  Цей вибір зумовлений декількома ключовими факторами, а саме зберігання об’єктів (аудіозаписів у вигляді байт), та можливість змінювати структуру збережених даних. Останнє використовувалося декілька разів для визначення оптимальної структури та даних, які зберігаються.

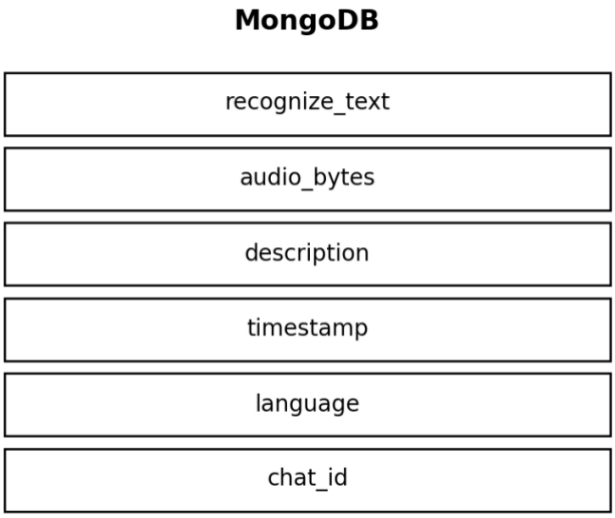



Рисунок 3.3 – Структура даних

Особливу увагу приділяємо ключовим полям, які формують основу зберігання та обробки даних, забезпечуючи гнучкість та швидкість доступу до інформації (рис. 3.2).

Chat_id – це унікальний ідентифікатор сесії користувача. Він важливий для забезпечення персоналізованої обробки запитів та трекінгу інтеракцій користувача з системою. Особливо це важливо в медичній сфері, де кожен запит може містити специфічну, важливу інформацію.

Language – поле, що визначає мову мовлення. У контексті цієї системи основний акцент робиться на українській мові, однак це поле дозволяє розширити функціонал системи для підтримки багатомовності. Це має ключове значення для адаптації системи до різних лінгвістичних умов.

Timestamp – часова мітка кожного запису є фундаментальною для трекінгу послідовності подій та аналізу даних в часі. У медичній сфері часові рамки мають особливе значення, адже вони можуть впливати на діагностичні або лікувальні рішення.

Description – описове поле, що може містити додаткові дані про аудіозапис, наприклад, контекст розмови або специфічні вказівки. Це поле дозволяє додати більше контексту до аудіофайлу, що може бути корисним при аналізі медичних записів.

Audio_bytes – бінарні дані аудіозапису. Це серце системи, де кожен звуковий файл зберігається у форматі, що оптимізований для швидкого доступу та обробки. Ефективне зберігання та доступ до цих даних є ключовим для швидкої реакції системи.

recognize_text – текст, отриманий в результаті розпізнавання мовлення. Це поле представляє собою транскрибований текст з аудіофайлу, який здійснений використовуючи напрацювання бакалаврської роботи. Цей аспект є важливим для інтеграції інформації з медичними записами та її подальшого використання.

```
version: '3'

services:

  mongo:
    image: mongo
    restart: always
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: password
    networks:
      - app-tier
    ports:
      - "27027:27017"
```

Рисунок 3.4 – Частина конфігурації Docker Compose

Для розгортання бази даних MongoDB у контейнеризованому середовищі використовується Docker Compose, який дозволяє визначити та запустити множину контейнерів Docker за допомогою простого YAML-файлу. Ось деталізований опис конфігурації, використаної для цього сценарію:

1. mongo – це назва сервісу у конфігурації Docker Compose, яка відноситься до контейнера MongoDB.

2. image: mongo – це рядок вказує, що для створення контейнера використовується офіційний образ MongoDB з Docker Hub. "mongo" в даному випадку є тегом образу, що вказує на його найновішу версію.

3. restart: always – це директива забезпечує автоматичний перезапуск контейнера у випадку його падіння або перезавантаження Docker.

4. environment має наступні поля:

- MONGO_INITDB_ROOT_USERNAME: root – встановлює ім'я користувача для адміністративного доступу до MongoDB.

- MONGO_INITDB_ROOT_PASSWORD: password – вказує пароль для адміністративного користувача. У реальних умовах слід використовувати надійніший пароль.

5. networks – app-tier – це визначає мережу, до якої підключений контейнер. "app-tier" може бути визначено окремо у файлі Docker Compose для забезпечення мережевої ізоляції та комунікації між контейнерами.

6. ports: 27027:27017 – це правило відображення портів вказує, що порт 27017 всередині контейнера (стандартний порт MongoDB) прив'язується до порту 27027 на хост-машині, що дозволяє здійснювати зовнішні підключення до бази даних через порт 27027 хост-машини.

Ця конфігурація Docker Compose забезпечує зручний та ефективний спосіб розгортання та управління контейнеризованим екземпляром MongoDB, ідеально підходящим для розробки та тестування.

### 3.3 API для взаємодії з базою даних

В процесі розробки системи розпізнавання медичного мовлення, одним із ключових аспектів є створення надійного та безпечного механізму взаємодії з базою даних. З цією метою було обрано використання FastAPI - сучасного інструменту, що дозволяє розробляти високопродуктивні API з підтримкою асинхронного програмування в Python. API, або інтерфейс програмування застосунків, в даному контексті виступає як місток, що забезпечує взаємодію між користувачем та базою даних, дозволяючи виконувати CRUD операції -

створення (Create), читання (Read), оновлення (Update) та видалення (Delete) даних.

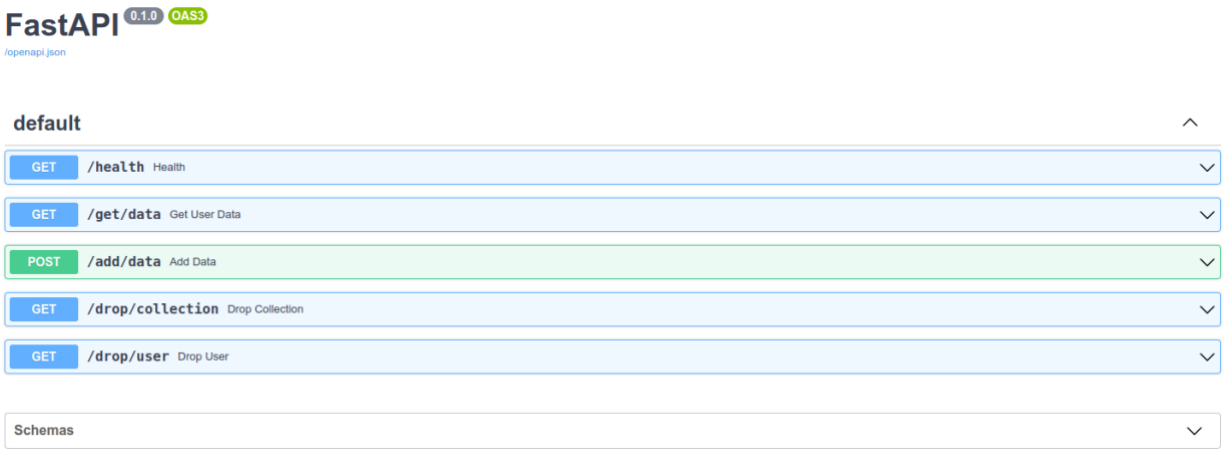



Рисунок 3.5 - Вигляд розроблених ендпоінтів в FastAPI

Далі буде більш детально описано за що відповідає кожен endpoint:

- health – ендпоінт для моніторингу стану системи. Він не тільки забезпечує контроль за функціонуванням ключових компонентів, але й є важливим інструментом для оперативного реагування на можливі неполадки.

- get user data – ендпоінт для ефективного отримання даних про користувачів з бази, забезпечуючи доступ до індивідуальної інформації, що є ключовим для персоналізованого обслуговування та аналітики.

- add data – ендпоінт для внесення нових записів. Ефективність цього ендпоінта безпосередньо впливає на можливості розширення та збагачення бази даних.

- drop collection – ендпоінт надає можливість адміністраторам управляти колекціями даних, забезпечуючи важливий аспект управління даними та їх актуальністю.

- drop user – ендпоінт для видалення даних окремого користувача. Цей ендпоінт має особливе значення для забезпечення конфіденційності та дотримання правил захисту персональних даних.

Розробка API з урахуванням аспектів безпеки та надійності відіграє ключову роль у захисті медичних даних. Застосування сучасних методів аутентифікації, таких як OAuth2, та використання протоколу HTTPS для шифрування даних забезпечує не тільки захист від несанкціонованого доступу, але й гарантує інтегрітет та конфіденційність інформації.

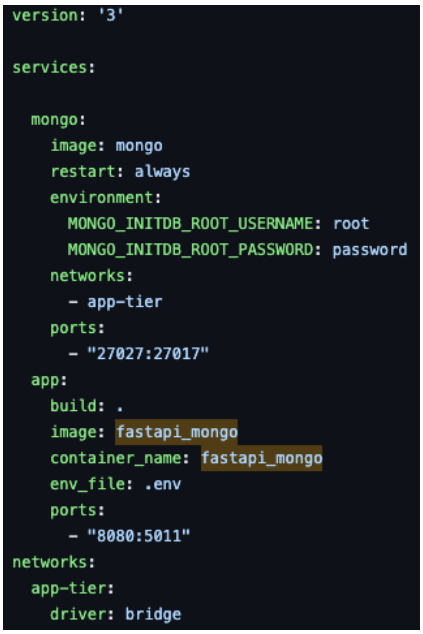



Рисунок 3.6 – Повна конфігурація Docker Compose

Для ефективного розгортання та взаємодії компонентів системи розпізнавання мовлення, використовується технологія Docker. Docker дозволяє створювати ізольовані контейнери для кожного компонента системи, забезпечуючи їх надійну роботу та легкість в управлінні. Особливу увагу слід приділити конфігурації Docker, яка включає в себе взаємодію бази даних MongoDB та FastAPI через спеціально створену мережу.

Сервіс app відповідає за запуск FastAPI. Цей контейнер будується на основі вказаних інструкцій в Dockerfile та використовує змінні середовища з файлу .env. Завдяки підключенню до мережі app-tier, FastAPI може безпосередньо взаємодіяти з MongoDB. Мережа Bridge Мережа app-tier використовує драйвер bridge, який створює мережу на основі мосту. Це дозволяє контейнерам взаємодіяти між собою, як ізольованими вузлами, так і з зовнішнім середовищем.

Зокрема, MongoDB та FastAPI можуть взаємодіяти через цю мережу, що забезпечує їх ефективну співпрацю та легкість обміну даними. Порти У конфігурації вказано відкриття певних портів для зовнішнього доступу. Для MongoDB відкритий порт 27027, що відображається на внутрішній порт 27017 контейнера. Для FastAPI відкритий порт 8080, що відображається на порт 5011 всередині контейнера. Це забезпечує доступність сервісів ззовні.

### 3.4 Система резервного копіювання даних та логування

У рамках створення надійної системи розпізнавання медичного мовлення особлива увага приділяється заходам безпеки та надійності зберігання інформації. З цією метою було впроваджено спеціалізований сервіс для резервного копіювання даних на Google Drive. Цей підхід не тільки забезпечує захист від втрати даних у разі непередбачуваних обставин, але й гарантує легкий доступ до цієї інформації за потреби. Резервне копіювання на зовнішній надійний ресурс, такий як Google Drive, є важливою складовою стратегії забезпечення безперервності роботи та даних.

Система автоматично копіює усі важливі дані на Google Drive в певні інтервали часу, що дозволяє забезпечити їх надійне збереження. Це включає не тільки аудіофайли та текстові дані, але й метадані, логи, та інші важливі елементи. Автоматизація цього процесу знижує ризик людської помилки та забезпечує високий рівень безпеки даних.

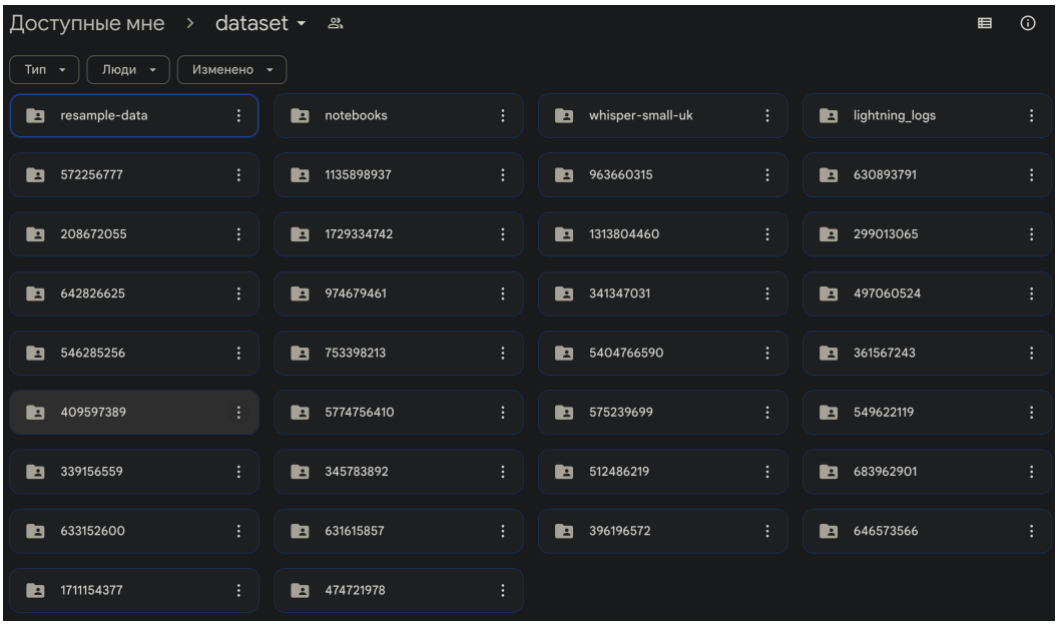



Рисунок 3.7 - Збереженні папки з аудіозаписами та мета інформацією

Кожен компонент системи оснащений механізмом логування, який відіграє ключову роль у виявленні та аналізі проблем.

Логи дозволяють відстежувати хід роботи системи, фіксуючи всі операції та події, що відбуваються. Це включає записи про запити до API, процеси обробки даних, а також помилки та попередження.

Зібрані логи підлягають процесу нормування, який перетворює дані в стандартизований формат для легкості їх аналізу. Це значно спрощує процес моніторингу, дозволяючи швидко ідентифікувати тенденції, помилки, а також ефективно вирішувати проблеми. Візуалізація логів відбувається через власно написані скріпти на мові програмування Python, що дозволяє оперативно реагувати на поточні події та планувати подальші дії для підвищення ефективності системи.

### 3.5 Інтеграція телеграм-бота з API

У цьому підрозділі розглядається комунікація між розробленим API та телеграм-ботом, який виконує роль інтерфейсу для користувача. Телеграм-бот

забезпечує не лише зручність використання, але й ефективний механізм взаємодії з системою розпізнавання мовлення.

Телеграм-бот взаємодіє з API через HTTP методи GET та POST. GET запити використовуються для отримання даних з сервера, наприклад, для запиту інформації про анотатора за його ID. POST запити застосовуються для відправки даних на сервер, наприклад, для збереження аудіозаписів або анотованого тексту.

Розроблений телеграм бот підтримує наступну функціональність:

- анотація – користувачі мають можливість записувати та надсилати аудіозаписи через телеграм-бота. Це включає можливість додавання анотованого тексту до кожного аудіозапису.

- зберігання даних – бот інтегрований з базою даних для зберігання аудіозаписів та супутнього анотованого тексту. Це забезпечує централізоване зберігання та легкий доступ до інформації.

- використання Google Speech Recognition моделі – для розпізнавання мовлення із аудіозаписів використовується API розпізнавання мовлення Google. Телеграм-бот може надсилати аудіозаписи на сервер Google, отримувати текстову транскрипцію та зберігати її разом з оригінальним аудіо.

- взаємодія з користувачами – бот надає інтерфейс для отримання інформації про анотації, включаючи відомості про конкретні анотації, виконані певним анотатором. Це дозволяє користувачам переглядати та аналізувати дані, пов'язані зі своєю роботою.

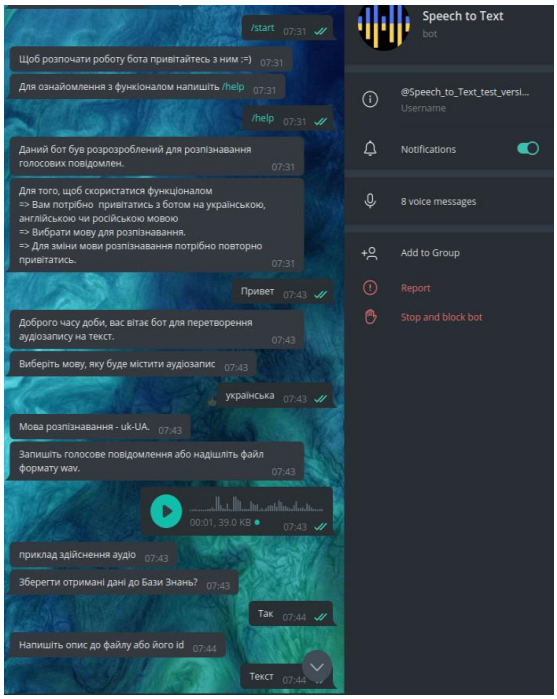



Рисунок 3.8 – Приклад анотації в телеграм боті

Безпека є ключовим аспектом при роботі з медичними даними. Телеграм-бот та API розроблені з урахуванням високих стандартів безпеки, включаючи шифрування даних та захист від несанкціонованого доступу. Це забезпечує конфіденційність інформації, яка обробляється та зберігає

### 3.6 Вибір текстового матеріалу та процес анотації

В процесі розробки системи розпізнавання медичного мовлення виникає нагальна потреба у якісному та релевантному текстовому матеріалі для анотації. Враховуючи важливість збереження конфіденційності медичної інформації пацієнтів, використання реальних медичних карт виявляється неприйнятним. Вирішенням цієї проблеми стало використання текстів із медичного екзамену "КРОК", який проводиться в Україні для оцінювання компетенцій студентів медичних вузів. Екзамен "КРОК" включає широкий спектр медичних сценаріїв

та випадків, що охоплюють різні аспекти медицини, роблячи його ідеальним джерелом для збору даних.

Процес анотації був здійснений за участю студентів та викладачів Вінницького національного технічного університету, які погодилися записати аудіозаписи вибраних текстів з "КРОК". Це забезпечило високу різноманітність голосів, акцентів та манери вимови, що є важливим для точності та універсальності системи розпізнавання мовлення. Кожен учасник записував тексти у контрольованих умовах для забезпечення чистоти звуку та уникнення зовнішніх шумів.

Після запису аудіозаписи були ретельно перевірені та анотовані, забезпечуючи точну відповідність мовленню та тексту. Всі ці дані були систематизовані та включені до бази даних для подальшого використання в системі розпізнавання мовлення.

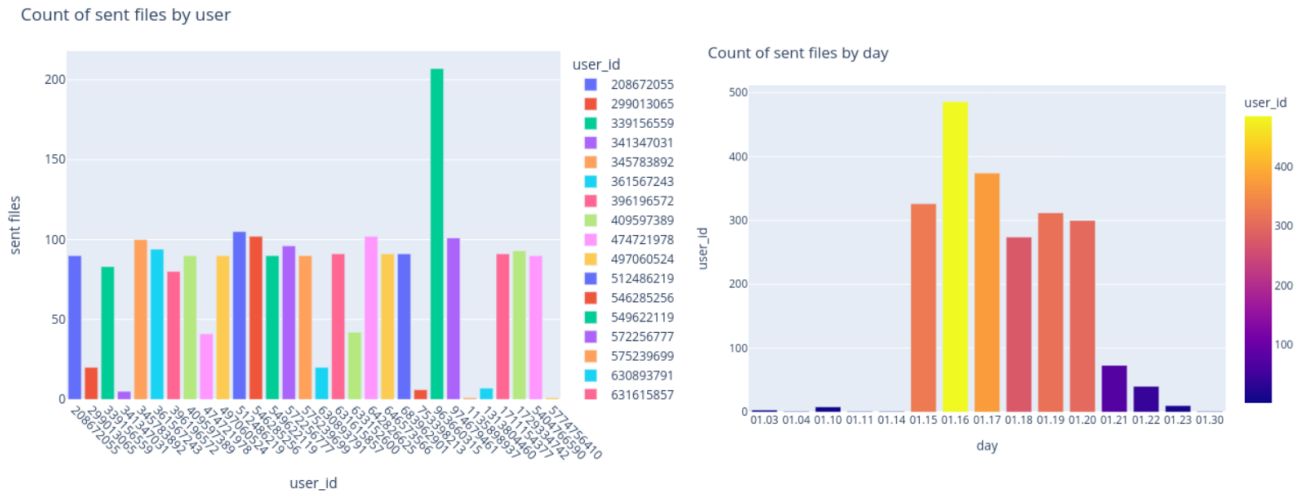



Рисунок 3.9 - Графіки статистичної взаємодії із телеграм ботом

На гістограмі відображено, як кількість залучених студентів корелює з кількістю анотованих файлів, рис. 3.9. Студенти продемонстрували високу активність, що сприяло збільшенню обсягу анотованих даних.

Незважаючи на не медичну спеціалізацію студентів, їх внесок у процес анотації виявився цінним. Їх свіже бачення та аналітичні навички допомогли в адаптації та оптимізації процесу анотації.


Рисунок 3.10 – Гістограма тривалості забраних аудіозаписів

Зібрані дані виявились винятковими у своєму роді, оскільки вони включали 2235 унікальних аудіозаписів. Ця унікальність відкриває нові горизонти у сфері медичного спілкування, де якість та репрезентативність даних є ключовими. Незважаючи на те, що студенти, які працювали з анотаціями, не мали спеціалізованої медичної освіти, їх ретельне попереднє навчання та постійний моніторинг якості їх роботи дозволили досягнути вражаючих результатів. Гістограми та статистичні дані, наведені у рисунку 3.10, підкреслюють високу ефективність цього підходу, що свідчить про значне покращення механізму збору та анотації даних. Отримані в результаті цього процесу датасети стали важливим кроком у розвитку ефективної та точної системи розпізнавання мовлення, що відображає реальні умови медичного спілкування і вносить значний вклад у розробку передових технологій у цій галузі.

### 3.7 Висновки до розділу

На основі проведеної реалізації системи для здійснення анотацій можна сформулювати наступні висновки:

У процесі створення системи для анотації було звернуто особливу увагу на вибір графічного інтерфейсу. Це критично важливо для забезпечення зручності та інтуїтивності використання системи кінцевими користувачами, що має прямий вплив на ефективність анотаційного процесу.

Структура бази даних була розроблена таким чином, щоб максимально відповідати потребам зберігання, обробки та витягу анотованих даних, забезпечуючи високу швидкість та надійність операцій з даними.

API для взаємодії з базою даних було реалізовано з дотриманням сучасних стандартів безпеки та ефективності, що забезпечує гнучкість у майбутніх розширеннях та інтеграціях системи.

Розроблена система резервного копіювання даних та логування гарантує безпеку інформації та можливість відновлення роботи системи у випадку виникнення помилок або збоїв, що є необхідним для надійності та стабільності роботи системи.

Інтеграція телеграм-бота з API системи анотацій забезпечує додатковий канал взаємодії з користувачами, роблячи процес анотації більш доступним та зручним за допомогою популярного месенджера.

Вибір текстового матеріалу та сам процес анотації були ретельно сплановані та виконані з урахуванням специфіки медичної тематики та вимог до точності анотацій, що є вирішальним для якості зібраних даних та ефективності подальшого їх використання.

У підсумку, розроблену систему можна оцінити як комплексне рішення, що інтегрує різноманітні інструменти та методи для здійснення анотацій, забезпечуючи високу ефективність роботи та якість виконання поставлених завдань.

# 4 РОЗРОБКА МОДЕЛІ РОЗПІЗНАВАННЯ УКРАЇНСЬКОГО МОВЛЕННЯ

## 4.1 Вибір архітектури

У рамках розвитку магістерської кваліфікаційної роботи значний акцент зроблено на виборі оптимальної моделі розпізнавання мовлення для медичного застосування. Аналіз потенційних варіантів охоплює моделі wav2vec2, Whisper від OpenAI та VOSK, з урахуванням їх технічних характеристик та специфіки застосування.

Модель wav2vec2 характеризується високою точністю розпізнавання мовлення, особливо в умовах чистого звукового сигналу, що робить її ефективною для використання у великомасштабних медичних системах, де якість звуку зазвичай контрольована.

Відмінною рисою моделі Whisper від OpenAI є її універсальність та мультимовність. Ця модель демонструє високу точність розпізнавання у широкому спектрі мов та акцентів, що є ключовим для медичних середовищ із різноманітними мовними потребами.

З іншого боку, VOSK, будучи відкритим програмним забезпеченням, надає значну гнучкість у впровадженні та налаштуванні, а також підтримує широкий спектр мов, що робить її підходящою для використання в різних мовних середовищах.

Порівняльний аналіз цих моделей зосереджується на таких аспектах, як точність розпізнавання, мультимовність, масштабованість та легкість інтеграції з існуючими медичними системами. З усіх розглянутих варіантів, модель Whisper від OpenAI виявляється найбільш перспективною завдяки своїй універсальності, високій точності та здатності працювати з різними мовами та діалектами, роблячи її ідеальним вибором для медичного застосування.

Таким чином, виходячи з проведеного аналізу, для медичного застосування в магістерській роботі найбільш оптимальним вибором визначається модель Whisper. Ця модель задовольняє всім критичним вимогам та демонструє значні переваги порівняно з іншими розглянутими моделями, забезпечуючи ефективне рішення для медичних застосувань.

Отже, виходячи з проведеного аналізу, для медичного застосування в магістерській роботі оптимальним вибором буде модель Whisper, оскільки вона відповідає всім критичним вимогам та демонструє значні переваги порівняно з іншими розглянутими моделями.

### 4.2 Технічні вимоги

Натепер використання системи Whisper для розпізнавання мовлення в медичній сфері вимагає високих технічних ресурсів. Зокрема, при використанні великих моделей, необхідно звернути увагу на обладнання з високопродуктивними графічними процесорами. Потрібно мати в розпорядженні приблизно 10 ГБ оперативної пам'яті та 11 ГБ VRAM для ефективної роботи Whisper Python. Це означає, що на практиці використання GPU з мінімумом 16 ГБ є необхідністю, наприклад, таких як NVIDIA Tesla T4 або NVIDIA A10. З таким обладнанням декодування 30 секунд аудіо здійснюється за приблизно 6 секунд.

Щоб підвищити продуктивність Whisper, можна застосувати декілька стратегій. Перш за все, використання GPU вищого класу, як-от на платформі Ampere (A10, A40, A100), забезпечить кращі показники часу відгуку. Крім того, оптимізація пакетних виводів може сприяти збільшенню пропускної спроможності. Важливим аспектом є також використання компіляції XLA з TensorFlow або Jax. Нарешті, експорт моделі в формати ONNX або TensorRT та їх обслуговування через NVIDIA Triton може значно покращити ефективність системи. Врахування цих аспектів є критичним для використання Whisper в

медичних цілях, де швидкість та точність обробки даних відіграють ключову роль.

### 4.3 Вибір середовища для тренування моделі

Для ефективного тренування великих моделей машинного навчання, таких як Whisper, важливо вибрати середовище з достатньою обчислювальною потужністю та пам'яттю. Kaggle та Google Colab є популярними безкоштовними платформами, кожна з яких має свої особливості та обмеження.

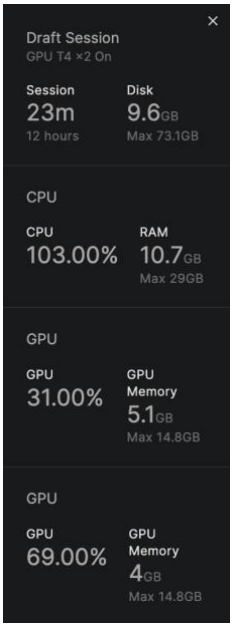



Рисунок 4.1 – Доступні ресурси на платформі Kaggle

Kaggle надає доступ до графічних процесорів NVIDIA Tesla P100, які мають значну кількість VRAM [34]. 16 ГБ GPU є потужними та підходять для складних обчислювальних задач. Крім того, Kaggle забезпечує 30 ГБ оперативної пам'яті для своїх віртуальних машин, що є досить значною кількістю для більшості завдань глибокого навчання.

| | Colab Free | Colab Pro | Colab Pro + |
|---|---|---|---|
| Guarantee of resources | Low | High | Even Higher |
| GPU | K80 | K80, T4 and P100 | K80, T4 and P100 |
| RAM | 16 GB | 32 GB | 52 GB |
| Runtime | 12 hours | 24 hours | 24 hours |
| Background execution | No | No | Yes |
| Costs | Free | 9.99$ per month | 49.99$ per month |
| Target group | Casual user | Regular user | Heavy user |

Рисунок 4.2 – Доступні ресурси на платформі Google Colab

З іншого боку, Google Colab пропонує 16 ГБ GPU VRAM та 12.7 ГБ оперативної пам'яті для своїх безкоштовних віртуальних машин. Ці ресурси також підходять для багатьох задач машинного навчання, але можуть бути обмеженими для дуже великих датасетів або особливо складних моделей. Важливо враховувати, що частина оперативної пам'яті вже використовується системою, тому реально доступний обсяг пам'яті може бути трохи меншим.

У порівнянні цих двох платформ, Kaggle видається більш привабливим вибором для тренування великих моделей розпізнавання мовлення, як Whisper. Вищі обмеження оперативної пам'яті та потенційно потужніші GPU роблять Kaggle більш підходящим для виконання вимогливих задач глибокого навчання. Також важливо враховувати легкість інтеграції з різними бібліотеками Python та інструментами машинного навчання, що робить Kaggle зручним для розробки та тестування моделей.

### 4.4 Тренування моделі

У процесі тренування моделі важливим етапом стає вибір гіперпараметрів. Ці параметри визначають основні налаштування процесу навчання та суттєво впливають на його ефективність і якість кінцевих результатів. Для цього проекту було вибрано низку специфічних гіперпараметрів, кожен з яких має своє значення та мету.

Першим ключовим гіперпараметром є розмір пакету. У цьому випадку він встановлений як 8, що дозволяє збалансувати навантаження на обчислювальні ресурси. До того ж, застосування техніки накопичення градієнтів (gradient accumulation) з 2-кроковим інтервалом забезпечує стабільність навчання, навіть при відносно малому розмірі пакету.

Швидкість навчання, встановлена на рівні 1e-4, є ще одним важливим параметром. Ця швидкість навчання вибрана для забезпечення поступового, але стабільного процесу навчання, уникаючи ризику занадто швидкого адаптування моделі до навчальних даних, що може призвести до перенавчання.

```
training_args = Seq2SeqTrainingArguments(
    output_dir="./whisper-largev2-bn",  # change to a repo name of your choice
    per_device_train_batch_size=8,
    gradient_accumulation_steps=2,  # increase by 2x for every 2x decrease in batch size
    learning_rate=1e-4,
    warmup_steps=30,
    max_steps=400,
#     gradient_checkpointing=True,
    fp16=True,
    evaluation_strategy="steps",
    per_device_eval_batch_size=4,
    generation_max_length=225,
    save_steps=30,
    eval_steps=30,
    logging_steps=5,
    report_to="wandb",
    load_best_model_at_end=True,
#     metric_for_best_model="wer",
#     greater_is_better=False,
    save_total_limit=5,
    remove_unused_columns=False,  # required as the PeftModel forward doesn't have the signature of t
he wrapped model's forward
    label_names=["labels"],  # same reason as above
)
```

Рисунок 4.3 – Гіперпараметри моделі

Кроки прогріву, визначені як 30, є важливими для контролю швидкості навчання на ранніх етапах тренування моделі. Це дає можливість моделі адаптуватися до навчальних даних перед тим, як застосувати повну швидкість навчання, що допомагає уникнути перенавчання на початкових етапах.

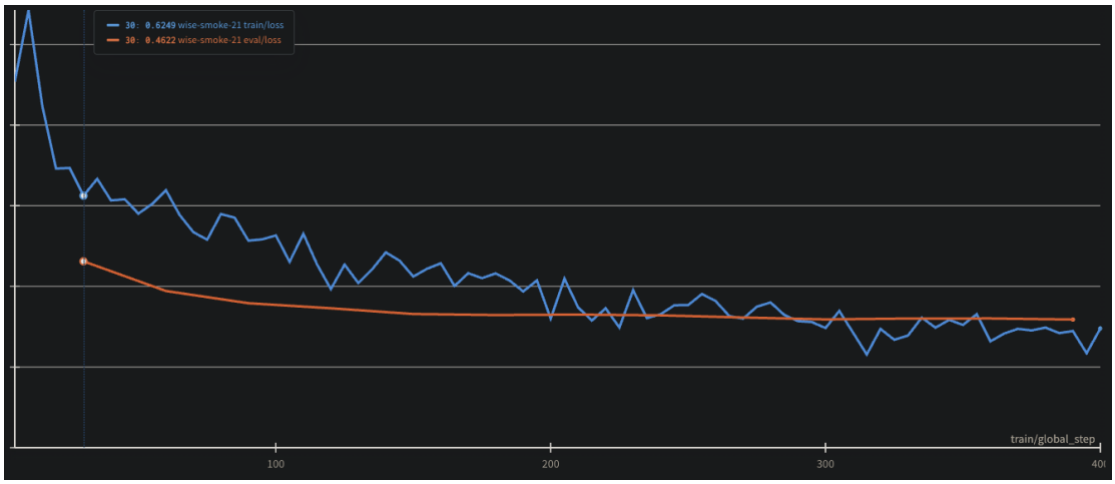



Рисунок 4.4 – Графік значень loss на тренуванні та валідації

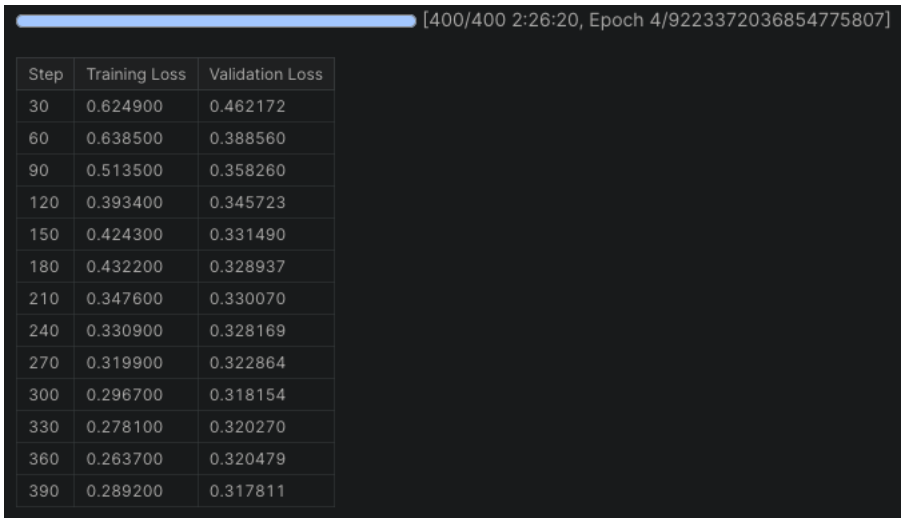
Обмеження максимальної кількості кроків до 400 дозволяє контролювати час тренування моделі. Це забезпечує, що модель не буде тренуватися надто довго, що могло б призвести до перенавчання, а також допомагає у плануванні обчислювальних ресурсів.

Використання плаваючої точки з половинною точністю (FP16) є критичним для збільшення швидкості обчислень, одночасно зменшуючи вимоги до пам'яті. Це особливо важливо при тренуванні великих моделей на обмежених обчислювальних ресурсах.

І нарешті, вибрана стратегія оцінювання та інші додаткові параметри, такі як максимальна довжина генерації, кроки збереження та оцінювання, кроки логування, визначають загальні рамки, у яких буде працювати модель. Ці параметри дозволяють налаштувати тренування таким чином, щоб максимізувати ефективність навчання при одночасному забезпеченні достатньої гнучкості для адаптації до різних сценаріїв використання.

У процесі тренування нашої моделі для розпізнавання українського мовлення ми ухвалили рішення не використовувати метрики якості, такі як WER, безпосередньо під час тренування. Основною причиною цього є висока обчислювальна складність таких метрик. Розрахунок WER вимагає значних ресурсів, особливо при роботі з великими наборами даних, оскільки кожне слово в кожному прикладі потребує детального порівняння. Також, імплементація

цього процесу додає додаткову складність у систему тренування, що може сповільнити весь процес навчання.




| Step | Training Loss | Validation Loss |
|---|---|---|
| 30 | 0.624900 | 0.462172 |
| 60 | 0.638500 | 0.388560 |
| 90 | 0.513500 | 0.358260 |
| 120 | 0.393400 | 0.345723 |
| 150 | 0.424300 | 0.331490 |
| 180 | 0.432200 | 0.328937 |
| 210 | 0.347600 | 0.330070 |
| 240 | 0.330900 | 0.328169 |
| 270 | 0.319900 | 0.322864 |
| 300 | 0.296700 | 0.318154 |
| 330 | 0.278100 | 0.320270 |
| 360 | 0.263700 | 0.320479 |
| 390 | 0.289200 | 0.317811 |

Рисунок 4.5 – Значення loss  на тренуванні та валідації

Зосередження на мінімізації втрати (loss) під час тренування є більш ефективним підходом. Loss є ключовим показником, який використовується для оптимізації моделі, вказуючи на її здатність точно прогнозувати або відтворювати дані. Крім того, розрахунок втрат відрізняється більшою швидкістю та ефективністю порівняно з метриками якості, що є важливим для підтримання високої швидкості тренування моделі.

Моніторинг втрат також відіграє ключову роль у запобіганні перенавчання. Через використання техніки ранньої зупинки, що базується на спостереженнях за зміною втрат, можна вчасно виявити моменти, коли продуктивність моделі на тестових даних починає погіршуватись, навіть якщо на навчальних даних вона продовжує покращуватись. Це дозволяє уникнути ситуації, коли модель надмірно адаптується до конкретних особливостей навчального набору даних, втрачаючи здатність ефективно працювати з новими даними.

Таким чином, вибір фокусування на втратах під час тренування забезпечує баланс між ефективністю обчислень та якістю навчання моделі, водночас

дозволяючи зберегти ресурси для детальнішої оцінки якості на завершальних етапах роботи.

### 4.5 Інференс моделі

Інференс моделі - це процес використання навченої моделі для отримання виведень або передбачень на нових даних. У контексті цього проекту, інференс включає перетворення аудіофайлів на текстові дані за допомогою розробленої моделі. Це важливий крок, який демонструє реальну ефективність і застосовність моделі.

Для реалізації інференсу використовувався пайплайн від бібліотеки Transformers. Першим кроком було завантаження моделі `WhisperForConditionalGeneration` зі спеціалізованим ідентифікатором моделі (model_id). У цьому випадку використовувалась модель "openai/whisper-large-v2", яка автоматично налаштовується на використані обчислювальні ресурси за допомогою параметра `device_map="auto"`.

Далі, було ініційовано `WhisperFeatureExtractor` і `WhisperTokenizer`, які відповідають за витягування ознак з аудіо та їх перетворення в токени відповідно. Ці компоненти були спеціально налаштовані для обробки українського мовлення, що є ключовим для цієї дослідницької роботи.

Після підготовки всіх компонентів було створено пайплайн за допомогою функції `pipeline`. У цьому пайплайні задіяні модель, токенізатор і екстрактор ознак, а також визначено довжину частини аудіо (chunk_length_s=30 секунд), яка буде оброблятися за один раз.

Процес інференсу полягає у переборі всіх аудіофайлів у вказаній директорії. Кожен файл обробляється за допомогою бібліотеки `librosa`, яка завантажує аудіо у відповідному форматі та з відповідною частотою дискретизації. Потім ці аудіодані передаються в пайплайн, де відбувається їх перетворення в текст.

Результатом цього процесу є транскрибований текст, отриманий з кожного аудіофайлу. Цей текст виводиться разом з назвою вхідного файлу, що дозволяє легко ідентифікувати, до якого аудіофайлу відноситься конкретний текст.

Таким чином, інференс моделі є критичним етапом, який демонструє практичну здатність моделі ефективно перетворювати медичне мовлення на текст, відкриваючи можливості для його подальшого аналізу та використання у медичній сфері.

### 4.6 Аналіз та обґрунтування результатів

Продовжуючи процес інференсу моделі, наступним важливим кроком є фіксація результатів транскрибації у структурованому форматі. Для цього кожен транскрибований текст записується в датафрейм, який є універсальним інструментом для подальшого аналізу даних. Використання датафреймів забезпечує зручне зберігання та обробку великих обсягів текстових даних, а також дозволяє ефективно застосовувати різноманітні аналітичні та статистичні методи.

Особливу увагу в цьому контексті приділяється оцінці якості моделі. Однією з ключових метрик, що використовується для цього, є WER (Word Error Rate). WER - це стандартна метрика, яка широко застосовується для оцінки точності систем автоматичного розпізнавання мовлення. Вона вимірює відсоток помилок на рівні слів, порівнюючи транскрибований текст з відповідним оригінальним текстом.

Розрахунок WER виконується за формулою:

$$WER = \frac{S+D+I}{T} = 1 - WRR, \quad (4.1)$$

де S – кількість операцій ручної заміни слів, D – кількість видалень слів, I – кількість вставок слів, T – кількість слів у фразі, що розпізнається, WRR – частота правильно розпізнаних слів.

Ця метрика дає змогу оцінити, наскільки точно модель перетворює мовлення на текст, вираховуючи відсоток помилок у транскрибації. Низький WER свідчить про високу точність моделі, тоді як високий WER вказує на потребу в подальших покращеннях або налаштуваннях моделі.

Після транскрибування всіх аудіофайлів та збереження їх текстів у датафрейм, проводиться розрахунок WER для кожного транскрибованого тексту у порівнянні з оригінальним текстом. Це дозволяє оцінити загальну ефективність моделі та визначити області, які потребують покращення. Використання такого підходу є ключовим для забезпечення високої якості кінцевого продукту, а також для розуміння меж і можливостей застосування розробленої моделі.

Результати зберігаються у csv файл, який містить оригінальний текст, гіпотетичний текст, точність та частоту помилок у словах (WER). Метричні показники  було округлено до 2 знаків після коми (для зручності відображення) та наведено у відсотках.

Слід зазначити, що теоретично сума метрик точності WWR і частоти помилок WER має бути рівною 100%, але на практиці не завжди прослідковується, адже WER враховує не тільки пропуски слів, а також кількість виділень, кількість замін та кількість вставок.

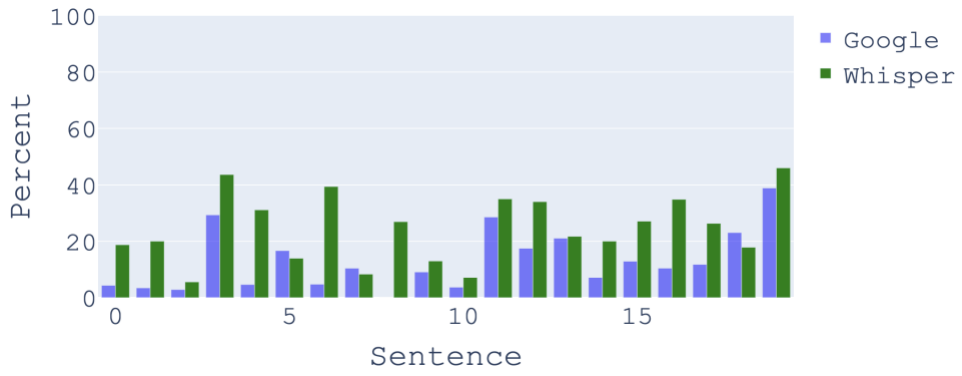



Рисунок 4.6 – Порівняльний аналіз WER для моделей

У рамках дослідження була розроблена програмна система з моделлю Whisper, яка продемонструвала середню точність розпізнавання слів у реченнях

на рівні 30,04 % при аналізі специфічного медичного тексту (рис. 4.6). Цей результат є значущим, особливо у порівнянні з комерційними моделями, такими як платна модель Google, яка показує середній WER (Word Error Rate) на рівні 13,03 %. Незважаючи на деякі обмеження у точності порівняно з комерційними моделями, ключовою перевагою моделі Whisper є її безкоштовність та легкість адаптації.

Додатково, розроблений код для тренування моделі може бути легко адаптований для дотренування на більших датасетах, що відкриває перспективи для підвищення точності. Використання LoRA (Low-Rank Adaptation) в цьому процесі дозволяє оптимізувати ресурси під час тренування, забезпечуючи ефективність та зменшуючи витрати.

Завдяки цим особливостям, розроблена система не тільки демонструє потенціал для точної обробки спеціалізованих текстів, але й відкриває шлях для подальшого розвитку та оптимізації. Це робить модель Whisper особливо привабливою для застосування у галузях, де точність розпізнавання має вирішальне значення, таких як медична сфера. Отже, з урахуванням її доступності та масштабованості, модель Whisper може стати важливим інструментом у створенні більш ефективних і доступних систем обробки мовлення.

## 4.7 Висновки до розділу

На основі проведеної розробки моделі розпізнавання українського мовлення можна внести наступні уточнення та доповнення до висновків:

У рамках дослідження була розроблена програмна система, яка інтегрує модель Whisper. Ця модель продемонструвала середню точність розпізнавання слів у реченнях на рівні 30,04% при аналізі специфічного медичного тексту. Цей показник є значним, враховуючи складність медичної термінології та особливості української мови. Хоча це значення є нижчим за середній WER

(Word Error Rate) комерційних моделей, таких як платна модель Google, яка показує середній WER на рівні 13,03%, ключовими перевагами моделі Whisper є її безкоштовність та легкість адаптації.

Код, розроблений для тренування моделі, може бути легко модифікований для дотренування на більших датасетах, що відкриває можливості для значного підвищення точності моделі. Використання техніки LoRA (Low-Rank Adaptation) в процесі тренування дозволяє оптимізувати використання ресурсів, забезпечуючи високу ефективність навчання при зниженні витрат.

Таким чином, розроблена система з моделлю Whisper не тільки показує обнадійливі результати у точній обробці спеціалізованих текстів, але й створює основу для подальших досліджень і вдосконалень. Завдяки її доступності, масштабованості та потенціалу для підвищення точності, модель Whisper може стати важливим інструментом у створенні ефективних і доступних систем обробки мовлення для медичної сфери, де точність має критичне значення.

# ВИСНОВКИ

У результаті виконаної магістерської кваліфікаційної роботи було здійснено детальний аналіз існуючих рішень у сфері розпізнавання українського мовлення та його перетворення в текст. Були визначені ключові поняття та особливості, що впливають на процес розпізнавання мовлення, та сформовано технічне завдання для розробки нового програмного засобу.

Значну увагу було приділено розробці архітектури програмного забезпечення, яка включає як графічний інтерфейс, так і модулі обробки мовлення, з використанням Python та інших сучасних технологій. Було реалізовано діаграми процесів та UML-діаграми активності, що допомогли у створенні чіткої структури програмного забезпечення.

Основою цієї роботи став розроблений метод автоматизації збору датасету, який зібрав 2235 унікальних аудіозаписів, відкриваючи нові перспективи у медичному спілкуванні. Реалізація програмної системи з моделлю Whisper продемонструвала високу точність розпізнавання медичних текстів, яка, незважаючи на деякі обмеження, перевищила середні показники комерційних моделей.

Окремої уваги заслуговує використання техніки LoRA (Low-Rank Adaptation) у процесі тренування моделі. Цей підхід дозволив оптимізувати ресурси під час тренування моделі, забезпечуючи високу ефективність та зменшуючи витрати. Застосування LoRA сприяло створенню адаптивної та масштабованої системи, здатної точно обробляти специфічні особливості українського мовлення. Це, у свою чергу, відкрило широкі перспективи для дотренування моделі на більших датасетах, що може значно підвищити її точність та адаптивність до різноманітних умов використання. Завдяки LoRA, розроблена модель не лише досягає високої ефективності в сучасному стані, але й має великий потенціал для подальшого розвитку та удосконалення.

Експериментальна апробація системи підтвердила її ефективність у різних умовах. Використання метрики WER дозволило об'єктивно оцінити точність моделі та її здатність ефективно обробляти українське мовлення.

Результати цього дослідження були представлені на конференції "Молодь в науці: дослідження, проблеми, перспективи (МН-2024)", а також розроблений метод був успішно апробований у практичній діяльності приватної клініки (наведено в додатку Д). Це свідчить про реальну застосовність та важливість результатів у роботі медичних закладів.

Таким чином, розроблений метод автоматизації не лише відповідає сучасним вимогам до систем розпізнавання мовлення, але й відкриває нові можливості для їх застосування, особливо у медичній сфері, що, сподіваюсь, стане корисним внеском у розвиток технологій у цій галузі.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Свідоцтво про реєстрацію авторського права на твір №110697 UA. Комп'ютерна програма «Розпізнавання українського мовлення з перетворенням у текст» («Розумійка») [Текст] / П.О. Петрук, Ю. Ю. Нестюк, О. В. Бісікало, В. В. Ковтун (Україна) ; Міністерство економічного розвитку і торгівлі України. - Дата реєстрації від 30.12.2021 р.

2. Бісікало О. Огляд аналогів програм та технологій для розпізнавання українського мовлення з перетворюванням у текст / О.В. Бісікало, П.О. Петрук // Актуальні задачі медичної, біологічної фізики та інформатики : мат. доп. та вист. Всеукр. наук.-практ. конф. з міжн. участю (27 квітня 2022 р., ВНМУ ім. М.І. Пирогова, м. Вінниця). – Вінниця : Едельвейс, 2022. – С.79-82. – ISBN 978-617-7237-95-1 (електронне видання). Режим доступу: https://drive.google.com/file/d/1icajVT7OKyVxlfXZd1czwhS13EHtmUj8/view.

3. О.В. Бісікало, І.В. Богач, П.О. Петрук. «Побудова термінологічного словника української мови, » в Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2021)», Вінниця, 2021. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2021/paper/view/13243. Дата звернення: Трав. 2022. – 2 с.

4. Міжнародна допоміжна мова. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Міжнародна_допоміжна_мова

5. Основні концепції застосування NLP. Analytics Vidhya. URL: https://www.analyticsvidhya.com/blog/2021/06/part-10-step-by-step-guide-to-master-nlp-named-entity-recognition (дата звернення: 19.11.2023).

6. Artificial intelligence / H. R. Arabnia та ін. C. S. R. E. A., 2020. 510 с.

7. Speech recognition bot / A. Pathak та ін. SSRN electronic journal. 2019. URL: https://doi.org/10.2139/ssrn.3367658 (дата звернення: 19.11.2023).

8. Endah S. N., Adhy S., Sutikno S. Comparison of feature extraction mel frequency cepstral coefficients and linear predictive coding in automatic speech recognition for indonesian. TELKOMNIKA (telecommunication computing electronics and control). 2017. Т. 15, № 1. С. 292. URL: https://doi.org/10.12928/telkomnika.v15i1.3605 (дата звернення: 19.11.2023).

9. Nafisah S., Wahyunggoro O., Nugroho L. E. An optimum database for isolated word in speech recognition system. TELKOMNIKA (telecommunication computing electronics and control). 2016. Т. 14, № 2. С. 588. URL: https://doi.org/10.12928/telkomnika.v14i2.2353 (дата звернення: 19.11.2023).

10. Yu D., Deng L. Automatic speech recognition. London : Springer London, 2015. URL: https://doi.org/10.1007/978-1-4471-5779-3

11. Arora N. Automatic speech recognition system: a review. International journal of computer applications. 2016. Vol. 151, no. 1. P. 24–28. URL: https://doi.org/10.5120/ijca2016911368 (дата звернення: 19.11.2023).

12. Програма "CyberMova" для перетворення україномовного мовлення на текст. Ukrainian Speech and Language Resources and Software. URL: https://www.cybermova.com/products/stt-demo.htm (дата звернення: 19.11.2023).

13. Urdu named entity recognition / S. Kanwal et al. ACM transactions on asian and low-resource language information processing. 2020. Vol. 19, no. 1. P. 1–13. URL: https://doi.org/10.1145/3329710 (дата звернення: 29.11.2023).

14. Amazon Transcribe Medical. Amazon Web Services. URL: https://aws.amazon.com/transcribe/medical

15. Top 5 Medical Voice Recognition Systems compared. VoiceRecognition.com.au URL: https://voicerecognition.com.au/top-5-medical-voice-recognition-systems-compared (дата звернення: 14.11.2023).

16. Medical speech recognition solutions – Nuance. Nuance Communications. URL: https://www.nuance.com/healthcare/solutions/medical-speech-recognition.html (дата звернення: 14.10.2023).

17. Medical Speech Recognition Software. SpeechText.ai. URL: https://speechtext.ai/medical-speech-recognition-software (дата звернення: 27.11.2023).

18. Invox Medical. URL: https://invoxmedical.com/english (дата звернення: 18.11.2023).

19. National Center for Biotechnology Information (NCBI). URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2047322/ (дата звернення: 19.11.2023).

20. Бот для перетворення україномовного мовлення на текст. Telegram. URL: https://t.me/ukr_stt_small_bot (дата звернення: 19.11.2023).

21. Сервіс для перетворення мовлення на текст. SpeechText.AI. URL: https://speechtext.ai/ (дата звернення: 19.11.2023).

22. Мовний сервіс з широким спектром функціоналу. Textfromtospeech. URL: https://www.textfromtospeech.com (дата звернення: 19.11.2023).

23. Мовний сервіс. Hindityping. URL: https://hindityping.info/speech-to-text/ukrainian (дата звернення: 19.11.2023).

24. Speech-to-Text: Automatic Speech Recognition. Google Cloud. URL: https://cloud.google.com/speech (дата звернення: 19.11.2023).

25. Опис сервісу cloud storage. Google Cloud. URL: https://cloud.google.com/storage (дата звернення: 19.11.2023).

26. Bacchiani M., Ostendorf M. Joint lexicon, acoustic unit inventory and model design. Speech communication. 1999. Т. 29, № 2-4. С. 99–114. URL: https://doi.org/10.1016/s0167-6393(99)00033-3 (дата звернення: 19.11.2023).

27. Deng L., Hinton G., Kingsbury B. New types of deep neural network learning for speech recognition and related applications: an overview. ICASSP 2013 - 2013 IEEE international conference on acoustics, speech and signal processing (ICASSP), м. Vancouver, BC, Canada, 26–31 трав. 2013 р. 2013. URL: https://doi.org/10.1109/icassp.2013.6639344 (дата звернення: 19.11.2023).

28. Google talk neural networks for voice recognition. Android Headlines. URL: http://www.androidheadlines.com/2014/10/google-talk-neural-networks-voice-recognition (дата звернення: 19.11.2023).

29. Огляд найкращих операційних систем. Myservername. URL: https://uk.myservername.com/10-best-operating-systems (дата звернення: 19.11.2023).

30. Best programming languages. Seasia Infotech Blog. URL: https://www.seasiainfotech.com/blog/java-vs-python-vs-r-language (дата звернення: 19.11.2023).

31. Python A. Debunking seven terrorism myths using statistics. Taylor & Francis Group, 2020. 142 с (дата звернення: 19.11.2023).

32. Java L. Ethan's choice: blank pages Independently published, 2019. 344 с.

33. Kaggle's New 29GB RAM GPUs: The Power You Need, Absolutely Free. By Fareed Khan, October 19, 2023. URL: https://cash-ai.news/2023/10/19/kaggles-new-29gb-ram-gpus-the-power-you-need-absolutely-free-by-fareed-khan-oct-2023/ (дата звернення: 19.11.2023).

34. Tutorial natural language processing. Datastart. URL: https://datastart.ru/blog/read/plavnoe-vvedenie-v-natural-language-processing-nlp

35. Pandas documentation. Pandas. URL: https://pandas.pydata.org/docs/user_guide/index.html#user-guide (дата звернення: 19.11.2023).

36. Willman J. M. Beginning PyQt: A Hands-on Approach to GUI Programming. Apress, 2020. 460 с.

37. Gupta A. Top Python IDEs. Simplilearn. URL: https://www.simplilearn.com/tutorials/python-tutorial/python-ide (дата звернення: 19.11.2023).

38. UML activity diagram tutorial. Lucidchart. URL: https://www.lucidchart.com/pages/uml-activity-diagram (дата звернення: 19.11.2023).

39. Діаграма розгортання. JAK. URL: https://jak.koshachek.com/articles/13-diagrami-rozgortannja.html (дата звернення: 19.11.2023).

40. Порівняльний аналіз патернів mvc, mvp і mvvm. Intkonf. URL: http://intkonf.org/razno-yu-yu-porivnyalniy-analiz-paterniv-mvc-mvp-i-mvvm/

41. Good K. W. Speech privacy measurements and metrics. The journal of the acoustical society of america. 2019. Т. 145, № 3. С. 1651. URL: https://doi.org/10.1121/1.5101055 (дата звернення: 19.11.2023).

42. Word recognition error analysis / L. J. Flynn та ін. Assessment for effective intervention. 2011. Т. 36, № 3. С. 167–178. URL: https://doi.org/10.1177/1534508411398649 (дата звернення: 19.11.2023).

43. Методология оценивания работы систем автоматического распознавания речи. информационные технологии автоматического анализа речи. Информационные технологии автоматического анализа речи. С. 38. URL: https://science.donntu.edu.ua/sii/onopko/library/10.pdf

44.Програмна реалізація експерименту. Google Colab. URL: https://colab.research.google.com/drive/1ocrp0-TdxQZjnJfzRTs1VYG5AHWUQpQk?usp=sharing (дата звернення: 19.11.2023).

45.О.В. Бісікало, П.О. Петрук. «Розробка методу розпізнавання українського мовлення медичного спрямування з перетворенням аудіозаписів у текст» в Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2024)», Вінниця, 2023. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/19684. Дата звернення: Грудень. 2023. – 2 с.

# ДОДАТКИ

**Додаток А**

**(обов'язковий)**

**Технічне завдання**

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації

**З А Т В Е Р Д Ж У Ю**
Завідувач кафедри АІІТ
д.т.н., професор Олег БІСІКАЛО
(підпис)
«12» жовтня 2023 р.

**ТЕХНІЧНЕ ЗАВДАННЯ**
на магістерську кваліфікаційну роботу
**РОЗРОБКА МЕТОДУ РОЗПІЗНАВАННЯ УКРАЇНСЬКОГО МОВЛЕННЯ МЕДИЧНОГО СПРЯМУВАННЯ З ПЕРЕТВОРЕННЯМ АУДІОЗАПИСІВ У ТЕКСТ**
08-31.МКР.011.02.000 ТЗ

Керівник магістерської кваліфікаційної роботи
д.т.н., проф., зав. каф. АІІТ
Олег БІСІКАЛО
«11» жовтня 2023 р.
Розробив студент гр. 3АКІТ-22М
Петро ПЕТРУК
«10» жовтня 2023 р.

Вінниця ВНТУ 2023

1. Назва та галузь застосування.

Розробка методу розпізнавання українського мовлення медичного спрямування з перетворенням аудіозаписів у текст. Автоматизація та комп'ютерно-інтегровані технології.

2. Підстава для проведення робіт.

Підставою для виконання роботи є наказ №247 по ВНТУ від «18» вересня 2023р., та індивідуальне завдання на МКР, затверджене протоколом №1 засідання кафедри АІІТ від «30» серпня 2023р.

3. Мета та призначення роботи.

Метою дослідження є підвищення ефективності перетворення аудіозаписів у текст шляхом розробки та апробації методу автоматизованого розпізнавання українського медичного мовлення.

4. Джерела розробки:

1) Python: The Most Advanced Programming Language for Computer Science Applications. Dhruv, A., Patel, R. and Doshi, N. Proceedings of the International Conference on Culture Heritage, Education, Sustainable Tourism, and Innovation Technologies (CESIT 2020), 2022, pages 292-299

2) Core.telegram – Документація по створенню Telegram-ботів. URL: https://core.telegram.org/bots (дата звернення 02.10.2023)

3) Setup Telegram bot to get alert notifications. URL: https://medium.com/linux-shots/setup-telegram-bot-to-get-alert-notifications-90be7da4444 (дата звернення 01.10.2023)

5. Показники призначення Основні технічні вимоги та мінімальні системні вимоги до програми:

- стабільний зв'язок з мережею Інтернет;
- встановлений Telegram бот;
- версія Android від 8, або IOS від 10, або Windows від 8.

Вихідні дані для проведення робіт:

78

Інстанс для розгортання, стек технологій для розробки продукту: месенджер Telegram та Telegram API.

Методи дослідження:

У даній роботі методи дослідження включають аналітичний огляд існуючих рішень, методи машинного навчання, технології побудови програмних модулів, методи критеріальної оцінки та планування експерименту, організації тестування на реальних медичних даних.

Результати роботи програми:

Збереження уділфайлу у базі даних, надання розпізнаного тексту, який міститься на аудіозаписі.

6. Стадії розробки:

а) Вибір, узгодження та затвердження теми МКР 25.09.23 – 05.10.23

б) Аналіз та обгрунтування розробки програмного продукту. Попередня розробка основних розділів 06.10.23 – 09.10.23

в) Розробка технічного завдання (ТЗ) 09.10.23 – 10.11.23

г) Аналіз та обгрунтування розробки методу 10.11.23 – 17.11.23

ґ) Розробка архітектури програмного засобу 17.11.23 – 24.11.23

д) Реалізація системи для здійснення анотацій 25.11.23 – 04.12.23

е) Розробка моделі розпізнавання українського мовлення 04.12.23 – 08.12.23

є) Аналіз результатів роботи 08.12.23 – 10.12.23

ж) Оформлення індивідуального завдання 10.12.23 – 11.12.23

7. Порядок контролю та приймання

Рубіжний контроль провести до «1» листопада 2023 р.

Попередній захист МКР провести «10» листопада 2023 р.

Захист МКР провести до «11» грудня 2023 р.

Розробив студент групи ЗАКІТ-22м _____ Петро ПЕТРУК

# Додаток Б
# (обов’язковий)
# Ілюстративна частина

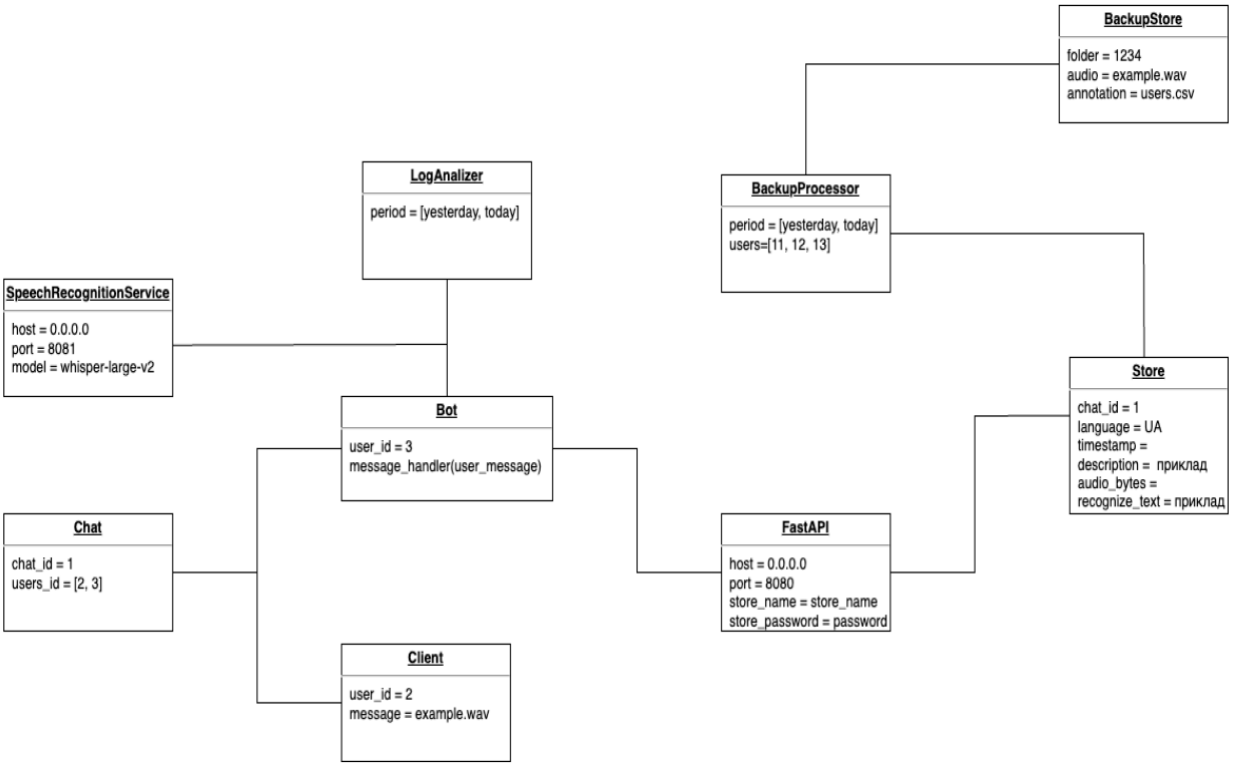



Рисунок Б.1 – UML-діаграма об’єктів

**Продовження додатку Б**

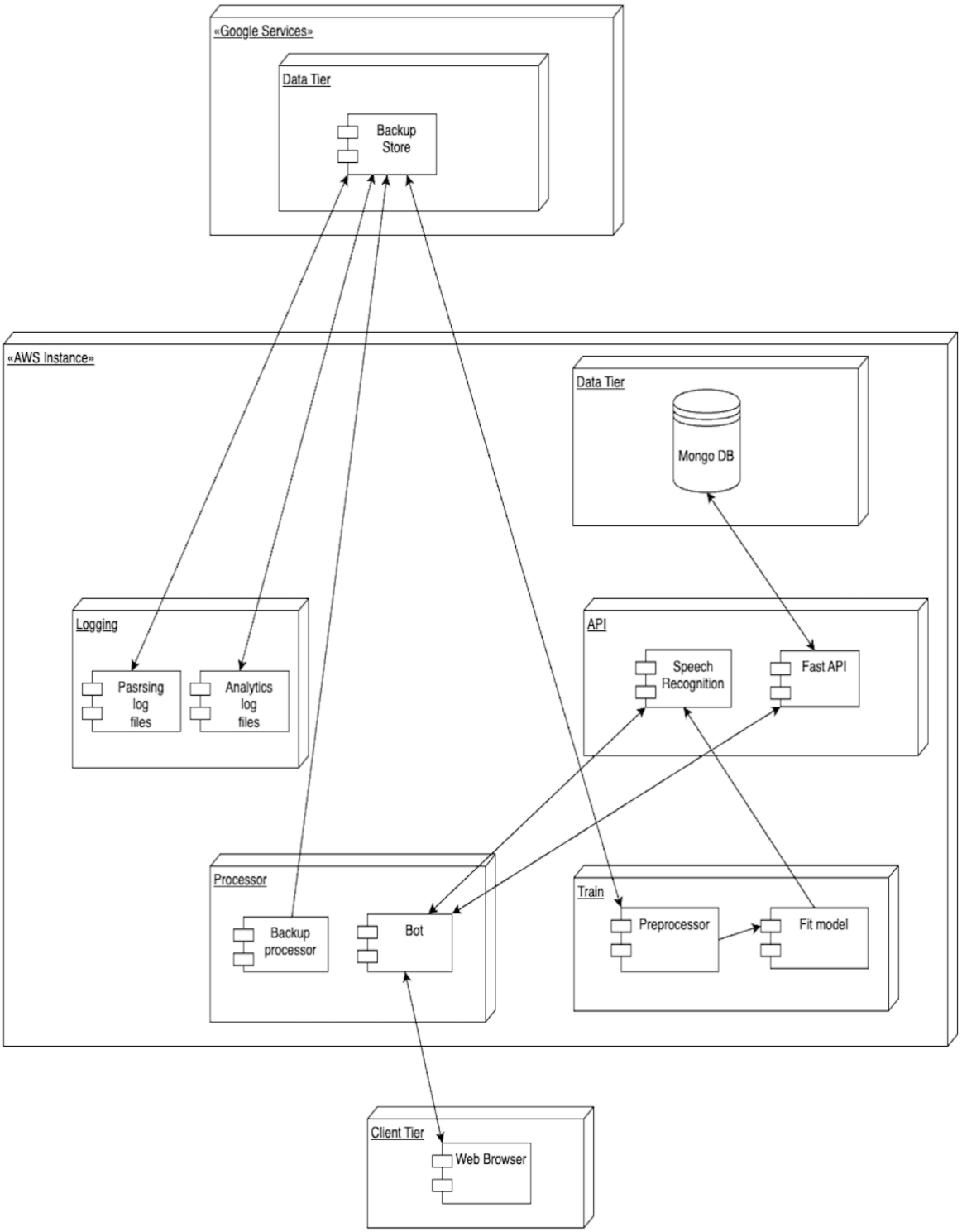



Рисунок Б.2 – UML-діаграма розгортання

**Продовження додатку Б**

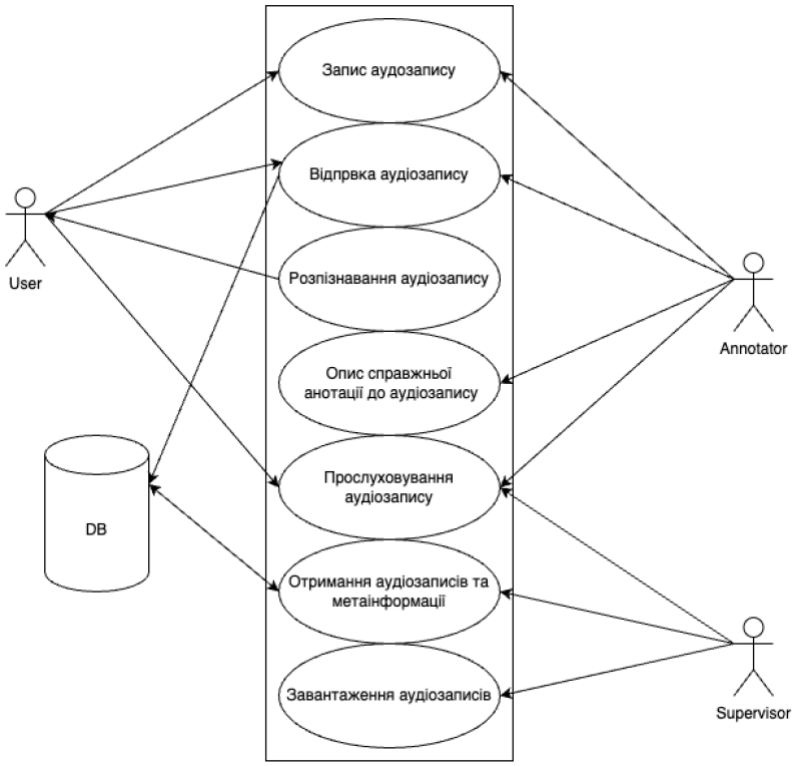



Рисунок Б.3 – – UML- Use Case diagram діаграма

**Продовження додатку Б**

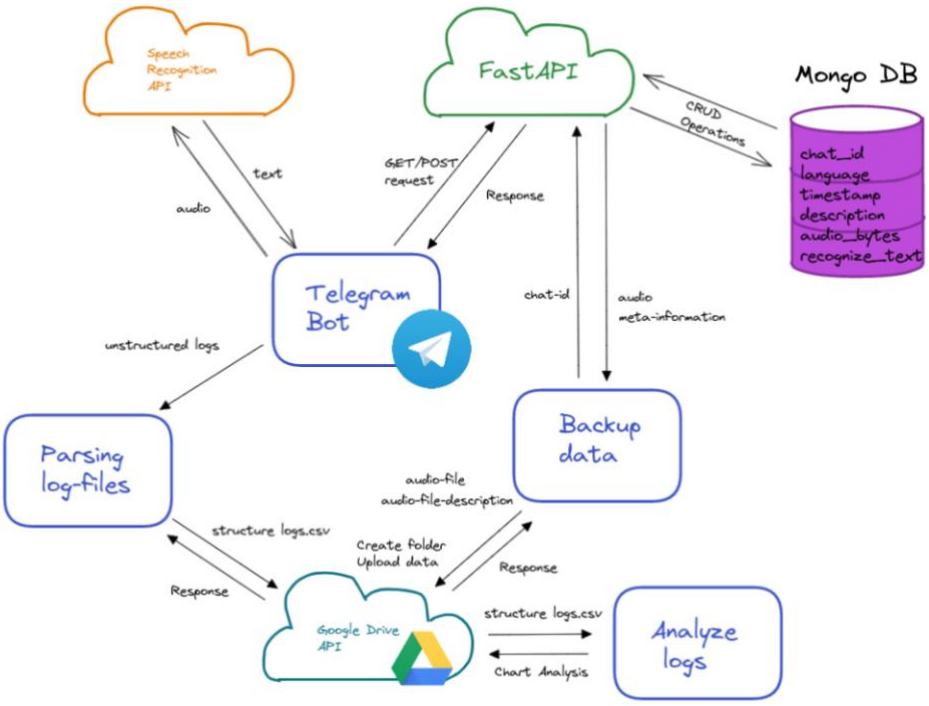



Рисунок Б.4 – Схема архітектури

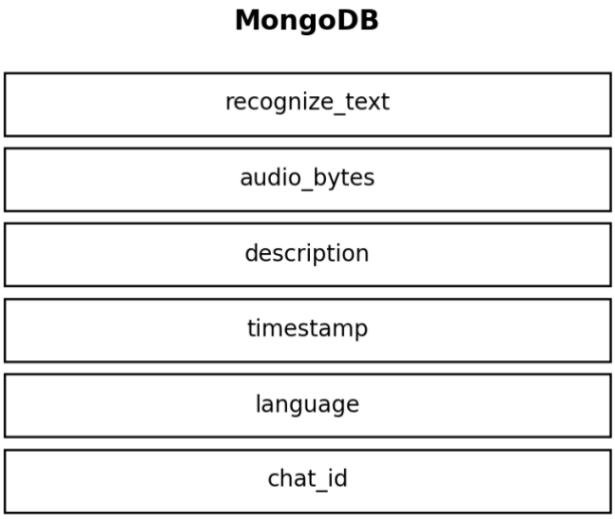



Рисунок Б.5 – Структура даних

**Продовження додатку Б**

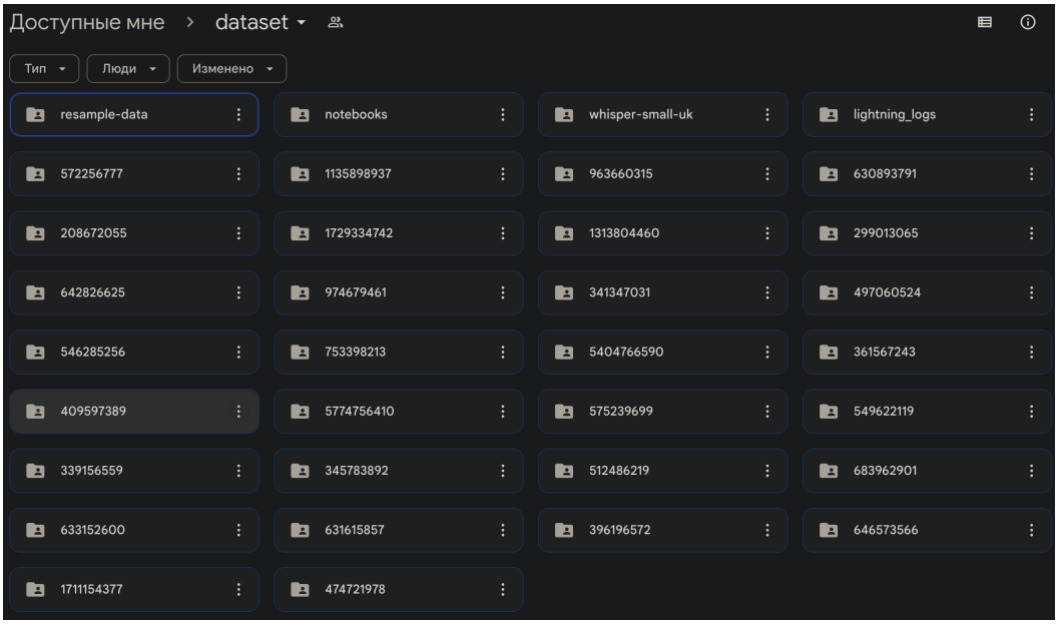



Рисунок Б.6 – Збереженні папки з аудіозаписами та мета інформацією

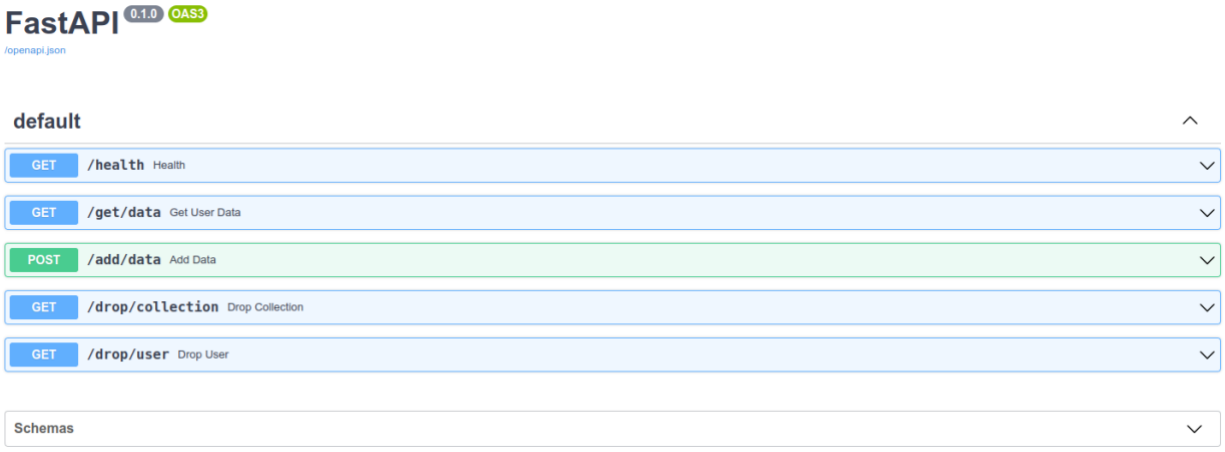



Рисунок Б.7 – Вигляд розроблених ендпоінтів в FastAPI

**Продовження додатку Б**

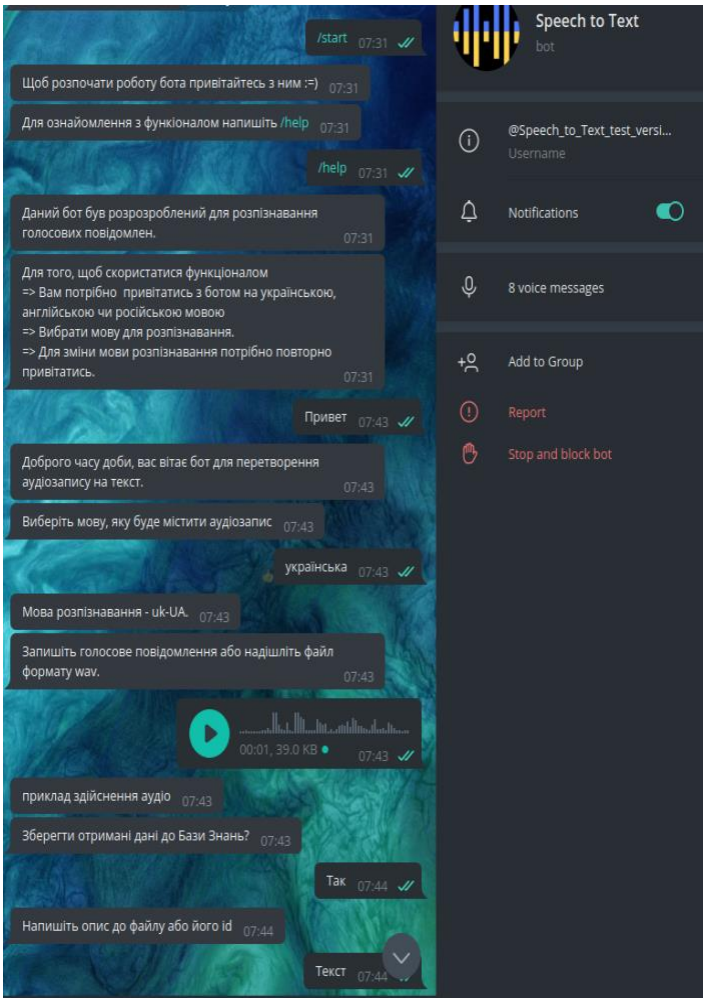



Рисунок Б.8 – Приклад анотації в телеграм боті

**Продовження додатку Б**

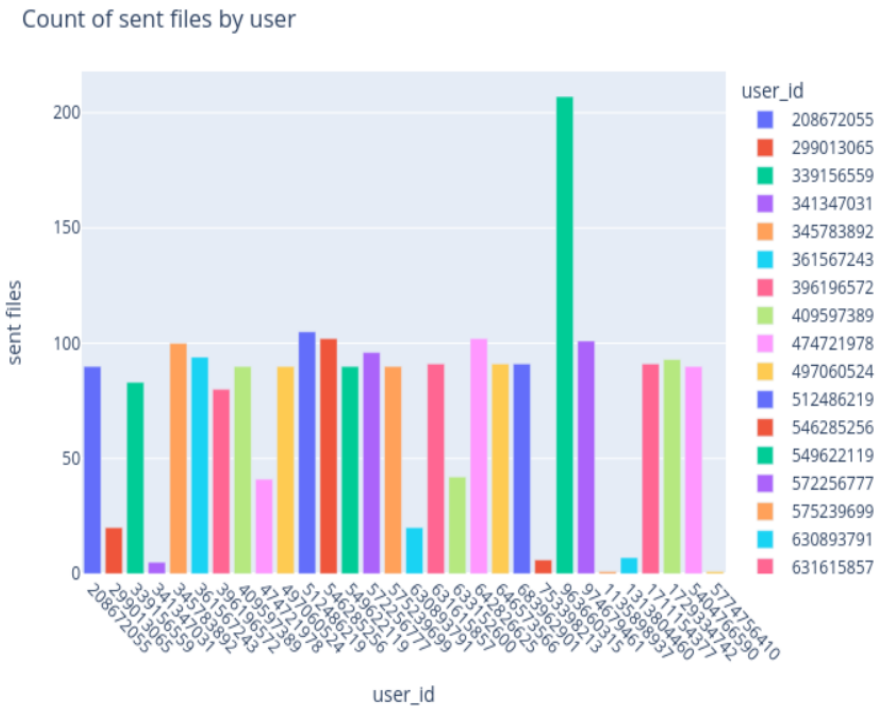



Рисунок Б.9 - Графіки статистичної взаємодії із телеграм ботом частина 1

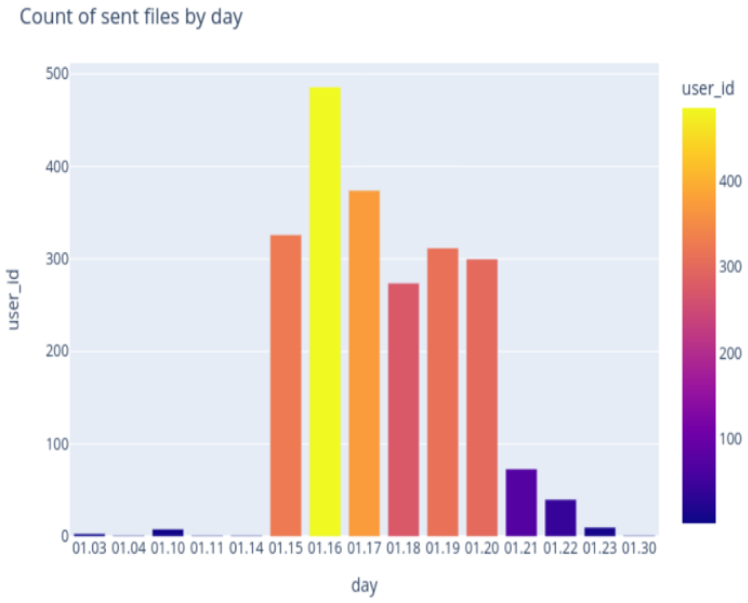



Рисунок Б.10 - Графіки статистичної взаємодії із телеграм ботом частина 2

**Додаток В**

**(обов’язковий)**

**Лістинг програмного коду**

```
import base64
import os
import sys
from datetime import datetime
from io import BytesIO

import telebot
from loguru import logger
from telebot import types

from modules.converter_manager import get_text_with_speech
from modules.database_manager import get_files_by_chat_id, get_save_data, save_to_database


logger.configure(
    handlers=[
        {"sink": sys.stderr, "level": "DEBUG"},
        dict(
            sink="logs/debug.log",
            format="{time} {level} {message}",
            level="DEBUG",
            rotation="1 weeks",
        ),
    ]
)
```

```
LANGUAGES_MAP = {
    "українська": "uk-UA",
    "російська": "ru-RU",
    "англійська": "en-US",
    "німецька": "de-DE",
}

SERVER_HOST = os.getenv("SERVER_HOST")
SERVER_PORT = os.getenv("SERVER_PORT")
TOKEN = os.getenv("TOKEN")

CFG = {"language": "uk-UA"}
utcnow = datetime.utcnow().strftime("%Y-%m-%dT%H:%M:%S.%fZ")

bot = telebot.TeleBot(TOKEN)
logger.info("Bot successfully launched")



@bot.message_handler(commands=["start", "help", "search"])
def handle_start_help(message: telebot.types.Message):
    """Handle start and help commands"""
    logger.info(f"User [{message.chat.id}] => asked for help")

    if message.text == "/start":
        bot.send_message(message.chat.id, "Щоб розпочати роботу бота привітайтесь з ним :=)")
        bot.send_message(message.chat.id, "Для ознайомлення з функіоналом напишіть /help")
```

```
    elif message.text == "/help":
        bot.send_message(
            message.chat.id, "Даний бот був розрозроблений для розпізнавання голосових повідомлен."
        )

        bot.send_message(
            message.chat.id,
            "Для того, щоб скористатися функціоналом \n"
            "=> Вам потрібно  привітатись з ботом "
            "на українською, англійською чи російською мовою \n=> Вибрати мову для розпізнавання.\n"
            "=> Для зміни мови розпізнавання потрібно повторно привітатись.",
        )

    elif message.text == "/search":
        bot.send_message(
            message.chat.id,
            "Надішліть часовий проміжок за який потрібно знайти записи. Слід зауважити час збережений за Coordinated Universal Time (UTC). Приклад формату:",
        )

        bot.send_message(
            message.chat.id, "2021-12-18T12:41:05.488441Z 2021-12-18T12:41:05.488441Z"
        )
        bot.send_message(message.chat.id, "Введіть часовий проміжок (або chat id):")
```

```
            bot.register_next_step_handler(message, search_files)

        elif word_search(message.text):
            say_hello(message)

        else:
            bot.send_message(
                message.chat.id,
                "Для того, щоб скористатися функціоналом \n"
                "=> Вам потрібно  привітатись з ботом "
                "на українською, англійською чи російською мовою \n=> Вибрати мову для розпізнавання.\n"
                "=> Для зміни мови розпізнавання потрібно повторно привітатись.",
            )


    def search_files(message: telebot.types.Message):
        """Handle search files command"""
        logger.info(
            f"User [{message.chat.id}] => search files by time interval or chat_id => {message.text}"
        )

        chat_id, time_from, time_to = None, None, None
        try:
            if len(message.text.split()) == 1:
                chat_id = int(message.text)
            else:
                time_from, time_to = message.text.split()
        except Exception:
```

```
            bot.send_message(
                message.chat.id,
                "Неправильний формат часу. Слід зауважити час збережений за "
                "Coordinated Universal Time (UTC). Приклад формату:",
            )
            bot.send_message(
                message.chat.id, "2021-12-18T12:41:05.488441Z 2021-12-
18T12:41:05.488441Z"
            )
            bot.send_message(message.chat.id, "Введіть часовий проміжок ще раз:")

            logger.error(f"User [{message.chat.id}] => Invalid time format => {message.text}")
            bot.register_next_step_handler(message, search_files)

        else:
            markup = types.ReplyKeyboardMarkup()

            markup.row("Так", "Ні")
            bot.send_message(
                message.chat.id,
                "Виконувати пошук серед користувачів, які надали анотацію до аудіозапису?",
                reply_markup=markup,
            )

            bot.register_next_step_handler(message, run_search_files, chat_id, time_from, time_to)
```

```
def run_search_files(message: telebot.types.Message, chat_id, time_from, time_to):
    collection_name = "user" if message.text.lower() == "так" else "undefined_user"
    logger.info(f"User [{message.chat.id}] => Asked search files with => {collection_name}")

    try:
        if chat_id:
            result = get_files_by_chat_id(SERVER_HOST, SERVER_PORT, collection_name, chat_id)[
                "result"
            ]
        else:
            result = get_save_data(SERVER_HOST, SERVER_PORT, collection_name, time_from, time_to)[
                "result"
            ]

        bot.send_message(
            message.chat.id,
            f"Знайдено записів: {len(result)}",
            reply_markup=telebot.types.ReplyKeyboardRemove(),
        )
        logger.info(f"User [{message.chat.id}] => Find files => {len(result)}")

        for data in result:
            user_id = data["user_id"]
            time = data["timestamp"]["$date"]
```

```
            decode_bytes = base64.b64decode(data["speech_bytes"])
            caption = (
                f'description ~ {data["description"]} \n'
                f"time ~ {time} \nuser_id ~ {user_id} \n"
                f'language ~ {data["language"]} \n'
                f'text ~ {data["text"]}'
            )

            # logger.info(
            #                 f"User [{message.chat.id}] => Send file => {size_b64_string(data['speech_bytes']) / 1024} kb")
            # logger.info(f"User [{message.chat.id}] => Length caption => {len(caption)} characters")

            if len(caption) > 1024:
                preview_message = bot.send_voice(
                    message.chat.id,
                    decode_bytes,
                    caption=caption[:1024],
                )

                bot.reply_to(preview_message, caption[1024:])

            else:
                bot.send_voice(
                    message.chat.id,
                    decode_bytes,
                    caption=caption,
                )
```

```
        except Exception as e:
            bot.send_message(
                message.chat.id,
                "Помилка з'єднання з сервером. Повідомте про це розробників.",
            )
            logger.error(f"User [{message.chat.id}] => {e}")


    def is_help(message: telebot.types.Message):
        """Check if message is help command"""
        if not message == "/help":
            bot.register_next_step_handler(message, say_hello)


    @bot.message_handler(content_types=["voice"])
    def voice_processing(message: telebot.types.Message):
        """Processing voice message"""
        logger.info(f"User [{message.chat.id}] => Sent audio message")

        file_info = bot.get_file(message.voice.file_id)
        downloaded_file = bot.download_file(file_info.file_path)

        try:
            text = get_text_with_speech(BytesIO(downloaded_file),
CFG["language"], logger, message)
        except Exception as e:
            bot.send_message(
                message.chat.id,
                "На етапі розпізнавання аудіозапису сталася помилка - сповістіть
про це розробників",
```

```
            )
            logger.error(f"User [{message.chat.id}] ~ Recognition-Error => {e}")

        else:
            if not text:
                bot.send_message(
                    message.chat.id,
                    "Текст невдалося розпізнати, спробуйте записати аудіозапис у менш шумному місці.",
                )
            else:
                bot.send_message(message.chat.id, text)
                logger.info(f"User [{message.chat.id}] => Recognition text => {text}")

                markup = types.ReplyKeyboardMarkup()

                markup.row("Так", "Ні")
                bot.send_message(
                    message.chat.id,
                    "Зберегти отримані дані до Бази Знань?",
                    reply_markup=markup,
                )

                bot.register_next_step_handler(message, is_save_to_db, text, downloaded_file)


    def is_save_to_db(message: telebot.types.Message, text, downloaded_file):
        logger.info(f"User [{message.chat.id}] => Asked for save data => {message.text}")
```

```
        time_utc = datetime.utcnow().strftime("%Y-%m-%dT%H:%M:%S.%fZ")

        if message.text.lower() == "так":
            bot.send_message(
                message.chat.id,
                "Напишіть опис до файлу або його id",
                reply_markup=telebot.types.ReplyKeyboardRemove(),
            )

            bot.register_next_step_handler(message, input_description, text, downloaded_file, time_utc)

        else:
            save_to_db_without_description(message, text, downloaded_file, time_utc)

            bot.send_message(
                message.chat.id,
                "Запишіть голосове повідомлення або надішліть файл формату wav.",
                reply_markup=telebot.types.ReplyKeyboardRemove(),
            )


    def save_to_db_without_description(
        message: telebot.types.Message, text, downloaded_file, time_utc
    ):
        """Save to database without description"""
```

```
        logger.info(f"User [{message.chat.id}] => Input description => None")

        try:
            status = save_to_database(
                SERVER_HOST,
                SERVER_PORT,
                "undefined_user",
                message.chat.id,
                time_utc,
                text,
                CFG["language"],
                downloaded_file,
                message.text,
            )

            logger.info(f"User [{message.chat.id}] ~ Request-Status => {status}")
        except Exception as e:
            logger.error(f"User [{message.chat.id}] ~ Save-Error => {e}")


    def input_description(message: telebot.types.Message, text, downloaded_file, time_utc):
        """Input description for file and save to db"""
        logger.info(f"User [{message.chat.id}] => Input description => {message.text}")
        save_config("description", message.text, message)

        try:
            status = save_to_database(
                SERVER_HOST,
```

```
                SERVER_PORT,
                "user",
                message.chat.id,
                time_utc,
                text,
                CFG["language"],
                downloaded_file,
                message.text,
            )

            bot.send_message(
                message.chat.id,
                "✅ Інформація успішно збережена до Бази Знань.",
            )
            logger.info(f"User [{message.chat.id}] ~ Request-Status => {status}")
        except Exception as e:
            logger.error(f"User [{message.chat.id}] ~ Save-Error => {e}")
            bot.send_message(
                message.chat.id,
                "❌ На етапі збереження даних сталася помилка - сповістіть про це розробників",
            )

        bot.send_message(
            message.chat.id,
            "Запишіть голосове повідомлення або надішліть файл формату wav.",
            reply_markup=telebot.types.ReplyKeyboardRemove(),
        )
```

```
@bot.message_handler(func=lambda message: True, content_types=["text"])
def event_handler(message: telebot.types.Message):
    """Handle all messages"""
    logger.info(f"User [{message.chat.id}] => sent a message => {message.text}")
    if word_search(message.text):
        say_hello(message)


def say_hello(message: telebot.types.Message):
    """Say hello to user and ask for language"""
    if word_search(message.text):
        bot.send_message(
            message.chat.id,
            f"Доброго часу доби, вас вітає бот для перетворення аудіозапису на текст. "
            f"Ваш унікальний ідентифікатор чату => {message.chat.id}.",
        )
        markup = types.ReplyKeyboardMarkup()

        markup.row("українська", "російська")
        markup.row("англійська", "німецька")
        bot.send_message(
            message.chat.id,
            "Виберіть мову, яку буде містити аудіозапис",
            reply_markup=markup,
        )

        bot.register_next_step_handler(message, get_language)
```

```
def get_language(message: telebot.types.Message):
    """Get language from user"""
    change_language = LANGUAGES_MAP.get(message.text, "uk-UA")
    save_config("language", change_language, message)

    bot.send_message(
        message.chat.id,
        f"Мова розпізнавання - {change_language}.",
        reply_markup=telebot.types.ReplyKeyboardRemove(),
    )
    bot.send_message(
        message.chat.id,
        "Запишіть голосове повідомлення або надішліть файл формату wav.",
    )


def save_config(key: str, value: str, message: telebot.types.Message) -> None:
    """Save data to config"""
    CFG.update({key: value})
    logger.info(f"User [{message.chat.id}] => Updated settings => {CFG}")


def word_search(text: str) -> bool:
    """Check if text contains hello words"""
    key_word = [
        "hello",
        "привет",
        "hi",
        "ку",
```

```
        "вітаю",
        "привіт",
        "доброго",
    ]

    array = list(map(lambda el: el.lower(), text.split()))
    result = True in list(map(lambda el: el in array, key_word))

    return result


if __name__ == "__main__":
    bot.polling(none_stop=True, interval=0)
```

**Додаток Г**

**(обов'язковий)**

**Протокол перевірки магістерської кваліфікаційної роботи на наявність текстових запозичень**

Назва роботи: «Розробка методу розпізнавання українського мовлення медичного спрямування з перетворенням аудіозаписів у текст»

Тип роботи: магістерська кваліфікаційна робота

Підрозділ: кафедра АІІТ

**Показники звіту подібності Unicheck**

Оригінальність 96,8 % Схожість 3,2 %

Аналіз звіту подібності (відмітити потрібне)

• Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

○ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.

○ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ (підпис) Роман МАСЛІЙ

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ (підпис) Петро ПЕТРУК

Керівник роботи _____ (підпис) Олег БІСІКАЛО

**Додаток Д**

**(необов'язковий)**

**Апробація програмного забезпечення**

УКРАЇНА

СВІДОЦТВО

про реєстрацію авторського права на твір

№ 110697

Комп'ютерна програма «Розпізнавання українського мовлення з перетворенням у текст» («Розумійка»)

(вид, назва твору)

Автор(и) Петрук Петро Олександрович, Нестюк Юлія Юріївна, Бісікало Олег Володимирович, Ковтун В'ячеслав Васильович

(повне ім'я, псевдонім (за наявності))

Дата реєстрації 30 грудня 2021 р.

Генеральний директор
Державного підприємства
«Український інститут
інтелектуальної власності» Андрій КУДІН

«УКРАЇНСЬКИЙ ІНСТИТУТ ІНТЕЛЕКТУАЛЬНОЇ ВЛАСНОСТІ»
ідентифікаційний код
31032378

Рисунок Д.1 – Авторське право на розроблене програмне забезпечення

**ВЛАСТАМЕД**
приватна медична практика
Вінницька область, м. Вінниця, 4 пров. Каденюка, 2,
тел. 068 60 85 241, 063 64 85 242, email: vlastamed2021@gmail.com

«14» грудня 2023 р.

# ДОВІДКА

про впровадження результатів магістерської кваліфікаційної роботи студента ВНТУ Петрука Петра Олександровича на тему «Розробка методу розпізнавання українського мовлення медичного спрямування з перетворенням аудіозаписів у текст»

Предметом дослідження у магістерській кваліфікаційній роботі "Розробка методу розпізнавання українського мовлення медичного спрямування з перетворенням аудіозаписів у текст" було створення інноваційного методу для автоматизації процесу розпізнавання мовлення. Основна мета полягала у розробці ефективного рішення, яке б дозволяло лікарям та медичному персоналу перетворювати аудіозаписи медичного спрямування в текст, сприяючи тим самим підвищенню ефективності медичної документації та обробки пацієнтських даних.

У ході дослідження було розроблено спеціалізований телеграм-бот https://t.me/Speech_to_Text_test_version_Bot, що використовує натреновану на медичних даних модель. Цей бот демонструє значні переваги у точності та швидкості перетворення мовлення на текст, що є особливо важливим у медичній сфері.

Впровадження цього методу відбувалося у приватній клініці «Властамед», де його застосовано для розпізнавання надиктованих аудіозаписів та оптимізації процесу документування медичної інформації. Застосування розробленого програмного забезпечення в клініці сприяло значному поліпшенню обробки даних, ефективності ведення медичних записів та забезпечення високої якості комунікації з пацієнтами.

Цим підтверджую, що основні результати магістерської роботи Петрука Петра Олександровича було успішно впроваджено у практичну діяльність приватної клініки. На мою думку, розширення такого впровадження у інших медичних закладах вплине на покращення медичного обслуговування громадян України за рахунок помітного вивільнення робочого часу лікарів, що витрачається на документування.

Головний лікар Сергій ЛЯХОВЧЕНКО