НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

"На правах рукопису"
УДК 004.8

ДО ЗАХИСТУ ДОПУЩЕНО
Завідувач кафедри

_____ Олександр РОЛІК

“ ____ ” _____ 2023 р.

МАГІСТЕРСЬКА ДИСЕРТАЦІЯ

на здобуття ступеня магістра

за освітньо-науковою програмою

«*Інтегровані інформаційні системи*»

зі спеціальності 126 «*Інформаційні системи та технології*»

на тему:

«Підсистема транскрибування голосових повідомлень на основі технології глибинного навчання»

Виконав:
студент 2 курсу, групи ІА-11мн
Драган Михайло Сергійович _____

Керівник:
к.т.н., доцент кафедри інформаційних систем
та технологій КПІ ім. Ігоря Сікорського,
Писаренко Андрій Володимирович _____

Рецензент:
професор кафедри обчислювальної техніки, д. ф.-м., с.н.с.
Гордієнко Юрій Григорович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2023 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій.

Рівень вищої освіти – другий (магістерський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-наукова програма «Інтегровані інформаційні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Олександр РОЛІК__.

«__» _____2023 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Драган Михайло Сергійович

1. Тема дисертації «Підсистема транскрибування голосових повідомлень на основі технології глибинного навчання»

науковий керівник дисертації Писаренко Андрій Володимирович, к.т.н., доцент кафедри інформаційних систем та технологій КПІ ім. Ігоря Сікорського, затверджені наказом по університету від «20» березня 2023 р. № 1275-с.

2. Строк подання студентом дисертації “ 12 ” травня 2023 р.

3. Об’єкт дослідження: підсистема транскрибування голосових повідомлень. Дослідження зосереджене на вивченні можливостей застосування технологій глибинного навчання для поліпшення якості транскрибування.

4. Предмет дослідження: процес транскрибування голосових повідомлень з використанням технологій глибинного навчання, який досліджується з метою покращення точності транскрибування та підвищення ефективності роботи з великим обсягом голосових даних.

5. Перелік завдань, які потрібно розробити:

проаналізувати наявні методи транскрибування голосових повідомлень; вивчити основи технологій глибинного навчання; розробити та навчити нейронну мережу для автоматичного транскрибування голосових повідомлень; застосувати методи попереднього оброблення аудіоінформації для підвищення якості транскрибування; провести експериментальне дослідження розробленої системи транскрибування на різноманітних даних та порівняти отримані результати з існуючими методами транскрибування; визначити ефективність розробленої системи транскрибування та її можливості для подальшого розвитку та використання в практичних задачах.

6. Перелік графічного (ілюстративного) матеріалу
Діаграма зв'язку сутностей тренувальних даних. Діаграма діяльності алгоритму фільтрації аудіофайлів від службових тегів. Діаграма компонентів. Діаграма зв’язку сутностей підсистеми. Діаграма прецедентів. Діаграма станів. Діаграма послідовності підсистеми.

7. Орієнтовний перелік публікацій

«Preprocessing of audio data for voice transcription systems» Драган М.С., Писаренко А.В. Міжвідомчий науково-технічний журнал «Адаптивні системи автоматичного управління». – 2023. № 1(42). С. 39–48.

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання “ 31 ” січня 2023 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Систематизація результатів огляду літератури	13.02	
2	Порівняльний аналіз існуючих методів та технологій оброблення мовленнєвої інформації	18.02	
3	Постановка та формалізація задачі транскрибування	14.03	
4	Покращення процесу транскрибування	28.03	
5	Розробка програмного забезпечення	10.04	
7	Проведення експериментальних досліджень розробленої моделі	16.04	
8	Оформлення документації	18.04	
9	Подання роботи на попередній захист	20.04	
10	Подання роботи на основний захист	17.05	

Студент _____ *Михайло ДРАГАН*

Науковий керівник _____ *Андрій ПИСАРЕНКО*

РЕФЕРАТ

Магістерська дисертація: 113 с., 24 рис., 7 табл., 36 джерел, 1 додаток.

Актуальність. Транскрибування голосових повідомлень є актуальним, оскільки з розвитком технологій та збільшенням обсягів аудіо- та відеоматеріалів, які виникають у різних сферах життєдіяльності, стає дедалі більш важливим мати ефективні методи їх автоматичної обробки та аналізу. Глибинне навчання у цьому процесі значно спрощує та поліпшує процес обробки даних.

Мета дослідження – підвищення точності результатів транскрибування або підвищення швидкості роботи з великим обсягом голосових даних без втрати точності за рахунок використання технологій штучного інтелекту.

Об’єкт дослідження – підсистема транскрибування голосових повідомлень.

Предмет дослідження – процес транскрибування голосових повідомлень з використанням технологій глибинного навчання, який досліджується з метою покращення точності транскрибування та підвищення ефективності роботи з великим обсягом голосових даних.

Методи дослідження, застосовані у даній роботі, базуються на методах глибинного навчання.

Наукова новизна одержаних результатів полягає у поєднанні технологій глибинного навчання та попереднього оброблення голосових повідомлень, що збільшує швидкості транскрибування за рахунок використання смугового фільтру у попередній фільтрації.

Публікації. Матеріали роботи опубліковані у статті на тему «Попереднє оброблення аудіоданих для систем транскрибування голосу» у журналі «Міжвідомчий науково-технічний журнал «Адаптивні системи автоматичного управління». – 2023. № 1(42). С. 39–48.

ТРАНСКРИБУВАННЯ АУДІОФАЙЛУ, ГОЛОСОВІ ПОВІДОМЛЕННЯ, ГЛИБИННЕ НАВЧАННЯ.

ABSTRACT

The master’s dissertation: 113 p., 24 fig., 7 tables, 36 sources, 1 appendix.

Relevance. Transcription of voice messages is relevant because with the development of technology and the increase in the volume of audio and video materials that arise in various spheres of life, it is becoming increasingly important to have effective methods for their automatic processing and analysis. Deep learning in this process greatly simplifies and improves the data processing process.

The purpose of the study is to improve the accuracy of transcription results or increase the speed of working with a large amount of voice data without losing accuracy through the use of artificial intelligence technologies.

The object of research is a voice message transcription subsystem.

The subject of the study is the process of transcribing voice messages using deep learning technologies, which is being investigated to improve transcription accuracy and increase the efficiency of working with large amounts of voice data.

The research methods used in this paper are based on deep learning techniques.

The scientific novelty of the results obtained is the combination of deep learning and voice message pre-processing technologies, which increases transcription speeds by using a bandpass filter in pre-filtering.

Publications. The materials of the work are published in the article on the topic "Preprocessing of audio data for voice transcription systems" in the " Interdepartmental scientific-technical journal «Adaptive systems of automatic control». – 2023. № 1(42). P. 39–48.

AUDIO FILE TRANSCRIPTION, VOICE MESSAGES, DEEP LEARNING.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ 7

ВСТУП..... 9

1 ТЕОРЕТИЧНІ АСПЕКТИ ТРАНСКРИБУВАННЯ ГОЛОСОВИХ ПОВІДОМЛЕНЬ..... 13

1.1 Аналіз існуючих методів транскрибування голосових повідомлень 14

1.2 Аналіз методів попереднього оброблення голосових повідомлень 22

1.3 Огляд технології глибинного навчання та її використання у задачах оброблення мовленнєвої інформації..... 27

1.4 Опис архітектури системи транскрибування голосових повідомлень на основі технології глибинного навчання..... 34

Висновки до розділу 1 39

2 РОЗРОБЛЕННЯ ПІДСИСТЕМИ ТРАНСКРИБУВАННЯ ГОЛОСОВИХ ПОВІДОМЛЕНЬ НА ОСНОВІ ТЕХНОЛОГІЇ ГЛИБИННОГО НАВЧАННЯ 40

2.1 Збір даних та підготовка набору даних для навчання моделі 41

2.2 Розроблення та навчання моделі системи транскрибування голосових повідомлень на основі технології глибинного навчання 56

2.3 Реалізація та тестування системи 63

2.4 Оцінка ефективності системи транскрибування..... 73

Висновки до розділу 2 79

3 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ 82

3.1 Порівняння розробленої системи транскрибування з існуючими 83

3.2 Покращення слабких місць моделі..... 89

Висновки до розділу 3 97

ВИСНОВКИ..... 99

ПЕРЕЛІК ПОСИЛАНЬ 102

ДОДАТОК А Графічниий матеріал..... **Ошибка! Закладка не определена.**

Плакат 1. Діаграма зв'язку сутностей тренувальних даних**Ошибка! Закладка не определена.**

Плакат 2. Діаграма діяльності алгоритму фільтрації аудіофайлів від службових тегів..... **Ошибка! Закладка не определена.**

Плакат 3. Діаграма компонентів..... **Ошибка! Закладка не определена.**

Плакат 4. Діаграма зв’язку сутностей підсистеми**Ошибка! Закладка не определена.**

Плакат 5. Діаграма прецедентів..... **Ошибка! Закладка не определена.**

Плакат 6. Діаграма станів..... **Ошибка! Закладка не определена.**

Плакат 7. Діаграма послідовності підсистеми**Ошибка! Закладка не определена.**

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер;

ФНЧ – фільтр низьких частот;

ФВЧ – фільтр високих частот;

AI – Artificial Intelligence;

API – Application Programming Interface;

ASR – Automatic Speech Recognition;

AWS – Amazon Web Services;

Blob – Binary Large Object;

CNN – Convolutional Neural Network;

CPU – Central Processing Unit;

DFT – Discrete Fourier Transform;

DOM – Document Object Model;

DNN – Deep Neural Network;

DSP – Digital Signal Processing;

FFT – Fast Fourier Transform;

GMM – Gaussian Mixture Models;

GPU – Graphics Processing Unit;

HMM – Hidden Markov Model;

HTTP – Hypertext Transfer Protocol;

LSTM – Long Short-Term Memory;

MFCC – Mel Frequency Cepstral Coefficients;

NLTK – Python Natural Language Toolkit;

RBF – Radial Basis Function;

RNN – Recurrent Neural Network;

SVM – Support Vector Machines;

TIMIT – Texas Instruments and Massachusetts Institute of Technology;

WER – Word Error Rate.

ВСТУП

Голосові повідомлення є одним з найпоширеніших засобів комунікації в сучасному світі. Проте, їх оброблення та аналіз можуть стикатися з багатьма труднощами, включаючи різні акценти та мовленнєві особливості, а також шумові фони. Для аналізу цієї інформації потрібно перетворити її у текстовий формат. Вручну це займає дуже багато часу і потребує багато зусиль, тому виникає потреба у системах автоматичного транскрибування голосових повідомлень. Тому задача якісного перетворення аудіо інформації в текстовий формат є дуже актуальною у наш час.

Одним з потужних інструментів для вирішення цієї задачі є глибинне навчання. Технології глибинного навчання можуть допомогти забезпечити точність і швидкість транскрибування, що може бути корисним для бізнесу, наукових досліджень та особистого використання.

Технології глибинного навчання, такі як глибинні нейронні мережі, дали величезний розгін у сфері оброблення голосу. Вони можуть допомогти зробити транскрибування голосових повідомлень більш точним і швидким. За допомогою глибинного навчання можна навчити комп'ютер розпізнавати різноманітні звуки, такі як голосові команди, мовні фрази, шуми і ефекти.

Транскрибування голосових повідомлень на основі технологій глибинного навчання використовується в різних сферах та індустріях, наприклад:

- комунікації – технології транскрибування голосових повідомлень можуть бути використані для покращення комунікації між людьми; наприклад, це може бути корисно для людей з проблемами слуху або для тих, хто працює в гучних середовищах;
- медіа – компанії медіа використовують технології транскрибування голосових повідомлень для створення текстових версій аудіо- та відеоконтенту; це дозволяє користувачам швидко знайти та переглянути цікаві моменти у вмісті;

- комп'ютерна безпека – технології транскрибування голосових повідомлень можуть бути використані в комп'ютерній безпеці для аналізу телефонних розмов та виявлення загроз;
- медицина – технології транскрибування голосових повідомлень можуть бути використані в медицині для створення записів прийому пацієнтів та звітів про лікування;
- освіта – технології транскрибування голосових повідомлень можуть бути використані в освіті для запису лекцій та створення текстових версій з метою полегшення навчання.

Отже, технології транскрибування голосових повідомлень на основі глибинного навчання можуть бути корисними в різних сферах та індустріях, де потрібно перетворити голосові повідомлення на текст. Далі буде детально розглянуто підходів та методів розв’язання задачі транскрибування голосових повідомлень, а також описано компоненти підсистеми транскрибування та розробка робочого алгоритму.

Об’єкт дослідження

Об’єктом дослідження магістерської дисертації є підсистема транскрибування голосових повідомлень. Дослідження зосереджене на вивченні можливостей застосування технологій глибинного навчання для поліпшення якості транскрибування.

Предмет дослідження

Предметом дослідження магістерської дисертації є процес транскрибування голосових повідомлень з використанням технологій глибинного навчання, який досліджується з метою покращення точності транскрибування та підвищення ефективності роботи з великим обсягом голосових даних.

Мета

Метою магістерської дисертації є підвищення точності результатів транскрибування або підвищення швидкості роботи з великим обсягом голосових даних без втрати точності за рахунок використання технологій штучного інтелекту.

Для досягнення мети роботи були поставлені та вирішені наступні завдання:

- проаналізувати наявні методи транскрибування голосових повідомлень;
- вивчити основи технологій глибинного навчання;
- розробити та навчити нейронну мережу для автоматичного транскрибування голосових повідомлень;
- застосувати методи попереднього оброблення аудіоінформації для підвищення якості транскрибування;
- провести експериментальне дослідження розробленої системи транскрибування на різноманітних даних та порівняти отримані результати з існуючими методами транскрибування;
- визначити ефективність розробленої системи транскрибування та її можливості для подальшого розвитку та використання в практичних задачах.

Практичне значення полягає у наступному:

1) покращення ефективності транскрибування голосових повідомлень; розроблена підсистема транскрибування з використанням технологій глибинного навчання дозволяє досягати більшої точності результатів порівняно з існуючими методами транскрибування голосу; це забезпечує збільшення ефективності та якості обробки голосової інформації;
2) автоматизація процесу транскрибування; розроблена підсистема транскрибування є автоматичною, що дозволяє значно зменшити час, необхідний для транскрибування голосових повідомлень, та зменшити ризик помилок, пов'язаних з людським фактором;

3) розширення можливостей роботи з голосовою інформацією; розроблена підсистема транскрибування може бути використана в різних галузях, де потрібна обробка голосової інформації, наприклад, у сфері медицини, правоохоронних органів, бізнесу тощо;
4) можливість подальшого розвитку та вдосконалення системи транскрибування; розроблена підсистема транскрибування може бути вдосконалена та розширена з використанням нових методів та технологій, це відкриває можливості для подальшого розвитку системи та її використання в нових галузях людської діяльності.

1 ТЕОРЕТИЧНІ АСПЕКТИ ТРАНСКРИБУВАННЯ ГОЛОСОВИХ ПОВІДОМЛЕНЬ

Транскрибування голосу – це процес перетворення аудіо- або відеозапису мовлення на письмовий текст [1]. Цей процес може бути корисним у багатьох різних контекстах, де необхідно перетворити мовлення на текст. Ось кілька прикладів, де використовують транскрибування голосових повідомлень:

- комунікація – деякі месенджери, такі як WhatsApp, Telegram та Viber, надають функціонал голосових повідомлень; транскрибування цих повідомлень може полегшити спілкування та зробити його більш доступним для людей з різними мовними навичками;
- документація – транскрибування голосових записів може бути корисним у наукових, медичних та юридичних дослідженнях, де необхідно мати точний запис мовлення у текстовому форматі;
- підтримка клієнтів – компанії можуть використовувати транскрибування голосових повідомлень для покращення обслуговування клієнтів; наприклад, якщо клієнт залишив голосове повідомлення зі скаргою на продукт, компанія може транскрибувати це повідомлення та аналізувати вміст для покращення продукту та обслуговування;
- медіа-контент – транскрибування голосових записів може бути корисним для виробництва аудіо- та відео-контенту, такого як подкасти або відео-інтерв’ю; транскрибування дозволяє створювати текстову версію вмісту, що полегшує доступ до нього для людей з різними особливостями сприймання.

Отже, транскрибування голосових повідомлень може бути корисним у багатьох різних сферах та контекстах, де необхідно перетворити мовлення на текстовий формат.

Задача транскрибування голосових повідомлень полягає у перетворенні мовлення, записаного у формі аудіо- чи відеофайлу, на текстовий формат. Ця

задача є досить складною через те, що мовлення може містити різноманітні мовні особливості, такі як акцент, діалект, швидкість мовлення, наявність шуму тощо. Тому існує безліч підходів та методів розв'язання цієї задачі. Існує декілька методів транскрибування, а саме:

- ручне транскрибування – цей метод передбачає вручну переписування тексту, що вимовляється на записі; природньо, що загалом цей метод достатньо точний, за виключенням специфічних випадків, однак й забирає максимально багато часу [2];
- метод автоматичного транскрибування з використанням програмного забезпечення використовує розпізнавання мовлення, що базується на алгоритмах машинного навчання, щоб автоматично перетворити аудіо- або відеозапис мовлення на текст; це перспективний метод, який має найбільшу швидкість роботи, при наявності відповідного обладнання [3], однак все ще має певні вади, при розпізнаванні;
- сполучення ручного та автоматичного транскрибування передбачає використання програмного забезпечення для автоматичного транскрибування, а потім редагування транскрипції вручну для поліпшення точності; цей метод поєднує переваги обох методів, що дозволяє швидко транскрибувати велику кількість аудіо- або відеоматеріалів з високою точністю [4].

Природньо, що в підсистемі доцільно реалізовувати саме автоматичний метод транскрибування, дані з якого можна буде вторинно обробити ручним способом.

1.1 Аналіз існуючих методів транскрибування голосових повідомлень

Існують різні методи транскрибування голосових повідомлень. Найпоширенішими з цих методів є автоматичне мовленнєве розпізнавання, ручне транскрибування, гібридний метод, що поєднує у собі ручне та автоматичне транскрибування, комбіноване транскрибування.

Автоматичне мовленнєве розпізнавання (ASR) – це технологія, яка дозволяє перетворювати голосовий сигнал у текстовий формат. ASR використовує навчальні алгоритми та моделі мови, щоб розпізнавати та перетворювати голосовий сигнал у текст [5]. Вона знаходить своє застосування в багатьох галузях, включаючи транскрибування голосових повідомлень, розпізнавання команд у голосових асистентах, відео- та аудіо-індексацію та багато іншого.

Основні компоненти ASR:

- захоплення звуку – аудіо-сигнал з мікрофону або іншого джерела зберігається в цифровому форматі;
- попередня обробка – сигнал обробляється з метою зменшення шуму та підготовки до подальшої обробки;
- розпізнавання мовлення – сигнал перетворюється на послідовність слів або речень за допомогою алгоритмів розпізнавання мовлення, таких як глибоке навчання або статистичні методи;
- після обробка – отримані результати піддаються після обробці, такі як корекція помилок або виправлення форматування.

При цьому перетворення звукових сигналів мовлення у відповідний текстовий формат в процесі може стикатись з наступними проблемами, що є найбільш розповсюдженими:

- різні акценти та діалекти – люди говорять з різними акцентами та діалектами, що може призвести до складнощів у процесі розпізнавання мовлення;
- шум – навколишній шум, такий як шум вулиці або шум вітру, може впливати на якість аудіо-сигналу та ускладнювати процес розпізнавання мовлення;
- розмовна швидкість – розмовники можуть говорити з різною швидкістю, що може також ускладнити процес розпізнавання мовлення;
- розмовники з різним голосом – розмовники можуть мати різний тембр голосу, що може впливати на якість аудіо-сигналу та ускладнювати процес розпізнавання мовлення;

- мовна модель – відповідність вимови слова та його написання залежить від мовної моделі, яку використовують у процесі розпізнавання мовлення;
- відсутність контексту – без контексту контекстуальна інформація може бути втрачена, що може впливати на якість розпізнавання мовлення та зробити його менш точним;
- ідентифікація мови може бути складною, особливо якщо розмовники спілкуються на декількох мовах або змінюють мову під час розмови;
- некоректна вимова – люди можуть вимовляти слова некоректно, що може призвести до неправильного розпізнавання мовлення та зробити його менш точним;
- недостатня якість аудіо-сигналу може бути недостатньою, якщо запис здійснюється на відстані від джерела звуку або з використанням недостатньої якості мікрофону, що може ускладнити процес розпізнавання мовлення.

Процес ASR складається з кількох етапів. По-перше, голосовий сигнал записується та аналізується для визначення його характеристик, таких як частота та амплітуда. Далі голосовий сигнал проходить через алгоритми Digital Signal Processing (DSP), що дозволяють його опрацювати та зменшити шум. Потім сигнал подається до нейронної мережі або моделі Hidden Markov Models (HMM), які використовуються для розпізнавання та класифікації звуків. Далі модель мови використовується для визначення послідовності слів, що найбільш вірогідно могли бути вимовлені згідно з отриманим голосовим сигналом. Модель мови може використовувати статистичні методи, наприклад, граматики, або машинне навчання, такі як нейронні мережі, для розпізнавання та класифікації слів. Зазвичай ASR використовується для розпізнавання мовлення на одній мові, але також може бути застосований для розпізнавання мовлення на декількох мовах. ASR використовується в багатьох додатках, таких як голосові помічники, системи автоматичного телефонного меню та системи диктовки тексту [6].

DSP – це обробка сигналів у цифровому форматі за допомогою алгоритмів та математичних методів. DSP використовується для опрацювання різноманітних

сигналів, включаючи аудіо, зображення, радіосигнали, біомедичні сигнали та інші [7]. У контексті автоматичного мовленнєвого розпізнавання, DSP застосовується для опрацювання голосових сигналів, з метою виявлення та виділення корисної інформації зі сигналу, такої як мовлення, зниження рівня шуму та інтерференції. Опрацьований сигнал подається до моделі мови для подальшого розпізнавання. DSP також використовується для покращення якості звуку в системах зв'язку, аудіо та відео зберігання та передачі даних. DSP алгоритми застосовуються в широкому спектрі пристроїв, таких як смартфони, медична техніка, аудіо та відео обладнання, радіо та телевізійне обладнання та інші.

HMM – це статистична модель, що використовується для аналізу послідовностей даних. У контексті автоматичного мовленнєвого розпізнавання, HMM використовуються для моделювання послідовності фонем або слів, що входять до мовленнєвого сигналу. HMM складається зі складових частин, таких як стани, спостереження та переходи між станами. Стани можуть бути видимими або прихованими, тобто модель може бути використана як для моделювання відкритої або закритої мовленнєвої послідовності [8]. HMM також використовуються для моделювання залежності між фонемами або словами в мовленнєвому сигналі. Однією з основних переваг HMM є їхній рівень гнучкості та можливість моделювання складних мовних послідовностей. HMM може працювати зі слабкими сигналами та шумом, що робить його ідеальним для застосування в області автоматичного мовленнєвого розпізнавання. Однак, HMM має деякі обмеження. Наприклад, вона може стикатись зі складністю у розпізнаванні мовлення з деякими акцентами або діалектами. Крім того, HMM потребує достатньої кількості тренувальних даних для побудови ефективної моделі.

Gaussian Mixture Models (GMM) – це статистична модель, що використовується для аналізу даних, які складаються зі змішання декількох гаусівських розподілів. У контексті автоматичного мовленнєвого розпізнавання, GMM використовуються для моделювання гаусівських розподілів звуків, що входять до мовленнєвого сигналу. GMM складається з декількох компонентів, які моделюються гаусівськими розподілами. Кожна компонента містить відомості про

середнє значення та дисперсію, що дозволяє визначити, наскільки далеко відхиляється звук від середнього значення. Кількість компонентів в GMM зазвичай визначається експериментальним шляхом або залежить від кількості звуків у мовленнєвому сигналі. Однією з основних переваг GMM є їхній високий рівень точності у розпізнаванні мовлення. GMM може працювати зі слабкими сигналами та шумом, що робить його ідеальним для застосування в області автоматичного мовленнєвого розпізнавання. Аналогічно до проблем пов'язаних з HMM, GMM погано справляється з діалектами та специфічними акцентами. Тож для вирішення цієї проблеми потребує великої кількості різноманітний даних, які б могли покриту широкий спектр варіацій мовлення [9].

Support Vector Machines (SVM) – це метод машинного навчання, що використовується для класифікації та регресії. SVM зазвичай використовують для бінарної класифікації, тобто для розділення даних на дві категорії. Однак, за допомогою розширення SVM можна використовувати для багатокласової класифікації. Принцип роботи SVM полягає у знаходженні гіперплощини, яка найкраще розділяє дані на дві категорії [10]. На рисунку 1 продемонстровано приклад розділення елементів на дві категорії. Гіперплощина визначається таким чином, щоб максимізувати відстань між гіперплощиною та ближніми точками даних двох класів. Це називається максимізацією ширини гіперплощини.

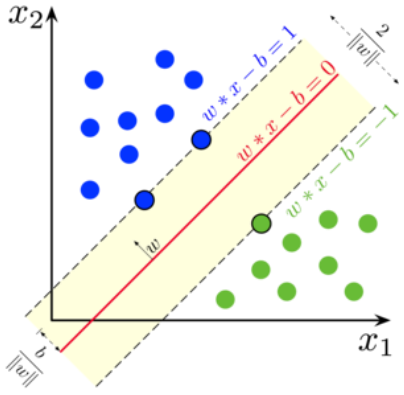

Рисунок 1 – SVM приклад бінарного розподілу

Якщо дані не можуть бути розділені за допомогою лінійної гіперплощини, SVM використовує трюк ядра, щоб перетворити дані в більш високорозмірний

простір, де вони можуть бути розділені за допомогою гіперплощини. Ядро може бути побудоване на основі різних функцій, наприклад, лінійних, поліноміальних, радіальних базисних функцій (RBF) та ін. Приклад розподілення даних у більш високомірному просторі наведений на рисунку 2 [11].

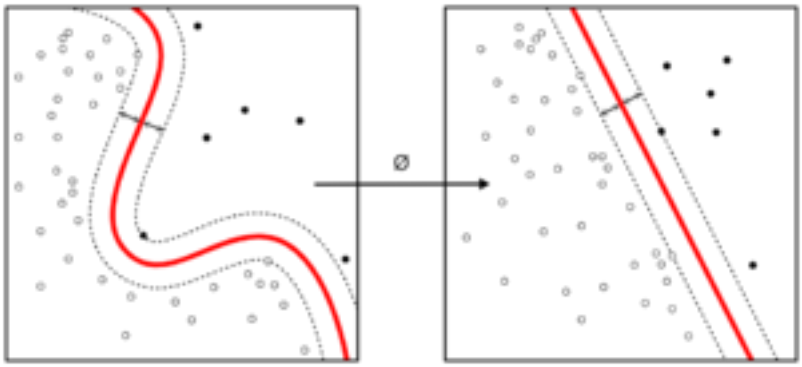

Рисунок 2 – Приклад використання RBF для розділення даних

SVM є ефективним методом класифікації та регресії, який використовується в різних галузях, включаючи обробку сигналів, комп'ютерне зорове сприйняття та розпізнавання мовлення. В контексті транскрибування аудіо файлів, SVM може використовуватися для класифікації різних звуків, таких як різні голоси, звуки фону та інші. Наприклад, SVM може бути використаний для автоматичного виявлення мовлення у фоновому шумі або розпізнавання окремих слів у мовленні.

Ручне транскрибування є одним з методів транскрибування голосових повідомлень, який полягає в тому, що людина слухає аудіозапис голосового повідомлення і відтворює його текстовий варіант.

Процес ручного транскрибування може включати такі етапи, як:

– підготовка – перед транскрибуванням потрібно підготуватися, зокрема, встановити якість звуку, налаштувати гучність або навушники, щоб краще чути голосове повідомлення;
– слухання – слухач прослуховує голосове повідомлення та записує його у текстовому вигляді;
– транскрибування – процес переведення голосового запису в текстовий формат;

– редагування – після транскрибування текст можна переглянути та виправити помилки.

Ручне транскрибування може бути корисним в таких випадках, як:

– голосове повідомлення містить специфічну термінологію, яку складно розпізнати програмними засобами;
– голосове повідомлення містить шуми або інші перешкоди, які ускладнюють розпізнавання мови програмним засобом;
– голосове повідомлення містить дуже особисту або конфіденційну інформацію, яку потрібно зберігати у таємниці.

Однак ручне транскрибування може займати багато часу та потребувати від людини високої концентрації та уваги. Крім того, цей метод може бути дорожчим, оскільки він потребує оплати людської праці. Ручне транскрибування має свої переваги та недоліки. Переваги даного методу:

Переваги ручного транскрибування голосових повідомлень:

– більш точний результат – людина здатна розрізняти нюанси мови, тону, інтонації та інших деталей мовлення, що може призвести до більш точного результату;
– гнучкість – людина може пристосуватися до різних видів мовлення, включаючи акценти, діалекти, швидкість мовлення та інші фактори, що можуть ускладнити автоматичне розпізнавання мови;
– незалежність від технологій – ручне транскрибування не потребує використання складних алгоритмів та комп'ютерної техніки, тому цей метод може бути доступним для використання в різних ситуаціях.

Недоліки ручного транскрибування голосових повідомлень:

– час – процес ручного транскрибування може зайняти багато часу, залежно від довжини голосового повідомлення та складності його мовлення;
– вартість – ручне транскрибування може бути дорожчим, оскільки він потребує оплати людської праці;

- суб'єктивність – ручне транскрибування може бути підв'язане до особистих думок та переконань людини, що може призвести до неточностей у транскрибуванні;
- помилки – люди можуть допускати помилки, особливо якщо голосове повідомлення містить незвичну термінологію або складне мовлення.

Отже, ручне транскрибування може бути корисним у деяких випадках, зокрема, коли потрібно більш точний результат, або коли мовлення має складні деталі. Але цей метод має свої недоліки, такі як часові та вартісні витрати.

Комбіноване транскрибування – це метод транскрибування голосових повідомлень, який поєднує в собі ручне та автоматичне транскрибування. Ідея полягає в тому, що спочатку автоматична система транскрибування виконує перетворення голосового запису в текст, а потім людина редагує і виправляє цей текст, досягаючи кращої якості транскрибування. Його використовують там, де точність автоматичної транскрипції недостатня, але при цьому повністю ручне транскрибування займає багато часу і ресурсів. Крім того, цей метод може використовуватись для швидкого попереднього огляду великої кількості голосових повідомлень, які потребують транскрибування, і визначення тих, які потребують додаткової уваги та ручного редагування.

Однією з переваг комбінованого транскрибування є те, що цей метод може бути більш ефективним за ручне транскрибування, але при цьому забезпечувати більш високу точність транскрипції, ніж автоматична система. Крім того, цей метод може бути більш економічним та швидким, ніж повністю ручне транскрибування.

Недоліками комбінованого транскрибування можуть бути високі вимоги до якості автоматичної системи транскрибування, яка має бути достатньо точною, щоб забезпечити якісне початкове транскрибування. Крім того, від редактора потрібно досить великої кваліфікації та досвіду для того, щоб виправляти транскрипції, забезпечуючи їх точність та стандартизацію. Тому використання комбінованого транскрибування може займати більше часу та коштувати більше, ніж інші методи, але може забезпечити більш точні результати.

Комбіноване транскрибування може бути особливо корисним у випадках, коли точність транскрибування має велике значення, наприклад у медичних або юридичних випадках, або в дослідженнях, де потрібна висока точність даних.

Загалом, цей метод є компромісом між точністю та ефективністю, що може бути корисним в багатьох сценаріях, де важливість точності даних компенсується більшою кількістю ресурсів, витрачених на транскрибування.

1.2 Аналіз методів попереднього оброблення голосових повідомлень

Методи попередньої обробки голосу використовуються для відновлення та покращення якості аудіо- та мовленнєвих сигналів. Деякі з основних методів обробки голосу включають фільтрацію шуму, енергетичний аналіз, часові та частотні аналізи, визначення параметрів голосу, таких як інтонація, тональність та гучність. Основна мета методів обробки голосу – це перетворити голосові сигнали на структуровані дані, що можуть бути використані для подальшого аналізу та інтерпретації. Основними методами попереднього оброблення голосових повідомлень є:

- фільтрація шуму;
- енергетичний аналіз;
- визначення параметрів голосу;
- зменшення обсягу даних;
- гібридне транскрибування.

Фільтрація шуму – це процес вилучення шуму з аудіо сигналу, який може виникнути під час записування голосу. Шум може бути спричинений різними факторами, такими як електромагнітні сигнали, перешкоди від інших пристроїв, акустичні ефекти тощо. Шум може спотворювати голосовий сигнал і знижувати якість запису.

Існують різні методи фільтрації шуму, такі як часова фільтрація, частотна фільтрація та фільтрація в хвильовому просторі. Основна мета фільтрації шуму

полягає у тому, щоб зменшити відхилення голосового сигналу від ідеального, без шуму. Існують різні методи фільтрації шуму, а саме:

– часова фільтрація – цей метод використовується для видалення адитивного білого гаусівського шуму з сигналу; зазвичай цей метод використовується для фільтрації мовлення або звукових ефектів, часова фільтрація використовує вейвлет-перетворення для виділення шуму та його фільтрування [12];
– частотна фільтрація – цей метод використовується для видалення шуму з певних діапазонів частот у спектрі сигналу; це може бути корисним, наприклад, якщо у сигналі присутній шум з високою частотою, такий як електричний шум від приладів [13];
– фільтрація в хвильовому просторі – цей метод використовується для видалення шуму з багатоканальних сигналів, наприклад, з мікрофонів або електроенцефалографічних сигналів; фільтрація в хвильовому просторі може використовувати просторову інформацію, щоб виокремити шумові джерела від корисного сигналу [14].

Основна мета фільтрації шуму полягає у тому, щоб зменшити відхилення голосового сигналу від ідеального, без шуму. Шум може бути викликаний різними факторами, такими як електронні пристрої, атмосферні умови, випадкові звуки та багато іншого. Фільтрація шуму може бути виконана з використанням різних методів, включаючи цифрову обробку сигналів та акустичну обробку. Цифрові методи можуть використовувати різні фільтри, які допомагають видалити шум з голосового сигналу. Акустичні методи можуть використовувати матеріали з поглинанням шуму, такі як спеціальні тканини або піни. Основними типами фільтрів для фільтрації шуму є фільтри нижніх частот, фільтри верхніх частот та фільтри кутових частот. Фільтри нижніх частот дозволяють пропускати тільки низькочастотні складові сигналу, фільтри верхніх частот - високочастотні складові, а фільтри кутових частот – ті, які перебувають у певному діапазоні частот. Однак фільтрація шуму не завжди може забезпечити повністю чистий звук, тому часто використовують комбінацію різних методів обробки голосу для досягнення кращих

результатів. Фільтрація шуму використовується в багатьох галузях, включаючи телефонію, аудіо- та відеопродукцію, спостереження за безпекою та багато іншого. Фільтрація шуму дозволяє покращити якість звуку та розуміння мови, що є важливим для багатьох застосувань обробки голосу.

Енергетичний аналіз є одним з основних методів обробки голосу. Цей метод використовується для визначення рівня енергії звукового сигналу в певному проміжку часу. Для здійснення енергетичного аналізу звукового сигналу, спочатку сигнал розбивається на невеликі часові інтервали, наприклад, у кожній мілісекунді. Далі, в кожному інтервалі обчислюється енергія сигналу, що знаходиться в цьому інтервалі. Для цього кожен зразок сигналу підноситься до квадрату, після чого всі квадрати сумуються, щоб отримати загальну енергію за даний інтервал. За результатами енергетичного аналізу можна визначити, на яких ділянках звукового сигналу є значна енергія, наприклад, коли людина говорить, а також визначити частотні характеристики звукового сигналу. Таким чином, енергетичний аналіз є корисним методом для визначення ділянок звукового сигналу, де присутній звук мовлення, а також для інших задач обробки голосу.

Визначення параметрів голосу включає в себе вимірювання та аналіз різних аспектів мовлення, таких як інтонація, тональність, гучність та інші. Ці параметри можуть бути використані для розпізнавання мовлення, синтезу мовлення, аналізу емоцій тощо. Основні параметри голосу:

- інтонація – це підйом та спад голосу в тоні та ритмі мовлення; інтонація може вказувати на емоційний стан та наміри мовця;
- тональність – це висота голосу; тональність може вказувати на стать, вік та емоційний стан мовця;
- гучність – це сила та інтенсивність звуку мовлення; гучність може вказувати на статус, емоційний стан та настрій мовця;
- тембр – це якість голосу, що залежить від форми та розміру голосових зв'язок; тембр може вказувати на стать, вік та фізичні характеристики мовця;

- ритм – це пульсація мовлення, що визначається швидкістю та паузами між словами та фразами; ритм може вказувати на емоційний стан та настрій мовця;
- темп – це швидкість мовлення, що визначається кількістю слів на одну хвилину; темп може вказувати на настрій та емоційний стан мовця;
- артикуляція – це якість та чіткість вимови звуків та слів; артикуляція може вказувати на мовленнєві недоліки та фізичні проблеми мовлення [15].

Для вимірювання та аналізу параметрів голосу використовуються різні методи, такі як спектральний аналіз, аналіз формантів, аналіз повітряного потоку та інші. Спектральний аналіз полягає у розкладі звукового сигналу на окремі частотні компоненти за допомогою дискретного перетворення Фур'є (DFT) або швидкого перетворення Фур'є (FFT) [16]. Цей аналіз дає інформацію про розподіл енергії звукового сигналу за частотою. Аналіз формантів використовується для визначення формантних частот голосу, які відповідають резонансним частотам порожнини рота та горла. Ці частоти визначають форму звукових гармонік та дозволяють встановити, наприклад, який саме звук вимовляється. Аналіз повітряного потоку використовується для вимірювання інтенсивності повітряного потоку, що проходить через голосові зв'язки при вимовлені звуків. Цей аналіз може допомогти визначити, наприклад, чи є порушення в роботі голосових зв'язок.

Також для обробки голосу використовуються методи зменшення обсягу даних, що дозволяє пришвидшити процес обробки голосу. Компресія даних – це процес зменшення обсягу даних за допомогою різних методів. Основна мета компресії даних полягає в ефективному збереженні, передачі та обробці даних за допомогою меншого обсягу ресурсів, таких як пропускна здатність мережі, дисковий простір та пам'ять. Існує два види компресії даних: втратна та безвтратна. У втратній компресії даних певна кількість інформації втрачається, щоб зменшити розмір даних. Цей метод зазвичай використовується для зменшення обсягу зображень та аудіо- та відеофайлів. Основна ідея поточних методів полягає в видаленні з аудіоданих частини інформації, яку людське вухо не може розрізнити. На рисунку 3 показано принцип кодування MP3, який використовує відсікання

високочастотних складових, а також використання психоакустичної моделі для визначення того, які частини звуку можуть бути видалені без помітної втрати якості [17].

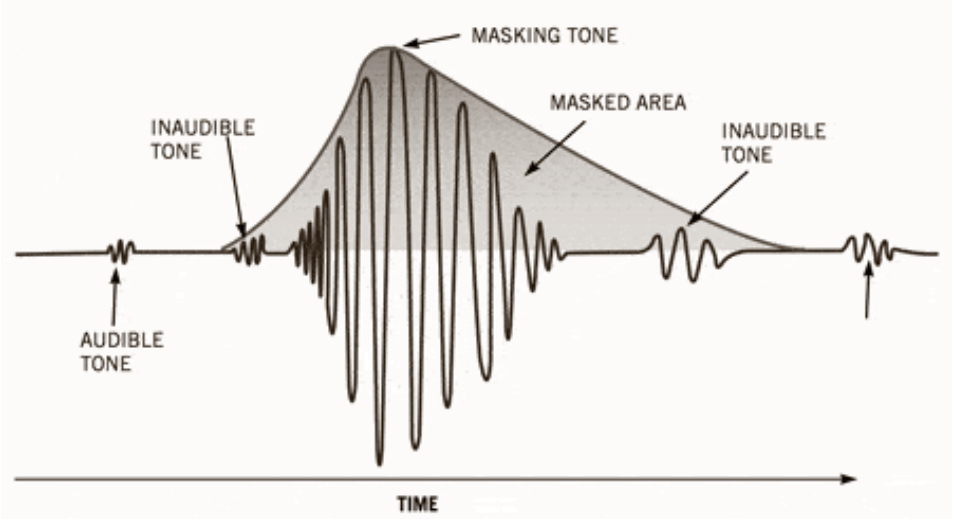

Рисунок 3 – Механізм часткового чи повного відкидання звуку в mp3 файлах [18]

У безвтратній компресії даних всі дані зберігаються точно, але застосовується алгоритм для зменшення обсягу даних, наприклад, кодування Хаффмена. Розуміння методів компресії даних є важливим для розробки ефективних алгоритмів передачі, обробки та збереження даних. Основні методи обробки голосу можуть використовуватися окремо або в по єднанні для досягнення більш точної інформації про голосовий сигнал. Наприклад, фільтрація шуму може бути застосована до аудіо запису перед тим, як він буде підданий аналізу енергії, щоб забезпечити точніші результати. Аналіз інтонації та інші параметри голосу можуть використовуватися разом з алгоритмами розпізнавання мови для покращення точності розпізнавання. Крім того, методи обробки голосу можуть бути використані в різних застосуваннях, таких як відновлення старих записів, розпізнавання емоцій, аналіз мовлення та багато іншого. Застосування цих методів може значно поліпшити розуміння та аналіз голосового сигналу в різних сферах, включаючи мовленнєве розпізнавання, музику та звукову техніку.

Наступним методом є гібридне транскрибування. Гібридне транскрибування - це метод транскрибування голосових повідомлень, який поєднує в собі ручне транскрибування та автоматичне розпізнавання мовлення [19].

У гібридному підході, спочатку використовується ASR для перетворення аудіофайлу на текст. Потім, отриманий текст перевіряється та виправляється редактором, який перевіряє, коригує та доповнює автоматично транскрибований текст за потреби. Цей підхід поєднує переваги обох методів - швидкість та ефективність автоматичного розпізнавання мовлення з точністю та розумінням людської мови, що дає можливість отримати якісний результат транскрибування.

Гібридне транскрибування може бути особливо корисним в ситуаціях, коли необхідно швидко та ефективно транскрибувати велику кількість голосових повідомлень, а також у випадках, коли ASR не може повністю розпізнати мовлення через акцент, діалект або складність мовлення. Однак, недоліками гібридного підходу є витрати на оплату редакторів та можливість помилок під час редагування тексту, що може знизити точність транскрибування.

У загальному, гібридне транскрибування може бути ефективним методом транскрибування голосових повідомлень, який поєднує переваги ручного транскрибування та автоматичного розпізнавання мовлення, зменшуючи недоліки обох підходів.

1.3 Огляд технології глибинного навчання та її використання у задачах оброблення мовленнєвої інформації

В останні роки технології глибинного навчання зазнали значного розвитку та знаходять все більше застосувань у різних галузях науки і техніки, в тому числі й у задачах оброблення мовленнєвої інформації. Великий обсяг даних, який сьогодні доступний, дозволяє створювати все більш складні та точні моделі, які здатні автоматично виконувати завдання з аналізу мовленнєвих даних, таких як розпізнавання мови, генерація мовленнєвих відповідей, транскрибування голосових повідомлень, сентимент-аналіз та багато інших. У даному огляді ми

розглянемо основні підходи до глибинного навчання, а також розглянемо приклади використання цих технологій у задачах оброблення мовленнєвої інформації.

Технології глибинного навчання – це нейромережеві архітектури, що використовуються для здійснення автоматичного вивчення рівнів абстракції в даних, що представлені в великій кількості [20]. Глибинне навчання може бути застосоване в задачах оброблення мовленнєвої інформації, таких як розпізнавання мови, машинний переклад, генерація мовлення, синтез мовлення, розпізнавання мовчання та інших.

У задачах оброблення мовленнєвої інформації, таких як розпізнавання мовленнєвих команд, розуміння мовленнєвого контексту, голосове керування та транскрибування голосових повідомлень, глибинне навчання може допомогти досягти кращих результатів порівняно з традиційними методами.

Використання глибинного навчання у задачах оброблення мовленнєвої інформації дійсно дозволяє досягати кращих результатів порівняно з традиційними методами. Це зокрема стосується таких задач як розпізнавання мови, синтез мовлення, машинний переклад, аналіз тональності, визначення емоцій тощо. Глибинне навчання дозволяє автоматично вивчати характеристики даних, що є важливим для задач оброблення мовленнєвої інформації, де дані мають складну структуру та можуть бути зв'язані з великою кількістю різних факторів. Наприклад, у задачі розпізнавання мови глибинне навчання дозволяє автоматично вивчати характеристики голосових сигналів, які залежать від голосу, акценту, мови та інших факторів, а також використовувати цю інформацію для покращення якості розпізнавання.

Штучний інтелект (AI) є однією з найбільш швидко розвиваючихся галузей сучасної науки та технологій. Ця технологія знаходить застосування у багатьох сферах життя, включаючи мовленнєвий аналіз [21]. Для ефективної роботи зі звуковими даними, такими як мовлення, було розроблено різні технології, такі як глибокі нейронні мережі, машинне навчання та аналіз даних. Використання цих технологій дозволяє отримати точні результати при аналізі мовлення, що має

важливе значення у таких галузях, як розпізнавання голосу, машинний переклад, аналіз тональності та багато інших.

Глибокі нейронні мережі (DNN) є однією з ключових технологій штучного інтелекту. Вони базуються на математичних алгоритмах, які дозволяють моделювати складні залежності між вхідними і вихідними даними. Глибокі нейронні мережі складаються з багатьох шарів, кожен з яких містить велику кількість нейронів. Завдяки цьому, вони можуть ефективно працювати з великою кількістю даних і виконувати складні завдання, такі як розпізнавання образів, голосів, мовлення та інших.

DNN – це клас нейронних мереж, який використовується для вирішення задач машинного навчання, зокрема задачі розпізнавання мовлення. DNN зазвичай складається з багатьох шарів нейронів, що дозволяє моделі здійснювати більш складні перетворення даних [22].

Для вирішення задачі транскрибування голосових повідомлень DNN використовується у складі систем автоматичного розпізнавання мовлення. Зазвичай система складається з декількох компонентів:

- попередньої обробки сигналу – фільтрації шуму та підготовки сигналу для подальшого аналізу;
- витягування ознак – вибір найбільш інформативних частин сигналу для подальшого аналізу, наприклад, мел-кепстральні коефіцієнти (MFCC);
- моделювання мовлення – створення моделі мовлення на основі отриманих ознак та навчання її на великому корпусі тексту та відповідних аудіозаписів;
- декодування – відповідна обробка сигналу та порівняння зі створеною моделлю мовлення для визначення найбільш ймовірної послідовності слів або фраз.

Такі системи мають високу точність та можуть бути успішно використані в різних сферах, де потрібна автоматична транскрипція голосових повідомлень, наприклад, в call-центрах, медіа-компаніях, розвідці тощо.

Основна ідея глибоких нейронних мереж полягає у тому, що кожен шар мережі розпізнає певні абстрактні ознаки вхідних даних і передає їх наступному шару, де вони комбінуються для отримання більш складних ознак. Перший шар глибокої нейронної мережі отримує вхідні дані, які можуть бути зображеннями, звуком або текстом. Далі дані проходять через послідовні шари, де вони обробляються з використанням ваг, які підлаштовуються в процесі навчання мережі. Кінцевий шар нейронів вирішує задачу, яку вона розроблялась для виконання, наприклад, класифікацію зображень або розпізнавання мовлення. На рисунку 4 продемонстрований типовий вигляд DNN, які наочно демонструє розмежування між вхідний шаром, прихованими шарами та вихідним шаром, які виконують вище описані функції [23]. Таким чином, глибокі нейронні мережі можуть розпізнавати складні шаблони в даних, що робить їх корисними для багатьох застосувань, включаючи комп'ютерне зорове сприйняття, обробку мовлення та інші.

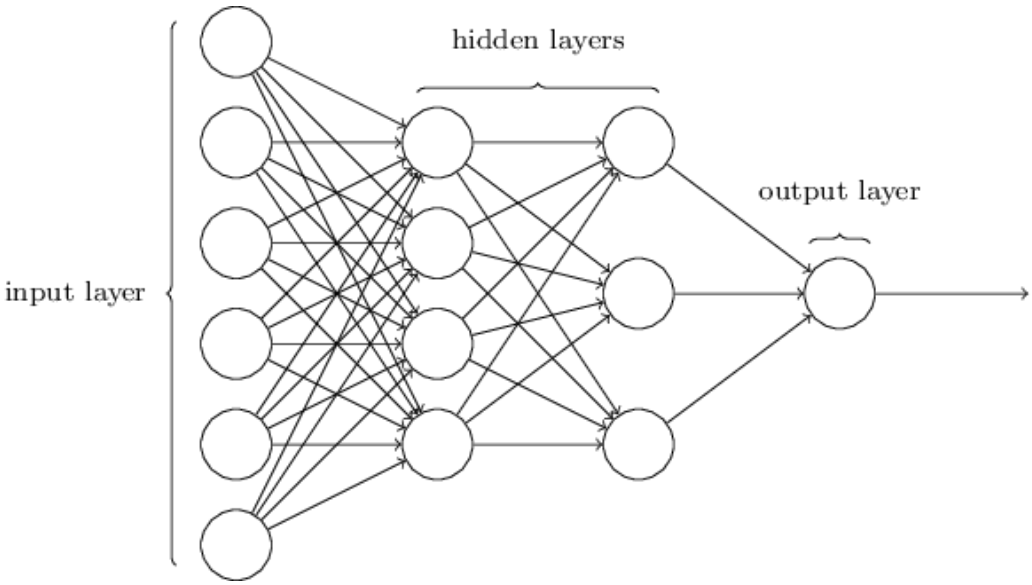

Рисунок 4 – Схематичне зображення DNN [24]

Для навчання глибоких нейронних мереж використовується метод машинного навчання, який передбачає подачу великої кількості прикладів даних і відповідей на них, після чого модель самостійно визначає залежності між вхідними і вихідними даними. Для цього використовуються спеціальні алгоритми

машинного навчання, такі як зворотне поширення помилок (backpropagation), який використовують для коригування вагів між нейронами. Однією з ключових переваг глибоких нейронних мереж є їх здатність до адаптації до нових даних, що робить їх корисними для застосувань в реальному часі, таких як розпізнавання голосу та обробка мовлення. Розробка та навчання глибоких нейронних мереж потребує великої кількості даних та великих обчислювальних ресурсів. Однак, коли навчання проведено і мережа готова, вона може швидко та ефективно обробляти нові дані, що робить її корисною в багатьох галузях, включаючи комп'ютерне зору, машинний переклад та голосовий асистент.

У випадку з транскрибуванням голосових повідомлень, велика кількість звукових записів, що зібрані з мікрофонів, можуть бути використані для тренування глибоких нейронних мереж. Навчання мережі здійснюється на основі пар звукових файлів та їх відповідних транскрипцій у текстовий формат. Мережа потім може бути використана для автоматичного транскрибування нових голосових повідомлень.

LSTM (Long Short-Term Memory) - це вид рекурентної нейронної мережі, яка дозволяє моделювати залежності в часових послідовностях з великим інтервалом між подіями. Ці мережі можуть бути використані для різноманітних завдань, таких як розпізнавання мови, машинний переклад, генерація тексту та інші.

Однією з основних переваг LSTM мереж є їх здатність до запам'ятовування довгострокових залежностей у послідовності. У звичайних рекурентних мережах, які використовуються для моделювання послідовностей, такі як послідовності слів у реченні, градієнт може дещо зникає при проходженні через кілька шарів мережі, що робить неможливим зберігання довгострокових залежностей. LSTM мережі вирішують цю проблему за допомогою використання спеціальних блоків пам'яті, які дозволяють зберігати та використовувати інформацію на довгі періоди часу.

У розпізнаванні мови, LSTM мережі можуть бути використані для моделювання залежностей між фонемами та словами у вимові, що дозволяє покращити точність розпізнавання мови.

Машинне навчання (machine learning) – це галузь штучного інтелекту, що досліджує розробку алгоритмів та моделей, які дають змогу комп'ютеру самостійно навчатися з даних і виконувати завдання без явного програмування. У машинному навчанні використовуються різноманітні алгоритми та моделі, такі як нейронні мережі, дерева рішень, метод опорних векторів та інші. Ці алгоритми навчаються на прикладах, що містять вхідні дані та правильні відповіді, і використовують отримані знання для прогнозування відповідей на нових даних. Машинне навчання застосовується в багатьох галузях, включаючи медицину, фінанси, автомобільну промисловість, рекламу та багато інших. Завдяки машинному навчанню комп'ютери здатні розв'язувати складні завдання, що раніше вважалися можливими тільки для людей, та зробити роботу більш ефективною та точною. Машинне навчання можна класифікувати на три типи:

– навчання з вчителем (supervised learning) – алгоритми навчання, які використовують набір прикладів, які містять вхідні дані та відповіді, для навчання моделі; після навчання модель може використовуватися для прогнозування відповідей на нових даних;
– навчання без вчителя (unsupervised learning) – алгоритми навчання, які працюють з набором даних без відповідей; вони використовуються для виявлення певних закономірностей та шаблонів у даних;
– навчання з підкріплення (reinforcement learning) – алгоритми навчання, які працюють за принципом "винагорода-покарання" (reward-penalty); основна ідея полягає в тому, щоб навчити агента приймати рішення у визначеній ситуації таким чином, щоб отримувати максимальну винагороду (reward) і уникати покарань (penalty), агент отримує інформацію про поточний стан (state) системи та виконує певну дію (action), після кожної дії, агент отримує винагороду або покарання, з часом агент навчається змінювати свої дії залежно від того, які наслідки вони мають; метою є максимізація очікуваної винагороди за допомогою оптимальних дій [25].

Аналіз даних – це процес виявлення корисної інформації з набору даних. В сучасному світі ми маємо доступ до величезної кількості даних, які можуть бути використані для отримання нових знань та розуміння різних явищ. В рамках мовленнєвого аналізу аналіз даних використовується для виявлення корисних зв'язків між різними аспектами мовлення, такими як інтонація, темп мовлення та емоційний стан особи. Для аналізу мовленнєвих даних використовуються різні інструменти та методи. Наприклад, статистичний аналіз може допомогти виявити зв'язки між певними аспектами мовлення та ефективністю комунікації, а методи машинного навчання можуть допомогти виявити складні зв'язки та шаблони, які не завжди очевидні. Важливим етапом аналізу даних є підготовка даних, яка може включати очищення даних від помилок та видалення зайвих або некорисних даних. Після підготовки даних використовуються різні методи, такі як кластерний аналіз, аналіз часових рядів, дискримінантний аналіз та інші, для виявлення корисної інформації та побудови моделей, які можуть допомогти в прогнозуванні майбутніх явищ. Аналіз даних є невід'ємною частиною різних галузей, включаючи мовленнєвий аналіз, медичну діагностику, бізнес-аналіз та багато інших. В сучасному світі, де дані грають все більш важливу роль, аналіз даних стає все більш потрібним та цінним.

Усі ці технології, такі як глибокі нейронні мережі, машинне навчання та аналіз даних, є складовими штучного інтелекту. Використовуючи ці технології, вдалося створити системи, які можуть аналізувати та розуміти людське мовлення, розпізнавати голос та здійснювати автоматичний переклад мов. Застосування штучного інтелекту в мовленнєвому аналізі може мати значний вплив на багато сфер, включаючи медицину, банківську справу, правоохоронну діяльність та освіту. Наприклад, системи автоматичного перекладу мов можуть полегшити комунікацію між людьми, які розмовляють на різних мовах, а системи розпізнавання голосу можуть виявити проблеми зі здоров'ям голосових зв'язок раніше, ніж це зможуть зробити люди. Однак, варто зазначити, що штучний інтелект не є універсальним рішенням для всіх проблем, пов'язаних з мовленнєвим аналізом. Деякі аспекти мовлення, такі як контекст та інтенції, можуть бути складні

для аналізу за допомогою AI. Тому, хоча штучний інтелект може бути дуже корисним для розуміння мовлення, він повинен використовуватися у поєднанні з іншими технологіями.

Застосування глибинного навчання в обробленні мовленнєвої інформації може допомогти знизити час та витрати на розробку та підтримку системи, а також підвищити її точність та надійність. В цілому, глибинне навчання є потужним інструментом для розв'язання складних задач у галузі мовленнєвого аналізу та в майбутньому може стати невід'ємною частиною багатьох додатків і технологій, які пов'язані з мовленнєвою інформацією, таких як розпізнавання мовлення, машинний переклад, аналіз настроїв і емоцій, стилізація мовлення та багато інших. Завдяки глибинному навчанню можна автоматично вивчати складні залежності між різними характеристиками даних та покращувати якість результатів в порівнянні з традиційними методами.

1.4 Опис архітектури системи транскрибування голосових повідомлень на основі технології глибинного навчання

Інструментальні засоби грають важливу роль у впровадженні технологій штучного інтелекту, включаючи мовленнєвий аналіз. Інструментальні засоби вирішення задачі транскрибування голосових повідомлень можуть бути різноманітними і залежать від використовуваних методів і технологій. Нижче наведені деякі інструменти, які можуть бути використані для реалізації великої кількості різних завдань мовленнєвого аналізу:

- бібліотеки мов програмування. Python та MATLAB є дуже популярними мовами програмування серед дослідників та розробників, що працюють у галузі штучного інтелекту; у цих мовах є багато бібліотек та модулів для обробки мовленнєвих даних, таких як NLTK та Scikit-learn для Python, а також Signal Processing Toolbox для MATLAB [26];
- пакети для розпізнавання голосу. Google Speech Recognition та CMU Sphinx є прикладами пакетів для розпізнавання голосу, які можуть бути

використані для перетворення аудіоданих на текст; ці пакети можуть бути корисними для автоматичного транскрибування мовлення та аналізу мовленнєвих даних [27];

– фреймворки для машинного навчання. TensorFlow та Keras є прикладами фреймворків для машинного навчання, які можуть бути використані для розробки моделей глибокого навчання для різних задач мовленнєвого аналізу, таких як класифікація звуків мовлення, розпізнавання мовленнєвих емоцій та генерація мовлення [28].

Інші інструменти, такі як бібліотеки для обробки сигналів, такі як librosa або soundfile, можуть бути використані для обробки аудіоданих перед їх подальшим аналізом. Також, пакети для аналізу даних, наприклад, Pandas або NumPy, можуть бути використані для ефективного обробки та маніпулювання даними перед їх використанням у моделях машинного навчання. Крім того, інші фреймворки для машинного навчання, такі як PyTorch, MXNet, та Caffe, можуть бути використані для побудови глибоких нейронних мереж для мовленнєвого аналізу [29]. Вибір конкретного інструменту залежить від потреб та вимог проекту, і він може бути вибраний на основі функціональності, швидкодії, складності використання, наявності документації та спільноти користувачів.

Візуальний інтерфейс системи може знаходитись як за посиланням у браузері, так і бути інстальованим окремим застосунком на пристрій. Однак через безумовну перевагу браузера в незалежності та наявності у будь-якого користувача. Та потенційне відокремлення сервера-обробника на окрему машину, через зазначені вище великі потреби в машинних ресурсах було прийнято рішення розробляти саме веб-застосунок.

Vue.js – це дуже популярний JavaScript фреймворк для розробки веб-інтерфейсів, який пропонує зручний та ефективний підхід до розробки веб-додатків. Ось кілька причин, чому можна обрати саме Vue.js. Достатньо простий в освоєнні фреймворк. Має швидкий віртуальний DOM, що дозволяє створювати ефективні додатки, які мають малий розмір та швидко завантажуються. Vue.js є багатокомпонентним фреймворком, а саме він дозволяє додавати бажані

компоненти до збірки [30]. Відповідно система має широкий спектр стандартних можливостей, які завантажуються тільки при потребі розробника. Крім того, на основі Vue.js існує Quasar Framework, який включає в себе великий набір завчасно зроблених веб-компонентів, які гарантують інтуїтивний процес використання системи, з підтримкою адаптивної роботи на різних пристроях. Саме через що, їх і було обрано.

В широкому сенсі, веб-застосунок приймає файл від користувача, попередньо перевіряючи тип та розмір файлу, на відповідність допустимим, та надсилає його на сервер, шляхом використання POST-запиту.

Маршрутизація запитів буде здійснена за допомогою Ocelot Gateway. Ocelot Gateway – це відкрите програмне забезпечення для управління API-інтерфейсами, яке дозволяє розробникам легко маршрутизувати та керувати вхідними запитами до фонових сервісів [31]. Воно розроблено для роботи з .NET Core та побудовано на основі ASP.NET Core. Таким чином, надісланий файл перенаправляється за допомогою Ocelot на вказаний порт. Це дозволяє автоматично та зручно, не прив’язуючись до конкретного сервера надсилати файл, а потім змінювати шлях, редагуванням одного єдиного поля. Відповідно все «спілкування» компонентів підсистеми відбувається за допомогою описаного вище фреймворку.

Ocelot не тільки підвищує зручність організації зв’язків, але й приховує реальні адреси фізичних серверів, що так чи інакше дозволяє убезпечитись від різного роду недоброзичливих осіб. Також можна використовувати функції контролю обмежень швидкості та обмеження запитів, що не тільки допомагає підлаштовуватися під наявні потужності, так і допомагає запобігти атакам на сервіс та зменшити вплив DDoS-атак.

Для збереження файлів було обране сховище Azure Blob Storage, котре дозволяє зберігати у собі данні у форматі Blob. Blob (Binary Large OBject) – це тип даних, який використовується для зберігання великих об'єктів, таких як зображення, відео, аудіо, документи тощо. Blob зазвичай зберігається в базі даних або в хмарному сховищі Відповідно, для зберігання надісланих файлів, використовується саме Azure Blob Storage, який при створені нової комірки даних

повертає унікальний згенерований Id, за яким надалі йде запит до сервера обробника. Запит до файлового сховища відбувається через налаштований Ocelot на сторінці веб-застосунку, з використанням TypeScript.

Користуючись маршрутизацією Ocelot, веб-застосунок надсилає наступний запит на сервер-обробник, який звертається до сховища файлів, за отриманням конкретного аудіофайлу.

Для безпосереднього процесу транскрибування тексту файли повинні бути приведені до наступного вигляду: аудіо інформація повинна подаватися у вигляді одноканального файлу з розширенням .WAV. Відповідно до цих потреб, у першу чергу потрібно привести отриманий аудіо- чи відеофайл до даного типу. Для цих та інших цілей обробки аудіофайлу чудово підходить сервер на мові програмування Python. Python має чистий та легко зрозумілий синтаксис, що дозволяє швидко розробляти та тестувати серверний код, підтримує багатопоточність, що дозволяє обробляти багато запитів одночасно та зменшує витрати часу на відповідь на запити та й на додачу має багато бібліотек та фреймворків для масштабування. Python має велику кількість модулів, що дозволяє додавати функціональність до сервера, тобто при потребі додавати більше методів обробки звуку. Python підтримує велику кількість платформ, що дозволяє запускати сервер на різних операційних системах, що дозволяє як піднімати збірку в Docker, чим значно полегшує процес інсталювання програмного забезпечення (ПЗ) на різних машинах, або ж дозволяє піднімати сервер на Windows машині, тобто на персональному комп’ютері (ПК).

Pydub – це бібліотека Python для роботи з аудіофайлами, яка дозволяє конвертувати аудіофайл з одного формату в інший, застосовувати фільтри, такі як еквалайзер, підсилювач, нормалізатор та інші. До того ж дозволяє змінювати швидкість файлу та обрізати його на основі часових маток або шаблонів. Разом з технологіями мовленнєвого аналізу та штучного інтелекту, дана бібліотека дозволяє не тільки привести файл до бажаного формату, а й позбутися зайвих проміжків часу та пришвидшити його на певний відсоток, це допоможе зменшити час транскрибування та обсяг файлу, не втрачаючи в якості отриманого тексту.

В залежності від поставлених задач та технічних потужностей системи, попередньо оброблений файл може бути або направлений в сховище, де він може перезаписати дані з попередньо файлу, чи отримати нову комірку для зберігання, або безпосередньо переданий на сервер-обробник, який займається транскрибуванням. Ці тонкощі не впливають на фінальний результат транскрибування, однак дозволяються налаштувати систему під наявні потужності.

Транскрибування підготовлених файлів буде здійснюватися за допомогою натренованої нейронної мережі. Для зручності формування GET-запиту до серверу написаному на Python було обрано до використання технологію Flask, яка дозволяє створювати веб-додатки швидко та ефективно, маючи при цьому просту структуру і розширювані можливості. Flask – це мінімалістичний веб-фреймворк для мови програмування Python. До основних компонентів Flask входять:

- шаблонізатор Jinja2;
- роутер – обробник маршрутів, який визначає, які дані повинні бути відображені на сторінці;
- обробник запитів, який приймає та обробляє HTTP-запити;
- механізм для збереження даних на сервері та управління сесіями.

Через це Flask чудово підходить для створення веб-серверу. Таким чином сервер транскрибування буде побудований на основі Flask, а також нейронної мережі, яку буде використано для транскрибування голосових повідомлень. Після попередньої обробки файла за допомогою Pydub, відбувається звертання ПЗ до сервера з нейронною мережею, яка і виконує фінальне транскрибування завчасно підготовленого аудіофайлу.

Як було вище зазначено, звертання до серверу відбувається через GET-запит, представлений нижче. Як видно, при запиті на сервер, він автоматично звертається до Azure Blob Storage, вказуючи завчасно повернений Id у запит. Таким чином, після виконання транскрибування сервер обробник повертає деяку службову інформацію, наприклад як ім’я файлу, час обробки чи девайс, тобто центральний процесор (CPU) або графічний процесор (GPU). Також повертається й шукана

інформація, а саме транскрибований текст, який приймається та відображається в браузері.

Висновки до розділу 1

У цьому розділі розглянуто існуючі методи транскрибування голосових повідомлень, методи попереднього оброблення голосових повідомлень та технології глибинного навчання у задачах оброблення мовленнєвої інформації.

Як зазначено у розділі, глибинне навчання є однією з найпотужніших технологій штучного інтелекту, що дозволяє автоматично вивчати характеристики даних та здійснювати складні обчислення за допомогою нейронних мереж, використання цієї технології може допомогти досягти кращих результатів порівняно з традиційними методами.

У якості нейронної мережі, яка буде використовуватись транскрибування голосових повідомлень, обрано LSTM, бо мережа цього типу є ефективною для транскрибування голосових повідомлень, оскільки вона може добре працювати з послідовностями даних, такими як звукові сигнали, та враховувати контекст при обробці цих даних. Також LSTM здатна зберігати та використовувати довготривалу залежність між вхідними даними, що є корисним для розпізнавання різних варіантів вимови та акцентів у голосових повідомленнях. Крім того, LSTM може зберігати контекст інформації, що допомагає покращити точність транскрибування та автоматичну обробку мовленнєвих даних.

У цьому розділі проаналізовані основні методи та підходи для розв’язання задачі транскрибування голосових повідомлень, та обрано методи, основні технології та інструментальні засоби реалізації транскрибування голосових повідомлень. До того ж описано основні параметри голосу та обрано напрямок покращення процесу транскрибування аудіофайлів. Основними обраними технологіями для побудови архітектури системи є мова Python з її бібліотеками NLTK, Pydub, Flask та Scikit-learn, Ocelot Gateway в поєднанні з ASP.NET Core, Vue.js, Quasar Framework та Azure Blob Storage.

2 РОЗРОБЛЕННЯ ПІДСИСТЕМИ ТРАНСКРИБУВАННЯ ГОЛОСОВИХ ПОВІДОМЛЕНЬ НА ОСНОВІ ТЕХНОЛОГІЇ ГЛИБИННОГО НАВЧАННЯ

Транскрибування голосових повідомлень на основі технологій глибинного навчання стає все більш актуальним і корисним інструментом в різних сферах діяльності. Це може бути корисно для полегшення комунікації, збільшення продуктивності роботи та забезпечення безпеки в різних областях. Для досягнення цих цілей необхідно мати підсистему транскрибування, яка складається з компонентів, таких як розпізнавання мови, сегментація сигналу, відшумлення та інші. У цьому пункті буде розглянуто опис компонентів підсистеми транскрибування та розробку робочого алгоритму, що дозволить досягти більш високої точності та ефективності транскрибування.

Кожен компонент підсистеми транскрибування виконує важливу роль у процесі перетворення голосового сигналу в текст. Наприклад, розпізнавання мови відповідає за ідентифікацію звуків та формування слів, сегментація сигналу визначає межі окремих фрагментів голосового запису, а відшумлення покращує якість сигналу та дозволяє отримати більш точний результат. Підсистема транскрибування голосових повідомлень повинна бути зручною для користування людям, без профільних знань у цій або дотичних галузях. Відповідно, будь-який користувач, із базовим розумінням процесу оперування комп’ютером повинен мати змогу працювати з продуктом. Таким чином система повинна надавати доступ до графічного інтерфейсу, з якого користувач може завантажити аудіо- чи відеофайл, а застосунок, в свою чергу повинен повернути транскрибований файл. При цьому процес повинен бути максимально можливо оптимізований, за рахунок зменшення об’ємів та часу файлу.

Розробка власної моделі для транскрибування голосових повідомлень на основі технології глибинного навчання може бути досить складною задачею, яка вимагає глибокого розуміння таких областей, як обробка природної мови, акустичне моделювання та машинне навчання. Одним з можливих підходів може бути використання різних архітектур глибинних нейронних мереж, таких як

рекурентні нейронні мережі (RNN), згорткові нейронні мережі (CNN) та трансформери, які були успішно використані в подібних завданнях. Для розробки моделі необхідно також мати наявність відповідних даних для навчання, які містять аудіофайли та відповідні текстові транскрипції. Ці дані можна отримати шляхом створення власної бази даних або використовувати готові набори даних, які доступні відкритою інтернет-спільнотою. Після збору даних необхідно провести попередню обробку даних, яка включає в себе такі етапи, як збирання фонетичних транскрипцій, вирізання сегментів з голосом та попередню підготовку даних для тренування моделі. Далі, можна використовувати різні бібліотеки для глибинного навчання, такі як TensorFlow, PyTorch або Keras, для розробки та навчання моделі на відповідних даних. Важливо також виконати післянавчальний аналіз та налаштування гіперпараметрів моделі для отримання оптимальних результатів. Крім того, важливо забезпечити оптимальну інфраструктуру для використання моделі в режимі реального часу, з достатньо потужними графічними процесорами та достатньою кількістю оперативної пам'яті. Для розгортання моделі можна використати різноманітні сервіси, такі як Amazon Web Services (AWS), Google Cloud або Microsoft Azure. Наприклад, можна використовувати сервіси обробки даних та машинного навчання, Amazon SageMaker або Google Cloud ML Engine, щоб розгорнути модель у контейнерах Docker та забезпечити масштабовану та надійну інфраструктуру для її використання. Або розгорнути модель на власному сервері, використовуючи різноманітні фреймворки для розгортання моделей, такі як TensorFlow Serving або Flask. У цьому випадку необхідно встановити всі необхідні бібліотеки та залежності для моделі, налаштувати веб-сервер, який буде обробляти запити на транскрибування та забезпечити безпеку та захист даних.

2.1 Збір даних та підготовка набору даних для навчання моделі

Збір даних та підготовка набору даних - це важливий етап у розробці будь-якої моделі машинного навчання, включаючи моделі для розпізнавання мови.

Нижче наведено загальну процедуру збору та підготовки даних для навчання моделі розпізнавання мови:

- визначення мети – першим кроком є визначення мети моделі розпізнавання мови; наприклад, це може бути розпізнавання голосових команд, розпізнавання тексту з аудіо або розпізнавання діалогів;
- збір даних – далі необхідно зібрати набір даних, який відповідає визначеній меті; це може бути записи голосу з мікрофону, аудіофайли, відеофайли зі звуковою доріжкою або інші джерела;
- перевірка та очистка даних – після збору даних необхідно перевірити їх на наявність помилок, шуму та інших проблем; наприклад, можна перевірити аудіофайли на наявність ділянок з нечітким звуком, перевірити записи на наявність шуму в тлі, тощо;
- анотування даних – для навчання моделі необхідно позначити правильну мітку або категорію для кожного запису або ділянки запису; наприклад, якщо метою моделі є розпізнавання тексту з аудіо, то необхідно перетворити аудіо в текст та позначити цей текст як правильну мітку для кожного запису;
- розділення даних – після анотування даних необхідно розділити їх на навчальний, тестовий та перевірочний набори; навчальний набір буде використовуватись для навчання моделі, тестовий – для перевірки точності роботи моделі.

Безперечно, етап збору даних є одним з найважливіших та основних етапів розробки будь-якої моделі, включаючи моделі для машинного навчання та обробки природної мови. Якість та точність роботи системи напряму залежить від якості та репрезентативності тренувальних даних, які використовуються для навчання моделі. Для забезпечення якості та репрезентативності тренувальних даних можна використовувати різноманітні варіанти збору даних, зокрема збір даних з різних джерел, створення власних даних за допомогою інструментів для анотування та маркування даних, а також використання публічно доступних даних та наборів даних.

На даний момент існує невелика кількість датасетів українською мовою, які на додачу ще й мають незначну кількість годин аудіо та й сама варіативність даних не покриває широкого спектру наукових термінів, що робить даний варіант тренування малоперспективним. Більшість існуючих датасетів мають обмежену кількість даних, або ж не містять важливих категорій, що необхідні для багатьох задач обробки природних мов. Недостатність даних може призвести до того, що результати роботи будуть недостатньо точними або ж неадекватними. В свою чергу особисте формування навчальних даних для тренування моделей штучного інтелекту є надзвичайно трудомістким і часоємким процесом, що потребує великого обсягу людської праці. Починаючи зі збору і анотування даних до підготовки їх для використання у моделі, кожен етап процесу потребує зайнятості експертів у відповідній галузі та досвідчених фахівців з обробки даних. Збір даних може включати в себе складання корпусу текстових даних, записів аудіо або відео, що вимагає багато ресурсів, часу та зусиль. Після збору даних необхідно їх анотувати, тобто позначити кожен запис за темою, категорією чи метою, щоб модель могла правильно їх інтерпретувати. Анотування може бути підсилено людською експертизою, що вимагає додаткових зусиль та коштів. Після збору та анотування даних потрібно провести попередню обробку, включаючи очищення та структурування даних. Наступним кроком є підготовка даних для використання у моделі, включаючи їх конвертацію в необхідний формат та подальшу обробку. У цілому, формування власних навчальних даних є складним та часоємким процесом, який може забрати багато ресурсів та зусиль.

Оптимальний варіант для формування тренувальних даних залежить від конкретної задачі та доступних ресурсів. Проте, завантаження високоякісних відео з субтитрами з платформи YouTube може бути ефективним методом отримання навчальних даних для розпізнавання мови та інших мовних завдань. YouTube містить велику кількість відео на різні теми, що може бути корисним для формування тренувальних даних. Більшість відео на YouTube мають субтитри, що дозволяє отримати текстові дані разом з відео. Ці дані можуть бути використані для тренування моделі для розпізнавання мови, покращення машинного перекладу та

інших завдань, пов'язаних з мовою. Однак, важливо пам'ятати, що якість даних може варіюватись на різних відео та можуть бути помилки в субтитрах. Тому, для отримання високоякісних тренувальних даних, необхідно використовувати тільки відео з якісними субтитрами та проводити додаткову обробку даних. Використання подкастів, у якості джерел тренувальних даних, забезпечує легкий доступ до контенту, але і дозволяє глибше підійти до аналізу різних наукових тем. Більшість подкастів мають якісний звук та адекватні субтитри, що робить їх чудовим джерелом тренувальних даних для мовних моделей. Особливо це стосується тих подкастів, які спеціалізуються на наукових темах, оскільки вони включають у себе багато термінології та технічної лексики, що може збагатити словниковий запас моделі. Таким чином, використання високоякісних подкастів як джерела тренувальних даних може значно покращити якість мовних моделей, особливо в тих галузях, де використовуються спеціалізовані терміни та технічна лексика.

Проте, не менш важливим етапом є підготовка та обробка даних перед їх використанням для навчання моделі. Цей етап може включати в собі такі процеси, як очищення та обрізання даних, видалення шуму та збалансування даних. Ці процеси допоможуть забезпечити репрезентативність та якість даних для навчання моделі. Таким чином, для повноцінного отримання відеоданих з субтитрами потрібно наступне:

- отримати посилання на відео;
- завантажити відео та субтитри в BLOB-storage;
- конвертувати відео у аудіофайл, для зменшення загальних обсягів зберігаємої інформації та приведення даних до стандартизованого вигляду;
- вилучення зайвих звуків, базуючись на інформації з субтитрів;
- позначення даних у якості «готових» для тренування.

Безумовно, процес попередньої обробки даних можна здійснювати без збереження проміжних результатів, однак розбиття процесу на декілька паралельних незалежних модулів дозволяє більш повно та ефективно використовувати наявні потужності, та уникає створення «вузьких місць», що

суттєво сповільнюють загальний час роботи системи. Відповідно до цього, розроблена діаграма компонентів елемента отримання тренувальних даних продемонстрована на рисунку 5.

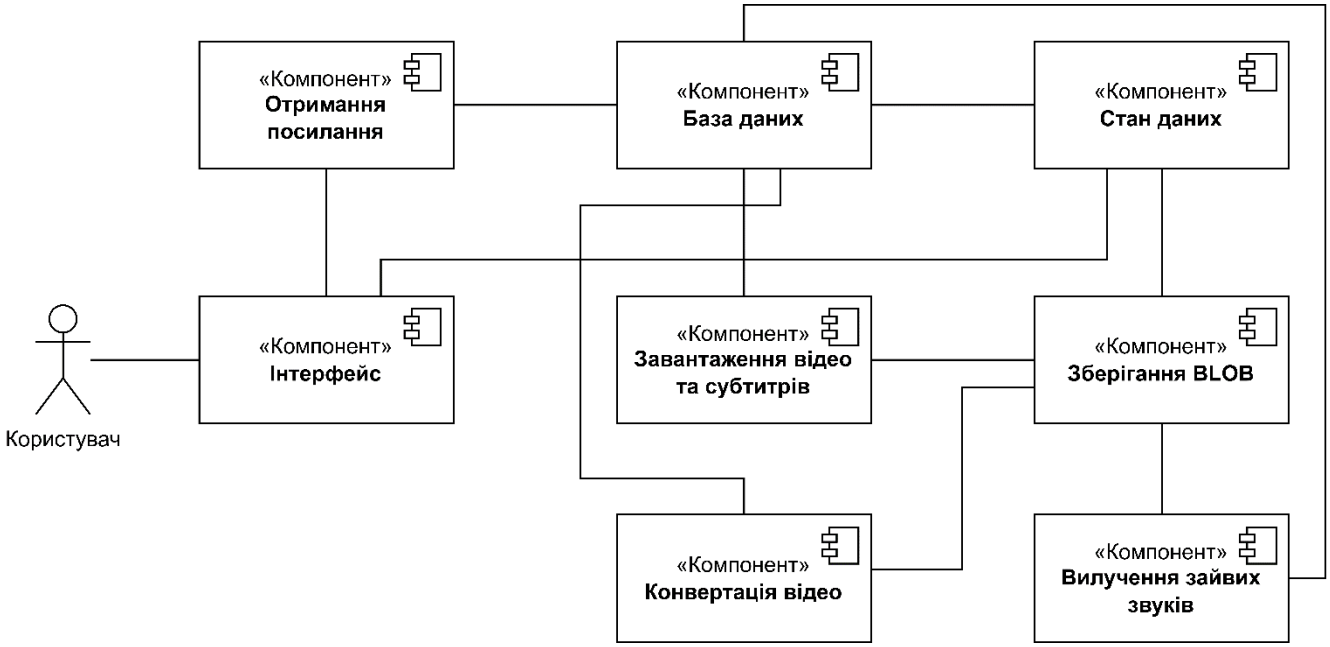

Рисунок 5 – Діаграма компонентів елемента отримання тренувальних даних

За допомогою інтерфейсу користувач може не тільки завантажувати окремі відео з YouTube, але і надавати списки відео для автоматичного завантаження. Це дозволяє користувачам з легкістю знайти відео на певну тему та швидко сформувати власний набір тренувальних даних. Крім того, такий інтерфейс дозволяє збирати тренувальні дані з великої кількості джерел, що робить навчальну модель більш репрезентативною та забезпечує її здатність працювати з різноманітними типами даних. Після чого посилання зберігаються в новостворених комірках БД. Новостворені комірки БД можуть містити різноманітну інформацію, таку як назву відео, опис, автора, тривалість, дату публікації та інші метадані. Збереження такої інформації дозволяє отримувати доступ до додаткової інформації, яка може бути використана для подальшого аналізу даних і покращення якості навчання моделі. Зокрема, можливо використати ці дані для формування різноманітних вибірок, створення тестових наборів даних для перевірки якості моделі, та інших цілей. Наступним кроком, для кожного відео, програма

автоматично завантажує відео та відповідні субтитри до нього в BLOB-storage, що забезпечує швидкий доступ до даних та зберігає їх у безпечному місці. Після завантаження відео та субтитрів, інформація про це також зберігається в базі даних, що дозволяє легко знаходити та використовувати ці дані для подальшого навчання мовної моделі. Після завантаження відеофайлу, він проходить процес конвертації у формат WAV. Отриманий WAV-файл зберігається в BLOB-storage, при цьому в БД вносяться відомості про місцезнаходження файлів у BLOB-storage.

Після того, як відеофайл успішно конвертовано в формат WAV та збережено в BLOB-storage, наступним кроком є вилучення зайвих звуків з аудіофайлу. Цей процес здійснюється за допомогою інформації з субтитрів. Субтитри, що були завантажені разом з відео, конвертуються в текстовий формат і розділяються на окремі рядки. Після цього проводиться аналіз тексту, який міститься у субтитрах, та визначаються зайві звуки, наприклад, шум зовнішнього середовища, паузи у мовленні, а також інші шуми, які не мають відношення до головного змісту відео. Після вилучення зайвих звуків з аудіофайлу, отримуємо «чистий» звук, який буде використовуватися для навчання мовної моделі. Для збереження отриманого аудіофайлу використовується BLOB-storage, при цьому, відповідна інформація зберігаються в БД для подальшої обробки та використання у тренуванні мовної моделі.

Для розробки описаної вище компоненти завантаження відео та субтитрів використано бібліотеку Pytube. Бібліотека дозволяє завантажувати відео в будь-якій якості та форматі, що підтримується платформою. Pytube працює на основі API YouTube та використовує бібліотеку requests для взаємодії з сервером. Бібліотека підтримує завантаження відео в пакетному режимі, що дозволяє ефективно використовувати ресурси мережі та прискорювати процес завантаження. Pytube є відкритою бібліотекою з ліцензією MIT та активно розвивається спільнотою розробників. Наданий програмний код дозволяє завантажувати відеозаписи та відповідні субтитри до них. Для завантаження відео скористалися методом "YouTube()" з бібліотеки Pytube. Цей метод приймає на вхід посилання на відео та дозволяє отримати відеозапис у форматі MP4. Завантаження відповідних субтитрів

до відео було здійснено за допомогою методу "caption_tracks" бібліотеки Pytube. Цей метод дозволяє отримати список усіх доступних субтитрів до відео. Обравши потрібний субтитр за його ідентифікатором, його було завантажено та збережено у форматі SRT.

```
# Завантаження відео
yt = YouTube(video_url)
video_stream = yt.streams.filter(progressive=True,
file_extension='mp4').order_by('resolution').desc().first()
video_file = video_stream.download()

# Завантаження субтитрів
caption = yt.captions.get_by_language_code('uk')
if caption:
    caption_content = caption.xml_captions
else:
    caption_content = ""
```

Окрім завантаження самого відео та субтитрів, важливим є ще й завантажити метадані, які в подальшому можуть бути використані для більш зручного аналізу, розуміння специфічної тематики контенту, пошуку подібних відео, чи навпаки, внесення даного автора до небажаних джерел інформації. Відповідно мінімальний та достатній набір метаданих включає в свій склад наступні змінні:

- посилання на відео;
- назва відео;
- автор відео;
- дата публікації;
- довжина відео;
- опис відео.

Для того, щоб отримати інформацію про відео з використанням бібліотеки Pytube, розроблено наступний програмний код:

```
# Отримання інформації про відео
title = yt.title
date = yt.date
author = yt.author
publish_date = yt.publish_date
description = yt.description
```

Для того, щоб отримати інформацію про довжину відео розроблено наступний програмний код з використанням бібліотеки moviepy. Наявність

інформації про першочергову довжину відео, дозволить порівняти його з обробленим аудіофайлом, що надає додаткових можливостей для аналізу отриманих даних та їх джерела.

```
video = VideoFileClip(video_stream)
duration = video.duration
```

Після того, як отримані метадані з відео збережені в базі даних, генерується унікальний ключ кортежа. Цей ключ буде використаний для зв'язку між записом в базі даних та файлами в BLOB-storage. На рисунку 6 відображено зв’язок між отриманими в ході завантаження та обробки сутностями.

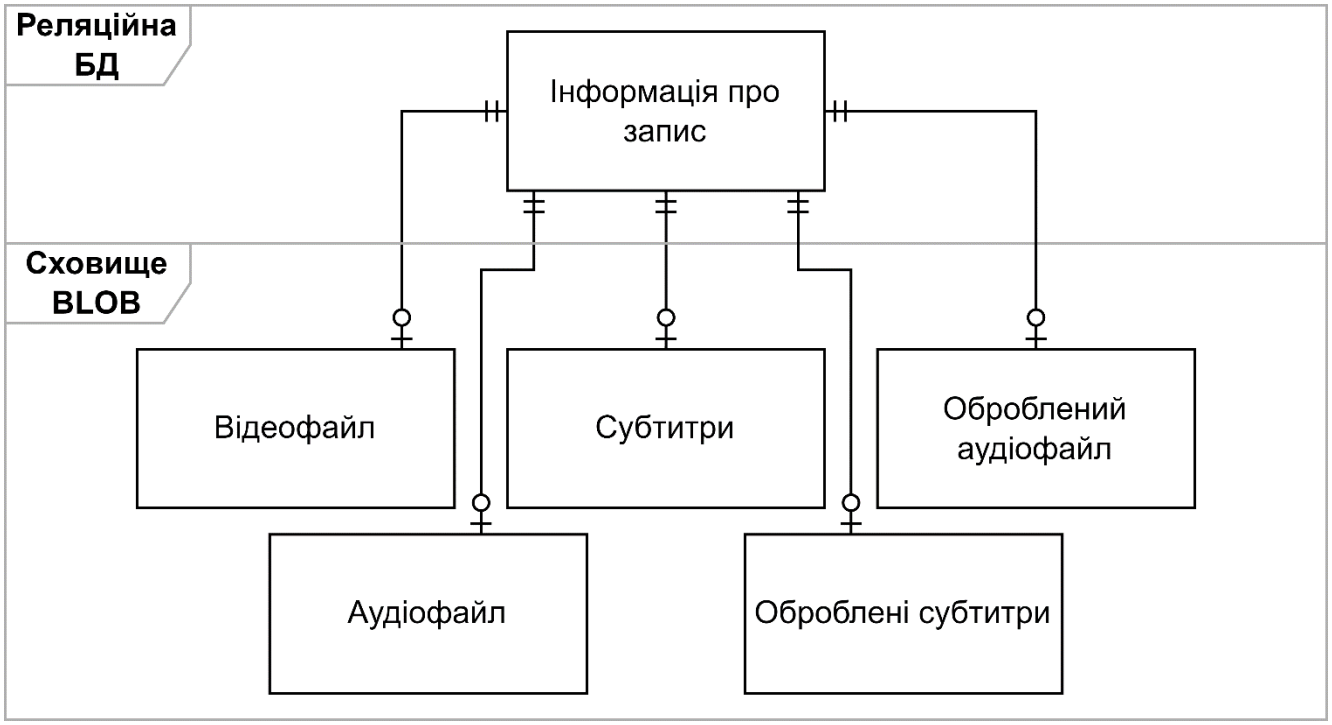

Рисунок 6 – Діаграма зв’язку сутностей даних тренувального набору

Інформація про відеозапис має жорсткий типізований формат, тож дана таблиця знаходиться в реляційній PostgreSQL базі. Однак, Відеофайли, субтитри, чи аудіофайли до та після обробки, знаходяться в сховищі BLOB, кожен в своєму контейнері. Таке рішення дозволяє реалізувати логічний зв’язок між елементами один до одного, що дозволяє відмовитись від розв’язуючих таблиць та пришвидшити загальний час роботи БД й знизити обсяги даних. Відповідно унікальний ідентифікатор формується при створені кортежу з інформацією в

реляційній БД, після чого надається при створені записів в сховище BLOB. Залежно від специфіки та поставлених завдань використання даних, файли з сховища можна видаляти, після планового використання. Така модель поведінки, дозволяє очистити дисковий простір від зайвих даних, завдяки збереженню метаданих та показників якості використаної інформації. На рисунку 7 зображено модель сутності-зв'язок таблиці інформації, яка використовується для зберігання інформації про відео.

Рисунок 7 – Діаграма зв’язку сутностей даних тренувального набору

Таблиця інформації про відео включає в себе не мільки такі метадані, як назва відео, дата публікації, опис відео, його довжина та посилання на нього, а й безліч інших службових даних, які дозволяють орієнтуватися в наявних даних. Безумовно, в таблиці присутній унікальний ідентифікатор, що формується автоматично, та в подальшому використовується для відповідних комірок в кожному з контейнерів сховища BLOB. Окрім цього, в таблицю можна ввести оцінку придатності вихідного відео для подальшого розуміння доцільності використання ресурсів даного автора. Ба більше, через асинхронність циклів збору й подальшої обробки даних та можливість видалення непотрібних даних, з будь якого контейнеру сховища, таблицею передбачено наявність наступних прапорців, що допоможуть

швидко зрозуміти, чи були данні вже занесені в сховище, для подальшого їх використання, та чи були вони видалені, щоб не звертатися до них, якщо з’явиться така потреба, а саме:

- прапорець завантаження відео;
- прапорець завантаження субтитрів;
- прапорець конвертації відеофайлу в аудіофайл;
- прапорець фільтрації аудіофайлу від зайвих звуків;
- прапорець фільтрації субтитрів від зайвого службового тексту;
- прапорець видалення субтитрів;
- прапорець видалення аудіофайлу;
- прапорець видалення відфільтрованого аудіофайлу;
- прапорець видалення відфільтрованих субтитрів.

Більш докладна модель структури зберігання тренувальних даних продемонстрована у додатку А на плакаті 1 «Діаграма зв'язку сутностей тренувальних даних». Як видно зі схеми, автора винесено в окрему сутність. Таке вдосконалення зумовлено розробкою більш докладної системи аналізу якості зібраних даних, що дозволяє звертатися до каналу автора, у разі позитивної оцінки інформативності та репрезентативності отриманих даних. Реалізовано класичний зв’язок «один до багатьох», тобто кожний запис в інформаційній таблиці має зовнішній ключ, що посилається на таблицю автора, таким чином, не тільки унеможливлюється зайве повторення особистих даних каналу, а й при потребі, легко редагується інформація про нього. Для того щоб звернутися до БД необхідно використовувати бібліотеку psycopg2.

```
# З'єднання з базою даних
conn = psycopg2.connect(
    host="localhost",
    database="InfoDB",
    user="Admin123",
    password="Admin123"
)
```

Після з’єднання з БД залишається лише створити курсор, передати всі отримані метадані та отримати назад унікальний ідентифікатор, створений

безпосередньо для даного кортежу. Дані передаються з використанням команди «INSERT INTO», а ідентифікатор отримується командою «RETURNING id». Після виконання запиту, ми отримуємо ідентифікатор запису з використанням методу fetchone().

```
# Створення курсора
cur = conn.cursor()

# Вставка даних в таблицю
cur.execute("""
    INSERT INTO videos (title, author, publish_date, description, date)
    VALUES (%s, %s, %s, %s, %s)
    RETURNING id;
""", (title, author, publish_date, duration, date))

# Отримання автоматично згенерованого ID
record_id = cur.fetchone()[0]
```

Завантажені таким чином відео та відповідні субтитри потребують подальшого зберігання в BLOB-storage, що реалізується наступним програмним кодом.

```
# Завантаження відео до BLOB-storage
video_container_name = "videos"
video_blob_client =
blob_service_client.get_blob_client(container=video_container_name,
blob=f"{record_id}.mp4")
with open(video_file, "rb") as f:
    video_blob_client.upload_blob(f, overwrite=True)

# Завантаження субтитрів до BLOB-storage
caption_container_name = "captions"
caption_blob_client =
blob_service_client.get_blob_client(container=caption_container_name,
blob=f"{record_id}.srt")
caption_blob_client.upload_blob(caption_content, overwrite=True)
```

Щоб зберегти відео та субтитри до BLOB-storage, можна скористатися бібліотекою azure-storage-blob. Спочатку відбувається підключення до BLOB-storage, використовуючи рядок підключення connection_string та ім'я контейнера, в який будуть завантажені файли. Наступним кроком створюється об'єкт BlobClient, який відповідає відео та субтитрам, та завантажує файли відповідними методами.

Для конвертації відеофайлу у аудіо формат скористуємося бібліотекою «moviepy». Таким чином відеофайл завантажується з BLOB-storage, після чого конвертується в потрібний формат WAV та зберігається у відповідний контейнер у

сховищі, з аналогічною назвою, яка сформована з унікального ключа-ідентифікатора.

Головною завадою в навчанні мовної моделі є музика. Однак в субтитрах з YouTube музика помічається окремим службовим тегом «[музика]», це означає, що маючи субтитри, можна обрізати музики по часовим проміжкам. У форматі SRT субтитри складаються з трьох частин: номеру, часового проміжку та тексту, відповідний об’єкт продемонстровано на рисунку 8.

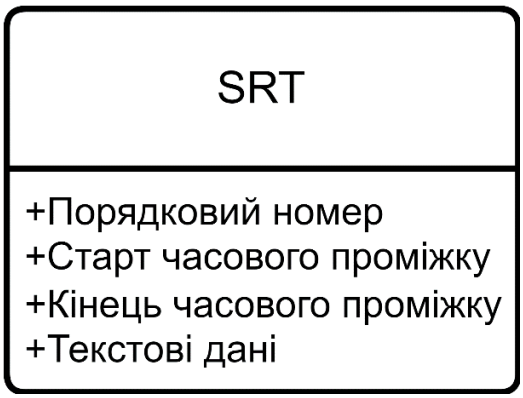

Рисунок 8 – Об’єкт SRT

Докладна схема процесу фільтрування аудіофайлів та субтитрів від музики та інших службових тегів, якщо такі зазначені, продемонстровано у додатку А на плакаті 2 «Діаграма діяльності алгоритму фільтрації аудіофайлів від службових тегів». Задля обробки субтитрів з подальшим видаленням музики, та й загалом іншого роду звуків, які так чи інакше не несуть інформацію для транскрибування, з аудіофайлу та субтитрів, потрібно в першу чергу завантажити субтитри без порядкових номерів до масиву даних. Після чого створюється додатковий масив, якому лише часові проміжки шуканих для видалення тегів. Пошук тегів відбувається в масиві з використанням регулярних висловів. Безумовно, можна видаляти всі теги, що мають структуру «[…]», де в замість трьох крапок назва звуку, який звучить, однак у даному випадку, будемо видаляти лише музику, так як системі не зайвим буде розуміти різноманітні звуки, які точно не мають в собі слів, на відміну від пісень. Відповідно масив субтитрів оброблюється з кінця, при знаходженні елемента з музикою часовий проміжок додається в кінець масиву

часових проміжків службових тегів, а з масиву субтитрів він видаляється. Наступним кроком, після обробки усього масиву субтитрів, є почергове видалення часових проміжків з аудіофайлу, за рахунок того, що масив часових проміжків сформовано за спаданням часу, то при видалення проміжків, аудіофайл не буду «зсуватись», що убезпечить аудіодані від колізій. Після повної обробки файлу, завчасно оброблений масив файлів трансформується в SRT документ, з послідовною нумерацією, починаючи від одного. Останнім кроком буде збереження в BLOB сховища, та встановлення відповідних поміток в інформаційну таблицю реляційної бази. На рисунку 9 продемонстровано діаграму діяльності, що графічно демонструє вище зазначений алгоритм.

Рисунок 9 – Діаграма діяльності алгоритму фільтрації аудіофайлів від музики

Задля зручного оперування даними субтитрів, їх потрібно розбити на масив окремих блоків. Даний код є функцією для розбиття SRT-файлу на окремі блоки, кожен з яких містить інформацію про одну субтитру. Використовуючи регулярні вирази бібліотеки «re», розбивається вміст SRT-файлу на окремі блоки за номером субтитру. Далі кожен блок перетворюється в словник за допомогою методу strip(), методу split(), та методу append(). У кінці функція повертає список словників, тобто шуканий масив вже приведених типізованих та готових до швидкої обробки даних.

Таким чином, з’являється змого швидко оперувати даними про субтитри, не витрачаючи зайвий час на парсинг текстового документу, яким по факту й є SRT-файл.

```
# Розбиваємо SRT файл на блоки за номером субтитру
srt_blocks = re.findall(r'\d+\n\d{2}:\d{2}:\d{2},\d{3} -->
\d{2}:\d{2}:\d{2},\d{3}\n.*?\n\n', str_content, re.DOTALL)

# Перетворюємо кожен блок в словник
srt_data = []
for block in srt_blocks:
    block_lines = block.strip().split('\n')
    block_dict = {
        'start_time': block_lines[1].split(' --> ')[0],
        'end_time': block_lines[1].split(' --> ')[1],
        'text': block_lines[2]
    }
    srt_data.append(block_dict)
```

Цей програмний код працює з масивом srt_data, який містить словники, що представляють блоки субтитрів з SRT файлу. Програма перевіряє кожен блок, щоб знайти місця, де є музика. Якщо в тексті блоку знайдено слова «music» або «song», то програма розглядає цей блок як частину музичного інтервалу. Якщо цей блок є початком нового музичного інтервалу, програма додає цей інтервал до списку музичних інтервалів music_intervals. Починаючи з останнього блоку srt, програма видаляє блоки, які належать до музичних інтервалів, до тих пір, поки не знайде блок, що не належить до музичного інтервалу.

```
    if re.search(r'\b(music|song)\b', block_dict['text'], re.IGNORECASE):
        if i == len(srt_data) - 1 or srt_data[i+1]['end_time'] !=
block_dict['start_time']:
            music_intervals.insert(0, (block_dict['start_time'],
block_dict['end_time']))
        srt_data.pop(i)
    else:
        if len(music_intervals) > 0 and music_intervals[0][1] ==
block_dict['start_time']:
            music_intervals[0] = (music_intervals[0][0], block_dict['end_time'])
            srt_data.pop(i)
```

Якщо програма знаходить блок, який не належить до музичного інтервалу і який є початком нового блоку srt, то програма додає цей новий блок до списку srt_data. Якщо програма знаходить блок, який не належить до музичного інтервалу і який не є початком нового блоку srt, то програма переходить до наступного блоку

srt. Цей код працює з SRT файлами, що містять субтитри і музику, та відділяє музику від тексту субтитрів, записуючи музичні інтервали в окремий список music_intervals.

Даний програмний код видаляє музичні проміжки з аудіофайлу, використовуючи інформацію про початок та кінець кожного музичного проміжку зі списку music_intervals.

```
for interval in music_intervals:
    start_time = int(interval[0].replace(',', '').replace(':', '')) # Перетворюємо
час у мілісекундах
    end_time = int(interval[1].replace(',', '').replace(':', '')) # Перетворюємо час
у мілісекундах
    audio = audio[:start_time] + audio[end_time:]
```

Цикл for ітерується по кожному елементу списку music_intervals. Для кожного елементу отримується початковий та кінцевий час проміжку у вигляді рядка. Ці рядки містять години, хвилини, секунди та мілісекунди, тому їх потрібно спочатку перетворити у мілісекунди, щоб вони могли бути використані для видалення проміжку з аудіофайлу. Для цього кожний рядок замінюється на рядок без ком і двокрапок та потім перетворюється в ціле число. Це дозволяє перетворити рядкове представлення часу в мілісекундах. Наступним кроком аудіофайл розбивається на дві частини: початкову та кінцеву. Початкова частина починається з початку аудіофайлу і закінчується за часом, який відповідає початковому часу музичного проміжку. Кінцева частина починається після часу, який відповідає кінцевому часу музичного проміжку і закінчується в кінці аудіофайлу. Нарешті, дві частини об'єднуються в один рядок, що становить оновлений аудіофайл без музичних проміжків.

Справа залишається з малим, перетворити масив очищених субтитрів назад у SRT-файл та зберегти аудіо та субтитри у сховище. Даний код проходить циклом по кожному блоку у списку srt_data, формуючи SRT-файл у змінній output. На кожній ітерації циклу спочатку додається номер блоку у форматі SRT (індекс елемента у списку srt_data плюс один), потім додається час початку та час закінчення блоку у форматі hh:mm:ss,fff --> hh:mm:ss,fff, і нарешті додається текст блоку. В кінці кожного блоку додається дві порожні стрічки (\n\n), як вимога

формату SRT. Отримана змінна output містить у собі відформатований SRT-файл, який можна зберегти у сховище.

```
output = ""
for i, block in enumerate(srt_data):
    output += str(i+1) + "\n"
    output += block['start_time'] + " --> " + block['end_time'] + "\n"
    output += block['text'] + "\n\n"
```

Даний процес код демонструє складний процес підготовки даних для навчання моделі з обробки природньої мови. Він дозволяє зібрати відео та субтитри з YouTube, конвертувати відео в аудіо формат, а потім обробляти субтитри та аудіо, щоб підготувати їх для навчання моделі. Для досягнення цього мети програмний код використовує різні модулі Python, включаючи pafy, moviepy, pydub, re, numpy, pymongo, та google-auth. Основна ідея полягає в тому, що збираються дані, потім вони обробляються та перетворюються у зручний формат для навчання моделі. Одним з основних етапів підготовки даних є видалення музичних проміжків з аудіофайлу, щоб зберегти тільки мовлення, яке необхідно для навчання моделі. Крім того, у процесі підготовки даних також встановлюються прапорці готовності у БД, що може бути корисним для моніторингу процесу та відновлення роботи у разі відмови системи. Загалом, даний етап виконує важливу роль у підготовці даних для навчання моделей з обробки природньої мови, і може бути корисним для дослідників та розробників у цій області.

2.2 Розроблення та навчання моделі системи транскрибування голосових повідомлень на основі технології глибинного навчання

Транскрибування аудіофайлів є важливим етапом у багатьох проектах, що пов'язані з обробкою мовлення. Моделі машинного навчання здатні автоматизувати цей процес, зменшивши навантаження на людину та забезпечуючи швидке та точне транскрибування. Однією з найпопулярніших архітектур для транскрибування є LSTM. LSTM є рекурентним шаром, що забезпечує здатність моделі запам'ятовувати попередній контекст та використовувати його при прийнятті рішень. Вони широко використовуються в задачах обробки мовлення,

таких як машинний переклад, розпізнавання мови та транскрибування. Однак, для досягнення більшої точності транскрибування, важливим є використання механізму уваги, тобто attention mechanism. Цей механізм забезпечує здатність моделі фокусуватися на певних ділянках вхідного аудіофайлу, що дозволяє їй зосередитися на більш важливих ділянках та ігнорувати менш важливі. Блок-схема нейромережевої моделі транскрибування аудіофайлів продемонстрована на рисунку 10.

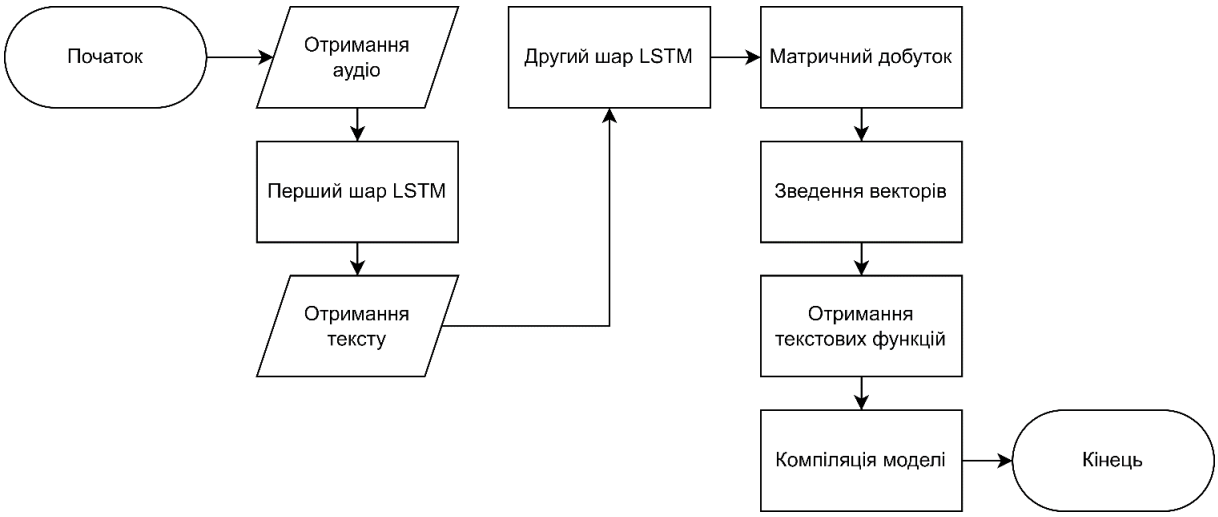

Рисунок 10 – Блок-схема нейромережевої моделі транскрибування

Блок-схема нейромережевої моделі відображає послідовність дій, які здійснюються для перетворення вхідних аудіоданих в вихідний текст. Основні етапи такої обробки розділяються на наступні кроки:

- введення вхідних даних – вхідні дані (аудіодані та текстові дані) вводяться в нейромережеву модель;
- кодування аудіоданих – аудіодані обробляються за допомогою LSTM-шару, який повертає послідовність векторів прихованого стану, які описують зміни стану мережі відповідно до вхідних даних;
- кодування текстових даних - текстові дані проходять через ще один LSTM-шар, щоб згенерувати послідовність векторів прихованого стану;
- використання матричного добутку для підрахунку вагових коефіцієнтів для кожного вхідного стану;

- побудова контексту – обчислення вектора контексту, який являє собою зважену суму векторів прихованого стану аудіоданих;
- декодування – використовуючи LSTM-шар декодера, щоб згенерувати послідовність векторів прихованого стану, на основі контексту та текстових даних;
- генерація вихідного тексту – на останньому етапі використовуючи TimeDistributed-шар, генерується вихідний текст для заданої аудіо-послідовності; цей процес відбувається шляхом передачі кожного вектору з вихідної послідовності декодеру, який на кожному кроці генерує наступний вектор тексту з урахуванням контексту і попереднього згенерованого вектора; ця процедура повторюється для кожного елементу вихідної послідовності, доки не буде згенерований повний текст; кінцевий результат – модель, яка може генерувати текст на основі аудіо вхідних даних.

Основною ідеєю LSTM є збереження довготривалих залежностей в послідовностях, що змінюються в часі. LSTM можна використовувати для транскрибування аудіо, де кожен елемент послідовності відповідає короткому фрагменту аудіо (наприклад, 1 секунді). За допомогою LSTM можна обробити ці фрагменти та отримати послідовність слів, які відповідають аудіофрагментам. Механізм attention дозволяє моделі приділяти більше уваги певним фрагментам аудіо при транскрибуванні. Це забезпечує більш точне визначення того, які слова відповідають певним фрагментам аудіо. У моделі з механізмом attention, кожен елемент послідовності LSTM має вагові коефіцієнти, які вказують на те, наскільки важливо цей елемент для кожного слова у вихідному тексті. Ці вагові коефіцієнти використовуються для обрахування зваженого середнього вектору, який відповідає певному слову у вихідному тексті.

Keras – це високорівнева бібліотека для глибокого навчання, яка дозволяє швидко і легко створювати нейронні мережі. Вона має простий та зрозумілий інтерфейс, що дозволяє швидко перевіряти нові ідеї та концепції. Керас має вбудовану підтримку великої кількості архітектур нейронних мереж та

оптимізаторів, що дозволяє швидко створювати та оптимізовувати нейронні мережі. Крім того, Keras має добре розвинуту спільноту, яка створює багато навчальних матеріалів, і дозволяє легко використовувати існуючі моделі та приклади. Keras є частиною TensorFlow, тому використання Keras забезпечує зручну інтеграцію з іншими бібліотеками, що використовують TensorFlow в якості бекенду. Отже, Keras є популярною та потужною бібліотекою для розробки моделей глибокого навчання, яка забезпечує швидке та зручне створення та навчання нейронних мереж. Саме через що, використано даний інструментарій при розробці механізму транскрибування. Наступний код імпортує необхідні класи з бібліотеки Keras для побудови моделі машинного навчання.

```
from keras.layers import Input, Dense, LSTM, TimeDistributed, Concatenate, Dot
from keras.models import Model
```

Ці класи використовуються для побудови моделі машинного навчання з LSTM шарами та механізмом attention, а саме:

- Input – клас для визначення вхідних даних моделі;
- Dense - повнозв'язний шар, що використовується для перетворення вхідних даних з одного простору розмірності в інший;
- LSTM – клас для побудови LSTM шарів;
- TimeDistributed – шар, який застосовує інші шари до кожного часового кроку вхідних даних;
- Concatenate – шар, який об'єднує вихідні дані з декількох шарів у один тензор;
- Dot – шар для обчислення дот-продукту між двома тензорами;
- Model – клас для створення моделі, визначаючи вхідні та вихідні шари.

Використання цих класів забезпечує швидку та зручну розробку моделі з високою точністю та ефективністю.

```
input_shape = (None, num_audio_features)
```

Цей рядок коду визначає форму вхідних даних для моделі. Змінна num_audio_features визначає кількість аудіо-функцій, які будуть використовуватись для аналізу звукового сигналу. Перший елемент в кортежі (None) вказує на те, що модель приймає вхідні дані різної довжини, що є корисною

функцією для обробки аудіо-файлів різної довжини. Остаточна форма вхідних даних буде виглядати як (кількість часових кроків, num_audio_features), де кількість часових кроків відповідає кількості секунд аудіо-файлу, яка була розбита на маленькі частини.

```
encoder_outputs, state_h, state_c = LSTM(units=encoder_units, return_sequences=True, return_state=True, name='encoder')(inputs)
```

Цей рядок програмного коду визначає вхідний шар моделі з вказаною формою вхідних даних. Змінна input_shape визначає форму вхідних даних, яку складається з None – яка означає, що розмір пакета може бути будь-яким, та num_audio_features - кількість аудіо-функцій, яку ми використовуємо як вхідні дані.

```
decoder_inputs = Input(shape=(None, num_text_features), name='text_input')
decoder_lstm = LSTM(units=decoder_units, return_sequences=True, return_state=True, name='decoder')
decoder_outputs, _, _ = decoder_lstm(decoder_inputs, initial_state=[state_h, state_c])
```

Наступний код використовує структуру декодеру, яка приймає на вхід розгорнутий вектор слів транскрипції та повертає послідовність векторів з прогнозованими імовірностями наступних слів. decoder_inputs – вхідний шар декодера, де num_text_features – це кількість можливих слів у транскрипції, які можуть бути передані на вхід. decoder_lstm – LSTM шар декодера, який має decoder_units нейронів. Шар повертає послідовність значень для кожного часового кроку, а також останній стан LSTM. decoder_outputs – послідовність вихідних векторів декодера, що містять прогнозовані імовірності наступних слів. state_h та state_c – це стани, які передаються від енкодеру до декодера як початкові стани LSTM для декодера.

```
attention = Dot(axes=[2, 2], name='attention')([decoder_outputs, encoder_outputs])
attention = Dense(units=num_audio_features, activation='softmax', name='attention_softmax')(attention)
```

Використано механізм уваги (attention) для транскрибування аудіо. Конкретніше, використано скалярний добуток між виходами декодера (decoder_outputs) та енкодера (encoder_outputs), щоб отримати вектор уваги (attention vector) на кожному кроці часу. Потім застосовується повнозв'язний шар (Dense) зі зворотним розповсюдженням помилок (softmax activation) до цього вектору уваги, щоб отримати розподіл ваг на кожному кроці часу, який відображає

важливість кожної частини аудіо в кожному кроці часу для транскрибування тексту.

```
context = Dot(axes=[2, 1], name='context')([attention, encoder_outputs])
decoder_combined_context = Concatenate(axis=-1,
name='decoder_combined_context')([context, decoder_outputs])
```

Виконуємо скалярний добуток між attention та encoder_outputs з заданими осями. Це створює контекст для кожного кроку часу, де кожен контекст є зваженою сумою вихідних даних encoder_outputs. Та об’єднуємо контекст з виходами декодера decoder_outputs по останньому (третьому) виміру, що відповідає часовим крокам, за допомогою конкатенації. Ось що робить параметр axis=-1. Отримана змінна decoder_combined_context містить об'єднані виходи декодера та контекст для кожного часового кроку. Це допомагає моделі зосередитися на тих частинах вхідного сигналу, які найбільш істотні для кожного кроку часу.

```
output = TimeDistributed(Dense(units=num_text_features, activation='softmax'),
name='output')(decoder_combined_context)
```

Даний програмний код визначає останній шар нашої моделі output. Він використовує TimeDistributed, щоб застосувати Dense шар до кожного часового кроку вхідних даних decoder_combined_context, вихідний контекст з декодера і контекст з увагою, повертаючи остаточний прогнозований текст. Отже, остаточний вихід моделі буде матрицею розміру (batch_size, max_text_len, num_text_features), яка містить розподіли ймовірностей для кожного можливого токену у тексті, що відповідає вхідному аудіо.

```
model = Model(inputs=[inputs, decoder_inputs], outputs=output)
```

Цей код створює об'єкт моделі, використовуючи попередньо визначені вхідні дані та вихідний шар. Це досягається за допомогою класу Model з бібліотеки Keras. Об'єкт моделі отримує список вхідних шарів та список вихідних шарів, а потім може бути скомпільований за допомогою методу .compile(), використовуючи певну функцію втрат та оптимізатор.

```
model.compile(optimizer='adam', loss='categorical_crossentropy')
```

Фінальний метод compile у Keras для складання моделі із вказаною функцією втрат та оптимізатором. Модель використовує оптимізатор «adam» та функцію втрат «categorical_crossentropy», яка використовується для багатокласової

класифікації з декількома категоріями. Оптимізатор «adam» є популярним алгоритмом градієнтного спуску з адаптивним коефіцієнтом навчання.

Щоб навчити модель, потрібно передати навчальні дані та підготувати їх для навчання. Підготувати навчальні дані у відповідному форматі. Для цієї моделі потрібно мати два типи даних - аудіо та текст, що відповідає цьому аудіо. Обидва типи даних мають бути у вигляді послідовностей, де кожна послідовність відповідає одному аудіофайлу та відповідному тексту. Що й було попередньо зроблено. Дані завантажуються зі сховища та передаються в модель для тренування, тобто встановлення певних важків між нейронами, на основі яких й будуть прийматись наступні рішення, стосовно транскрибування аудіо доріжок, наданих для транскрибування.

```
model.fit(x=[X_train_audio, X_train_text], y=y_train, batch_size=batch_size, epochs=num_epochs)
```

Цей код використовує функцію fit для навчання моделі. Функція fit приймає на вхід декілька параметрів, що дозволяють налаштувати процес навчання, а саме:

- x – список з вхідними даними, які подаються на вхід моделі; у випадку даної моделі, список містить дві матриці: X_train_audio – дані аудіофіч, та X_train_text – текстові дані; обидві матриці мають форму (num_samples, sequence_length, num_features);
- y – цільові значення, до яких модель має збігатися; у даному випадку, y_train – це матриця форми (num_samples, sequence_length, num_text_features);
- batch_size – розмір пакета даних, що використовуються для навчання моделі; пакети використовуються для того, щоб модель можна було навчити ефективніше;
- epochs – кількість епох, які модель буде навчатися, тобто кількість проходження по всіх навчальних прикладах.

Розроблений програмний код створює модель машинного навчання з використанням нейронних мереж LSTM та механізмом уваги. Вона складається з двох частин: енкодеру та декодеру. Енкодер приймає на вхід аудіофайли та перетворює їх у послідовність векторів за допомогою LSTM шару. Декодер

приймає на вхід послідовність векторів, які складаються з розпізнаного тексту та генерує на їх основі текстовий опис аудіофайлу. Декодер також використовує LSTM шар, який приймає на вхід текстові вектори та використовує увагу, щоб враховувати контекст аудіофайлу та згенеровані раніше слова.

Механізм уваги обчислюється за допомогою точкового множення та конкатенації векторів, після чого він подається на вхід повторному LSTM шару декодеру разом із згенерованим раніше текстом. На виході декодеру отримуємо остаточний результат, який проходить через TimeDistributed Dense шар з функцією активації softmax. Після створення моделі, ми компілюємо її з оптимізатором 'adam' та функцією втрат 'categorical_crossentropy'. Далі модель можна навчати на тренувальних даних.

Розроблена модель є комбінованою рекурентною нейронною мережею, яка призначена для перетворення аудіофайлів у відповідний текст. Вона складається з двох рекурентних LSTM шарів, які обробляють вхідні дані та зворотно-поширення вивчається вагові коефіцієнти моделі. Також в моделі використовується механізм уваги , який дозволяє фокусувати увагу на певній частині вхідних даних. За допомогою функції активації softmax на виході моделі отримується розподіл ймовірностей наступного символу тексту. Для навчання моделі використовується категоріальна хрест-ентропія як функція втрат.

2.3 Реалізація та тестування системи

Розробка застосунку транскрибування аудіофайлів – це складний і багатоетапний процес. Однак, з використанням наявних бібліотек та інструментів, можна значно спростити цей процес. Транскрибування аудіо- та відеофайлів потребує певної інфраструктури та інструментів обробки цих даних. Для реалізації системи транскрибування аудіо- та відеофайлів було використано наступні технології:

- Azure – це платформа хмарних обчислень, яка забезпечує хмарні ресурси для розгортання та виконання застосунків;

- Vue.js – це фреймворк для створення інтерфейсів користувача на основі JavaScript, використовується для створення фронтенду застосунків веб-додатків;
- Python – це високорівнева мова програмування, яка використовується для написання бекенду застосунків, а також для аналізу даних та машинного навчання; в розробленій системі використовується для розробки алгоритму транскрибування аудіофайлів;
- Kubernetes – це інструмент для автоматизації розгортання, масштабування та керування контейнерами; в розробленій системі використовується Kubernetes для керування контейнерами зі складовими нашого застосунку;
- PostgreSQL – це реляційна база даних з відкритим вихідним кодом, яка використовується для зберігання даних у нашій систем, використовується для зберігання історії.

Структуру розробленої системи представлена у додатку А на плакаті 3 «Діаграма компонентів», який складається з наступних елементів:

- візуальний інтерфейс;
- Azure AD;
- кластер Kubernetes;
- послуги ядра;
- сервіси платформи
- сервер обробник.

Ця діаграма описує архітектуру системи на рівні компонентів та взаємодії між ними. Вона демонструє, як компоненти системи взаємодіють один з одним та як вони розміщені на різних фізичних пристроях [32]. На розробленій діаграмі видно, як користувач через візуальний інтерфейс автоматично надсилає запит у Azure AD на отримання унікального токену-доступу для авторизації користувача у системі, що дозволяє захистити сервер від проникнення несанкціонованих осіб, після чого користувач має повний доступ до системи. Додаючи файл для транскрибування через візуальний інтерфейс веб-застосунка, користувач передає його спочатку у кластер Kubernetes, через Ocelot Gateway, який дозволяє

налаштувати якісну маршрутизацію запитів, приховавши реальні порти та адреси розподілених елементів системи, що підвищує її захищеність. Ocelot, в свою чергу, надсилає переданий користувачем файл в сховище, структура якого продемонстрована у додатку А на плакаті 4 «Діаграма зв’язку сутностей системи», після чого повертається унікальний ключ запису, який надає змогу звернутись до сховища, та отримати дані. Отриманий ключ надсилається серверу, що займається обробкою та транскрибуванням файлі, після чого він звертається до сховища, користуючись послугами маршрутизатора, отримує звідти дані та після обробки, з використанням Ocelot, повертає їх на сторінку користувача. Візуальна частина підсистеми була розроблена на базі бібліотеки Vue.js, а логічна частина на мові Python. Транскрибування самого ж файлу відбувається за допомогою створеної нейронної мережі.

Розроблена система має одну сторінку, на якій відбуваються всі процеси. Ця сторінка є основною в системі. На ній користувач може виконати наступні дії:

- переглянути історію транскрибованих файлів;
- завантажити файл для транскрибування;
- скасувати процес транскрибування;
- вийти з системи;
- переглянути транскрибований текст;
- скопіювати транскрибований текст;
- зберегти результати у файл формату .PDF;
- очистити сторінку.

Ця сторінка надає основний функціонал розробленої системи, її загальний вигляд зображено на рисунку 11. Справа зверху знаходяться дві кнопки:

- вийти з системи;
- згорнути панель.

Панель, що знаходиться на лівій стороні сторінки, може бути згорнута натисканням описаної кнопки. Ця панель містить інформацію про юзера, меню конвертації та історію транскрибованих файлів. Самі файли зберігаються у сховищі Azure Blob Storage у форматі Blob, а історія міститься у реляційній базі даних.

Файли в історії відображуються відповідно до дати їх завантаження та відсортовані за її спаданням. При натисканні на поле «Завантажити файл» користувачеві надається можливість обрати файл для транскрибування. Також варто зазначити, що цей процес налаштований лише на вибір аудіо- чи відеофайлів, інші файли вставити неможна. При цьому модель транскрибування працює безпосередньо з форматом .WAV, однак система конвертує всі аудіо чи відео в цей формат, якщо це необхідно, тобто юзер може завантажити аудіо- чи відеофайл у будь-якому форматі, а система в свою чергу за необхідності зробить конвертацію.

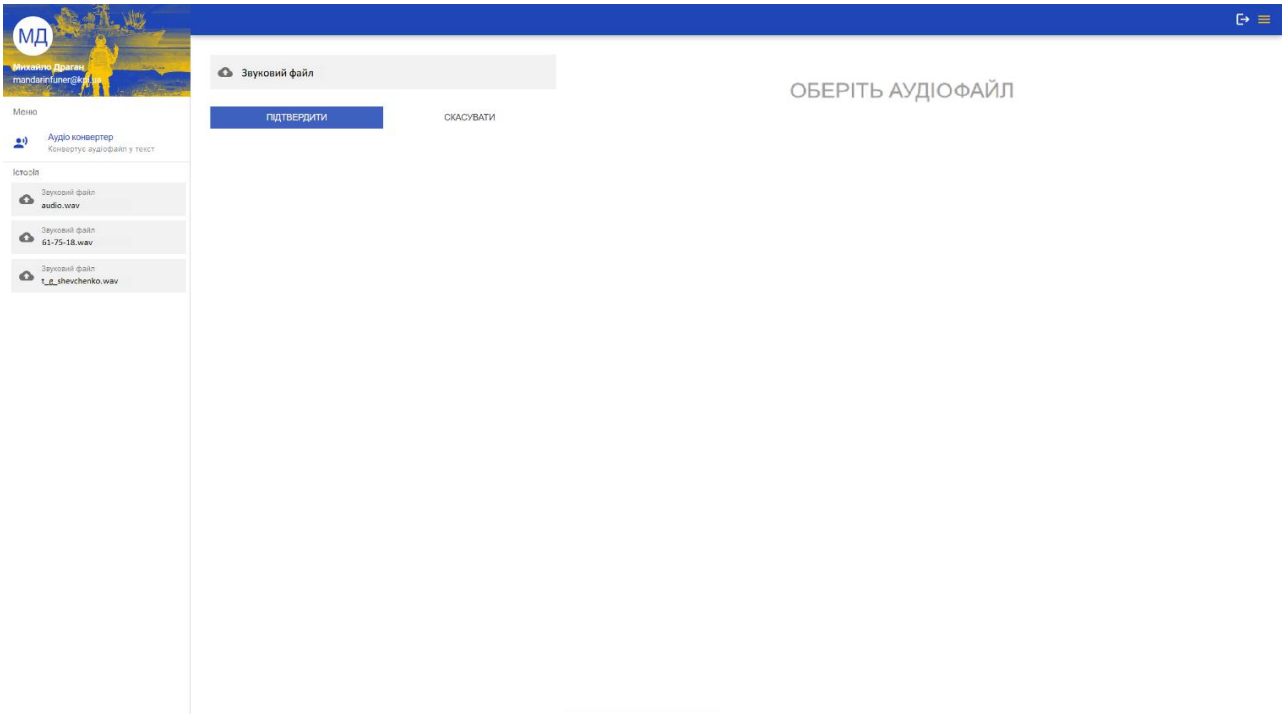

Рисунок 11 – Основна сторінка розробленої системи

Після вибору файлу, який обмежений лише аудіо чи відео форматами, для завантаження спочатку текст на сторінці замінюється на «Підтвердіть чи скасуйте вибір», після натискання юзером кнопки «Підтвердити» внизу сторінки з’являється повідомлення з текстом «Запит відправлено, очікуйте», він означає, що файл було відправлено на сервер обробник для проведення транскрибування та повернення транскрибованого тексту юзеру на сторінку. У випадку якщо ж станеться якась помилка, тобто не вийде звернутись до сервера, повідомлення буде підсвічено червоним кольором та зазначено «Сталася помилка, спробуйте пізніше». В момент,

коли сервер повернув результат, його одразу буде відображено на сторінку, в цей момент з’являється транскрибований текст, програвач аудіо та дві кнопки біля нього. Цю сторінку зображено на рисунку 12. Якщо юзер вмикає завантажений файл через програвач аудіо, рядки тексту послідовно підсвічуються відповідно моменту, який програється на доріжці. Це дозволяє йому відслідкувати, що і коли було сказано у завантаженому файлі. Якщо юзер натисне на якийсь з рядків тексту, вказівник на доріжці програвача перейде на момент, де у файлі звучали ці слова. Також біля програвача знаходяться кнопки «Експортувати текст у файл» та «Очистити». Перша завантажує транскрибований текст у файл з розширенням .PDF та назвою, що відповідає назві завантаженого файла. Друга кнопка очищує робочу зону сторінки та пропонує завантажити новий файл для транскрибування, на сторінці знову з’являється текст «Оберіть аудіофайл».

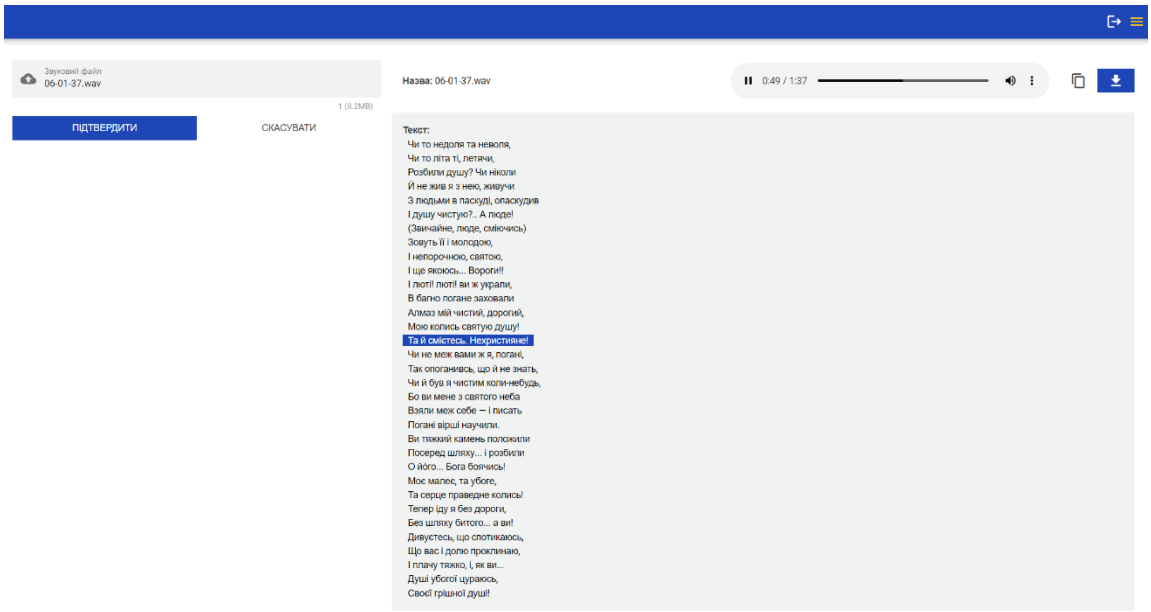

Рисунок 12 – Результат процесу транскрибування завантаженого файлу на сторінку

Також варто зазначити, що без авторизації користувач не має доступу до цієї сторінки. Замість неї він бачить вікно авторизації у акаунт Microsoft, що зображено на рисунку 13, але вже після авторизації користувач отримує повний доступ до функціоналу. Якщо користувач немає цього акаунту, він може його створити та

мати доступ до системи. Також це вікно буде показано після того, як юзер вийде з системи, тобто без авторизації користувач не зможе навіть переглянути загальний вигляд основної сторінки.

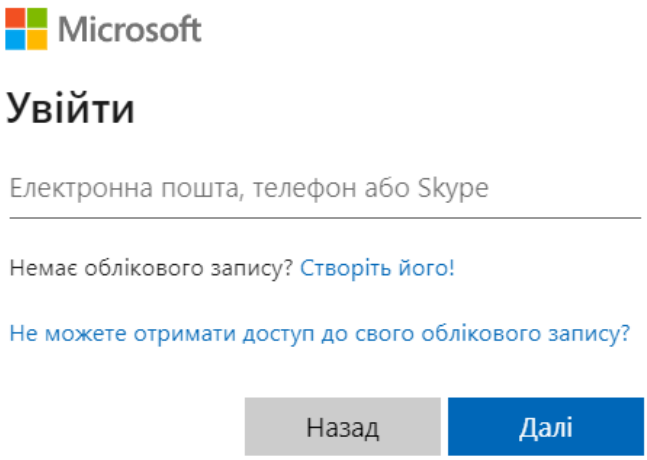

Рисунок 13 – Вікно входу у акаунт Microsoft

Більш детально усі функції системи користувача у розробленій системі та сценарії її використання представлені у додатку А на плакаті 5 «Діаграма прецедентів». Цей тип діаграм використовується для опису взаємодії користувачів з системою та її функціональністю. Ця діаграма дозволяє виділити основні функції системи та показати, як користувачі взаємодіють з цими функціями [33]. Актором розробленої системи є користувач, який взаємодіє з нею. Користувачем цієї системи може стати будь хто та використовувати її у своїх власних цілях. Після авторизації користувач має декілька сценаріїв використання системи:

– перегляд основної сторінки підсистем;
– вихід із системи;
– перегляд історії транскрибованих файлів;
– завантаження файла для транскрибування.

Сценарій «перегляд історії транскрибованих файлів» дозволяє юзеру повторно транскрибувати файл, що проходив цей процес в минулому та

автоматично веде до наступного сценарію. Останній сценарій використання дає юзеру доступ до таких сценаріїв, як:

- зупинити процес транскрибування файлу;
- очистити дані зі сторінки;
- переглянути транскрибований текст;
- прослухати завантажений файл;
- перегляд виділеного тексту.

При цьому після перегляну транскрибованого тексту користувач може виконати наступні дії:

- очистити дані зі сторінки;
- скопіювати транскрибований текст;
- зберегти транскрибований текст у форматі .PDF;
- натиснути на транскрибований текст;
- прослухати завантажений файл.

Описаний набір сценаріїв використання системи надає користувачеві повний доступ до функціональності системи та дозволяє транскрибувати аудіо- та відеофайли з метою задоволення потреб користувача в розробленій системі.

Для описання поведінки системи відповідно до її різних станів в залежності від дій, які виконує користувач, було розроблено діаграму, яка представлена у додатку А на плакаті 6 «Діаграма станів». На цій діаграмі можна побачити, як реагує система на різні дії користувача та переходи між різними станами системи [34]. Першим станом системи є відображення сторінки авторизації. Як вже було зазначено, користувач не може отримати доступ до функціоналу системи без попередньої авторизації у акаунт Microsoft. Після входу в акаунт система переходить до стану перевірки даних авторизації, якщо дані, які ввів користувач є невірними, система відобразить помилку авторизації та повернеться до першого стану, а користувач побачить помилку. Якщо ж дані є вірними, система відображує основну сторінку, де користувач може завантажити аудіо- чи відеофайл для транскрибування. Після натискання кнопки «Обрати файл» система виведе стандартне вікно для вибору файла, якщо користувач відхилить цю дію, він знову

побачить основну сторінку, але якщо він обере файл, який необхідно транскрибувати, система почне процес транскрибування файла та по його завершенню відобразить на екрані транскрибований текст та аудіодоріжку з завантаженого файлу. В цьому випадку користувач може виконати декілька дій:

- очистити дані зі сторінки;
- увімкнути аудіодоріжку на програвачі;
- натиснути на рядок транскрибованого тексту;
- скопіювати текст;
- натиснути на кнопку «Експортувати текст у файл».

Кожна з цих дій переведе систему у певний стан. При очищенні даних зі сторінки система очистить сторінку від результатів транскрибування та повернуться у стан відображення основної сторінки. При увімкненні користувачем аудіодоріжки з завантаженого файлу на програвачі система запустить аудіо та почне виділяти фрагменти транскрибованого тексту відповідно до моменту на аудіодоріжці. Ця функція працює і в зворотний бік, якщо користувач натисне на якийсь з рядків транскрибованого тексту, він підсвітиться системою, а час на програвачі зміниться на час, що відповідає виділеному тексту, таким чином користувач зможе переслухати потрібні йому моменти. При копіюванні транскрибованого тексту система завантажить його в буфер обміну для подальших дій користувача. Натиснувши на кнопку «Експортувати текст у файл», користувач побачить стандартне вікно для вибору місця збереження файлу з результатом транскрибування. При підтвердженні цієї дії системі переходить у стан формування файла у форматі .PDF з транскрибованим текстом та збереження сформованого файла у обране користувачем місце.

Для зображення процесу взаємодії між об’єктами в системі в певний момент часу було розроблено діаграму, яка представлена у додатку А на плакаті 7 «Діаграма послідовності системи». Діаграма послідовності дозволяє описувати порядок виконання дій між об'єктами та комунікації між ними [35]. Її об’єктами у даній системі є:

- користувач;

- візуальний інтерфейс;
- Azure Blob Storage;
- сервер маршрутизації;
- сервер обробник.

На цій діаграмі зображено, як користувач завантажує файл для транскрибування, та які процеси після цієї дії відбуваються у системі. Після того, як користувач додав файл для транскрибування, у візуальному інтерфейсі формується POST-запит на додавання цього файлу у Azure Blob Storage, але для того, щоб запит дійшов до місця призначення, він спочатку потрапляє на сервер маршрутизації, який аналізує надісланий POST-запит та відправляє його у Azure Blob Storage, де відбувається збереження файлу у сховище. У якості відповіді зі сховища повертається Id запису з файлом, що було створено, ця відповідь спочатку також потрапляє на сервер маршрутизації, після чого він надсилає відповідь назад до візуального інтерфейсу, де формується GET-запит до сервера обробника з метою транскрибування файла. Цей запит містить у собі Id запису у сховищі. Перш ніж GET-запит дійде до сервера обробника, він потрапить знову до сервера маршрутизації, який направить цей запит до місця призначення. Для того, щоб на сервері обробку почався процес транскрибування, він створить новий GET-запит та звернеться до Azure Blob Storage за файлом. Цей запит потрапляє на сервер маршрутизації, після чого сервер аналізує запит на направляє його у Azure Blob Storage. Сховище обробляє запит та у відповідь надсилає файл, який знаходиться у записі, що відповідає надісланому Id. Відповідь потрапляє на сервер маршрутизації, після чого доходить до сервера обробника, де і починається процес транскрибування файла нейронною мережею на цьому сервері. Цей процес проводиться для того, щоб дати відповідь на GET-запит, який було надіслано з візуального інтерфейсу. Відповіддю буде транскрибований текст, який спочатку потратить на сервер маршрутизації, після чого він дійде до візуального інтерфейсу та буде відображений користувачеві як результат транскрибування доданого їм файлу.

При тестуванні розробленої системи не було виявлено помилок. Для транскрибування були використані різні аудіо- та відеофайли, які в подальшому були конвертовані у формат .WAV та транскрибовані натренованою моделлю. Текст був коректно транскрибований з завантажених файлів та відображений на сторінці. При перегляді історії файлів, що були транскрибовані раніше, як і було зазначено, користувач може повторно відкрити файл для транскрибування та отримати коректні результати. При спробі завантажити результати транскрибування, система сформувала файл, який показано на рисунку 14. Він має розширення .PDF та містить у собі транскрибований текст, назва файла співпадає з назвою аудіофайлу, який був наданий системі для оброблення, ці дані представлено на рисунку 12.

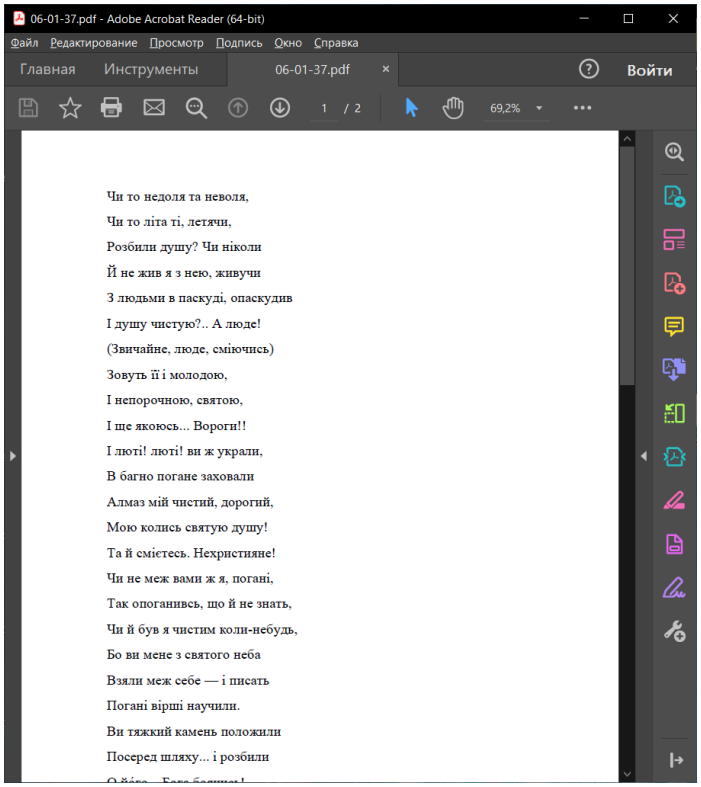

Рисунок 14 – Результати транскрибування, завантажені у окремий файл

Розроблена система може бути вдосконаленою за необхідності, наприклад, за рахунок розширення функціоналу, а саме за рахунок імплементації аналізу транскрибованого тексту за певними критеріями. Також виділення головної інформації з тексту може дати юзеру додаткову інформацію про завантажений

текст. Також у систему можуть бути додані нові функції, наприклад, вивантаження аудіо- чи відеофайлів, які були транскрибовані в минулому. Загалом, система може набути вдосконалень чи змін за потреби.

2.4 Оцінка ефективності системи транскрибування

Важливим етапом розробки системи транскрибування є оцінка якості та визначення ефективності системи. Це дозволяє зрозуміти, наскільки точно та швидко вони працюють, та визначити можливі обмеження при їх використанні. Різноманітні метрики, графіки та механізми статистичного аналізу дозволяють прозоро та тверезо оцінити «дитячі хвороби» та сильні сторони системи, після чого вони можуть бути змінені на краще, або ж зазначені в особливостях поведінки системи.

Однією з основних метрик є точність транскрипції, яка визначається порівнянням отриманого тексту з оригінальним аудіофайлом або з ручною транскрипцією. Чим більше співпадінь, тим вища точність системи. Крім того, важливими метриками є час транскрипції системи, який впливають на швидкість обробки та кількість файлів, що можуть бути оброблені протягом певного періоду часу. Важливість тестування системи транскрибування на різних рівнях зашумленості полягає у тому, що звукові сигнали в реальних умовах можуть містити різний рівень шуму, що може вплинути на точність транскрибування. Тому для ефективної роботи системи транскрибування необхідно перевіряти її ефективність в різних умовах. доцільності у різних сферах застосування. Якщо система показує високу точність транскрибування при різних рівнях зашумленості, це говорить про її високу надійність та здатність працювати в різних умовах. Однак, якщо результати тестів показують низьку точність транскрибування в певних умовах, наприклад, при дуже високому рівні шуму, це може свідчити про необхідність проведення додаткового дослідження, на тему впливу шуму на процес транскрибування, та пошуку алгоритму позбавлення, чи бодай зниження, негативного ефекту від різного типу завад.

Графік залежності функції втрат від кількості ітерацій навчання, продемонстровано на рисунку 15.

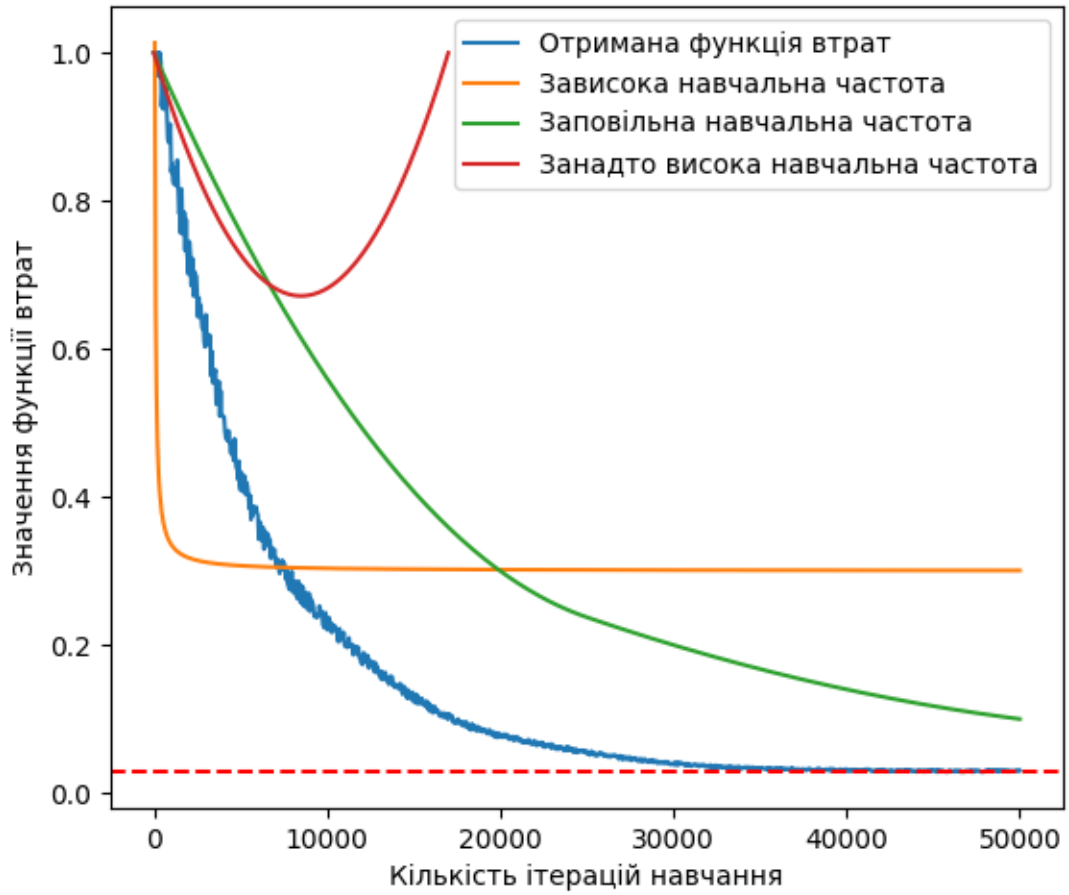

Рисунок 15 – Графік залежності функції втрат від кількості ітерацій навчання

Графік показує, як швидко зменшується значення функції втрат під час навчання моделі. Функція втрати визначає, наскільки добре модель підійшла під навчальні дані шляхом порівняння прогнозів моделі з правильними відповідями. Рисунок демонструє поведінку функції втрат близьку до гіперболічної функції, це означає, що частота тренування моделі була обрана правильно. Функція втрат спадає, на відміну від випадків з неправильним навчанням, коли втрати лише ростуть з плином часу, що демонструє адекватність прийняття рішення нейронною мережею. Також функція втрат зменшується по кривій, що вказує на достатню частоту навчання, на відміну від випадків, коли повільного навчання, де графік радше схожий на пряму. Важливо підкреслити, що показники втрат прямують до

позначко 0.3, на противагу випадкам, коли мінімальне значення зупиняється, на певному проміжку та перестає спадати, це призводить до виникнення сталої похибки система, яка зменшить відсотки адекватного прийняття рішення.

Графік точності моделі в залежності від кількості ітерацій навчання продемонстровано на рисунку 16.

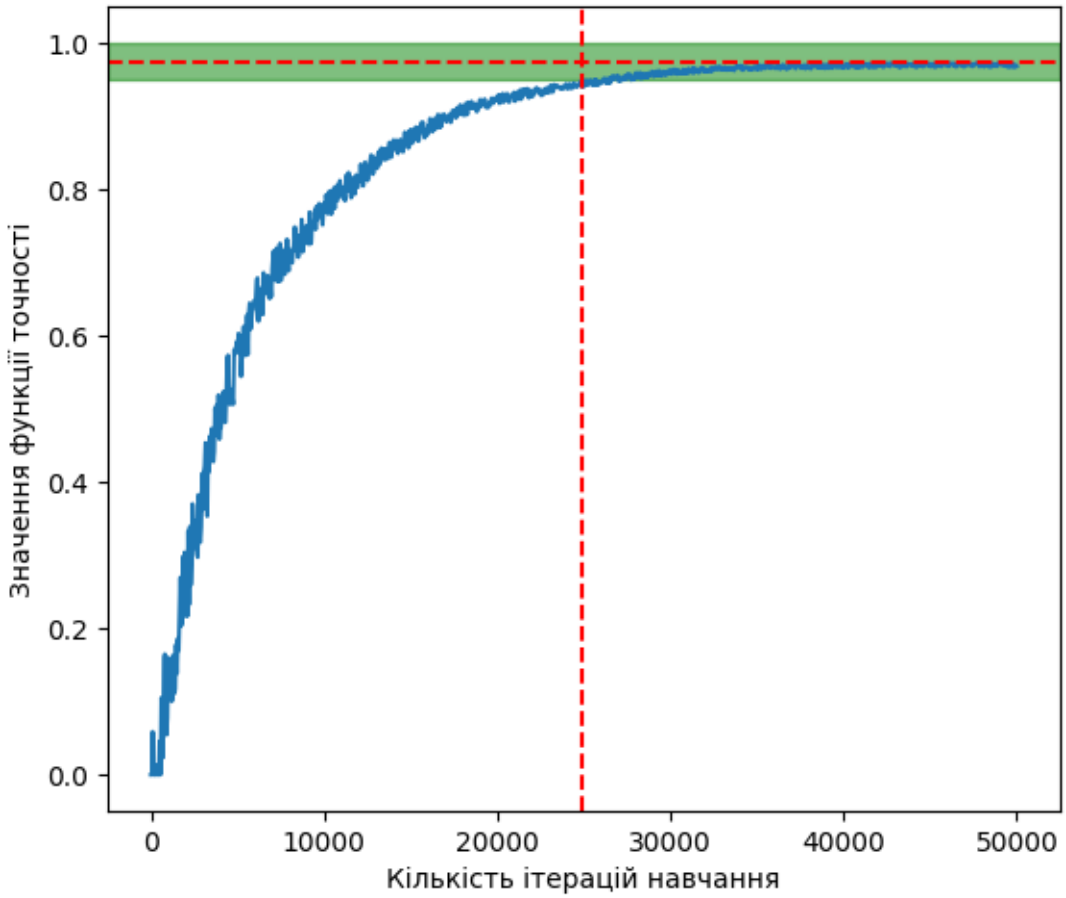

Рисунок 16 – Графік залежності функції точності від кількості ітерацій навчання

Цей графік показує, як швидко зростає точність моделі під час навчання. Модель долає показник точності в 95% близько 24700 ітерації, подальше навчання збільшить показники якості системи до 97%. Це означає, що загальна точність системи сягає 97%, тобто 97 з 100 рішень, які приймає модель, а саме, яке слово транскрибувати, з певного проміжку аудіо, є правильними. Подальша спроба навчати модель позбавлена сенсу, так як ефективність покращення точності моделі

даним процесом близька нулю, однак час та розрахункові потужності, які використовуються в ході таких операцій, суттєво ускладнюють процес навчання системи.

Час транскрибування у відриві від часу аудіо та точності транскрибування, не дає жодної об’єктивної картини якості роботи системи, як і відсоток точності, не зважаючи на витрачений час, не може бути єдиним метриком, для прийняття рішення, стосовно якості продукту. Через що, на рисунку 17 продемонстровано одразу три типи даних, які дозволяють повноцінно оцінити картину.

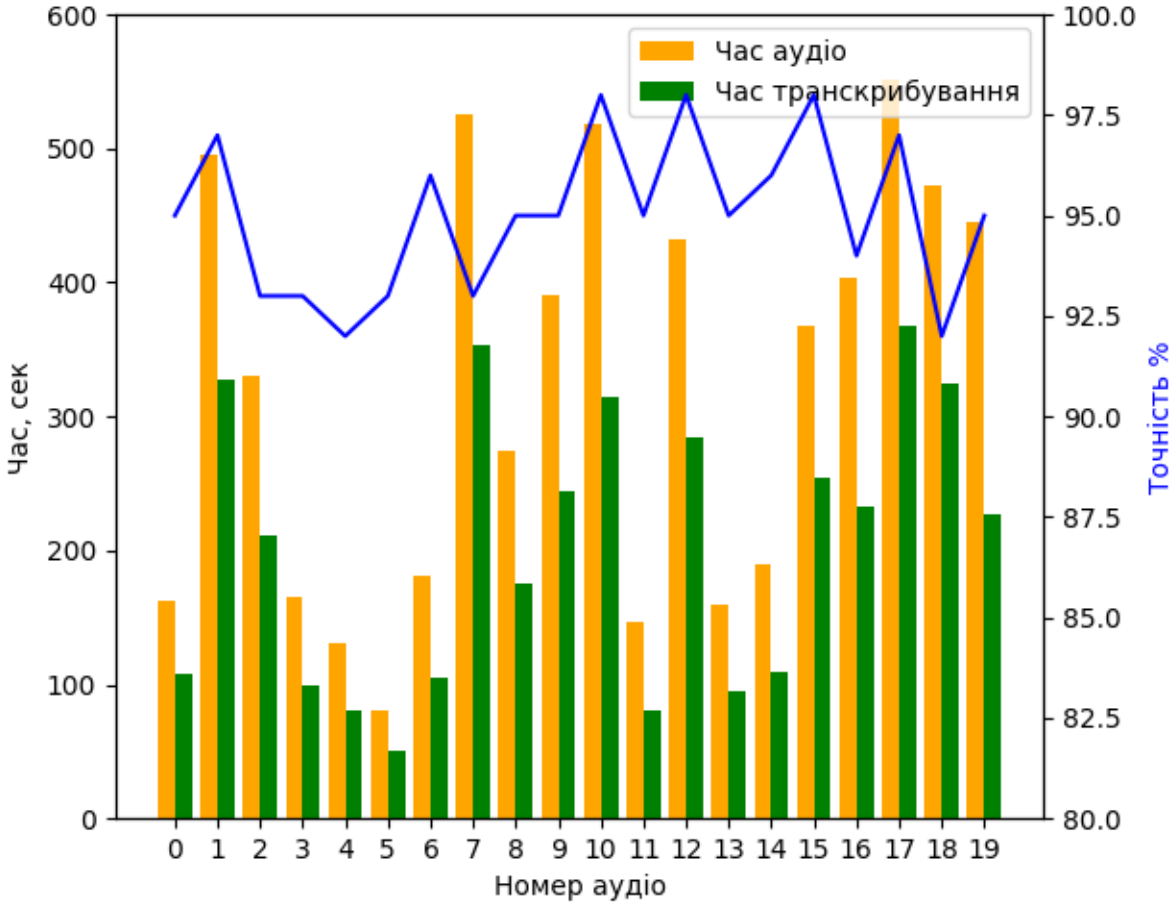

Рисунок 17 – Графік точності та часу транскрибування аудіо, порівняно з його часом

При перевірці було використано 20 аудіофайлів, з часом в проміжку від 1 до 10 хвилин. Помаранчеві стовпці демонструють час відео, в свою чергу зелені – час транскрибування. Яв видно з результатів, точності, продемонстрованої на жовтому

графіку, найменший її показник 92%, а найбільший, в свою чергу – 98%. Таким чином, транскрибування відео автоматично не лише зменшую витрачений час, порівняно з тим, який би оператор витратив лише на прослуховування, а й має пристойний показник якості, в неменше за 92 відсотки.

Відношення $\frac{t_{\text{аудіо}}}{t_{\text{транскрибування}}}$ демонструє, у скільки разів час машинного транскрибування, перевищує час аудіофайла витрачений лише на його прослуховування. На рисунку 18 продемонстровані значення таких співвідношень, між раніше обробленими аудіо.

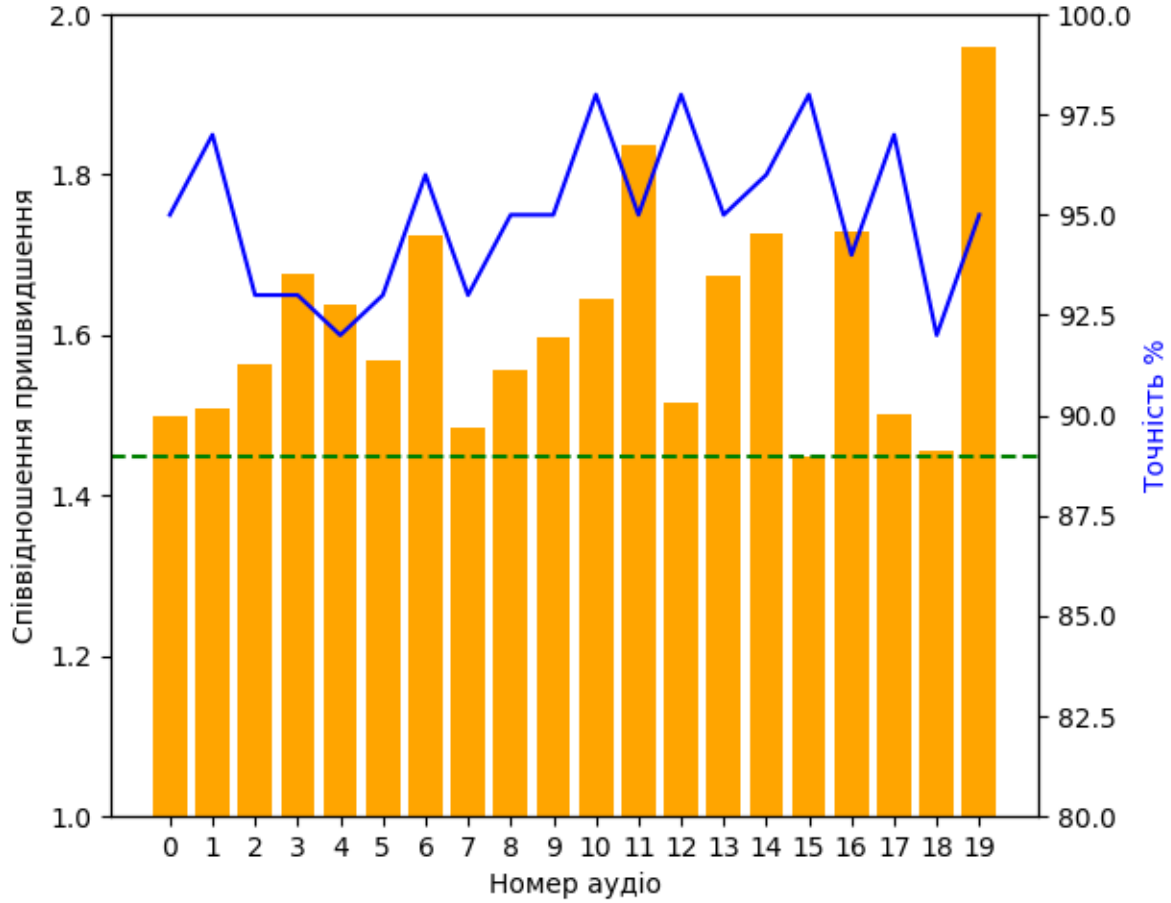

Рисунок 18 – Графік співвідношення часу транскрибування аудіо та його точності

Графік демонструє, незалежність зміни співвідношення між часом транскрибування та його точністю. Проаналізувавши графік, наочно видно, що мінімальне пришвидшення часу транскрибування, порівняно часом

прослуховування аудіофайлу, сягає близько 1.45 раз, в свою чергу максимальне пришвидшення, сягає позначки майже в 1.95. Дані результати, безумовно залежать від потужності сервера, який виконує обробку аудіофайлів, однак високі показники точності та пришвидшення транскрибування, мінімум, майже в півтора рази, наочно демонструють правильність розробки системи.

Природно, що в реальних умовах аудіо, з якими може працювати система транскрибування, буде містити шум різної природи та інтенсивності. Тому, щоб перевірити ефективність моделі на більш реалістичних даних, проведено додаткові тестування на аудіофайлах з різним відсотком постійних шумів. Результати перевірки наведено на рисунку 19.

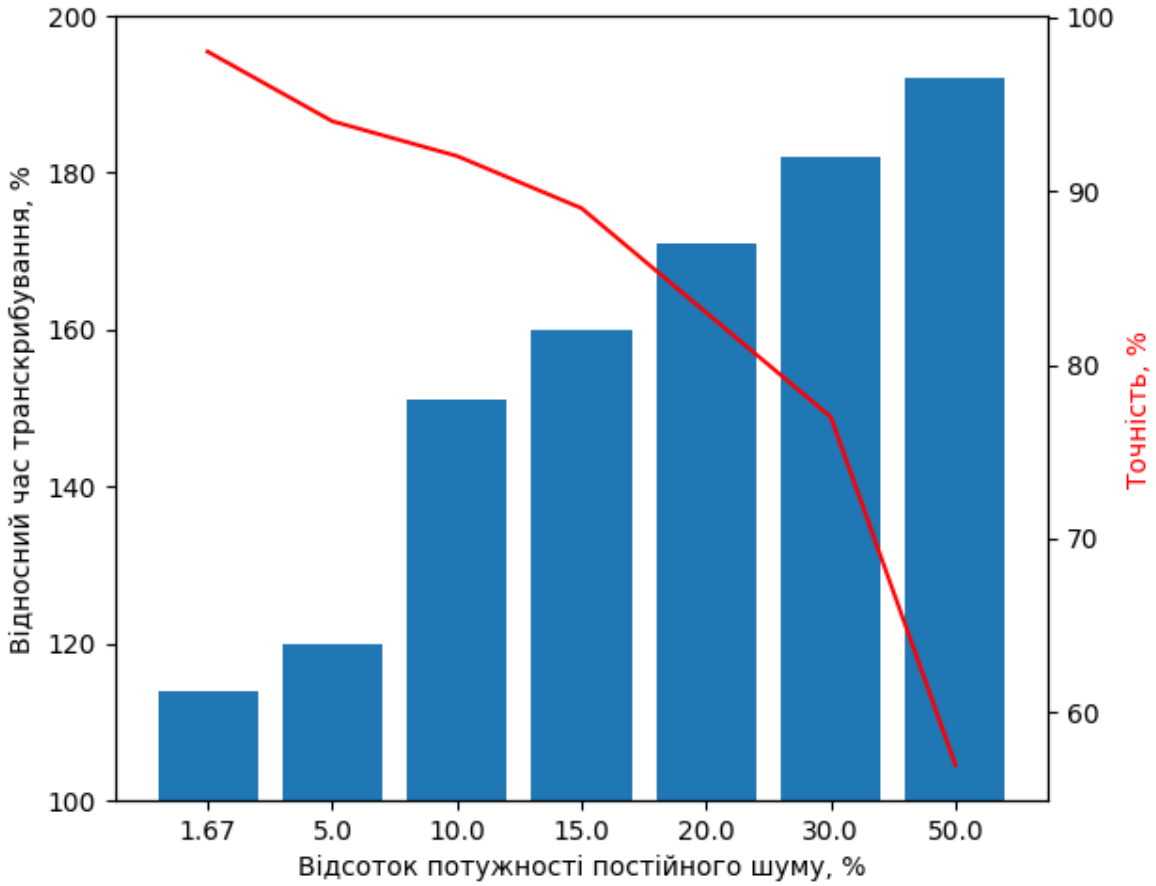

Рисунок 19 – Графік відносного часу транскрибування та точності, в залежності від інтенсивності постійного шуму

Дослідження проводились з використанням еталонного не зашумленого файлу, після чого створювались копії з накладання постійного шуму різної

потужності. Потужність шуму була в межах [1.67%:50%], відповідно час транскрибування зріс у проміжку від 112% до 195%, в свою чергу якість транскрибування знизилась з 97% еталонного файлу, до 57% з потужністю постійного шуму в 50% від інформаційного сигналу. Ці дані можуть бути корисними для вдосконалення системи транскрибування, з метою забезпечення більш точного результату для аудіофайлів з високим рівнем шуму. Такі дослідження також можуть бути корисними для визначення потреби в застосуванні інших методів обробки звуку, які можуть допомогти зменшити ефект постійного шуму в аудіофайлах.

Отримані результати перевірки системи транскрибування свідчать про її високу ефективність та продуктивність. Зокрема, точність транскрибування нашої моделі сягає понад 90%. Однак, необхідно зазначити, що точність транскрибування може бути підвищена за рахунок додаткової обробки та підготовки аудіофайлів перед їх поданням на вхід системі. Крім того, важливо зазначити, що час моделі є майже в півтора рази менше, ніж час аудіофайла, що дозволяє швидко та ефективно обробляти великі обсяги аудіоінформації. Отже, можна зробити висновки, що розроблена модель для транскрибування аудіофайлів на основі LSTM-шарів та механізму уваги є ефективною та продуктивною, та може бути використана в різних сферах, де необхідна швидка та точна система транскрибування аудіоінформації.

Висновки до розділу 2

Збір даних є однією з найважливіших складових процесу навчання моделі транскрибування. Для забезпечення високої якості транскрипції необхідно мати достатньо велику та репрезентативну вибірку аудіофайлів для тренування моделі. Наприклад, дані можуть бути зібрані з різних голосів, акцентів, швидкостей мовлення та інших варіантів, щоб покрити якомога більший спектр можливих вхідних даних. Крім того, важливо забезпечити достатній обсяг даних, щоб модель мала достатню кількість прикладів для навчання та могла розпізнавати різні

варіанти вхідних даних. Окрім цього, необхідно забезпечити належну якість аудіофайлів, що включає в себе якість запису та чіткість звуку. Важливим кроком у процесі збору даних є їх анотування. Це означає, що кожен аудіофайл повинен бути супроводжуваний текстовим файлом з транскрипцією мовлення. З дотриманням зазначених вимог розроблено програмну компоненту, що дозволяє зберігати відеофайли з веб-сервісу YouTube та дотичну інформацію, для подальшого аналізу доцільності використання подібних відео, каналу автора, тощо. Для анотування даних використовуються субтитри, надані ресурсом, отримані субтитри слугують джерелом інформації, для фільтрації аудіо, взятого з завантаженого відео. Таким чином відбувається автоматичний збір різноманітних високоякісних даних.

Розроблена модель машинного навчання, яка працює з аудіофайлами та генерує на їх основі текстовий опис, складається з енкодеру та декодеру. Енкодер трансформує аудіофайли у послідовність векторів за допомогою LSTM шару, а декодер генерує текстовий опис на основі цих векторів та розпізнаного тексту. Механізм уваги використовує точкове множення та конкатенацію векторів, щоб зосередитись на контексті аудіофайлу та згенерованих раніше словах. Остаточний результат проходить через шар з функцією активації softmax. Модель має два рекурентних LSTM шари та використовує категоріальну хрест-ентропію як функцію втрат для навчання.

На онові розробленої нейронної мережі спроектовано та розроблено інфраструктуру, що дозволяє транскрибувати аудіо- чи відеофайли. Розроблений застосунок включає в себе візуальну компоненту, реалізовану через веб-браузер, з використанням технології Vue.js, сховище BLOB-storage, яке використовується для зберігання аудіо- чи відеофайлів, реляційну базу даних, яка слугує для зберігання історії користування застосунком, Python сервер на основі Flask, який у відповідь на запит завантажує бажаний файл зі сховища та повертає транскрибовану інформацію, використовуючи розроблену та навчену нейронну мережу, Ocelot маршрутизатор, який дозволяє уникнути механічного задавання портів та адресів програмних компонент, та приховує фізичні адреси від користувачів, та Azure AD,

що надає токени для доступу в систему, захищаючи користувачів та інфраструктуру від несанкціонованого доступу, що може призвести до втрати особистих даних, або ж ускладнення роботи продукту.

В ході перевірки показників якості застосунку продемонстровано на графіках статистики, що точність транскрибування якісних аудіофайлів сягає мінімум 90%, а час транскрибування, порівняно з часом вхідного файлу, мінімум зменшується в 1.45 разів. Ба більше, доведено, в ході аналізу статистики, представленої на графіках, час вхідного аудіофайла не впливає, на точність отриманої транскрипція, тобто аналіз великих аудіофайлів не потребує зміни алгоритму. До того ж проведено дослідження впливу постійного шуму різної відносної потужності. При накладанні шуму до 20% від потужності інформаційного сигналу, що насправді суттєво заважає людському вуху сприймати інформацію, точність системи спадає максимум на 15%. Однак при подальшому збільшені потужності шуму, показники точності можуть впасти до 50%. До того ж, час транскрибування аудіо може збільшуватись майже вдвічі. Що наочно доводить наступне твердження: забезпечення максимальної точності транскрибування рекомендується використовувати якісний звуковий запис.

Таким чином розроблена та протестована система включає в себе розгалужену, захищену від різного роду несанкціонованого доступу, інфраструктуру. Показники якості системи вказують на її цілковиту ефективність, у випадках використання якісного та зашумленого, не більш ніж на 20% потужності, файлу. Для покращення системи варто провести додаткові дослідження з фільтрації аудіо. Та, при позитивних результатах, імплементувати відповідну компоненту.

3 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

Транскрибування голосових повідомлень набуває дедалі більшої актуальності в світі та є важливим інструментом для збору та аналізу даних, зокрема у галузях науки та досліджень. Так, шляхом транскрибування аудіо- або відеозаписів можна зібрати значну кількість даних у мовознавстві, соціології, політичних науках, маркетингу та багатьох інших галузях. Крім того, транскрибування може стати основним інструментом спілкування для людей з порушеннями слуху, що дозволяє їм легше розуміти оточуючих.

Однак, використання систем автоматичного розпізнавання мовлення передбачає наявність якісного звукового ряду. Це означає, що будь-яке шумове середовище, з яким зазвичай доводиться мати справу, може значно сповільнити роботу системи. Таким чином, для забезпечення ефективності та точності автоматичного транскрибування, необхідно забезпечити належну якість вхідного аудіо- або відеозапису.

Це викликає потребу в порівнянні розробленого продукту, з уже існуючими системами, виділення слабких та сильних сторін кожного з них та порівняння стійкості до різного роду завад. Важливим етапом є проведення дослідження, задля перевірки механізмів покращення аудіофайла, та вплив отриманих результатів покращення, на подальшу ефективність роботи системи транскрибування.

Не менш важливим є репрезентативність моделі, тобто наскільки точно модель відображає реальний світ або вибірку даних, на яких вона була навчена. Якщо модель недостатньо репрезентативна, вона може працювати погано на нових даних, які вона не бачила раніше. Основною проблемою сучасних систем транскрибування є їх цільова сфера застосування, транскрибування здебільшого пропонується у якості інструменту створення штучних асистентів, через що системи орієнтуються на повсякденний набір слів, у який не входять складні наукові терміни, скорочення та абревіатури. У відповідність цьому, варто перевірити ефективність роботи існуючих систем та розробленої моделі транскрибування, яка навчалась на різноманітних науково популярних текстам,

через що повинна мати змогу видавати точне текстове розшифрування подібного роду аудіофайлів.

3.1 Порівняння розробленої системи транскрибування з існуючими

У сучасному світі машинного навчання та обробки природних мов існує велика кількість різних інструментів та технологій, які дозволяють створювати моделі для аналізу тексту, розпізнавання мови та інших задач. Одним з таких інструментів є WhisperAI – інноваційна платформа для аналізу мови, яка використовує штучний інтелект для розуміння тексту та мовлення. Однак, не завжди існуючі моделі відповідають конкретним потребам користувачів, тому часто необхідно створювати власні моделі. У даному дослідженні створено власну модель

В першу чергу проведемо порівняння точності роботи обох інструментів та часу, відносному часу витраченому на транскрибування, порівняно з часом аудіозапису. Перший порівняльний експеримент проводиться на основі типових стандартних слів та речень, що зазвичай використовується в людському побуті, у випадку використання розробленої моделі в якості асистента, якість та швидкодія реакції на подібні стандартні дані, повинна бути якомога швидша. Тестові дані не несуть в собі відчутних постійних шумових завад, задля коректності проведення первинного дослідження. Для проведення повноцінного експерименту використано голоси різних груп людей, такі як: дитячі голоси, жіночі голоси, чоловічі голоси та голоси людей поважного віку. Зібрані голоси розбито на три групи за гучністю їх мовлення, а саме:

- шепіт, з амплітудою мовлення 0.075-0.2 мм, дані такого роду не повинні мати вирішального впливу на оцінку якості роботи застосунку, однак їх аналіз має право на існування;
- буденна мова, з амплітудою 0.5-2 мм, тобто такі, з якими система транскрибування зустрічається найчастіше;

– крик, який має потужність в 5-10 мм, який, на відміну від шепоту, часто зустрічається при транскрибуванні файлів.

Для зручності аналізу та читання графіків, дві метрики, тобто точність та відносний час розбито на два графіка, що не перенасичувати інформацією один рисунок. На рисунку 20 продемонстровано порівняння відносного часу транскрибування трьох тестових груп.

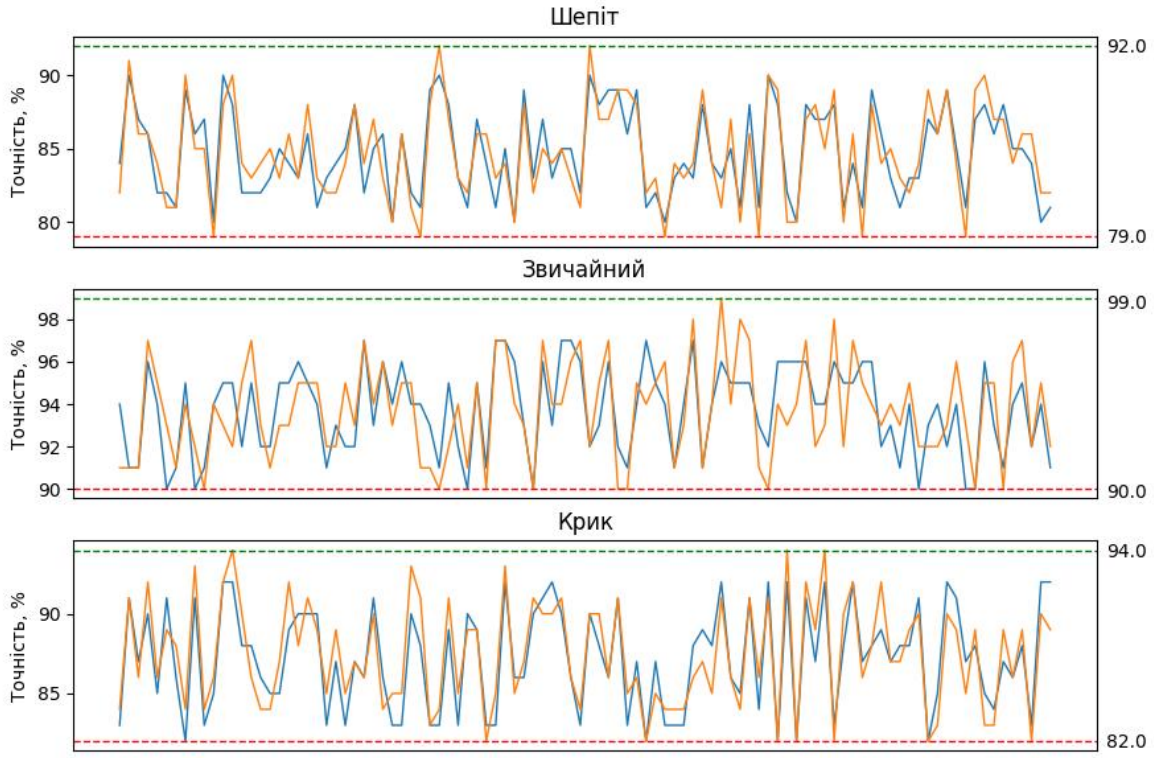

Рисунок 20 – Графік порівняння точності транскрибування розробленої моделі та WhisperAI

Синім кольором на графіках представлені дані зібрані з власної моделі, у свою чергу помаранчевим – отримані за допомогоюWhisperAI. На рафіках представлена наведена точність системи, по осі Y, в свою чергу, по осі X відображені різні номери експериментів. Графіки демонструю, подібність точності роботи застосунків, з типовими наборами даних. Як і очікувалось, точність трансрибування шепоту знаходить в діапазоні 79-92%, з середнім показником точності в 86%. Найліпші випадки транскрибування, в 92%, виділені зеленою

лінією, продемонстровано технологією WhisperAI, однак найгірші, в 79, виділені червоною лінією, теж продемонстровані при транскрибуванні з використанням WhisperAI. Звичайний голос же, природньо, продемонстрував більш високі показники точності, а саме проміжок в 90-99%, де загалом, WhisperAI надавав трошки кращі кіпи, іноді сягаючи 3-5%, однак траплялись і зворотні випадки, що вказує на подібність рівня точності моделей, при перевірці звичайних слів. Графік крику демонструю трішки гіршу картину, з точність в межах 80-94%, однак обидві моделі продемонстрували схожі результати, що свідчить про високий рівень навченості українській мові розробленої моделі, враховуючи популярність WhisperAI. У свою чергу, графік порівняння часу транскрибування $\frac{t_{\text{аудіо}}}{t_{\text{транскрибування}}}$ на різних тестових наборах представлено на рисунку 21.

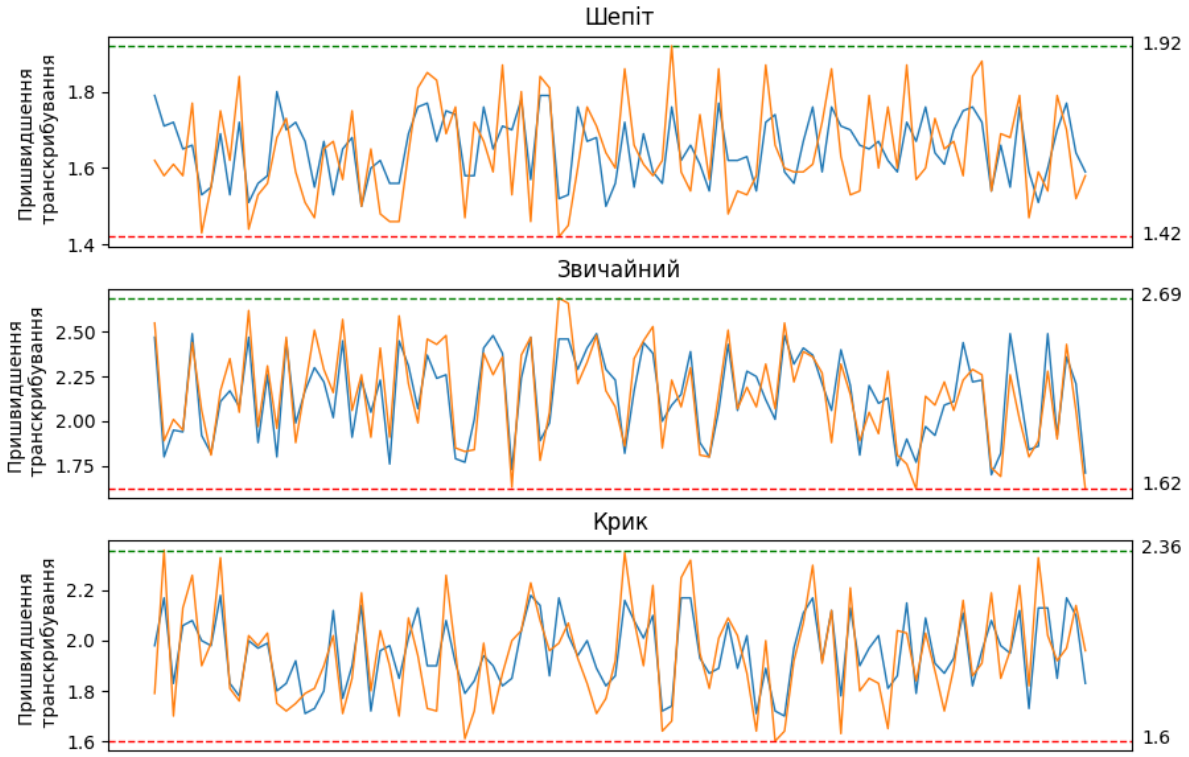

Рисунок 21 – Графік порівняння відносного часу транскрибування розробленої моделі та WhisperAI

На графіку наведено дані проведення експерименту з пришвидшенням часу транскрибування, відносно базової довжини аудіо. Дані зібрані паралельно, з

даними точності, тож аналогічно представлено три графікі, тій самій вибірці, де синя ламана – власна розроблена модель, а помаранчева – WhisperAI. Аналогічно до показників точності, найменше пришвидшення часу отримано на вибірці з шепотом, наступний за швидкістю – крик, а, природньо, найкращий – звичайний голос. Така поведінка зумовлена специфікою роботи нейронних мереж, у яких реалізовано механізм ігнорування певних даних, що заважають прийняти рішення, через що й збільшується загальний час аналізу. Як видно з графіка, пришвидшення транскрибування відносно часу аудіозапису з шепотом сягає проміжку в 1.42-1.92 рази, в свою чергу крик має відповідні показники на рівні 1.6-2.36 разів, а звичайний голос має показники в 1.62-2.69 раз. Варто підкреслити, що середня різниця між результатами звичайного голосу – найменша. Це може свідчити про великі набори тренувальних даних саме за таких умов. Розкид же шепоту та крику більш великий, хоча й діапазон та середні значення подібні, що може відображати, меншу кількість та репрезентативність використаних тренувальних даних, на що в майбутньому, за потреби аналізувати такі звуки, варто звернути увагу, задля покрашення швидкодії та точності моделі. Дані експерименти наочно демонструють, що рівень точності розробленої моделі сягає значень технології WhisperAI, з невеликою різницею, на рівні статистичної похибки, в 2-5%, а відносний час транскрибування

Експеримент продемонстрував подібність, з мінімальними відхиленнями, в швидкості та точності роботи власної моделі та моделі WhisperAI, однак чи буде ситуація подібна з аналізом науково популярних файлів, що мають в собі велику кількість специфічних слів та термінів. Даний експеримент проведено лише з вибіркою звичайного голосу в межах амплітуди 0.5-2мм та без істотних завад. На рисунку 22 продемонстрованого графіки точності та пришвидшення часу автоматичного транскрибування, відносно часу аудіофайлів. Вибірка сягає, як і в попередньому дослідженні 100 аудіофайлів різної довжини. Графік точності наочно демонструє суттєву різницю між якістю роботи власної розробленої моделі та WhisperAI. В той час, як пік якості транскрибування науково популярних файлів розробленою системою сягає 94.81%, найменша якість обробки текстів з

використанням WhisperAI може дорівнювати 54%. Безумовно, вершини та низи не дають повноцінної картини, однак загалом, середнє значення точності роботи розробленої моделі на науково популярних текстах дорівнює 84.83%, на тестовій вибірці. В той самий час, як аналогічна вибірка оброблена з середньою точністю 72.63% за допомогою WhisperAI.

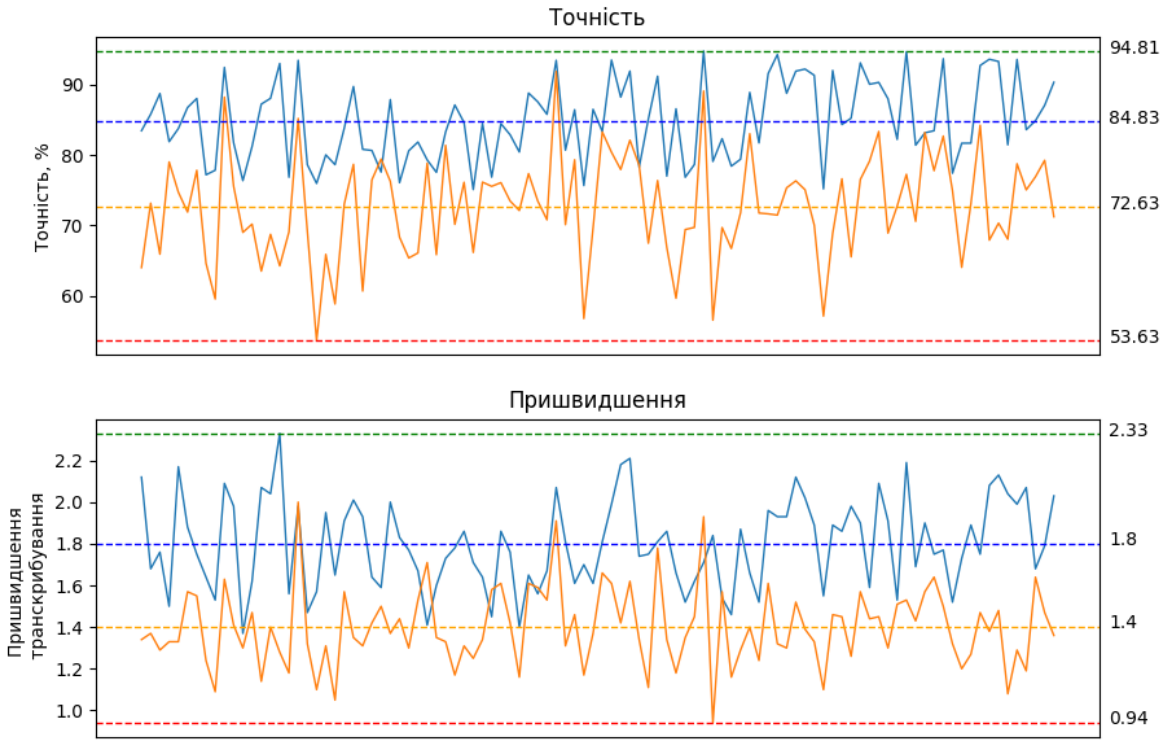

Рисунок 22 – Графік порівняння точності та відносного часу транскрибування розробленої моделі та WhisperAI на вибірці науково популярних даних

В свою чергу, різниця між часом транскрибування не настільки велика, у відсотковому значенні. Однак все ще, піки пришвидшення розробленої моделі сягають 2.33 рази, в той самий час, як низи WhisperAI – 0.94 рази. Таким чином, WhisperAI витратив навіть більше часу на транскрибування погано відомих йому слів, ніж час самого аудіозапису. Хоча, середній час пришвидшення WhisperAI сягає 1.4 рази, а розробленої моделі – в 1.8 разів.

Дана поведінка зумовлена специфічністю набору тренувальних даних, що були зібрані на першому етапі підготовки до розробки системи, та в подальшому

використані для навчання моделі. Що явно демонструє перевагу розробленої моделі, для транскрибування науково популярних, чи навіть наукових записів. Однак так чи інакше, обидві системи дозволяють пришвидшити загальний час роботи з даними та позбутись людського фактору, оглянутих показників якості, цілком може бути достатньо, для специфічних завдань.

Крайній експеримент спрямовано на порівняння показників якості роботи розробленої моделі та моделі WhisperAI при різному відсотку потужності шуму. На рисунку 23 продемонстровано графіки збільшення часу транскрибування розробленої моделі та моделі WhisperAI, порівняно з часом транскрибування файлу без накладання постійного шуму, та графік точності моделей, в залежності від рівня шуму.

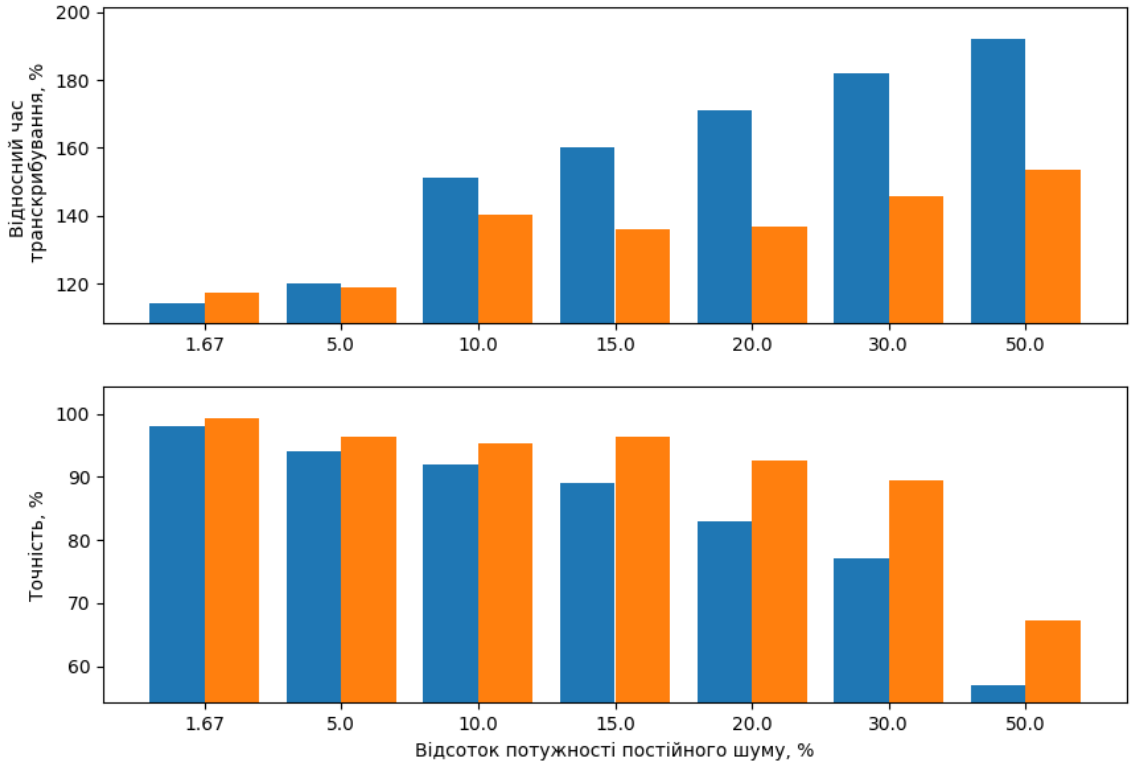

Рисунок 23 – Графік порівняння точності та відносного часу транскрибування розробленої моделі та WhisperAI на вибірці з накладанням шуму

Графік зростання відносного часу транскрибування, до часу транскрибування файлу без шуму демонструє, більшу пристосованість моделі WhisperAI, ніж

розробленої власної моделі. На що прямо вказує значно менше збільшення відносного часу транскрибування в проміжках [115%:153%] для технології WhisperAI та відповідно [113%:191%] для розробленої власноруч системи. А точність, в свою чергу спадаю більш повільно. Таким чином початкова точність сягає 99% у готової моделі, порівняно з 98% у власноруч розробленої, що більше схоже на статистичну похибку, ніж на реальні розбіжність, а от точність транскрибування файлу з 50% потужності шуму відрізняється більш помітно, 67% для моделі WhisperAI та 57% для розробленої.

Відповідно до проведених експериментів можна зробити наступні висновки, а саме: розроблена власноруч модель має схожі показники якості, до готової моделі транскрибування WhisperAI, при порівнянні на типових даних. Безперечною перевагою розробленої моделі є її високий рівень точності та швидкодії, при аналізі науково популярних записів, які в широкому обсязі використовувались під час збору даних та навчанні нейронної мережі, що робить дану розробку перспективною для транскрибування спеціалізованих записів українською мовою. Вочевидь транскрибування записів з високим вмістом шуму складно дається нейронній мережі, що призводить до думки стосовно доцільності проведення додаткового аналізу по вилучення шуму з наступним транскрибуванням отриманих файлів. У разі успіх зазначеного дослідження, модель можна вважати кращою, за існуючі аналоги.

3.2 Покращення слабких місць моделі

Задля підвищення якості обробки зашумлених записів, прийнято рішення виконання попередньої фільтрації аудіофайлів. Безумовно, можна збільшити кількість тренувальних даних, на додачу використовуючи замушлені зразки, однак це може викликати перенавченість системи, тобто її здатність якісно вирішувати подібні де тренувальних даних задачі, однак нездатність ефективно приймати рішення з небаченими до цього даними. До того ж, профільне спрямування моделі на різноманітні сфери науково популярного контенту, спричинить потреби в

додаткових різноманітних зашумлених даних, що може бути важко реалізуємо. Відповідно для оцінки ефективності роботи фільтрів було сформовано три набори аудіоданих:

– «чистий звук» – це звук, який найбільш наближений до оригіналу, тобто не містить шуму, завад, спотворень, артефактів тощо, тобто такий, де відсоток потужності постійного шуму не перевищує 0.1% відсотка від потужності інформаційного сигнала, цей звук людині здається майже ідеальним;
– «частково зашумлений» – цей звук містить складові шуму, але все ще зберігає деяку чіткість та розрізнення між різними звуками, тобто такий, де потужність шуму не перевищує 10% від потужності вхідного аудіо; переважну більшість слів можна ідентифікувати з першого разу, за винятком деяких, іноді можуть виникати деякі випадкові звуки;
– «відчутно зашумлений» – цей звук містить наскрізний шум, такий що потужність шуму, відносно інформаційного сигналу сяє мінімум 30%; слова хоча і можуть бути розпізнані, але вимагають додаткових зусиль, цей звук може бути досить неприємним для слуху людини.

Для відсоткового порівняння швидкості транскрибування двох файлів в парі використовується формула (1).

$$x = \left[1 - \frac{t_ф + t_{mф}}{t_m}\right] * 100\%, \quad (1)$$

де $t_ф$ – час попередньої фільтрації файлу;

$t_{тф}$ – час транскрибування відфільтрованого файлу;

$t_т$ – час транскрибування початкового файлу.

Природньо, що чим більше отриманий показник, тим результат фільтрування кращий. При проведені експериментів усі фільтри є фільтрами Баттерворда 6 порядку та коефіцієнтом підсилення 1.

На основі груп аудіоданих, які були описані вище, було проведено перевірку якості роботи фільтра високих частот (ФВЧ). Людський голос може займати частотні проміжки від 20 Гц до 20 кГц. Спочатку фільтрація була виконана з

нижньої межі в 50 Гц, а далі поступово збільшувалися значення частот, доки не було помічено погіршення результату.

Результати фільтрації подані у вигляді відсоткового співвідношення швидкості транскрибування відфільтрованих файлів, до початкових даних наведені в таблиці 1.

Таблиця 1 – Результати роботи ФВЧ

Частота ФВЧ, Гц	Чистий звук, %	Частково зашумлений, %	Відчутно зашумлений, %
50	2	-17	29
100	6	5	35
150	3	5	23
200	-5	12	21
250	-8	-49	20

Як видно з таблиці, в усіх випадках спостерігається пришвидшення роботи системи транскрибування. Як зазначалось вище, фільтрація може призводити до збільшення загального часу фільтрування, наприклад як в ситуаціях з чистим та частково чистим звуками, на частоті в 250 Гц.

Аналогічний експеримент було виконано з фільтром низьких частот (ФНЧ), результати якого представлені в таблиці 2.

Таблиця 2 – Результати роботи ФНЧ

Частота ФНЧ, Гц	Чистий звук, %	Частково зашумлений, %	Відчутно зашумлений, %
3000	-3	-25	7
3500	-10	-19	13
4000	1	-24	33
4500	0	-18	7
5000	-5	8	22
5500	-1	-21	20

Продовження таблиці 2

6000	-6	-20	41
6500	0	-13	7
7000	-3	-21	33

З отриманих результатів видно, що вплив ФНЧ значно менш передбачуваний, ніж вплив ФВЧ. Це може бути зумовлено специфікою навчання нейронної мережі. Ключові особливості слів, по яким мережа приймає рішення, значно відрізняються від людського сприйняття. Крім того, високочастотний шум зазвичай не зустрічається в побуті, на відміну від низьких глухих звуків. Тому ФНЧ, хоч і пришвидшує роботу системи, в ситуації відчутного зашумлення, псує результат в інших випадках. Тому доцільно розглянути роботу смугових фільтрів.

В таблиці 3 представлено результати роботи смугового фільтра з нижньою частотою 50 Гц та верхньою частотою смуги пропускання від 3000 Гц до 7000 Гц.

Таблиця 3 – Результати роботи смугового фільтра з нижньою частотою 50 Гц

Верхня частота смуги пропускання, Гц	Чистий звук, %	Частково зашумлений, %	Відчутно зашумлений, %
3000	4	-2	21
3500	-38	-10	-5
4000	-1	-25	26
4500	2	-20	45
5000	-1	-19	34
5500	4	-20	1
6000	1	-20	4
6500	7	-22	23
7000	-2	-2	-12

На підставі результатів, що представлені в таблиці, можна зробити висновок, при нижній частоті 50 Гц та частоті смуги пропускання 4-5 кГц смуговий фільтр сприяє поліпшенню результатів.

Результати роботи смугового фільтра з нижньою частотою смуги пропускання 100 Гц наведено в таблиці 4.

Таблиця 4 – Результати роботи смугового фільтра з нижньою частотою смуги пропускання 100 Гц

Верхня частота смуги пропускання, Гц	Чистий звук, %	Частково зашумлений, %	Відчутно зашумлений, %
3000	3	-21	26
3500	-7	-21	25
4000	-5	13	24
4500	3	7	8
5000	-5	8	19
5500	-8	5	25
6000	4	-19	34
6500	-7	5	20
7000	2	-23	22

Як видно з таблиці 4, смуговий фільтр з нижньою частотою смуги пропускання 100 Гц показав себе значно краще, оскільки наявні як суто додатні рядки, так і помітне пришвидшення в двох з трьох груп, при незначному погіршенні третьої.

Наступним кроком розглянемо роботу смугового фільтра з нижньою частотою смуги пропускання 150 Гц, результати досліджень наведені в таблиці 5.

Таблиця 5 – Результати роботи смугового фільтра з нижньою частотою смуги пропускання 150 Гц

Верхня частота смуги пропускання, Гц	Чистий звук, %	Частково зашумлений, %	Відчутно зашумлений, %
3000	-7	-24	24
3500	0	12	31
4000	-4	3	8

Продовження таблиці 5

4500	-6	7	24
5000	5	7	23
5500	-6	7	33
6000	-6	12	23
6500	-4	5	23
7000	-6	4	25

Як видно з результатів роботи смугового фільтра з нижньою частотою смуги пропускання 150 Гц, отримано суттєве пришвидшення при верхній частоті смуги пропускання 3500 Гц та 5000 Гц. Однак загалом, варто зазначити, що починаючи з верхньої частоти смуги пропускання 3500 Гц включно, система видавала вищу швидкість роботи в зашумлених випадках, порівняно з сповільненням для чистого звуку. Наявність чистого звуку, враховуючи розповсюдженість використання систем транскрибування – малоймовірна. Тож в широкому розумінні, смуговий фільтр з нижньою частотою 150 Гц та верхньою частотою смуги пропускання 3500-7000 Гц, наразі можна вважати універсальним рішенням.

Розглянемо вплив смугового фільтра з нижньою частотою смуги пропускання 200 Гц (таблиця 6).

Таблиця 6 – Результати роботи смугового фільтра з нижньою частотою смуги пропускання 200 Гц

Верхня частота смуги пропускання, Гц	Чистий звук, %	Частково зашумлений, %	Відчутно зашумлений, %
3000	-41	2	8
3500	-6	7	7
4000	2	7	51
4500	4	7	25
5000	1	1	49
5500	-7	1	24

Продовження таблиці 6

6000	-6	4	25
6500	-1	3	23
7000	-7	7	24

Як видно з результатів отриманих від роботи смугового фільтра з нижньою частотою смуги пропускання 200 Гц, однозначно можна відкинути варіант використання верхньої частоти смуги пропускання 3000 Гц, однак всі інші варіанти, особливо діапазон 4000-5000 Гц показав дуже високі результати при значній зашумленості, та не тільки не сповільнив, а й пришвидшив роботу при транскрибування інших двох груп.

І, нарешті, розглянемо результати роботи смугового фільтра з нижньою частотою смуги пропускання 250 Гц, наведені в таблиці 7.

Таблиця 7 – Результати роботи смугового фільтра з нижньою частотою смуги пропускання 250 Гц

Верхня частота смуги пропускання, Гц	Чистий звук, %	Частково зашумлений, %	Відчутно зашумлений, %
3000	-6	9	23
3500	-5	-13	43
4000	-1	10	24
4500	3	-45	45
5000	-7	13	32
5500	-5	16	24
6000	-4	2	18
6500	-25	-11	24
7000	-18	1	25

Як видно з результатів роботи, смуговий фільтр з нижньою частотою смуги пропускання 250 Гц, як був сумнівним варіантом при використанні лише ФВЧ з частотою зрізу 250 Гц, так ним й залишився.

Для перевірки точності та часу транскрибування використаємо випадково обрані фільтри в діапазоні смуги пропускання 150 - 3500 Гц, 150 - 5000 Гц, 200 - 4000 Гц, 200 - 4500 Гц, 200 - 5000 Гц. Таким чином, кожну секунду до аудіозапису застосовується фільтр з вищезазначеного діапазону, це потрібно для того, щоб не проводити більш складний попередній аналіз вхідного звуку, на предмет використання конкретного з фільтрів. На рисунку 24 продемонстровано порівняння показників якості роботи моделі з вхідним зашумленим звуком та зі звуком, що був попередньо профільтрований.

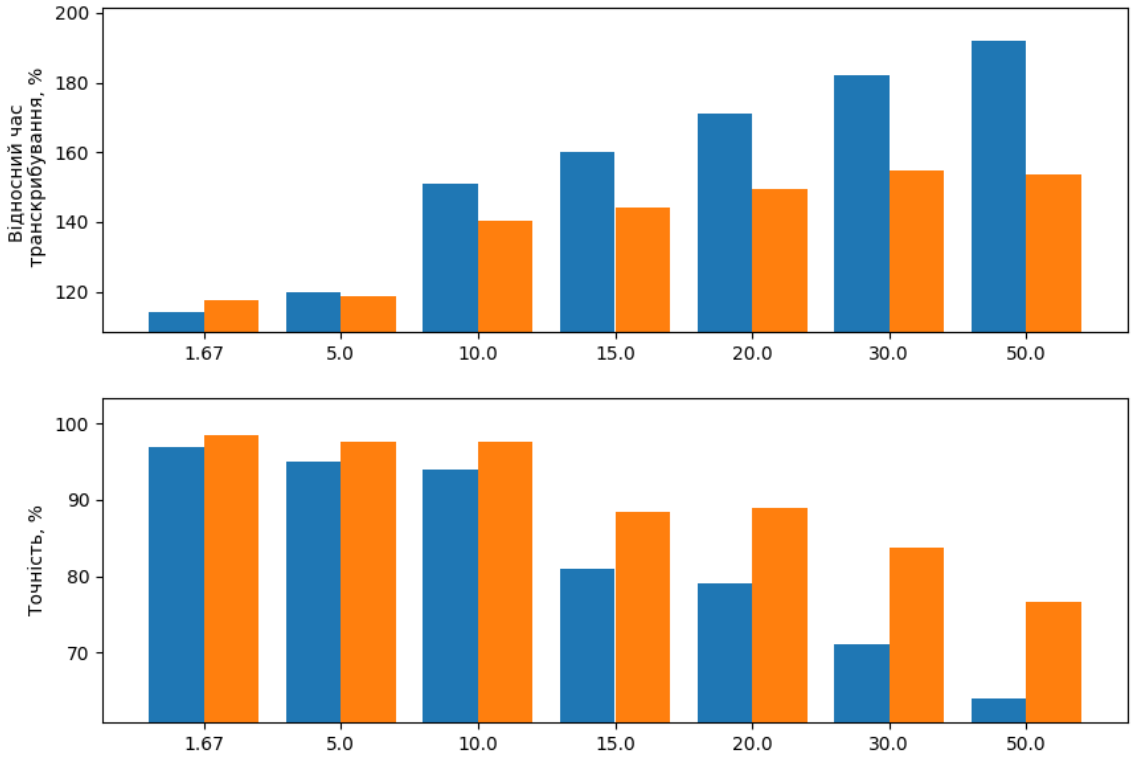

Рисунок 24 – Графік порівняння точності та відносного часу транскрибування розробленої моделі з та без використання фільтрації

Графік демонструє порівняння роботи моделі з та без фільтрації. Синім кольором позначено роботу власної моделі, а помаранчевим – власної моделі з реалізованим механізмом фільтрації. Графік наочно демонструють покращення показників якості з використанням фільтрів, а саме: зменшення відносного часу транскрибування з 192%, при потужності постійного шуму в 50% від

інформаційного сигналу, до 153% з використанням фільтрів. Також точність зросла з 64% до 76%, що наочно демонструє доцільність та ефективність використання розробленого алгоритму використання попередньої фільтрації аудіофайлів. Однак варто зазначити, що при використанні фільтрів до «чистого» звуку, результат може бути зворотній, через нехтування певною глибиною людського голосу.

Результати експериментальних досліджень було подано у статті «Preprocessing of audio data for voice transcription systems» Драган М.С., Писаренко А.В. у міжвідомчому науково-технічному журналі «Адаптивні системи автоматичного управління». – 2023. № 1(42). С. 39-48 [36].

Висновки до розділу 3

У ході проведення експериментів з тестування та порівняння моделі розробленого застосунку було виявлено кілька сильних та слабких сторін продукту, у порівнянні з відомими системами на ринку. У процесі тестування було наочно продемонстровано наступні сильні та слабкі сторони розробленого продукту. Розроблена модель має подібні показники якості до готової моделі на типових даних. Однак, перевагою розробленої моделі є висока точність та швидкодія при аналізі науково-популярних записів, які широко використовувалися під час навчання нейронної мережі. Це робить розробку перспективною для транскрибування спеціалізованих записів українською мовою. Однак, транскрибування записів з високим рівнем шуму складно дається нейронній мережі, тому були проведені додаткові експерименти саме з використанням попередньої фільтрації та подальшим транскрибуванням отриманих файлів.

В результаті попереднього фільтрування груп аудіофайлів, було наочно продемонстровано переваги використання смугового фільтра з нижньою частотою смуги пропускання в діапазоні 150-200 Гц та верхньою частотою смуги пропускання в діапазоні 3500-7000 Гц. Безумовно були виділені найкращі варіанти, а саме: діапазон смуги пропускання 150 - 3500 Гц, 150 - 5000 Гц, 200 - 4000 Гц, 200 - 4500 Гц, 200 - 5000 Гц. Однак, загалом використання зазначених фільтрів в

описаних діапазонах дозволяє пришвидшити процес транскрибування тексту. Таким чином, можна досягти збільшення швидкості транскрибування не тільки за рахунок використання відеокарт, але й використання центральних процесорів та попередньої фільтрації. Надалі можливо більш детально розглянути використання смугових фільтрів з нижньою частотою смуги пропускання в діапазоні 150-200 Гц верхньою частотою смуги пропускання в діапазоні 3500-7000 Гц. До того ж розглянути видалення пустих сегментів по енергетиці сигналу та вплив прискорення голосу, на час його транскрибування, з використанням вищезазначених фільтрів.

Таким чином розроблено та протестовано механізм вилучення шуму з аудіофайлів. Випадковим чином застосовується один із сформованих у список фільтрів та застосовується до запису. У результаті проведення такого експерименту вдалося значно підвищити рівень точності та швидкість транскрибування аудіофайлів. Доцільно провести додаткові дослідження у сфері фільтрації аудіосигналів, виявлені шуму та методах боротьби з ним. Однак розробленого наразі механізму боротьби з завадами достатньо, щоб стверджувати наступне: недолік розробленої моделі, в якості транскрибування фалів з високим відсотком потужності шуму компенсовано.

У ході експериментів довели перспективність розробки для транскрибування спеціалізованих науково популярних записів українською мовою. Підвищили завадостійкість отриманої моделі до завад різної сили потужності. Та продемонстрували схожість показників якості транскрибування української мови, з існуючими на ринку продуктами.

ВИСНОВКИ

Досягнуто основну мету дослідження, а саме розробка інформаційної підсистеми, що здатна відфільтрувати файл, від завад та представити відфільтрований файл у текстовому вигляді. Описано загальні підходи та методи розв’язання задачі транскрибування голосових повідомлень, підходи включають у себе: автоматичного розпізнавання мовлення, технологій мовленнєвого аналізу, технологій штучного інтелекту, декотрі інструментальні засоби. Основними методами є: фільтрація шуму, енергетичний аналіз, визначення, вимірювання та аналізу параметрів голосу, а також компресія звуку.

Впродовж виконання роботи проведено аналіз наявних методів транскрибування голосових повідомлень було виявлено, що існують різноманітні підходи до розв'язання цієї задачі, розробка спрямована на реалізацію моделі автоматичного транскрибування. Цей підхід базується на використанні комп'ютерних програм, які аналізують звукові дані та перетворюють їх у текст. Основними перевагами цього підходу є швидкість та ефективність процесу, а також можливість обробки великої кількості аудіо записів.

Під час вивчення технологій глибинного навчання виявлено, що це одна з найбільш ефективних технологій машинного навчання, яка використовується для розв'язання різноманітних завдань. Використання глибинного навчання дозволяє автоматизувати багато процесів, зокрема розпізнавання образів, розпізнавання мови, класифікацію текстів та інше. Окрім того, досліджено різні бібліотеки та інструменти для розробки глибинних нейронних мереж, зокрема TensorFlow, PyTorch, Keras, Theano та інші. Виявлено, що TensorFlow та PyTorch є найбільш популярними бібліотеками для розробки глибинних нейронних мереж, оскільки вони мають широкий функціонал та підтримуються великою спільнотою розробників. Навчання моделей глибинного навчання вимагає значних обчислювальних ресурсів, зокрема використання графічних процесорів. Також важливим кроком при розробці моделей глибинного навчання є оптимізація параметрів моделі, щоб досягти найкращих результатів.

Однією з ключових складових успішного навчання моделі транскрибування є збір відповідних даних. Для досягнення високої точності транскрипції необхідно забезпечити належну якість та репрезентативність вибірки аудіофайлів, що використовується для тренування моделі. Це означає, що дані повинні включати різні голоси, акценти, темп мовлення та інші можливі варіації, що забезпечить можливість розпізнавання різних вхідних даних моделлю. Важливим кроком у зборі даних є їх анотування, яке включає транскрипцію мовлення в текстовому форматі для кожного аудіофайлу. При зборі даних також необхідно дотримуватися вимог до якості аудіофайлів, включаючи якість запису та чіткість звуку. Для забезпечення ефективного збору та анотування даних розроблено програмну компоненту, яка дозволяє зберігати відеофайли з веб-сервісу YouTube та збирати відповідну інформацію для аналізу доцільності використання подібних відео, каналу автора тощо. Анотування даних виконується з використанням субтитрів, що надаються ресурсом, які служать джерелом інформації для фільтрації аудіо з завантаженого відео. Застосування таких методів дозволяє зібрати різноманітні високоякісні дані для тренування моделі транскрибування.

Створена модель машинного навчання, яка працює з аудіофайлами та генерує на їх основі текстовий опис. Модель складається з двох головних компонент: енкодеру та декодеру. Енкодер перетворює аудіофайли у послідовність векторів, використовуючи LSTM шар. Декодер генерує текстовий опис на основі цих векторів та розпізнаного тексту. Для зосередження на контексті аудіофайлу та згенерованих раніше словах модель використовує механізм уваги з точковим множенням та конкатенацією векторів. Остаточний результат проходить через шар з функцією активації softmax. Для навчання модель використовує категоріальну хрест-ентропію як функцію втрат та має два рекурентних LSTM шари.

У даній роботі запропоновано використання технологій, що найбільш оптимально справляються з задачею транскрибування голосових повідомлень, а також для створення візуальної складової підсистеми. Ці технології включають в себе: візуальну компоненту Vue.js, Ocelot Gateway, Azure Blob Storage, а також бібліотека Python Pydub для підготовки файлу до транскрибування. Цей набір

технологій та обрані методи для фільтрування та транскрибування аудіофайлів надають широкий спектр можливостей для створення підсистеми транскрибування голосових повідомлень на основі технологій глибинного навчання.

Таким чином, було створено та перевірено механізм для ефективного вилучення шуму з аудіофайлів. Він включає список фільтрів, один з яких випадковим чином використовується для обробки кожного запису. Цей експеримент значно покращив точність та швидкість транскрибування аудіофайлів. Хоча дослідження у галузі фільтрації аудіосигналів, виявлення шуму та методів його боротьби може бути продовжене, створений механізм є достатньо ефективним, щоб стверджувати, що недоліки моделі, пов'язані з транскрибуванням файлів з високою потужністю шуму, успішно виправлені. Під час тестування застосунку було виявлено, що точність транскрибування якісних аудіофайлів досягає не менше 90%, що підтверджується графіками статистики. Крім того, час транскрибування значно зменшується, якщо порівнювати його з часом вхідного файлу, а саме мінімум в 1.45 рази. Дослідження показали, що час вхідного аудіофайлу не має впливу на точність отриманої транскрипції. Отже, для аналізу великих аудіофайлів не потрібно змінювати алгоритм застосунку. Модель показали свою перевагу при порівнянні з WhisperAI, при транскрибування науково популярних записів. Така поведінка моделі обумовлена особливостями тренувальних даних, які були зібрані та використані для навчання.

ПЕРЕЛІК ПОСИЛАНЬ

1. Transcribing Audio: Tips and Best Practices. URL: https://transcriptionhub.com/blog/transcribing-audio-tips-and-best-practices (дата звернення 14.02.2023).
2. How to Transcribe Audio to Text: A Guide to Audio Transcription and Speech Recognition. URL: https://www.clickworker.com/customer-blog/transcribe-audio-to-text/ (дата звернення 15.02.2023).
3. Watson Speech to Text. URL: https://www.ibm.com/cloud/watson-speech-to-text (дата звернення 15.02.2023).
4. The Ultimate Guide to Audio Transcription URL: https://www.rev.com/blog/transcription/the-ultimate-guide-to-audio-transcription (дата звернення 15.02.2023).
5. QS World University Rankings. URL: https://www.topuniversities.com/university-rankings/world-university-rankings/2022 (дата звернення 15. 02.2023).
6. Daniel Jurafsky, James H. Martin. Speech and Language Processing. Prentice Hall, 2003. 1024 с.
7. A Beginner's Guide to Digital Signal Processing (DSP). URL: https://www.analog.com/en/design-center/landing-pages/001/beginners-guide-to-dsp.html#:~:text=What%20is%20a%20DSP%3F,and%20%22divide%22%20very%20quickly. (дата звернення 16. 02.2023).
8. Markov and Hidden Markov Model. URL: https://towardsdatascience.com/markov-and-hidden-markov-model-3eec42298d75. (дата звернення 16. 02.2023).
9. Unsupervised Learning of Gaussian Mixture Models on a SELU auto-encoder (Not another MNIST). URL: https://towardsdatascience.com/unsupervised-learning-of-gaussian-mixture-models-on-a-selu-auto-encoder-not-another-mnist-11fceccc227e (дата звернення 17.02.2023).

10. Hastie, Tibshirani, Friedman. The Elements of Statistical Learning. Springer, 2009. 767 с.
11. Template:Machine learning. URL: https://en.wikipedia.org/wiki/Template:Machine_learning (дата звернення: 17. 02.2023).
12. Audio Signal Processing- Understanding Digital & Analog Audio Signal Processing. URL: https://www.pathpartnertech.com/audio-signal-processing-understanding-digital-analog-audio-signal-processing/ (дата звернення 17.02.2023).
13. HOW TO BUILD AUDIO FILTER CIRCUITS. URL: https://www.circuitbasics.com/audio-filters/ (дата звернення 20.02.2023).
14. Фильтрация шума сигнала. URL: https://habr.com/ru/post/588270/ (дата звернення 17.02.2023).
15. Характеристики голосу. URL: https://spe.org.ua/blog/sp/kharakterystyky-holosu/ (дата звернення 17.02.2023).
16. Difference Between FFT and DFT. URL: http://www.differencebetween.net/technology/difference-between-fft-and-dft/ (дата звернення 17.02.2023).
17. Г. О. Добрушкін, В. Я. Данилов. ОСНОВНІ ПІДХОДИ ДО РОЗПІЗНАВАННЯМОВЛЕННЄВОЇ ІНФОРМАЦІЇ (ЧАСТИНА 1). Вісник Вінницького політехнічного інституту. 2009 15 с.
18. WAV or MP3: What's the Difference? URL: https://www.audiobuzz.com/blog/wav-or-mp3-whats-the-difference/ (дата звернення 17.02.2023).
19. Transformer-based Acoustic Modeling for Hybrid Speech Recognition. URL: https://mip-frontiers.eu/2020/10/07/transformer_hybridASR.html (дата звернення 17.02.2023).
20. What is Deep Learning? URL: https://machinelearningmastery.com/what-is-deep-learning/ (дата звернення 18.02.2023).

21. ЯК ДІЄ ШТУЧНИЙ ІНТЕЛЕКТ І ПЕРСПЕКТИВИ ЙОГО ВИКОРИСТАННЯ. URL: https://aiconference.com.ua/uk/news/printsipi-raboti-iskusstvennogo-intellekta-i-perspektiva-ego-ispolzovaniya-92238 (дата звернення 18.02.2023).
22. Basic knowledge of deep neural network (DNN). URL: https://wiki.sipeed.com/ai/en/basic/dnn_basic.html (дата звернення 18.02.2023).
23. What's Next In Neural Networking? URL: https://semiengineering.com/whats-next-in-neural-networking/ (дата звернення 18.02.2023).
24. Chemception: Deep Learning from 2D Chemical Structure Images. URL: https://depth-first.com/articles/2019/02/04/chemception-deep-learning-from-2d-chemical-structure-images/ (дата звернення 18.02.2023).
25. Reducing the gender gap in academic activities: a 10-year progress report by the Japan Endocrine Society Women Endocrinologists Association (JES-We-Can) URL: https://www.jstage.jst.go.jp/article/endocrj/66/4/66_EJ18-0501/_html/-char/en (дата звернення 18.02.2023).
26. Signal Processing Toolbox. URL: https://www.mathworks.com/products/signal.html (дата звернення 27.02.2023).
27. Speech Recognition in Python using CMU Sphinx. URL: https://www.geeksforgeeks.org/speech-recognition-in-python-using-cmu-sphinx/ (дата звернення 27.02.2023).
28. Обзор Keras для TensorFlow. URL: https://habr.com/ru/post/482126/ (дата звернення 27.02.2023).
29. Open Source Python Library for Speech Recognitions Python Speech Features (PSF). URL: https://products.fileformat.com/audio/python/speechpy/ (дата звернення 27.02.2023).
30. Vue.js. URL: https://www.jetbrains.com/help/webstorm/vue-js.html (дата звернення 27.02.2023).

31. Building API Gateway on .NET with Ocelot. URL: https://arbems.com/en/building-api-gateway-on-net-with-ocelot/ (дата звернення 27.03.2023).
32. UML 2 Component Diagrams: An Agile Introduction. URL: http://www.agilemodeling.com/artifacts/componentDiagram.htm (дата звернення 9.04.2023).
33. UML 2 Use Case Diagrams: An Agile Introduction. URL: http://www.agilemodeling.com/artifacts/useCaseDiagram.htm (дата звернення 9.04.2023).
34. UML 2 State Machine Diagrams: An Agile Introduction. URL: http://www.agilemodeling.com/artifacts/stateMachineDiagram.htm (дата звернення 9.04.2023).
35. UML 2 Sequence Diagrams: An Agile Introduction. URL: http://www.agilemodeling.com/artifacts/sequenceDiagram.htm (дата звернення 9.04.2023).
36. Drahan M., Pysarenko A. Preprocessing of audio data for voice transcription systems. Міжвідомчий науково-технічний журнал «Адаптивні системи автоматичного управління». – 2023. – №1(42). – С. 39–48.