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

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«На правах рукопису»
УДК 004.4'24

«До захисту допущено»
Завідувач кафедри
_____ Євгенія Сулема
«___»_____2023 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-науковою програмою
«Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

зі спеціальності 121 Інженерія програмного забезпечення
на тему: «Метод та програмне забезпечення для розпізнавання англомовних акцентів»

Виконав:
студент ІІ курсу, групи КП-11мн
Манохін Андрій Віталійович _____

Керівник:
Доцент кафедри ПЗКС, к.т.н., доцент
Рибачок Наталія Антонівна _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
Доцент кафедри ЕІ, к.т.н., доцент,
Вунтесмері Юрій Володимирович _____

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 прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-наукова програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Євгенія СУЛЕМА

«___»_____2021 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Манохіну Андрію Віталійовичу

1. Тема дисертації «Метод та програмне забезпечення для розпізнавання англомовних акцентів» науковий керівник дисертації Рибачок Наталія Антонівна, к.т.н., доцент, затверджена наказом по університету від «30» березня 2023 р. №1359-С.
2. Термін подання студентом дисертації «19» травня 2023 р.
3. Об'єкт дослідження: процес виокремлення інформації із аудіоданих записаних англомовними користувачами.
4. Предмет дослідження: метод виокремлення інформації із аудіоданих записаних англомовними користувачами.
5. Перелік завдань, які потрібно розробити:
   - дослідити існуючі методи аналізу аудіозаписів для виокремлення важливих даних;
   - на основі результатів дослідження розробити власний метод видобування необхідних даних із аудіо файлів, що призведуть до кращих результатів класифікації методами машинного навчання;
   - розробити програмне забезпечення, що реалізує даний метод;
   - проаналізувати ефективність розробленого методу, шляхом порівняння результатів класифікації даних, отриманих за його допомогою і за допомогою вже існуючих методів.
6. Перелік ілюстративного матеріалу:
   - діаграма бази даних;

- узагальнена схема методу;
- діаграма прецедентів;
- діаграма роботи LPC;
- діаграма дерева проблем;
- схема архітектури систем.

7. Перелік публікацій:
   - Тези доповіді «Методи розпізнавання англомовних акцентів»
8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент кафедри ПЗКС		

9. Дата видачі завдання «21» жовтня 2021 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Ґрунтовне ознайомлення з предметною галуззю	09.12.2021	
2.	Визначення структури магістерської дисертації; вивчення літератури, пошук додаткової літератури, патентний пошук	30.01.2021	
3.	Робота над першим розділом магістерської дисертації; проведення наукового дослідження	11.03.2022	
4.	Проведення наукового дослідження; робота над другим розділом магістерської дисертації; розроблення програмного забезпечення; підготовка матеріалів доповіді ПМК-2022	08.05.2022	
5.	Проведення наукового дослідження; робота над статтею за результатами наукового дослідження	14.08.2022	
6.	Проведення наукового дослідження; робота надтретім розділом магістерської дисертації	19.09.2022	
7.	Завершення роботи над основною частиною магістерської дисертації; підготовка ілюстративного матеріалу	09.02.2023	
8.	Оформлення текстової і графічної частини магістерської дисертації	08.04.2023	

Науковий керівник Наталія РИБАЧОК

Студент Андрій МАНОХІН

РЕФЕРАТ

Актуальність теми. Створення комбінованого методу, який об’єднує кілька методів аналізу мови, таких як MFCC, LPC, PLP і DWT, має важливе значення в області обробки мови. Кожен із цих методів має свої сильні та слабкі сторони, і, об’єднавши їх, можемо використовувати їхні переваги, щоб подолати обмеження та підвищити точність і ефективність завдань аналізу мовлення.

Однією з ключових причин розробки комбінованого методу є потреба в більш точних і надійних системах розпізнавання мовлення. Мовні сигнали складні та багатогранні мають різні часові, спектральні та перцептивні характеристики. Інтегруючи кілька методів, можемо охопити більш повне представлення мовлення. Це дозволяює нам отримувати більш багатий набір функцій і покращувати дискримінаційну здатність системи. Наслідком цього буде прогрес в таких процесах як автоматичне розпізнавання мови, ідентифікація мовця та класифікація акцентів.

Крім того, комбінований метод може вирішити обмеження, притаманні окремим методам. Наприклад, хоча MFCC широко використовується та ефективний для розпізнавання мовлення, він може мати проблеми із захопленням дрібних спектральних деталей або точним представленням нестаціонарних аспектів мовлення. Завдяки використанню LPC, який моделює характеристики голосового тракту, можемо краще вловлювати форму та динаміку голосового тракту, що призводить до покращеного моделювання та розпізнавання мовлення.

Загалом, актуальність створення комбінованого методу полягає в прагненні вдосконалити сферу аналізу та розпізнавання мовлення, забезпечуючи можливість створення більш точних і надійних систем, які можуть мати глибокий вплив на додатки, починаючи від голосових помічників та мовних інтерфейсів до охорони здоров’я та безпеки. системи.

В даній магістерській дисертації описано метод отримання ознак із

аудіоданих для покращення їх подальшого аналізу.

Об’єктом дослідження є процес виокремлення інформації із аудіоданих записаних англомовними користувачами.

Предметом дослідження є метод виокремлення інформації із аудіоданих записаних англомовними користувачами.

Метою дослідження є підвищення ефективності класифікації англомовних акцентів в середньому на 5% шляхом виокремлення більш детальних даних із аудіозаписів.

Методи дослідження: в роботі використовуються методи аналізу аудіоданих, методи машинного навчання із використанням згорткової моделі.

Наукова новизна роботи полягає в наступному: уперше запропоновано метод розпізнавання англомовних акцентів, який відрізняється від існуючих унікальним порядком виконання процедур оброблення мовленнєвого сигналу, а саме, застосуванням алгоритму LPC після оброблення сигналу методом DWT, що дозволяє підвищити точність передбачень англомовних акцентів у середньому на 5%.

Практичне значення одержаних результатів полягає у тому, що запропонований метод розпізнавання англомовних акцентів дозволяє підвищити якість програмного оброблення мовленнєвого сигналу, що у свою чергу підвищує якість відповідного програмного продукту.

Апробація роботи. Основні положення і результати роботи доповідалися та обговорювалися на XV науковій конференції магістрантів та аспірантів “Прикладна математика та комп’ютинг” ПМК-2022.

Структура та обсяг роботи. Магістерська дисертація складається з вступу, чотирьох розділів та висновків.

У вступі подано загальну характеристику роботи, зроблено оцінку сучасного стану проблеми, обґрунтовано актуальність напрямку досліджень.

У першому розділі проаналізовано існуючі методи аналізу

аудіоданих, проведено порівняння алгоритмів кожного процесу, їх переваг та недоліків.

У другому розділі описано комбінований метод аналізу аудіоданих. Запропонований метод полягає у використанні поетапно двох популярних методів для аналізу аудіоданих LPC та DWT.

У третьому розділі обґрунтовано вибір технологій для розробки системи. Описано загальну архітектуру системи та розглянуто модулі програмного забезпечення.

У четвертому розділі було проаналізовано ефективність роботи методу в порівнянні з іншими методами.

У висновках проаналізовано отримані результати дослідження.

Робота виконана на 83 аркушах, містить 2 додатки та посилання на список використаних літературних джерел з 23 найменувань. У роботі наведено 15 рисунків та 4 таблиці.

Ключові слова: ознаки аудіоданих, LPC, DWT, MFCC, форманта, кепстр, віконування, кадр.

ABSTRACT

Actuality of theme. Creating a combined method that integrates multiple speech analysis techniques, such as MFCC, LPC, PLP, and DWT, holds significant importance in the field of speech processing. Each of these techniques has its own strengths and weaknesses, and by combining them, we can leverage their respective advantages to overcome limitations and improve the accuracy and effectiveness of speech analysis tasks.

One of the key motivations for developing a combined method is the demand for more accurate and robust speech recognition systems. Speech signals are complex and multifaceted, encompassing various temporal, spectral, and perceptual characteristics. By integrating multiple techniques, we can capture a more comprehensive representation of speech, enabling us to extract a richer set of features and improve the discriminative power of the system. This can lead to advancements in applications such as automatic speech recognition, speaker identification, and accent classification.

Furthermore, a combined method can address the limitations inherent in individual techniques. For example, while MFCC is widely used and effective for speech recognition, it may struggle with capturing fine spectral details or accurately representing non-stationary aspects of speech. By incorporating LPC, which models the vocal tract characteristics, we can better capture the shape and dynamics of the vocal tract, resulting in improved speech modeling and recognition.

Overall, the actuality of creating a combined method lies in the pursuit of advancing the field of speech analysis and recognition, enabling more accurate and reliable systems that can have a profound impact on applications ranging from voice assistants and speech-based interfaces to healthcare and security systems.

This master's thesis describes a method of obtaining features from audio data to improve their further analysis.

Object of research is the process of extracting information from audio data recorded by English-speaking users.

Subject of research is the method of extracting information from audio data recorded by English-speaking users.

Goal of the work is to improve the classification efficiency of English accents by an average of 5% by extracting more detailed data from audio recordings.

Methods of research: the work uses audio data analysis methods, machine learning methods using a convolutional model.

Scientific novelty of the work is as follows**:** for the first time, a method for recognizing English accents is proposed, which differs from the existing ones in the unique order of performing speech signal processing procedures, namely, the use of the LPC algorithm after processing the signal by the DWT method, which allows to increase the accuracy of predictions of English accents by an average of 5%.

Practical value of the results obtained in the work is that the proposed method of recognizing English accents allows to increase the quality of software processing of the speech signal, which in turn increases the quality of the corresponding software product.

Approbation. The main provisions and results of the work were reported and discussed at the XV Scientific Conference of Master's and Postgraduate Students "Applied Mathematics and Computing" PMK-2022.

Structure and scope of work. The master's thesis consists of an introduction, four chapters and conclusions.

In the introduction a general description of the work is presented, an assessment of the current state of the problem is made, and the relevance of the research direction is substantiated.

The first section analyzes the existing methods of audio data analysis are, analyzed, a comparison is made in the algorithmic process of each method.

The second section describes the algorithmic-combined method of audio

data analysis. The proposed method is to use in stages two popular methods for analyzing audio data, LPC and DWT.

The third section substantiates the choice of technologies for system development. The general architecture of the system is described and the software modules are considered.

In the fourth section, the effectiveness of the method was analyzed in comparison with other methods.

The conclusion contains brief summary of study results.

The work is completed on 83 sheets, contains 2 appendices and links to the list of used literary sources from 23 names. The work contains 22 figures and 4 tables.

Keywords: features of audio data, LPC, DWT, MFCC, formant, kepstrum, windowing, frame.

ЗМІСТ

СПИСОК ВИЗНАЧЕНЬ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ACID (Atomicity, Consistency, Isolation, Durability) – набір властивостей, які гарантують надійність виконання транзакцій у БД.

API – комплект методів, які складають інтерфейс для швидкого розроблення програмного забезпечення.

CSRF – тип веб-атаки, що полягає у виконанні певних дій від імені користувача на веб-сторінці де останній авторизований.

Cookie – це невеликий блок корисних даних, розміщений на комп'ютері користувача чи іншому пристрої веб-браузером.

GraphQL – строго типізована мова запитів для побудови API на серверній частині.

IELTS – англомовний тест для інтернаціонального навчання.

JIT компіляція – just in time, динамічна компіляція.

MFCC`s – коефіцієнти, які описують спектр потужності звуку на нелінійній шкалі Мел.

LPC – широко використовуваний метод аналізу мовлення для представлення спектральної обвідної мовного сигналу.

PLP – техніка аналізу мовлення, яка використовується для виділення функцій для розпізнавання мовлення.

DWT – є технікою обробки сигналів, яка використовується для аналізу та розкладання сигналів на різні частотні компоненти.

ORM – техніка ООП для передачі даними між несумісними по типам системами.

SQL Injection – вид атаки на веб-сайти, який полягає у впровадження в запит довільного SQL-коду, який буде оброблений веб-сервером.

SQL – структурована мова запитів до БД.

UI – інтерфейс користувача.

XSS – тип вразливості інформаційних систем у мережі Інтернет, який полягає у впровадженні користувацьких скриптів у сторінки, віддані

веб-сервером.

drag'n'drop – спосіб керування UI за допомогою перетаскування графічних елементів.

urql – GraphQL клієнт.

БД – база даних.

Вікно Хеммінга – мінімізує бічну пелюстку сигналу (небажане випромінювання). Таким чином, поліпшується якість або гармоніки звуку.

Димове тестування – мінімальний набір тестів на явні помилки.

Мел – психофізична одиниця висоти звуку, застосовується головним чином в музичній акустиці. Назва походить від слова «мелодія».

Кепстр – є результатом обчислення зворотного перетворення Фур'є (IFF) логарифма оціненого спектру сигналу.

Ознаки аудіоданих – є описом звуку або аудіосигналу, який, в основному, можна вводити в статистичні моделі або моделі ML для створення інтелектуальних аудіосистем.

Форманта – це широкий спектральний максимум, який є результатом акустичного резонансу голосового тракту людини.

Віконування – аналіз невеликих підмножин загальних даних.

Мультипарадигмальна мова програмування – мова, яка підтримує більше ніж одну парадигму програмування

НМ – нейронна мережа.

ООП – об'єктно-орієнтовне програмування.

ПЗ – програмне забезпечення.

ПК – персональний комп'ютер.

СУБД – система управління базами даних.

Скриптова мова програмування – мова програмування, розроблена для запису скриптів, послідовностей операцій, які інтерпретуються, а не компілюються.

Телеграм – месенджер.

Телеграм-бот – це спеціальні програми, що виконують різні функції

і спрощують життя їх користувачів.

Транзакція – група атомарних послідновних операцій із БД, може бути виконана або цілком і успішно, або ніяк.

ВСТУП

Обробка мовлення – це сфера, яка швидко розширюється, і має декілька застосувань, зокрема ідентифікацію мовця, синтез голосу та розпізнавання мовлення. Виділення ознак, яке передбачає вибір відповідних властивостей або характеристик мовних сигналів для майбутнього аналізу, є вирішальним кроком у процесі обробки мовлення. Оскільки це дозволяє вивчати мовні сигнали, які є складними та змінними, виділення ознак має вирішальне значення в сучасній обробці мовлення.Голосові сигнали є безперервними сигналами, і через їх нестаціонарність їх аналіз може бути складним. Ці сигнали перетворюються на набір корисних функцій, які можна використовувати для додаткового аналізу за допомогою методів виділення ознак. У порівнянні з самими необробленими голосовими сигналами ці функції часто є більш яскравими та легшими для оцінки.

Наприклад, виділення ознак є важливим для розпізнавання мовлення, оскільки воно використовується для вилучення відповідних даних із голосового сигналу, які можна використовувати для розрізнення різних звуків або фонем. Вибір підходу до виділення ознак впливає на точність і надійність системи розпізнавання голосу. Деякі з методів, які часто використовуються в розпізнаванні мовлення для виділення ознак, це MFCC, LPC і PLP.

Інша програма, де виділення функцій є важливим, це ідентифікація динаміка. Процес виділення ознак використовується для точного визначення відповідних голосових рис, які можна використати, щоб відрізнити одного мовця від іншого. Висота тону, форманти, а також інші спектральні властивості можуть бути серед цих характеристик. Ці функції виділяються за допомогою методів виділення ознак, які потім можна використовувати для створення моделей для ідентифікації мовця.

Ще одна область, де виділення ознак є вирішальним, це синтез

мовлення. У синтезі мовлення виділення ознак використовується для запису інтонації, ритму та просодії людського голосу. Для створення голосових синтезаторів, які звучать автентично та природно, використовуються ці елементи.

Підсумовуючи, виділення ознак має вирішальне значення для сучасної обробки мовлення, оскільки воно дає змогу аналізувати складні та різноманітні мовні сигнали. Мовні сигнали перетворюються на набір корисних функцій, які можна використовувати для додаткового аналізу за допомогою методів виділення ознак. Ці характеристики важливі для таких програм, як синтез голосу, ідентифікація мовця та розпізнавання мовлення. На точність і надійність цих додатків може суттєво вплинути обраний метод виділення ознак.

Отже, щоб покращити функціональні можливості систем обробки мови, необхідно розробити та вдосконалити підходи до виділення ознак.

1. ОГЛЯД ІСНУЮЧИХ МЕТОДІВ

MFCC LPC і PLP широко використовуються в обробці мовлення та виділенні ознак. Незважаючи на те, що всі три методи спрямовані на виділення релевантних характеристик із мовних сигналів, вони відрізняються підходом, сильними та слабкими сторонами.

1.1. MFCC

MFCC – це широко використовувана техніка в обробці мовлення для виділення ознак. Він заснований на сприйнятті звуку слуховою системою людини, який моделюється за допомогою банку фільтрів із розділеними частотами Mel [1]. MFCC обчислюються шляхом дискретного перетворення Фур'є (ДПФ) віконного мовного сигналу, застосування групи фільтрів Mel до результуючого спектру потужності та логарифмування енергій групи фільтрів. Отриманий кепстр Mel-частоти потім обробляється за допомогою таких методів, як нормалізація середнього кепстра, розрахунок дельта, дельта від дельти та вибір ознак. Мелчастотний кепстр дуже ефективний у розпізнаванні аудіо та моделюванні суб'єктивної висоти та частотного вмісту аудіосигналів. Шкала Mel розраховується за такою формулою

$$\mathrm{Mel}(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right), \tag{1.1}$$

де Mel(*f*) – логарифмічний масштаб нормальної частотної шкали *f*. Шкала Mel має постійний інтервал частоти Mel і охоплює діапазон частот від 0 Гц до 20050 Гц. Кепстральні коефіцієнти обчислюються з коефіцієнтів потужності FFT, які фільтруються трикутним банком смугових фільтрів. Блок фільтрів складається з 12 трикутних фільтрів. MFCC розраховуються за формулою 1.2

$$C_n = \sqrt{\frac{2}{k}} \sum_{k=1}^{K} (\log \mathrm{S_k}) \cos\left[n(k-0.5)\frac{\pi}{k} \right], \quad (1.2)$$

де $\mathrm{S_k}$ (k = 1, 2, ⋯ K) – вихідний сигнал банків фільтрів, а N – загальна кількість вибірок у аудіо блоці 20 мс.

MFCC виводяться зі спектра потужності мовного сигналу та використовуються для захоплення спектральних характеристик сигналу. Процес отримання MFCC включає кілька кроків, які описані нижче: попереднє виділення, кадрування, вікна, перетворення Фур'є, банк фільтрів Mel, логарифмічне стиснення, дискретне косинусне перетворення (DCT) і нормалізацію кепстрального середнього.

1.1.1. *Попереднє виділення мовного сигналу*

Першим кроком у вилученні MFCC є попереднє виділення мовного сигналу відровідно до формули 1.3. Це робиться для посилення високочастотних компонентів сигналу та компенсації загасання, яке виникає під час створення та передачі мови.

$$y[n] = x[n] - \alpha \cdot x[n-1], \quad (1.3)$$

де $x[n]$ – вхідний мовний сигнал, $y[n]$ – вихідний сигнал, а α – коефіцієнт попереднього наголосу. Коефіцієнт попереднього наголошення зазвичай становить 0,97 або 0,95. Ця операція допомагає покращити співвідношення сигнал/шум і покращити спектральні характеристики, важливі для розпізнавання мовлення. Значення α визначає силу акценту, при вищих значеннях виробляється сильніший акцент високочастотних компонентів.

1.1.2. *Кадрування сигналу*

Другим кроком мовний сигнал розділяється на короткі кадри, що перекриваються, зазвичай довжиною 20 – 40 мс. Такі короткі кадри беруться через припущення про стаціонарність сигналу на коротких проміжках часу. Хоча мовні сигнали не є абсолютно стаціонарними, вони

виявляють квазістаціонарну поведінку протягом коротких інтервалів часу. Це означає, що статистичні властивості сигналу, такі як спектр потужності, залишаються приблизно постійними в межах кожного кадру, що дозволяє проводити більш точний аналіз і моделювання сигналу. Нестаціонарні властивості натомість нікуди не зникають, тому спектральні характеристикию сигналу, такі як форманти та резонанси, відносно постійні в кожному кадрі, що дозволяє більш надійно оцінювати ці характеристики.

Окрім цього шляхом поділу мовного сигналу на кадри зменшується обчислювальна складність процесу виділення ознак, оскільки кожен кадр можна обробляти незалежно та паралельно. Це дозволяє швидше обробляти й аналізувати великі масиви мовних даних. Причому кадрування може допомогти підвищити стійкість процесу виділення ознак до шуму, спотворень каналів та інших варіацій сигналу, оскільки характеристики, витягнуті з кожного кадру, усереднюються або комбінуються для отримання більш стабільного представлення сигнал.

1.1.3. *Віконування*

Наступним кроком кожен кадр помножується на функцію вікна, щоб зменшити спектральний витік і мінімізувати краєві ефекти. Найбільш часто використовуваною функцією вікна є вікно Хеммінга. Математичною формулою для вікна Хеммінга виступає

$$\mathrm{w}(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), \tag{1.4}$$

де $w(n)$ – значення вікна Хеммінга в позиції n, n – індекс поточної вибірки у вікні в діапазоні від 0 до $N - 1$, а N – довжина вікна. Дана формула генерує симетричне вікно з рельєфною формою косинуса, яке плавно звужує краї вхідного сигналу до нуля, зменшуючи спектральний витік і покращуючи точність перетворення Фур'є. Вона забезпечує хороший баланс між шириною основного пелюстка та придушенням бічного

пелюстка, що допомагає зберегти спектральні характеристики сигналу, мінімізуючи ефект шуму та інших спотворень.

Коли сигнал ділиться на кадри та аналізується за допомогою перетворення Фур’є, результуючий спектр може бути спотворений ефектом спектрального витоку, який виникає внаслідок різкого розриву на межі кадру. Цей ефект може спричинити «витік» енергії сигналу в сусідні діапазони частот, знижуючи точність спектрального аналізу. Помноживши кожен кадр на віконну функцію, яка плавно звужує краї кадру до нуля, спектральний витік може бути значно зменшений, підвищуючи точність спектрального аналізу. До того ж, вікна можуть покращити частотну роздільну здатність спектрального аналізу шляхом збільшення ефективної довжини кадру. Більша довжина кадру відповідає вужчому головному пелюстку вікна перетворення Фур’є, що, у свою чергу, забезпечує кращу роздільну здатність по частоті. Однак збільшення довжини кадру також зменшує часову роздільну здатність аналізу, тому потрібно зробити компроміс між частотною та тимчасовою роздільною здатністю. Окрім цього, використання вікон може допомогти зменшити вплив шуму на спектральний аналіз шляхом зменшення амплітуди шуму за межами головної пелюстки функції вікна. Це покращує відношення сигнал/шум спектрального аналізу та призводить до більш точних оцінок спектральних характеристик.

Зменшуючи спектральний витік, покращуючи частотну роздільну здатність і зменшуючи вплив шуму, вікна можуть допомогти підвищити якість спектральних характеристик, витягнутих із мовного сигналу, покращуючи продуктивність завдань обробки мовлення.

1.1.4. *Дискретне перетворення Фур’є*

Перетворення Фур'є застосовується до кожного віконного кадру, щоб отримати спектр потужності сигналу. Цей крок створює спектр, який

показує розподіл енергії на різних частотах у сигналі, що визначається наступною формулою:

$$\mathrm{X}(m,k) = \sum_{i=1}^{n} x[n] \cdot \mathrm{w}(n) \cdot e^{\frac{-2\pi ikn}{N}}, \qquad (1.5)$$

де X(m, k) представляє сигнал частотної області в частотному діапазоні k і кадрі m;

$x[n]$ представляє сигнал часової області у зразку n;

w(n) представляє віконну функцію, застосовану до сигналу в зразку n;

e представляє основу натурального логарифма;

i представляє уявну одиницю ($\sqrt{(-1)}$);

k представляє індекс діапазону частот;

n представляє індекс вибірки;

N представляє загальну кількість вибірок у сигналі.

Формула 1.5 обчислює дискретне перетворення Фур'є (DFT) сигналу $x[n]$ шляхом застосування віконної функції w(n) до сигналу, множення її на комплексну експоненціальну з частотою k і підсумовування всіх вибірок у сигналі. Результуюче значення X(m, k) представляє амплітуду та фазу сигналу в діапазоні k частоти та кадрі m.

Віконна функція w(n) використовується для зменшення ефекту спектрального витоку, а комплексна експонента використовується для зміщення частоти сигналу до потрібного діапазону частот. Зменшення амплітуди біля країв сигналу викликає «витік» спектральної енергії в сусідні діапазони частот, що призводить до явища, відомого як спектральний витік. Цей витік може спотворювати частотний вміст сигналу, що призводить до неточних оцінок спектральних властивостей сигналу, таких як частота його формант або резонансів.

Завдяки перетворенню сигналу з часової області в частотну область, дискретне перетворення Фур'є дозволяє точніше та ефективніше аналізувати спектральні характеристики мовного сигналу.

1.1.5. *Банк фільтрів Mel*

Спектр потужності потім пропускається через банк Mel-частотних фільтрів. Ці фільтри створені для імітації нелінійної частотної характеристики людського вуха. Вихід кожного фільтра представляє енергію в певному діапазоні частот.

Етап банку фільтрів Mel передбачає перетворення спектра потужності мовного сигналу з лінійної шкали в нелінійну шкалу Mel за допомогою набору трикутних фільтрів, які рівномірно розподілені на осі частот Mel.

Шкала Мела – це частотна шкала сприйняття, яка базується на реакції слухової системи людини на звук. Шкала Мела є нелінійною, що означає, що сприймана відстань між двома частотами не пропорційна їх різниці в Герцах (формула 1.1).

Щоб перетворити спектр потужності мовного сигналу в шкалу Мела, до спектру застосовують набір трикутних фільтрів, кожен з яких охоплює певний діапазон частот. Фільтри розподілені рівномірно за шкалою Мела, а їхня смуга пропускання пропорційна відстані між фільтрами за шкалою Мела. Форма кожного фільтра є трикутною, з піком на центральній частоті, а його сторони нахилені до нуля на частотах, охоплених сусідніми фільтрами.

На виході кроку банку фільтрів Мел є набір коефіцієнтів банку фільтрів, які представляють енергії в кожному з трикутних фільтрів. Ці коефіцієнти обчислюються шляхом множення спектра потужності мовного сигналу з кожним із трикутних фільтрів і підсумовування результату. Отримані коефіцієнти банку фільтрів потім пропускаються через логарифмічну функцію для перетворення їх у шкалу децибел, яка більше відповідає сприйняттю гучності слуховою системою людини.

Крок банку фільтрів Мел є важливим кроком у обчисленні MFCC, оскільки він допомагає усунути надмірність у спектрі потужності та

зосереджується на смугах частот, які є найбільш релевантними для сприйняття мови.

1.1.6. *Логарифмічна компресія*

Вихідні дані кожного фільтра потім перетворюються на логарифмічну шкалу, щоб імітувати логарифмічне сприйняття гучності людським вухом. Логарифмічну операцію стиснення можна представити наступним чином:

$$Y_m = \log_{10}\left(\sum_{k=1} P[k] \cdot H_m[k]\right), \tag{1.7}$$

де Y_m – коефіцієнт виходу банку фільтрів Мел, $P[k]$ – спектр потужності, а $H_m[k]$ – k-й фільтр у банку фільтрів Мел. Тут сума береться по всіх частотних діапазонах k, для яких спектр потужності $P[k]$ відмінний від нуля, а відповідний фільтр $H_m[k]$ перекривається з k-м частотним діапазоном. Потім обчислюється логарифм (за основою 10) суми, щоб стиснути динамічний діапазон виходів банку фільтрів, що робить їх більш придатними для подальшої обробки.

1.1.7. *Дискретне косинусне перетворення (DCT)*

Вихідний сигнал банку фільтрів Mel проходить через дискретне косинусне перетворення (DCT). DCT використовується для декореляції виходів банку фільтрів і визначається наступною формулою:

$$C_n = \sqrt{\frac{2}{n}} \sum_{n=1}^{N} Y_m \cos\left(\frac{\pi n(m-0.5)}{M}\right), \tag{1.8}$$

де C_n – n-й коефіцієнт MFCC, Y_m – вихідний сигнал m-го фільтра Mel, M – кількість фільтрів у банку фільтрів Mel, N – довжина сигналу, а n – індекс DCT.

У контексті обробки мови DCT використовується для вилучення кепстральних коефіцієнтів Mel-частоти (MFCC) із вихідних даних банку фільтрів, отриманих на попередніх етапах процесу обчислення MFCC.

DCT приймає послідовність з N дійсних чисел x_0, x_1, …, x_N як вхідні дані та створює послідовність з N дійсних чисел X_0, X_1, …, X_N як вихід. DCT – це лінійне перетворення, яке відображає вхідну послідовність у вихідну послідовність, так що кожне вихідне значення X_k є зваженою сумою вхідних значень x_0, x_1, …, x_N з вагами, визначеними за функція косинус.

DCT – це тип перетворення Фур'є, оптимізований для вхідних послідовностей дійсного значення. DCT працює з послідовністю дійсних чисел, тоді як перетворення Фур'є працює з послідовністю комплексних чисел. DCT має кілька бажаних властивостей для додатків обробки мовлення, включаючи його здатність охоплювати спектральну обвідну мовного сигналу та його здатність декорелювати виходи банку фільтрів, що зменшує розмірність простору ознак і покращує продуктивність наступних етапів обробки [2].

У контексті обчислення MFCC DCT застосовується до енергій блоку логарифмічних фільтрів, отриманих на попередньому етапі. Першим кроком є віднімання середнього значення енергій банку логарифмічних фільтрів від кожного коефіцієнта, щоб зробити вхідну послідовність нульовим середнім. DCT потім застосовується до результуючої послідовності з N коефіцієнтів, створюючи послідовність з N MFCC як вихід. Як правило, лише кілька перших MFCC (наприклад, перші 12-13) використовуються в програмах обробки мовлення, оскільки вони містять найбільш відповідну інформацію для розпізнавання мовлення та інших завдань.

1.1.8. *Нормалізація кепстрального середнього значення*

Нарешті, перший коефіцієнт MFCC (який представляє середнє значення логарифмічної енергії) віднімається з усіх інших коефіцієнтів MFCC, щоб зменшити вплив змін каналу та середовища.

Перший коефіцієнт MFCC відомий як енергетичний коефіцієнт і представляє загальну енергію мовного сигналу. Оскільки на енергію мовного сигналу впливають характеристики мовця, він вважається інформацією, що залежить від мовця і не є корисною для розпізнавання мовлення. Тому загальною практикою є видалення цієї інформації шляхом віднімання першого коефіцієнта MFCC з усіх інших коефіцієнтів MFCC.

1.1.9. *Характеристики MFCC*

Переваги:

- MFCC стійкі до шуму та змінності динаміків, що робить їх придатними для програм розпізнавання мовлення.
- Вони фіксують як спектральні, так і часові характеристики мовних сигналів.
- Вони широко використовуються і добре зарекомендували себе в області обробки мови.

Недоліки:

- MFCC чутливі до обрізання сигналу та насиченості, що може призвести до спотворення у представленні ознак.
- Вони не вловлюють усі аспекти сигналу, наприклад інформацію про фазу та висоту.

Загалом, MFCC найбільше підходять для завдань, які вимагають розпізнавання та класифікації мовних сигналів на основі їх спектрального вмісту, що робить їх популярним і широко використовуваним методом виділення ознак у сфері обробки мовлення.

1.2. LPC

LPC – це техніка, яка використовується для моделювання мовних сигналів як лінійної комбінації минулих зразків. Він працює шляхом оцінки лінійних прогнозних коефіцієнтів, які мінімізують помилку

прогнозування сигналу. Потім ці коефіцієнти можна використовувати для виділення характеристик із сигналу. LPC зазвичай використовується для кодування мови, перетворення голосу та розпізнавання мовця [2].

І MFCC, і LPC включають кадрування мовного сигналу в короткі вікна, що перекриваються, причому кожен кадр вважається нерухомим протягом своєї тривалості. Кадри зазвичай вікнаються за допомогою функції вікна, такої як вікно Хеммінга, щоб зменшити спектральний витік, спричинений різкими переходами на краях кадру.

Алгоритм методу включає нижче наведені кроки у відповідному порядку: алгоритм Левінсона-Дурбіна, зворотнє перетворення Фур'є та виділення залишкового сигналу.

1.2.1. *Алгоритм Левінсона-Дурбіна*

Першим виконується крок моделювання коефіцієнтів, що передбачає оцінку параметрів багатополюсного фільтра, який моделює мовний сигнал як лінійну комбінацію минулих зразків. Коефіцієнти фільтра оцінюються за допомогою методу автокореляції або коваріаційного методу. Метод автокореляції є найбільш широко використовуваним методом і базується на такому рівнянні:

$$\mathrm{R}(i) = \sum_{i=0}^{n} x[n] \cdot x[n-1], \tag{1.9}$$

де R(i) – функція автокореляції на крок i, $x[n]$ – мовний сигнал, а n – від 0 до N – 1. Функція автокореляції використовується для оцінки коефіцієнтів LPC за допомогою рекурсивного алгоритму Левінсона-Дурбіна, який рекурсивно оцінює коефіцієнти всеполюсного фільтра.

Алгоритм названий на честь Нормана Левінсона та Джеймса Дурбіна, які незалежно один від одного розробили його в 1940-х роках.

Вхідними даними для алгоритму рекурсії Левінсона-Дурбіна є послідовність точок даних, таких як зразки мовного сигналу. Мета полягає

в тому, щоб знайти коефіцієнти моделі AR, які можуть апроксимувати послідовність даних. Модель AR – це математична модель, яка описує часовий ряд як лінійну комбінацію його минулих значень, причому кожен член зважений коефіцієнтом. Порядок моделі AR визначає кількість минулих значень, які використовуються в лінійній комбінації.

Алгоритм рекурсії Левінсона-Дурбіна починається з початкової оцінки першого коефіцієнта AR, відомого як коефіцієнт помилки передбачення. Цей коефіцієнт обчислюється шляхом ділення коваріації першої та другої точок даних на дисперсію першої точки даних. Коефіцієнт похибки передбачення потім використовується для обчислення першого набору коефіцієнтів AR.

Потім алгоритм ітеративно обчислює додаткові набори коефіцієнтів AR, використовуючи попередній набір коефіцієнтів і помилку передбачення. Цей процес повторюється, доки не буде досягнуто бажаного порядку моделі AR. Кожна ітерація алгоритму включає серію обчислень, включаючи інверсію матриці та векторне множення.

Алгоритм можна виразити математично наступним чином:

Дано послідовність даних $x[n]$ довжини N, метою є знайти коефіцієнти $a[k]$ моделі AR порядку p, де $p < N$. Початкова оцінка першого коефіцієнта AR, $a[1]$, визначається як:

$$a[1] = -\frac{r[1]}{e[0]}, \tag{1.10}$$

де $r[1]$ – автокореляція $x[n]$ із затримкою 1, а $e[0]$ – енергія $x[n]$.

Рекурсія для обчислення коефіцієнтів AR відбувається наступним чином:

Ініціалізація:

k=1

$$a[1] = -\frac{r[1]}{e[0]}, \tag{1.11}$$

$$E[1] = e[0]\left(1 - a[1]^2\right), \tag{1.12}$$

Для k=2 до p:

$$\lambda[k] = 0, \tag{1.13}$$

для j=1 до $k - 1$

$$\lambda[k] += a[j] \cdot r[k - j], \tag{1.14}$$

$$\lambda[k] += r[k], \tag{1.15}$$

$$a[k] = -\left(\frac{\lambda[k]}{E[k-1]}\right), \tag{1.16}$$

для j=1 до $k - 1$

$$a[j] = a[j] + a[k] \cdot a[k - j], \tag{1.17}$$

$$E[k] = E[k-1]\left(1 - a[k]^2\right), \tag{1.18}$$

де $r[k]$ – автокореляція $x[n]$ із затримкою k, а $E[k]$ – помилка передбачення на ітерації k.

Результатом роботи алгоритму рекурсії Левінсона-Дурбіна є набір коефіцієнтів AR $a[k]$, які можна використовувати для оцінки спектральної обвідної сигналу. Коефіцієнти LPC можуть бути отримані з коефіцієнтів AR шляхом застосування зворотного фільтра, який є рекурсивним фільтром з коефіцієнтами, що дорівнюють негативним значенням коефіцієнтів AR.

Алгоритм рекурсії Левінсона-Дурбіна має кілька переваг перед іншими методами оцінки коефіцієнта AR. Це обчислювально ефективний і чисельно стабільний алгоритм.

Опісля як MFCC, так і LPC передбачають виконання перетворення Фур’є віконного кадру для отримання його спектрального вмісту. Однак конкретний тип використовуваного перетворення Фур’є відрізняється між цими двома методами. MFCC використовує модифіковане дискретне косинусне перетворення (DCT), тоді як LPC використовує зворотне перетворення Фур’є (IFT).

1.2.2. *Зворотне перетворення Фур'є*

Зворотне перетворення Фур’є (IFT) – це процес перетворення представлення сигналу в частотній області назад у представлення у часовій області [3]. У обробці мовлення IFT часто використовується для реконструкції вихідного мовного сигналу з його подання в частотній області, наприклад, мел-частотних кепстральних коефіцієнтів (MFCC) або лінійних коефіцієнтів передбачення (LPC).

IFT виконується за допомогою формули зворотного перетворення Фур’є, яка є протилежністю формули перетворення Фур’є. Для сигналу з дискретним часом $x[n]$ зворотне перетворення Фур’є можна записати так:

$$x[n] = \frac{1}{N} \sum_{k=1}^{N} X[k] \cdot e^{\frac{2\pi jkn}{N}}, \tag{1.19}$$

де N – довжина сигналу, $X[k]$ – k-та частотна складова сигналу, j – уявна одиниця.

На практиці IFT зазвичай виконується за допомогою такого алгоритму, як зворотне швидке перетворення Фур’є (IFFT), який є обчислювально ефективним методом для обчислення IFT сигналу. Алгоритм IFFT базується на тому факті, що перетворення Фур’є та його обернене є симетричними операціями, і їх можна обчислити за допомогою одного набору операцій.

IFT є вирішальним кроком у багатьох програмах обробки мовлення, таких як синтез мовлення, де метою є генерація синтетичного мовлення з набору спектральних характеристик.

Як MFCC, так і LPC зазвичай включають етап нормалізації кепстрального середнього (CMN), який віднімає середнє значення кожного кепстрального коефіцієнта для всіх кадрів у мовному сигналі. Цей крок важливий для нормалізації та зменшення мінливості для різних динаміків і умов запису.

1.2.3. *Залишковий сигнал*

Останнім кроком вилучення LPC є обчислення залишкового сигналу, який є різницею між оригінальним мовним сигналом і виходом фільтра синтезу LPC. Залишковий сигнал містить інформацію, яка не фіксується моделлю LPC, і представляє глухі або шумоподібні компоненти мовного сигналу

Залишковий сигнал можна обчислити за такою формулою:

$$e[n] = x[n] \cdot y[n], \tag{1.20}$$

де $x[n]$ – вихідний мовний сигнал, $y[n]$ – вихідний сигнал фільтра синтезу LPC, а $e[n]$ – залишковий сигнал.

Залишковий сигнал можна використовувати для різних програм, таких як кодування мови, розпізнавання мовця та виявлення емоцій. У кодуванні мови залишковий сигнал може бути квантований і переданий разом з коефіцієнтами LPC для реконструкції вихідного мовного сигналу в приймачі. Під час розпізнавання мовця залишковий сигнал можна використовувати як джерело розрізнювальних ознак для розрізнення мовців. При виявленні емоцій залишковий сигнал можна використовувати для захоплення емоційного змісту мови, який не вловлюється моделлю LPC.

Підводячи підсумок, останній крок у вилученні LPC включає обчислення залишкового сигналу, який містить інформацію, яка не фіксується моделлю LPC, і може використовуватися для різних програм обробки мови.

1.2.4. *Характеристики LPC*

Загалом, хоча методи вилучення ознак, що використовуються в MFCC і LPC, відрізняються, вони мають спільні кроки, такі як кадрування мовного сигналу, вікна та застосування перетворень Фур’є. Ці кроки дозволяють виділити важливі функції з мовного сигналу, які можна

використовувати в різноманітних програмах обробки мовлення, таких як розпізнавання мовлення, ідентифікація мовця та виявлення емоцій.

Сильні сторони:

- LPC є обчислювально ефективним і простим у реалізації.
- Він може точно моделювати спектральну обвідну мовного сигналу, що робить його корисним для синтезу мовлення та програм кодування.

Слабкі сторони:

- LPC дуже чутливий до шуму та спотворень у сигналі.
- Може бути складно оцінити оптимальну кількість коефіцієнтів передбачення, що може вплинути на точність виділення ознак.

1.3. PLP

PLP (перцептивне лінійне передбачення) – це техніка, яка поєднує аспекти як MFCC, так і LPC. Він моделює спектральну обвідну мовних сигналів за допомогою лінійного передбачення, але банк фільтрів розроблено на основі сприйняття звуку людською слуховою системою, подібно до MFCC. PLP використовується для розпізнавання мовлення, перевірки мовця та кодування мовлення [4].

Принцип видобування даних не сильно відрізняється від алгоритму отримання MFCC. А саме, подібно до підходу MFCC, першим кроком у PLP є застосування фільтра попереднього акцентування до мовного сигналу, щоб підкреслити високочастотні компоненти сигналу. Після чого мовний сигнал аналогічно ділиться на короткі кадри, зазвичай тривалістю 20-30 мс, із 50% перекриттям між сусідніми кадрами, після чого кожен кадр помножується на функцію вікна, таку як вікно Хеммінга або Ханнінга для зменшення спектрального витоку. Перетворення Фур’є застосовується до кожного віконного кадру, щоб отримати його представлення в частотній області, а спектр потужності кожного кадру обчислюється шляхом

зведення квадрата величини коефіцієнтів перетворення Фур'є. На кшталт MFCC, спектр потужності потім пропускається через банк фільтрів Мел, щоб отримати набір виходів банку фільтрів масштабу Мел. На кроці логарифмування відбувається перша відмінність із шляхом отримання MFCC. А саме, замість логарифмування виходів банку фільтрів, як у MFCC, PLP застосовує нелінійне відображення до виходів банку фільтрів, використовуючи техніку, що називається деформацією частоти.

1.3.1. *Деформація частоти*

Деформація частоти – це техніка, яка використовується в цифровій обробці сигналу для зміни масштабу частотної осі з метою кращого відображення людського сприйняття звуку. У контексті PLP частотна деформація використовується для того, щоб зробити групу фільтрів більш розбірливою на нижчих частотах, де людське вухо більш чутливе.

Стандартна частотна шкала Mel, яка використовується в MFCC, базується на сприйнятті людським вухом гучності, але вона не враховує різницю в тому, як людина сприймає висоту на різних частотах. Це означає, що для низькочастотних звуків шкала Мела може бути занадто грубою, а для високочастотних звуків – занадто точною. Щоб подолати цю проблему, PLP використовує функцію нелінійної деформації частоти, яка стискає частотну шкалу на нижчих частотах і розширює її на вищих частотах. Ця нелінійна функція розроблена для кращого відображення того, як людське вухо сприймає висоту звуку на різних частотах.

Функція деформації частоти, яка використовується в PLP, базується на шкалі Барка, яка є психоакустичною шкалою, спочатку запропонованою Еберхардом Цвікером у 1960-х роках. Шкала Барка ділить діапазон чутних частот на критичні смуги однакової ширини, які базуються на частотній вибірковості людського вуха. Шкала Барка – це нелінійне перетворення частоти, яке точніше відображає діапазон низьких частот, ніж діапазон

високих частот, відображаючи спосіб, у який людське вухо більш чутливе до змін висоти на нижчих частотах.

Формула, яка використовується для розрахунку частоти шкали Барка:

$$z = 6\operatorname{arcsin}\left(\frac{6000}{f}\right), \tag{1.21}$$

де z – частота Барка, а f – частота в Гц.

Коли частоту шкали Барка обчислено, вона використовується для отримання коефіцієнтів банку фільтрів для процесу виділення ознак PLP. Банк фільтрів зазвичай будується з використанням трикутних фільтрів, які однаково розподілені за частотою Барка. Отриманий набір фільтрів розташовано ближче в діапазоні низьких частот і ширше у діапазоні високих частот, що краще відображає те, як людське вухо сприймає висоту звуку в діапазоні частот.

Таким чином, деформація частоти є важливою технікою, яка використовується в PLP для кращого відображення нелінійного способу, яким людське вухо сприймає висоту тону на різних частотах. Використання шкали Барка як основи для функції викривлення частоти дозволяє банку фільтрів PLP бути більш дискримінаційним у нижньому діапазоні частот, де людське вухо більш чутливе.

1.3.2. *Лінійне передбачення*

Наступним застосовується аналіз лінійного передбачення до виходів банку фільтрів із викривленням частоти, щоб отримати набір коефіцієнтів лінійного передбачення (LPC).

Лінійне передбачення (LP) – це техніка, яка використовується для моделювання сигналу часового ряду шляхом прогнозування поточного значення вибірки як лінійної комбінації минулих вибірок. У випадку виділення ознак PLP, коефіцієнти LP використовуються для моделювання спектральної обвідної сигналу.

Коефіцієнти LP зазвичай отримують за допомогою методу автокореляції, який передбачає обчислення функції автокореляції (ACF) сигналу та вирішення рівнянь Юла-Уокера для отримання коефіцієнтів LP. Однак у виділенні ознак PLP використовується модифікована версія методу автокореляції, яка передбачає застосування фільтра попереднього виділення до сигналу перед обчисленням ACF.

Фільтр попереднього підсилення зазвичай є фільтром високих частот першого порядку, заданим рівнянням:

$$y[n] = x[n] - \alpha \cdot x[n-1], \tag{1.22}$$

де $x[n]$ – вхідний сигнал, $y[n]$ – вихідний сигнал, а α – коефіцієнт фільтра, який контролює рівень високочастотного акценту. Фільтр попереднього акцентування застосовується до сигналу, щоб підвищити відносну важливість високочастотних компонентів, які мають тенденцію бути більш важливими для завдань розпізнавання мовлення.

Після застосування фільтра попереднього акцентування коефіцієнти LP отримують шляхом обчислення ACF відфільтрованого сигналу та вирішення рівнянь Юла-Уокера. ACF задається рівнянням:

$$r[k] = \frac{1}{N} \sum_{k=1}^{N} x[n] \cdot x[n-k], \tag{1.23}$$

де $r[k]$ – функція автокореляції, $x[n]$ – попередньо підкреслений вхідний сигнал, k – затримка, а N – кількість вибірок у сигналі.

Рівняння Юла-Уокера задано рівнянням:

$$\mathbf{R} \cdot \mathbf{a} = \mathbf{r}, \tag{1.24}$$

де $\mathbf{R}$ – матриця Тепліца, побудована зі значень ACF, $\mathbf{a}$ – вектор коефіцієнтів LP, $\mathbf{r}$ – вектор значень ACF.

Коефіцієнти LP можна отримати, розв’язуючи рівняння Юла-Уокера за допомогою різноманітних методів, включаючи алгоритм рекурсії Левінсона-Дурбіна.

Після отримання коефіцієнтів LP їх можна використовувати для оцінки спектральної обвідної сигналу, яка потім використовується для обчислення характеристик PLP.

1.3.3. *Кепстральне згладжування*

Потім LPC згладжуються за допомогою техніки кепстрального згладжування, такої як підйом або спектральне віднімання, щоб зменшити вплив варіацій спектральної огинаючої.

Кепстральне згладжування – це техніка, яка використовується для усунення ефекту короткочасної спектральної зміни мовного сигналу. Ця техніка використовується в обробці мовлення для виділення основних характеристик мовного сигналу, які залишаються незмінними з часом.

Кепстральне згладжування включає перетворення Фур'є логарифму спектра величини сигналу. Цей процес подібний до виділення мелчастотних кепстральних коефіцієнтів (MFCC), де для виділення спектральної обвідної береться логарифм спектра величини мовного сигналу.

Спектральна огинаюча, отримана в процесі кепстрального згладжування, згладжується шляхом фільтрації за допомогою фільтра низьких частот. Цей процес згладжування усуває короткочасну спектральну зміну мовного сигналу, залишаючи лише довгострокову спектральну обвідну. Згладжуючий фільтр зазвичай реалізується як лінійно-фазовий фільтр із кінцевою імпульсною характеристикою (КІХ).

Процес кепстрального згладжування можна описати математично так:

Обчисліть короткочасне перетворення Фур’є (STFT) мовного сигналу x(n), щоб отримати спектр величини $|X(k, l)|$, де k – індекс частоти, а l – індекс часового кадру.

Візьміть логарифм спектра величини, щоб отримати спектр логарифмів величини: $\ln(|X(k, l)|)$

Проводимо обчислення зворотного перетворення Фур'є спектра логарифмічної величини, щоб отримати кепстр:

$$\mathrm{c}(n,l) = \mathrm{IDFT}[\ln(|\,\mathrm{X}(k,l)\,|)], \tag{1.25}$$

Згладжуємо кепстр вздовж осі часу за допомогою фільтра низьких частот:

$$\mathrm{c}'(n,l) = \mathrm{filter}(\mathrm{c}(n,l)\cdot \mathrm{h}(n)), \tag{1.26}$$

де $\mathrm{h}(n)$ – фільтр згладжування.

Обчисліть зворотне перетворення Фур'є згладженого кепстра, щоб отримати згладжений спектр:

$$\mathrm{S}(k,l) = \mathrm{IDFT}[\mathrm{c}'(n,l)], \tag{1.27}$$

Згладжений спектр $\mathrm{S}(k, \mathrm{l})$ можна використовувати як вхідні дані для процесу виділення PLP для отримання коефіцієнтів PLP. Процес кепстрального згладжування допомагає усунути короткочасну спектральну зміну мовного сигналу, роблячи його більш стійким до шуму та спотворень каналу.

Нарешті, DCT застосовується до згладжених LPC для отримання функцій PLP.

Функції PLP подібні до функцій MFCC тим, що вони захоплюють спектральні характеристики мовного сигналу компактним і релевантним для сприйняття способом. Однак PLP, як відомо, більш стійкий до шуму та спотворень каналу, ніж MFCC, оскільки використовує більш складну техніку спектральної обробки.

Процес виділення функції PLP можна підсумувати такою формулою:

$$\mathrm{PLP} = \mathrm{DCT}\left(\log\left(\mathrm{E}\left(\mathrm{x}[m]\right)\right) + R\cdot\cos\left(\frac{\pi m}{M}\right)\right), \tag{1.28}$$

де $\mathrm{E}(x[m])$ – це енергія m-го діапазону частот, відфільтрованого Mel, R – кількість кепстральних коефіцієнтів, які потрібно зберегти, M – кількість діапазонів частот, відфільтрованих Mel, і функція косинуса є терміном нормалізації DCT.

1.3.4. *Характеристики PLP*

Сильні сторони:

- PLP менш чутливий до шуму, ніж LPC, що робить його корисним для шумного середовища.
- Він фіксує як спектральні, так і часові характеристики мовних сигналів, що робить його придатним для програм розпізнавання мовлення.

Слабкі сторони:

- PLP є обчислювально дорожчим, ніж MFCC, що може бути недоліком у програмах реального часу.
- Він може бути не таким широко використовуваним або добре відомим, як MFCC у сфері обробки мови.
- Загалом, вибір методу, який використовувати, залежить від конкретного застосування та характеристик мовного сигналу, що аналізується. MFCC є найбільш часто використовуваною технікою, яка підходить для широкого спектру застосувань. LPC корисний для кодування мови та перетворення голосу, тоді як PLP підходить для шумного середовища та програм розпізнавання мови.

1.4. DWT

Дискретне вейвлет-перетворення (DWT) – це техніка аналізу сигналів, яка передбачає розкладання сигналу на набір базових функцій, які називаються вейвлетами. Цей метод особливо корисний для сигналів, які демонструють нестаціонарну поведінку, тобто їхні статистичні властивості змінюються з часом [5].

DWT включає серію операцій фільтрації та субдискретизації, які створюють серію наближень і деталей сигналу на різних рівнях роздільної

здатності. Розкладання зазвичай виконується ітераційно, причому кожна ітерація забезпечує вищий рівень роздільної здатності.

DWT можна представити математично за допомогою банків фільтрів. Банк фільтрів – це набір фільтрів, які використовуються для розділення сигналу на різні діапазони частот. Банк фільтрів DWT складається з двох типів фільтрів: фільтри аналізу та фільтри синтезу. Фільтри аналізу використовуються для розкладання сигналу, тоді як фільтри синтезу використовуються для реконструкції сигналу з вейвлет-коефіцієнтів.

DWT можна визначити математично таким чином:

Нехай $x[n]$ – дискретний сигнал довжини N. DWT $x[n]$ визначається як:

$$\mathrm{DWT}(\mathrm{x}[n]) = \left\{ w_{\{j,k\}}, v_{\{j,k\}} \right\}, \tag{1.29}$$

де $w_{\{j,k\}}$ і $v_{\{J,K\}}$ – вейвлет-коефіцієнти та коефіцієнти масштабування відповідно на рівні j і позиції k. Масштабні коефіцієнти представляють низькочастотні компоненти сигналу, а вейвлет-коефіцієнти представляють високочастотні компоненти.

DWT можна обчислити за допомогою серії операцій фільтрації та субдискретизації. Фільтрація виконується шляхом згортання сигналу за допомогою набору фільтрів аналізу, які зазвичай вибираються як фільтр високих частот і фільтр низьких частот. Підвибірка виконується шляхом вибору кожного другого зразка з відфільтрованого виходу.

Алгоритм дискретного вейвлет-перетворення (DWT) можна розділити на такі кроки:

Розкладання

Вхідний сигнал пропускається через фільтр низьких частот і фільтр високих частот, що призводить до двох наборів коефіцієнтів, відомих як коефіцієнти апроксимації та коефіцієнти деталізації відповідно.

Коефіцієнти апроксимації додатково субдискретизуються з коефіцієнтом два.

Той самий процес рекурсивно застосовується до коефіцієнтів апроксимації, доки не буде досягнуто бажаного рівня розкладання.

Реконструкція

Коефіцієнти деталізації на кожному рівні декомпозиції підвищуються в два рази.

Коефіцієнти деталізації з підвищеною дискретизацією пропускаються через фільтр низьких частот і фільтр високих частот для отримання коефіцієнтів апроксимації на наступному вищому рівні розкладання.

Коефіцієнти апроксимації на кожному рівні декомпозиції поєднуються з коефіцієнтами деталізації підвищеної дискретизації на наступному нижчому рівні, щоб отримати коефіцієнти апроксимації на поточному рівні декомпозиції.

Етапи декомпозиції та реконструкції можна представити математично так:

Розкладання

Для заданого вхідного сигналу x[n] коефіцієнти апроксимації A_j і коефіцієнти деталізації D_j на рівні j можна отримати за допомогою таких рівнянь (низькочастотна фільтрація та зниження частоти дискретизації та фільтрація високих частот і зниження частоти дискретизації відповідно):

$$A_{j-1}[k] = H[k] \cdot (A_j[k] + D_j[k]), \tag{1.30}$$

$$D_{j-1}[k] = G[k] \cdot (A_j[k] + D_j[k]), \tag{1.31}$$

де H[k] і G[k] – фільтри нижніх і високих частот відповідно, а * позначає згортку.

Декомпозицію можна виконувати ітеративно, поки не буде досягнуто бажаного рівня декомпозиції

Реконструкція

Для заданого набору коефіцієнтів апроксимації A_j і коефіцієнтів деталізації D_j на рівні j, коефіцієнти апроксимації A_{j+1} на наступному

вищому рівні декомпозиції можна отримати за допомогою такого рівняння (підвищена дискретизація та фільтрація):

$$A_{j+1}[k] = H[k] \cdot A_j[k] + G[k] \cdot D_j[k], \quad (1.32)$$

Реконструкцію можна виконувати ітераційно, доки не буде отримано вихідний сигнал $x[n]$.

У цих рівняннях $H[k]$ і $G[k]$ є синтезованими фільтрами низьких і високих частот відповідно, які пов'язані з фільтрами аналізу, що використовуються на етапі декомпозиції. Операція підвищення дискретизації полягає в простому вставленні нулів між сусідніми зразками сигналу.

1.4.1. *Характеристики DWT*

DWT має кілька переваг перед іншими методами обробки сигналів, такими як перетворення Фур'є. Однією з переваг є те, що DWT може надавати інформацію про частотно-часову локалізацію сигналу, що важливо для аналізу нестаціонарних сигналів. Ще одна перевага полягає в тому, що DWT можна використовувати для стиснення даних, оскільки вейвлет-коефіцієнти зазвичай мають розріджене представлення.

2. ОПИС РОЗРОБЛЕНОГО МЕТОДУ

Аналіз мовлення є важливою галуззю дослідження, яка займається виділенням і аналізом акустичної та лінгвістичної інформації з мовних сигналів. Для цього були розроблені різні методики, включаючи MFCC, LPC, PLP і DWT. Кожен із цих методів має свої сильні та слабкі сторони, і дослідники намагаються розробити нові методи, поєднуючи сильні сторони різних методів.

MFCC є одним із найбільш часто використовуваних методів аналізу мовлення. Він заснований на спостереженні, що людське вухо сприймає звукові хвилі не лінійно, а швидше логарифмічно. MFCC намагається змоделювати це логарифмічне сприйняття звуку, розділивши частотний спектр на серію частотних смуг шкали Mel. Потім він виділяє спектр потужності кожної смуги та застосовує логарифмічне перетворення, щоб отримати кепстральні коефіцієнти Mel-частоти. Сильні сторони MFCC включають його здатність охоплювати спектральну огинаючу мови, його стійкість до шуму та його обчислювальну ефективність.

З іншого боку, LPC – це метод, який моделює голосовий тракт як лінійний фільтр і використовує коефіцієнти цього фільтра для представлення мовного сигналу. Коефіцієнти фільтра оцінюються за допомогою авторегресійної моделі, а отримані коефіцієнти LPC представляють спектральну обвідну мовного сигналу. Сильні сторони LPC включають його здатність точно моделювати спектральну обвідну, його простоту та ефективність у програмах кодування мови.

PLP є модифікацією техніки MFCC, яка намагається усунути деякі її обмеження. Він використовує функцію нелінійної деформації частоти для трансформації частотної шкали та зменшення спектрального розмиття, яке відбувається з MFCC. PLP також використовує фільтр попереднього виділення, щоб підкреслити високочастотні компоненти мовного сигналу.

Сильні сторони PLP включають покращену спектральну роздільну здатність, стійкість до шуму та здатність вловлювати гармоніки мови.

DWT – це метод частотно-часового аналізу, який розкладає мовний сигнал на ряд вейвлет-коефіцієнтів. Це дозволяє представити сигнал як у часовій, так і в частотній областях, що робить його потужним інструментом для аналізу мови. Сильні сторони DWT включають його здатність вловлювати перехідні особливості мови, його високу частотно-часову роздільну здатність і його обчислювальну ефективність.

Незважаючи на свої сильні сторони, кожен із цих методів має обмеження, які не дозволяють використовувати їх ізольовано для певних програм. Наприклад, і MFCC, і LPC чутливі до змін у спектральній огинаючій мови, але можуть не вловлювати найдрібніші деталі мовного сигналу. PLP намагається вирішити деякі з цих обмежень, але не підходить для всіх програм. DWT, незважаючи на потужну здатність представляти мову як у часовій, так і в частотній областях, може не підходити для програм реального часу через свої обчислювальні вимоги.

Таким чином, існує попит на нові методи, які поєднують сильні сторони цих окремих методів, мінімізуючи їх обмеження. Одним із можливих підходів є поєднання MFCC і LPC, оскільки вони обидва моделюють спектральну оболонку мови, але різними способами. Інший підхід полягає в поєднанні PLP і DWT, оскільки вони обидва намагаються вирішити обмеження MFCC різними способами. Третій підхід полягає в поєднанні всіх чотирьох методів у гібридному методі, який використовує сильні сторони кожного методу.

На додаток до об’єднання існуючих методів, існує також потреба в розробці нових методів, які використовують останні досягнення в машинному навчанні та обробці сигналів. Наприклад, методи глибокого навчання показали багатообіцяючі результати в задачах аналізу мовлення, таких як розпізнавання мовлення та ідентифікація мовця. Ці методи можна

поєднати з традиційними методами для створення нових, більш потужних методів аналізу мовлення.

Отже, аналіз мовлення – це складна галузь, яка потребує комбінації методів для більш точного охоплення різноманітних характеристик, присутніх у мовних сигналах. Хоча такі методи, як MFCC, LPC, PLP і DWT, мають свої індивідуальні переваги, вони також мають обмеження, які не дозволяють використовувати їх ізольовано для певних програм.

Таким чином, беручи до уваги сильні та слабкі сторони розглянутих методів, було прийнято рішення на основі досліджень розробити відповідно комбінований метод, який найкраще підходить для отримання ознак із аудіозаписів з англомовними акцентами із ціллю отримання покращення в подальшій класифікації аудіозаписів із відповідним англійським акцентом.

2.1. Розроблений метод

Аналіз мовлення є важливою областю дослідження з численними практичними застосуваннями. Одним із таких застосувань є класифікація акцентів. Протягом багатьох років були розроблені різні методи, такі як MFCC, LPC, PLP і DWT, щоб витягти особливості з мовних сигналів. Проте кожен метод має свої обмеження, що ускладнює точну класифікацію мовних сигналів. У цьому есе я запропоную новий метод класифікації акцентів, який поєднує в собі сильні сторони вищезгаданих методів, долаючи їхні обмеження.

По-перше, метод Mel-Frequency Cepstral Coefficients (MFCC) був найбільш широко використовуваним методом для виділення ознак мовлення через його ефективність у захопленні спектральних характеристик мовного сигналу. Однак одним із його недоліків є те, що він передбачає, що спектральна обвідна мовного сигналу є лінійною. У деяких випадках це припущення може не виконуватися, що призводить до викривлення характеристик. Ще одна проблема з MFCC полягає в тому, що

він сприйнятливий до шуму, який може заважати точності вилучення ознак.

По-друге, метод лінійних коефіцієнтів прогнозування (LPC) був використаний для моделювання функції передачі голосового тракту шляхом прогнозування поточного зразка з попередніх зразків. LPC ефективний у захопленні формант мовного сигналу, що робить його цінним інструментом для виділення ознак мовлення. Однак одним із його недоліків є те, що він дорогий з точки зору обчислень, особливо при аналізі довгих мовних сигналів. Крім того, LPC чутливий до наявності шуму, який може значно знизити його точність у виділенні ознак. Метод дещо нагадує MFCC, тому за наявних вище переваг доцільно використати саме його.

По-третє, метод перцептивного лінійного прогнозування (PLP) є розширенням LPC, яке розроблено для подолання деяких його обмежень. PLP моделює слухову систему людини, використовуючи деформацію частоти для зменшення спектральних спотворень, викликаних LPC. Цей метод також включає додаткові функції, такі як стиснення динамічного діапазону та спектральна нормалізація, що робить його більш стійким до шуму. Однак PLP є дорогим з обчислювальної точки зору та вимагає ретельного вибору функції викривлення, щоб уникнути перепідбору, тому було вирішено не використовувати його.

Нарешті, метод дискретного вейвлетного перетворення (DWT) використовувався для виділення ознак у різних програмах завдяки його здатності фіксувати як частотну, так і часову інформацію. DWT використовувався для вилучення характеристик із мовних сигналів для класифікації емоцій, розладів мови та ідентифікації мовця. Однак DWT має обмеження щодо чутливості до вибору вейвлет-функції, яка використовується в перетворенні, що може вплинути на точність виділення ознак.

Запропонований комбінований метод полягає у поєднанні сильних сторін вище згаданих методів, долаючи їхні обмеження та включає такі кроки:

1. Попереднє виділення: цей крок є звичайним у більшості методів аналізу мовлення та передбачає посилення високочастотних компонентів мовного сигналу для зменшення шуму та підвищення точності виділення ознак.

2. Сегментація кадрів: проводиться сегментація мовного сигналу на кадри однакової довжини, щоб забезпечити ефективне виділення ознак, використовується сегментація мовного сигналу на кадри однакової довжини. Цей крок є ключовим у всіх методах аналізу мовлення та передбачає накладання суміжних кадрів на 50%, щоб уникнути втрати важливих функцій на краях кадрів.

3. Вилучення функції DWT: використовується метод DWT для захоплення інформації про час і частоту з кожного кадру мовного сигналу.

4. Після застосування DWT і вилучення коефіцієнтів наступним кроком є застосування LPC до цих коефіцієнтів. LPC є кращим перед PLP та MFCC, оскільки він виконує деформацію частоти та застосовує плавний фільтр для усунення невеликих флуктуацій у спектрі. Це допоможе зменшити вплив шуму та зробить характеристики більш надійними для класифікації. Крім того, PLP використовує нелінійну частотну шкалу, яка може захоплювати більш релевантну інформацію з мовного сигналу.

5. Після об’єднання методів наступним кроком є застосування техніки нормалізації. Проводиться нормалізація кепстрального середнього (CMN) або масштабування ознак, щоб нормалізувати функції. CMN є кращим, оскільки він усуває ефекти каналів і робить функції більш надійними для класифікації. Після нормалізації об’єкти готові до класифікації.

6. Щоб оцінити ефективність нового методу, використовується набір даних мовних сигналів з різними акцентами, розділяється набір даних на

набори для навчання та тестування та використовувати цей набір для навчання алгоритму машинного навчання. Потім використовується тестовий набір для оцінки точності алгоритму. Наступним порівнюється точність нового методу з точністю існуючих методів (LPC і DWT), щоб визначити, чи новий метод перевершує їх.

На закінчення, запропонований метод поєднання DWT і LPC для класифікації акцентів має потенціал для підвищення точності існуючих методів. Використання DWT забезпечує більш компактне представлення мовного сигналу, тоді як LPC дає більш повне представлення мовного сигналу. Використання CMN для нормалізації додатково покращує стійкість ознак для класифікації. Запропонований метод можна оцінити за допомогою набору даних мовних сигналів з різними акцентами та порівняти з існуючими методами для визначення його ефективності.

Для класифікації використовується алгоритм машинного навчання, такий як згорткова нейронна мережа (CNN). Алгоритм машинного навчання вивчатиме закономірності в об’єктах і використовуватиме їх для класифікації акцентів.

2.2. Математична репрезентація розробленого методу

Як і в умовах розгляду будь-якого методу для видобування ознак, для розробленого методу використання кроку попередньої обробки, та всі наступні кроки, що пов’язані із попередньою обробкою аудіоданих не є виключенням. Отже, аудіосигнал попередньо обробляється для видалення будь-яких шумів або спотворень, які можуть вплинути на точність виділення ознак. Цей крок передбачає фільтрацію сигналу, видалення будь-якого зміщення постійного струму та нормалізацію амплітуди.

Те ж саме стосується сегментації, а саме аудіо сигнал ділиться на менші кадри однакової тривалості. Зазвичай кожен кадр має довжину 20-30

мс із 50% перекриттям між сусідніми кадрами. Цей крок допомагає вловити часову динаміку мовного сигналу.

Наступним в будь-якому алгоритмі видобування ознак слугує застосування віконної функції. Віконна функція застосовується до кожного кадру, щоб зменшити спектральний витік, який виникає через розриви на границях кадру. Найбільш часто використовуваною віконною функцією є вікно Хеммінга (формула 1.4).

Опісля швидке перетворення Фур'є (FFT) застосовується до кожного віконного кадру, щоб отримати його частотний спектр. БПФ перетворює сигнал із часової області в частотну. Отриманий спектр є комплексним вектором із N / 2+1 розділів частоти, де N є довжиною FFT. Формула для FFT є дещо модифікованою версією формули DFT:

$$X[k] = \sum_{k=0}^{\frac{N}{2}-1} x[n] \cdot e^{\frac{-2\pi ikn}{N}}, \qquad (2.1)$$

Наступним застосовується банк фільтрів Mel до спектру, щоб імітувати нелінійне сприйняття звуку людиною. Шкала Мела – це нелінійне перетворення частоти, яке відображає перцептивну відстань між частотами, а не їхню фізичну відстань (формула 1.1).

Банк фільтрів Mel – це набір трикутних фільтрів, рівномірно розподілених на шкалі Mel. Фільтри застосовуються до спектру, і енергія всередині кожного фільтра підсумовується для отримання набору частотних кепстральних коефіцієнтів Mel (MFCC).

MFCC нормалізуються шляхом віднімання середнього значення кожного коефіцієнта для всіх кадрів мовного сигналу. Цей крок допомагає зменшити вплив мінливості каналів і гучномовців. Нехай $C_t(m)$ буде m-м коефіцієнтом MFCC t-го кадру мовного сигналу, і нехай mu(m) буде середнім значенням m-го коефіцієнта MFCC для всіх кадрів у мовному сигналі. Тоді нормалізація CMN m-го коефіцієнта MFCC t-го кадру мовного сигналу, позначеного як $C_t'(m)$, обчислюється як:

$$C_t^{'}(m) = C_t(m) - \mathrm{mu}(m), \tag{2.2}$$

Іншими словами, нормалізація CMN віднімає середнє значення кожного коефіцієнта MFCC для всіх кадрів у мовному сигналі з відповідного коефіцієнта в кожному кадрі. Це допомагає усунути вплив різних рівнів енергії та довжини голосового тракту в різних кадрах мовного сигналу та може покращити продуктивність завдань аналізу мовлення, таких як розпізнавання мовлення та ідентифікація мовця.

DCT застосовується до нормалізованих MFCC для декореляції коефіцієнтів і отримання компактного представлення спектральної огинаючої. Коефіцієнти DCT зазвичай скорочуються, щоб зберегти найбільш важливі компоненти.

Формула для дискретного косинусного перетворення (DCT) сигналу x(n) довжини N визначається так:

$$X[k] = 2\sum_{k,n=0}^{N-1}\left(x[n]\cdot\cos\left(\frac{\pi k(n+0.5)}{N}\right)\right), \tag{2.3}$$

де $X[k]$ – k-й коефіцієнт DCT, $x[n]$ - вибірка сигналу у часовій області в момент часу n, k – індекс коефіцієнта DCT, N – довжина сигналу.

Дана формула представляє DCT II типу, яка є найбільш часто використовуваною формою DCT.

Для кроку кепстрального підйому перший коефіцієнт MFCC віднімається з усіх інших коефіцієнтів MFCC, щоб усунути вплив довжини голосового тракту. Потім отримані коефіцієнти множаться на кепстральний ліфтер, який підкреслює високочастотні компоненти спектру. Формула кепстрального підйому визначається так:

$$L(n) = 1 + \frac{L}{2}\sin\left(\frac{\pi n}{L}\right), \tag{2.6}$$

де L – коефіцієнт підйому, а n – індекс коефіцієнта.

Наступним мовний сигнал розкладається за допомогою дискретного вейвлет-перетворення (DWT) на набір коефіцієнтів, які представляють

різні масштаби та орієнтації. Цей крок допомагає захопити як спектральну, так і часову інформацію в мовному сигналі.

Формула для дискретного вейвлет-перетворення (DWT) сигналу $x[n]$ щодо вейвлет-функції $\psi(t)$ і функції масштабування $\varphi(t)$ виглядає наступним чином:

Для j-го рівня декомпозиції коефіцієнти DWT обчислюються за допомогою наступних рівнянь:

$$c_j[k] = < x[n], \psi_{j,k[n]} > = \frac{1}{\sqrt{2^j}} \sum x[n] \cdot \psi_{j,k[n]}, \quad (2.7)$$

$$d_j[k] = < x[n], \varphi_{j,k[n]} > = \frac{1}{\sqrt{2^j}} \sum x[n] \cdot \varphi_{j,k[n]}, \quad (2.8)$$

де j – рівень розкладання, k – індекс коефіцієнта DWT, $\psi_{j,k[n]}$ – вейвлет-функція $\psi(t)$, яка була масштабована на 2^j і зміщена на k, $\varphi_{j,k[n]}$ – функція масштабування $\varphi(t)$, яка була масштабована на 2^j *і* зміщена на k;
$<x[n], \psi_{j,k[n]}>$ та $<x[n], \varphi_{j,k[n]}>$ позначають скалярний добуток $x[n]$ на $\psi_{j,k[n]}$ та $\varphi_{j,k[n]}$, відповідно.

Коефіцієнти DWT $c_j[k]$ представляють високочастотні компоненти сигналу на j-му рівні розкладання, тоді як коефіцієнти $d_j[k]$ представляють низькочастотні компоненти. Процес обчислення коефіцієнтів DWT можна рекурсивно повторювати на низькочастотних компонентах, щоб отримати розкладання сигналу з різною роздільною здатністю.

PCA застосовується до коефіцієнтів DWT для зменшення їх розмірності та отримання компактного представлення мовного сигналу. Цей крок допомагає зменшити обчислювальну складність завдання класифікації.

Аналіз основних компонентів (PCA) – це статистичний метод, який використовується для зменшення розмірності набору даних, зберігаючи при цьому якомога більшу частину вихідної варіації. Маючи набір даних із n змінних, PCA перетворює змінні на набір нових змінних, які називаються головними компонентами, які є лінійними комбінаціями вихідних змінних.

Перший головний компонент відповідає за найбільшу кількість варіацій у даних, а кожен наступний компонент – якомога більшу частину варіацій, що залишилися.

Формула для обчислення головних компонент така:

1. Стандартизуємо дані шляхом віднімання середнього значення кожної змінної від кожного спостереження в цій змінній, після чого поділимо кожне спостереження на стандартне відхилення цієї змінної.
2. Обчислюємо коваріаційну матрицю стандартизованих даних:

$$\mathbf{S} = \frac{1}{n-1}\left(\left(\mathbf{X} - \bar{\mathbf{x}} \right)^T \left(\mathbf{X} - \bar{\mathbf{x}} \right) \right), \tag{2.9}$$

де $\mathbf{X}$ – стандартизовані дані для спостереження i, $\bar{\mathbf{x}}$ – середнє значення стандартизованих даних для всіх спостережень, а n – кількість спостережень. Коваріаційна матриця $\mathbf{S}$ – це симетрична матриця, яка описує зв'язки між змінними в наборі даних.

3. Обчислимо власні вектори та власні значення коваріаційної матриці:

$$\mathbf{S} \cdot \mathbf{v_i} = \lambda_i \cdot \mathbf{v_i}, \tag{2.10}$$

де λ_i – і-е власне значення, а $\mathbf{v_i}$ – відповідний власний вектор. Власні вектори представляють напрямки, у яких дані мають найбільшу варіацію, а власні значення представляють величину варіації даних у цих напрямках.

4. Відсортовуємо власні вектори за їхніми власними значеннями в порядку спадання. Перший головний компонент – це власний вектор, що відповідає найбільшому власному значенню, другий головний компонент – власний вектор, який відповідає другому за величиною власному значенню, і так далі.
5. Обчислюємо перетворені дані:

$$\mathbf{X}^* = \mathbf{XV}, \tag{2.11}$$

де **X** – вихідна матриця даних, а **V** – матриця власних векторів, розташованих у порядку спадання власних значень. $\mathbf{X}^*$ – це трансформована матриця даних із такою ж кількістю рядків, що й **X**, але з меншою кількістю стовпців, що дорівнює кількості вибраних головних компонентів.

PCA є корисною технікою для багатьох застосувань, включаючи зменшення розмірності, стиснення даних і вилучення функцій. Його можна застосовувати до різних типів даних, включаючи числові дані, дані зображень і текстові дані.

3. СТРУКТУРНО-АЛГОРИТМІЧНА ОРГАНІЗАЦІЯ СИСТЕМИ

3.1. Вибір мови програмування для розроблення серверної частини

Щоб задовольнити поставлені вимоги, необхідно вибрати мову програмування, яка буде максимально гнучкою та універсальною. Основними вимогами є наступне:

- мова програмування повинна бути мультипарадигмальною;
- необхідні функціональні можливості, які забезпечують спеціальні бібліотеки, для побудови веб-сервера;
- має бути можливість вибору між жорсткими та динамічними типами;
- перевагою буде вибір такої мови, яка може використовуватися на стороні клієнта.

3.1.1. *C++*

C++ – це мова програмування високого рівня, яка підтримує кілька парадигм програмування, зокрема об'єктно-орієнтовану, узагальнену та процедурну.

C++ є прямим нащадком мови C, тому вона має всі її особливості синтаксису та компіляції. Однак, вона також надає розробникам вихід у величезний простір допоміжних бібліотек [6].

Мова не є дуже популярною для розробки веб-серверів, проте вона пропонує низку бібліотек, що використовують її сильні сторони та надають перевагу перед аналогами у швидкості та протидії високому навантаженню, наприклад, CppCMS.

Переваги використання C++ для розробки бекенду полягають у тому, що виконання програм на цій мові є найшвидшим серед аналогів. Крім того, екосистема мови є однією з найбільших серед усіх мов програмування, що означає, що є величезне ком'юніті для підтримки

найрізноманітніших бібліотек. Також мова дозволяє доступатися до пам'яті, адрес та портів на найнижчому рівні, що дає можливість максимально оптимізувати роботу програми.

Проте, є також недоліки. Наприклад, C++ вимагає значну кількість часу для компіляції. Крім того, мова є дуже складною та вимагливою до новачків, оскільки потребує обережності при роботі із пам'яттю та бракує "збирача сміття". Хоча C++ є мультипарадигмною мовою програмування, дійсної підтримки або виразів функціонального програмування їй браку.

3.1.2. *Python*

Python – це інтерпретована мова програмування високого рівня, яка забезпечує ефективність та простоту для розробки. Python є досить популярною мовою для розробки бекенду, оскільки має велику кількість фреймворків та бібліотек, які полегшують розробку. Найбільш відомий фреймворк для розробки бекенду на Python – це Django [7].

Переваги:

- Python є однією з найпопулярніших мов для програмування, що означає, що величезне ком'юніті підтримує його. Це дозволяє розробникам легко знайти відповідні бібліотеки та документацію.
- Python – інтерпретована мова, що дозволяє розробникам бачити результат своєї роботи майже миттєво.
- Python має велику кількість бібліотек, які забезпечують розробку бекенду, включаючи NumPy, Pandas, Flask, Pyramid та багато інших.
- Python має простий та лаконічний синтаксис, що дозволяє розробникам зосередитися на реалізації функціональності, а не на деталях синтаксису.

Недоліки:

- Python може бути повільнішим у порівнянні з іншими мовами програмування, такими як C++ або Java.
- Python, як і будь-яка інша інтерпретована мова, потребує великої кількості ресурсів комп'ютера, що може призвести до витоку пам'яті і зниження продуктивності.
- Python не є найкращим вибором для проєктів з високими вимогами до безпеки, оскільки не володіє такою ж ступенем ефективності, як мови, написані на C++.

3.1.3. *JavaScript*

JavaScript є мовою програмування, що зазвичай використовується для розробки веб-фронтенду. Однак, завдяки технології Node.js, JavaScript можна використовувати для розробки бекенду [8].

Переваги:

- JavaScript має велику кількість різноманітних бібліотек, що дозволяє значно спростити і прискорити процес розробки бекенду.
- Node.js дозволяє використовувати JavaScript на стороні сервера, що забезпечує більш гладкий процес зв'язку між фронтендом та бекендом.
- JavaScript є досить простою мовою програмування, що робить її досить доступною для новачків.
- Використання однієї мови програмування на фронтенді та бекенді дозволяє зменшити витрати на навчання та розробку проекту.

Недоліки:

- JavaScript має свої особливості, які можуть бути неприємними для розробників, зокрема, асинхронний код та callback-функції.
- Мова програмування JavaScript не є дуже швидкою, що може призвести до проблем з продуктивністю.

- JavaScript не має сильної типізації, що може призвести до помилок в роботі програми, особливо при роботі з великими проєктами.

3.1.4. *TypeScript*

TypeScript – це мова програмування, яка є синтаксичним розширенням JavaScript, що надає додаткові можливості інструментів розробки, таких як перевірка типів та відстеження помилок під час розробки. TypeScript стає все більш популярним для використання на бекенді разом з фреймворками, такими як Node.js [9].

Переваги TypeScript для бекенду:

- TypeScript дозволяє під час розробки бекенду виявляти помилки раніше, завдяки перевірці типів.
- TypeScript має розширену систему типів, що робить код більш безпечним та робить його легше для зрозуміння та розширення.
- TypeScript підтримує багато функцій з ES6 та ES7, що дозволяє використовувати нові функції, такі як async / await та інші, що полегшують написання асинхронного коду.
- TypeScript забезпечує зручний інтерфейс для написання класів та інтерфейсів, що дозволяє створювати більш структурований та підтримуваний код.

Недоліки TypeScript для бекенду:

- TypeScript може бути складним для вивчення для розробників, які раніше не мали досвіду з мовою.
- TypeScript вимагає компіляції до JavaScript, що може займати деякий час, але це компенсується швидкістю розробки та меншою кількістю помилок від тестування.
- TypeScript зазвичай використовується разом з Node.js та іншими фреймворками, що може бути не зручним для деяких розробників.

3.1.5. *Kotlin*

Kotlin є мовою програмування, яка була створена для розв'язання питань, які можуть виникнути під час розробки платформ Android. Однак, з часом вона стала популярною і для розробки бекенду [10].

Основні переваги Kotlin для розробки бекенду включають:

- Безпека: Kotlin є безпечною мовою програмування, оскільки має систему типів, що дозволяє виявляти помилки на етапі компіляції.
- Простота: Kotlin має простий і легкий для вивчення синтаксис, що дозволяє розробникам швидко створювати код і підтримувати його.
- Інтероперабельність: Kotlin може легко взаємодіяти з іншими мовами програмування, такими як Java, що робить його дуже привабливим для розробки бекенду.
- Функціональна парадигма: Kotlin підтримує функціональну парадигму програмування, що дозволяє розробникам створювати більш простий, чистий і лаконічний код.
- Велика підтримка з боку Google та JetBrains: Kotlin розробляється компанією JetBrains, яка є одним з провідних розробників інструментів для розробки програмного забезпечення. Google також підтримує Kotlin і включив його в свої офіційні інструменти для розробки Android.

Хоча Kotlin може бути більш привабливим для розробки мобільних додатків, він також підходить для розробки бекенду завдяки своїм перевагам та підтримці.

3.1.6. *Висновки*

Розглянувши популярні мови розробки серверної частини вебдодатків та проаналізувавши їх характеристики і деталі, було вирішено вибрати найдоцільніший з перерахованих інструментів – Node.js, мовою програмування – TypeScript, оскільки не тільки мова, а й частина фреймворку може бути застосована в розробці клієнтської частини додатку.

Насправді, вибір постав між TypeScript та Kotlin, оскільки обидва компілюються в JavaScript та пишуться під Node.js, але саме варіативність TypeScript мотивує до використання для розробки всього додатку, а також Телеграм бота.

3.2. Вибір інструментів для розроблення клієнтської частини

JavaScript є домінуючою мовою програмування для розробки вебсайтів, а TypeScript може бути вибраним як альтернатива для клієнтської частини. Проте, вибір мови є лише першим кроком у вирішенні проблеми вибору технологій для веб-розробки.

Для розробки веб-сторінок існує велика кількість можливостей, від простого відображення результатів із сервера до складніших розрахунків на стороні клієнта, використовуючи сучасні фреймворки. Серед трьох основних фреймворків для розробки клієнтської частини, а саме Angular.js, React.js та Vue.js, вибір може бути зроблений на основі таких критеріїв, як швидкість роботи і лаконічність написання, наявність бібліотек для представлення сторінки та масштабованість.

Критерії вибору можуть залежати від специфіки проекту і можуть варіюватися залежно від потреб користувачів та команди розробників. Важливо також враховувати підтримку та спільноту, які можуть бути корисні при вирішенні проблем, які виникають під час розробки.

3.2.1. *Angular.js*

AngularJS – це фреймворк JavaScript з відкритим вихідним кодом, призначений для створення односторінкових додатків, що складаються з HTML, CSS і JavaScript. Він використовує шаблон архітектури MVC, що дозволяє розділяти логіку, представлення і дані програми, щоб створювати веб-додатки з однією сторінкою [11].

AngularJS має велику кількість готових рішень, які дозволяють вирішувати різноманітні задачі, використовуючи готові модулі. Частини

програми розміщуються в модулях Angular.js, що дозволяє легко керувати ними. Така розбивка на модулі дозволяє завантажувати тільки потрібні служби і ефективно виконувати автоматичне тестування.

Однак, AngularJS має деякі недоліки. Наприклад, він застосовує двостороннє зв'язування даних, що змушує слухачів реагувати на будь-які зміни в інтерфейсі, що може сповільнювати роботу. Також, різноманітність структур може ускладнювати вивчення AngularJS в порівнянні з React і Vue.js, які використовують виключно компоненти.

3.2.2. *React.js*

React – це відкрита JavaScript бібліотека, яка дозволяє розробляти високоякісний інтерфейс для веб-додатків з використанням компонентної архітектури [12].

Переваги:

- Простий дизайн та документація, що дозволяє швидко вивчити та розпочати роботу з фреймворком.
- Швидкий та ефективний завдяки використанню React Virtual DOM та оптимізацій рендеринга.
- Розширюваність та повторне використання компонентів з допомогою React компонентів, що дозволяє забезпечувати швидкість та стабільність великих додатків.
- Можливість завантажувати HTML на стороні сервера, що покращує швидкість завантаження сторінок з великою кількістю даних та дозволяє забезпечити підтримку SEO.
- Одностороння прив'язка даних, що спрощує розробку та уникнення багатьох проблем з плутаниною в даних.

Недоліки:

- Використання JSX може здатися незвичним для розробників без досвіду з React.

- Відсутність чіткого контролю над станом компонентів може призвести до проблем з управлінням станом за допомогою Redux або Flux.
- Значний обсяг коду може бути потрібний для розробки складних компонентів, що може призвести до складнощів з розумінням коду та утриманням додатку.

3.2.3. *Vue.js*

Vue.js – це відкритий фреймворк JavaScript з високою продуктивністю для розробки інтерфейсів користувача та односторінкових додатків [13].

Переваги:

- Легкий і простий у вивченні, має досконалу документацію та зрозумілу структуру.
- Має вбудовану підтримку шаблонів та компонентів, що сприяє швидкій розробці та створенню перевикористовуваних блоків коду.
- Реалізує двостороннє зв'язування даних, що дозволяє зручно та ефективно керувати станом програми.
- Можливість побудови віртуального DOM та асинхронного рендерингу, що дозволяє розробникам досягати високої продуктивності.
- Досить гнучкий фреймворк, який можна використовувати в різних сферах, включаючи розробку веб-додатків, мобільних додатків, додатків для стільних комп'ютерів та телевізорів.
- Має велику кількість різноманітних плагінів та доповнень, що дозволяє легко розширювати функціонал.
- Висока спільнота розробників та підтримка від компанії, що забезпечує постійне оновлення фреймворку та виправлення помилок.

Недоліки:

- Є проблеми з оптимізацією під час використання більш складних додатків.
- Можливі проблеми зі сумісністю з деякими сторонніми бібліотеками та плагінами.
- Збільшення обсягу проєкту зі зростанням кількості компонентів та модулів.
- Не включає підтримку середовища розробки, тому може бути складним у встановленні
- Vue.js молодший фреймворк порівняно з React та Angular, тому він може мати менший екосистему і менше документації.
- Підтримка Vue.js може бути менш ефективною, оскільки фреймворк не так популярний як React і Angular.
- Деякі функції Vue.js можуть бути менш гнучкими, ніж у React, зокрема при роботі зі структурами даних.
- Vue.js може бути менш придатним для великих проектів зі складними потребами, порівняно з Angular, через менший набір функцій та інструментів.

У цілому, Vue.js є привабливим вибором для розробників, які шукають простий, легкий та гнучкий фреймворк для створення веб-додатків. Однак, він може мати обмеження, якщо вам потрібні великі та складні проекти з великою кількістю даних та інтеграцією з багатьма іншими інструментами та бібліотеками.

3.2.4. *Висновки*

Аналізуючи три найпопулярніші фреймворки для розробки клієнтської частини додатків, можна зробити висновок, що Angular не надає необхідної гнучкості, тоді як Vue.js є ще досить молодим. Тому, було вирішено віддати перевагу React.js як ідеальному балансу між простотою

та швидкодією. Його простий дизайн, використання JSX для шаблонів та дуже добре документована система дозволяють легко вчитися. Крім того, React.js є дуже швидким фреймворком завдяки реалізації Virtual DOM та різним оптимізаціям рендерингу. Використання односторонньої прив'язки даних та підтримка функціонального програмування також є перевагами React.js. Однак, змішування шаблонів з логікою може бути не дуже інтуїтивним для розробників без попереднього досвіду з React.js.

3.3. GraphQL, як заміна REST

При використанні REST, якщо клієнт потребує отримати інформацію про користувачів з бази даних, він отримує всю доступну інформацію про кожного користувача, включаючи ім'я, прізвище, електронну пошту та телефон. Це може бути надмірним для потреб клієнта, особливо якщо він потребує лише ім'я та прізвище. З GraphQL, клієнт може вибрати, що саме йому потрібно, тобто він може запросити лише ім'я та прізвище, і отримає тільки цю інформацію. Це дозволяє скоротити час на передачу даних та зменшити навантаження на сервер [14].

Однак, як і у будь-якій технології, GraphQL також має свої недоліки. Наприклад, погано спроектований GraphQL API може призвести до проблем, таких як зменшення продуктивності та збільшення часу відповіді. Крім того, GraphQL може бути складним для вивчення та розуміння, особливо для початківців. Також GraphQL не підходить для всіх випадків використання, іноді REST може бути більш придатним варіантом для певних проектів.

Отже, підсумовуючи, GraphQL є молодою, проте потужною технологією, яка вирішує деякі з проблем REST архітектури. Однак, вибір між GraphQL та REST залежить від потреб проекту та умінь розробника.

```
query Audios($limit: Int!, $cursor: String) {
  audios(cursor: $cursor, limit: $limit) {
    hasmore
    audios {
      id
      name
      accent
      data
      createdAt
      creator {
        id
        username
        email
        createdAt
      }
    }
  }
}
```

Дійсно, GraphQL має декілька переваг порівняно з REST. Окрім переваг, які були вже названі, слід зазначити також наступні:

- Запити до сервера в GraphQL можуть бути складнішими і більш точними, ніж в REST, оскільки GraphQL дозволяє вбудовувати запити один в одного і створювати більш складні запити до бази даних;
- Можливість розробки автономних клієнтів. Оскільки клієнт може запитувати саме ті дані, які йому потрібні, він може бути автономним і не залежити від сервера. Крім того, GraphQL дозволяє клієнтам отримувати лише необхідну кількість даних, що може покращити продуктивність;

- GraphQL працює добре з неструктурованими та змінюваними даними, тоді як REST підходить тільки для структурованих даних.

Однак, слід зазначити, що GraphQL не є універсальним рішенням для всіх випадків. Наприклад, якщо API має статичну структуру, REST може бути більш ефективним вибором. Крім того, GraphQL має свої недоліки, зокрема, він може стати складним у використанні, якщо необхідно створити складні запити до бази даних. Тому перед вибором між GraphQL та REST слід ретельно проаналізувати вимоги до API і визначити, яке рішення підходить найкраще в конкретному випадку.

3.4. Вибір інструментів для розроблення модулю підсистеми розпізнавання англомовних акцентів

Обидві мови програмування – JavaScript та Python відомі своєю широкою популярністю в галузі машинного навчання та глибокого навчання, тому вибір обох мов є розумним. JavaScript використовується для фронтенду та має велику кількість бібліотек для візуалізації даних, тоді як Python використовується для бекенду та має багато популярних бібліотек, таких як TensorFlow, Keras, PyTorch та інші, які підтримують глибоке навчання. Python також має великий екосистему бібліотек та інструментів для роботи з даними, включаючи бібліотеки для аналізу даних, обробки текстів та зображень.

Також важливим критерієм є легкість в опануванні необхідних бібліотек, і в цьому відношенні Python може мати перевагу, оскільки багато бібліотек машинного навчання та глибокого навчання були розроблені саме для Python та мають документацію та приклади використання на цій мові.

Отже, використання JavaScript та Python у спільній системі може забезпечити гнучкість та ефективність в роботі з великими обсягами даних та реалізації алгоритмів глибокого навчання.

3.4.1. *Python*

Дійсно, Python є дуже популярною мовою програмування для машинного навчання та аналізу даних завдяки великій кількості бібліотек та інструментів, доступних для розробників. Бібліотека pandas дозволяє легко і ефективно працювати з великими наборами даних, а numpy надає можливість операцій над матрицями, що є необхідним для багатьох алгоритмів машинного навчання. Бібліотека matplotlib дозволяє візуалізувати дані у вигляді графіків, що полегшує їх аналіз та інтерпретацію.

Бібліотека librosa є корисною для обробки аудіоданих, наприклад, для екстракції акустичних особливостей, які можуть використовуватися у задачах розпізнавання голосу та інших завданнях машинного навчання.

Інтерактивний режим Python дозволяє швидко виконувати експерименти, перевіряти різні алгоритми та моделі та вивчати мову програмування, що є дуже корисним для новачків.

Однак, також варто згадати, що JavaScript є популярною мовою програмування для веб-розробки та може бути використана для написання складних веб-додатків, включаючи додатки, що використовують машинне навчання. Також існують спеціалізовані бібліотеки для машинного навчання, такі як TensorFlow.js та Brain.js, що дозволяють використовувати машинне навчання прямо в браузері.

3.4.2. *JavaScript*

Дійсно, JavaScript має свої аналоги бібліотек для машинного навчання, такі як TensorFlow.js та Brain.js, але на відміну від Python, JavaScript більше використовується для розробки веб-додатків та взаємодії з користувачами в браузері. І хоча JavaScript може бути використаний для розробки сторонніх програм на сервері, Python має більше інструментів, що дозволяють зручно працювати з машинним навчанням та аналізом даних. Інтерактивний режим Python також дає можливість швидко

перевіряти результати і досліджувати дані в реальному часі, що зручно для розв'язання невеликих задач та навчання.

3.4.3. *Висновки*

Вважаючи наявність досвіду роботи з бібліотеками для Python перевагу було надано саме цій мові програмування.

3.5. Вибір СКБД

Для зберігання інформації про користувачів та їх дій, а також даних про аудіофайли, можна використовувати реляційні бази даних, такі як MySQL, PostgreSQL або SQLite. Ці бази даних дозволяють легко зберігати та організовувати структуровані дані, а також забезпечують можливість швидкого пошуку та фільтрації інформації.

Однак, якщо дані, з якими користувачі працюють, є великими аудіофайлами, то можна використовувати такі бази даних, як MongoDB або Cassandra, що базуються на нереляційних моделях даних. Вони дозволяють ефективно зберігати та опрацьовувати великі об'єми даних, такі як аудіофайли, і забезпечують високу продуктивність при операціях з даними.

Вибір бази даних залежить від різноманітних факторів, таких як обсяг даних, вимоги до продуктивності, тип даних та інші фактори. Отже, важливо детально вивчити всі можливі варіанти баз даних та їх можливості, щоб вибрати оптимальний варіант для даного проекту.

3.5.1. *Вибір моделі баз даних*

Бази даних можуть бути класифіковані за багатьма ознаками, але загалом їх можна поділити на дві основні групи в залежності від способу представлення та зберігання даних. Реляційні бази даних (SQL) примушують користувачів зберігати дані в явних типах та у вигляді таблиць, тоді як неструктуровані бази даних (NoSQL) не вимагають жорсткої структури зберігання даних у вигляді таблиць, але дозволяють

зберігати дані у вигляді документів (MongoDB), построково (Cassandra) та інших форматах.

До переваг реляційних баз даних типу SQL можна віднести високу швидкість отримання великої кількості записів з бази даних за допомогою запитів SQL, чітко визначені стандарти та можливість використання інтерактивної мови.

Нереляційні бази даних (NoSQL) – це інша категорія БД, яка відрізняється від реляційних БД тим, що не використовує схему з таблицями і стовпцями. Вони можуть бути збережені як документи, ключі-значення, графи тощо, залежно від конкретної системи управління базами даних. Ці БД зазвичай використовуються для зберігання великих обсягів даних, що не мають чіткої структури, таких як файли журналів, соціальні мережі тощо.

Основні переваги NoSQL БД включають в себе:

- Інтеграція з великими обсягами нереляційних даних;
- Горизонтальне масштабування даних з великою кількістю серверів;
- Більш прості засоби зберігання і забезпечення доступу до даних;
- Відсутність необхідності відображати дані у таблицях.

Незважаючи на це, NoSQL БД мають свої недоліки. Іноді вони можуть бути менш надійними і менш безпечними, так як вони не мають традиційної схеми, яка допомагає забезпечити консистентність даних. Також, незважаючи на те, що вони зазвичай забезпечують великі обсяги даних і горизонтальне масштабування, вони можуть бути менш ефективними в обробці запитів на вибірку даних.

У кінцевому підсумку, вибір між реляційними і нереляційними БД залежить від конкретних потреб користувача. Якщо потрібна висока швидкість запитів на вибірку даних, реляційні БД, такі як MySQL, можуть

бути кращим варіантом. Але, якщо обсяги даних великі і не мають чіткої структури.

3.5.2. *Вибір ядра SQL БД*

При розгляданні використання реляційних баз даних, SQL є одним з найпоширеніших інтерфейсів для роботи з ними. Ця мова запитів дозволяє створювати, редагувати, видаляти та вибирати дані з бази даних. Вибір конкретного представника SQL баз даних залежить від вашої конкретної ситуації і потреб. Наприклад:

- простота використання;
- наявність індексації даних;
- безкоштовність.

3.5.3. *SQLite*

SQLite – це реляційна система управління базами даних (RDBMS) типу SQL, яка дозволяє зберігати дані у вигляді файлу на локальному пристрої. Вона є одним з найбільш поширених та відомих рішень для використання в різноманітних проектах, зокрема веб-сайтах, мобільних додатках та десктопних програмах [15].

Однією з головних переваг SQLite є її легкість використання та простота встановлення. SQLite не вимагає складних конфігурацій чи налаштувань, що робить її досить привабливою для новачків в області розробки. Вона також підтримує більшість стандартних функцій SQL, таких як запити SELECT, INSERT, UPDATE та DELETE, тому розробникам легше працювати з SQLite, особливо якщо вони вже мають досвід роботи з SQL.

До інших переваг SQLite можна віднести:

- Швидкість та ефективність: SQLite працює досить швидко і може легко обробляти великі обсяги даних, що робить її відмінним варіантом для проектів з великою кількістю транзакцій.
- Невисока вартість: SQLite безкоштовна, вона доступна на різних платформах і не вимагає значних витрат на розгортання.

Недоліки SQLite полягають у тому, що вона не є найкращим вибором для проектів, де потрібна висока масштабованість і обробка великих обсягів даних, так як вона не підтримує розподілені обчислення та шаринг даних. Також, SQLite може бути менш ефективною у проектах, де часто відбувається одночасний доступ до бази даних, так як вона має деякі обмеження на кількість одночасних підключень.

Отже, SQLite – це надійний і простий у використанні інструмент.

3.5.4. *PostgreSQL*

PostgreSQL – це одна з найпотужніших та надійних відкритих SQL-систем управління базами даних, яка підтримує багато функцій, що відсутні у більшості інших систем. Вона має відкритий код, що дає можливість користувачам вносити внески до розвитку цієї БД, а також забезпечує безкоштовний доступ до програмного забезпечення [16].

До переваг PostgreSQL можна віднести:

- Надійність та безпека. PostgreSQL має потужну систему безпеки, яка забезпечує захист баз даних від несанкціонованого доступу, а також дозволяє відновлювати дані в разі аварії.
- Підтримка транзакцій. PostgreSQL має підтримку транзакцій, що дозволяє користувачам гарантувати цілісність даних у випадку збоїв системи.
- Масштабованість. PostgreSQL може працювати з великою кількістю даних і користувачів, що робить її ідеальним варіантом для великих проектів.

- Розширюваність. PostgreSQL має велику кількість додаткових модулів та розширень, що дозволяють розширювати її функціональність та пристосовувати до конкретних потреб користувача.

Недоліками PostgreSQL є:

- Складність. PostgreSQL може бути складною для використання для користувачів з обмеженим досвідом роботи з СУБД.
- Високі вимоги до апаратного забезпечення. PostgreSQL вимагає відносно потужного апаратного забезпечення для ефективної роботи з великою кількістю даних.
- Обмежена підтримка розподіленої обробки даних. PostgreSQL може працювати тільки на одній машині.

Загалом, PostgreSQL є потужним та надійним рішенням для зберігання та опрацювання даних, яке має великий спектр функцій та можливостей, а також відкритий вихідний код. Однак, перед використанням PostgreSQL, слід враховувати його складну настройку та обмежені можливості масштабування.

3.5.5. *Висновки*

Після розгляду переваг та недоліків вище згаданих баз даних, було вирішено обрати PostgreSQL. Це пов'язано з її більш гнучким повнотекстовим пошуком та тим, що PostgreSQL не потребує завантаження стороннього програмного забезпечення, завдяки своїй клієнт-серверній архітектурі.

Також слід зазначити, що для розробки веб-додатку потрібна база даних для зберігання значень для валідації користувачів. Для цієї мети підходить база даних для зберігання невеликих неструктурованих даних у вигляді пари ключ-значення в пам'яті, наприклад, Redis, який буде використовуватися в проєкті.

3.6. Архітектура розробленого програмного забезпечення

Буде створено систему для оцінювання акценту англомовних користувачів з використанням алгоритмів машинного навчання для визначення одного з восьми найбільш розповсюджених акцентів англійської мови з використанням аудіосегментів, які користувач записує.

Додаток також забезпечує функції для відстеження прогресу користувачів у покращенні їхньої вимови для кожного конкретного акценту. Крім цього, користувачі можуть зареєструватися в системі як репетитори, отримувати заявки від студентів та коментувати їх записи. Графічне представлення аудіозаписів дозволяє користувачам переслухати свою вимову повністю та отримати її візуалізацію. Система також надає докладний опис специфіки кожного акценту та його відмінностей від інших.

3.6.1. *Формулювання вимог до програмного забезпечення*

Вимоги до програмного забезпечення – це набір потреб потенційних користувачів щодо властивостей, якості та функцій програмного продукту, який потрібно розробити або модифікувати [18]. Виділяють наступні види вимог.

Функціональні вимоги:

- опис вимог замовника та його бачення програмного забезпечення без втручання в реалізацію;
- опис вимог користувача, під які розроблюється ПЗ.

Нефункціональні вимоги:

- сумісність із зовнішніми системами незалежно від екосистеми ПЗ;
- опис особливостей продукту, які характеризують його якість;
- границі можливостей системи.

Перші характеризують конкретний результат системи та компонентів, які показують, наскільки точно програмне забезпечення відповідає вимогам замовника.

Нефункціональні вимоги, натомість, задають поведінку системи або її компонентів, за якою можна зробити висновок про рівень надійності розробленого ПЗ.

Беручи до уваги вище перераховані цілі та результати системи, можна описати перелік вимог до неї. Виконання описаних нижче вимог в застосунку для оцінювання акценту англомовних користувачів призведе до остаточного результату розробки та вирішення описаної проблематики проєкту.

Таблиця 3.1

Вимоги до продуктивності

Код вимоги	Опис вимоги
PER -1	Вебсторінка має завантажуватися менше ніж за 3 секунди після завантаження аудіофайлу в браузер користувача.
PER-2	Система має відповідати на запити менш ніж за 1 секунду
PER-3	Система має коректно працювати при навантажені в 1000 запитів на секунду.

Таблиця 3.2

Вимоги до безпеки

Код вимоги	Опис вимоги
SEC-1	Система має бути захищена від поширених атак типу XSS, CSRF, SQL Injection
SEC-2	Система має зберігати паролі користувачів у закодованому вигляді, який не піддається зворотній трансформації

Таблиця 3.3

Вимоги до реалізації

Код вимоги	Опис вимоги
IMP-1	Серверна частина веб-додатку має бути побудована на основі Apollo Sever для реалізації GraphQL API
IMP-2	Збереження даних аудіозаписів користувачів має виконуватися за допомогою бази даних PostgreSQL.
IMP-3	Вебдодаток має коректно працювати в оточенні сучасних браузерів.

Для формування функціональних вимог, необхідно спершу описати ключові прецеденти користування вебдодатком користувачами за допомогою діаграм.

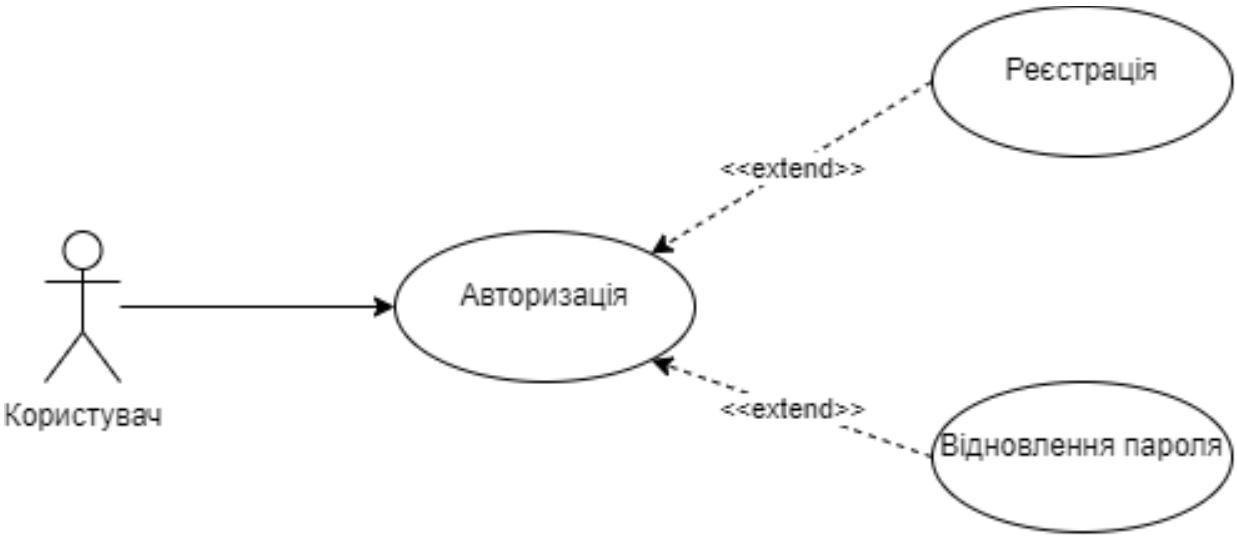


Рис. 3.1. Прецедент: Авторизація або реєстрація нового користувача

Прецедент 1: «Авторизація або реєстрація нового користувача» (рис. 3.4).

Користувач на сторінці із вебдодатком.

Користувач на сторінці авторизації, або, якщо не зареєстрований в системі, на сторінці реєстрації.

Користувач вводить дані.

Користувачу надається можливість відтворити свій пароль через електрону пошту..

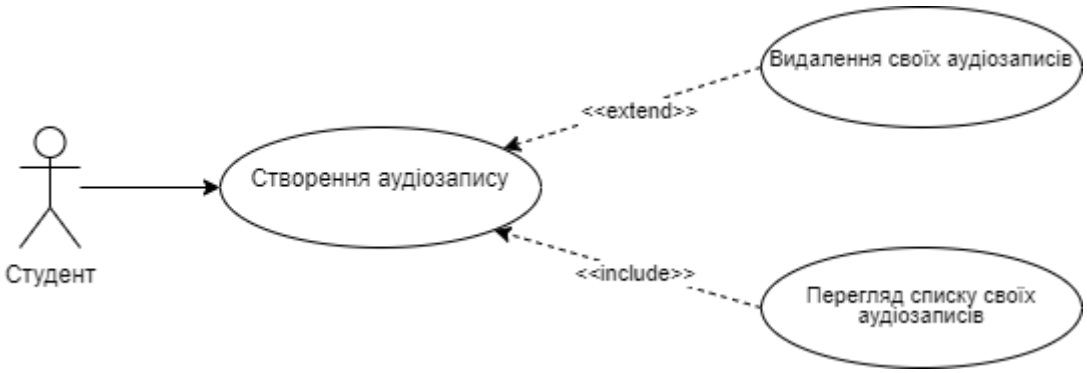


Рис. 3.2. Створення нового запису студентом

Прецедент 2: «Створення нового аудіозапису студентом».

Передумова: студент авторизований.

1. Студент на сторінці створення аудіозапису.
2. Студент натискає на кнопку початку запису.
3. Студент мовить у диктофон декілька слів.
4. Студент зупиняє аудіо за допомогою кнопки зупинки.
5. Студент може прослухати отриманий фрагмент, та зберегти, або видалити запис.

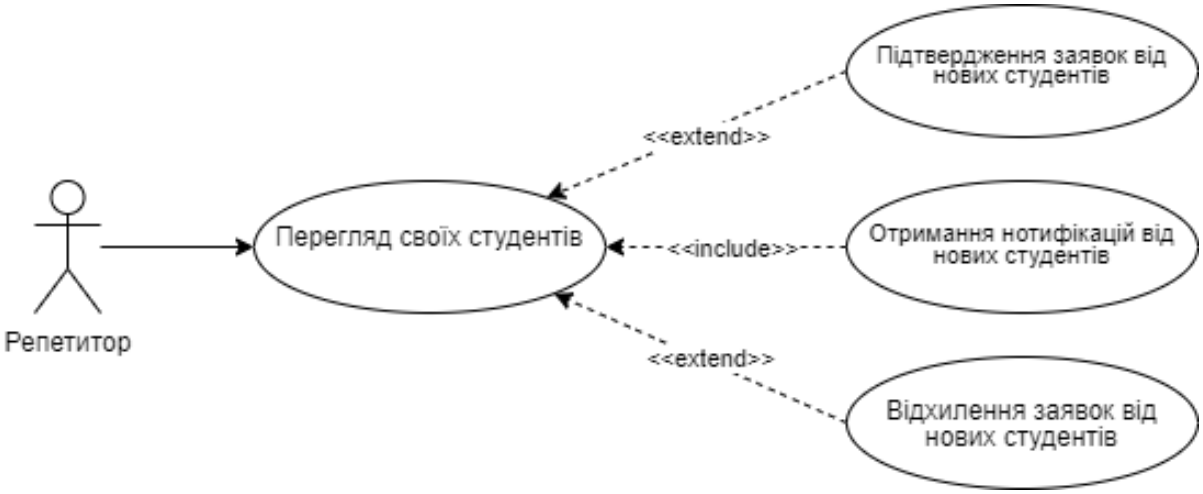


Рис. 3.3. Перегляд репетитором студентів

Прецедент 3: «Перегляд студентів репетитором» (рис. 3.6).

Передумова: репетитор авторизований.

1. Репетитор на сторінці перегляду своїх студентів.
2. Репетитор отримує нові повідомлення про запит від студента.
3. Репетитор затверджує або відхиляє запит студента.

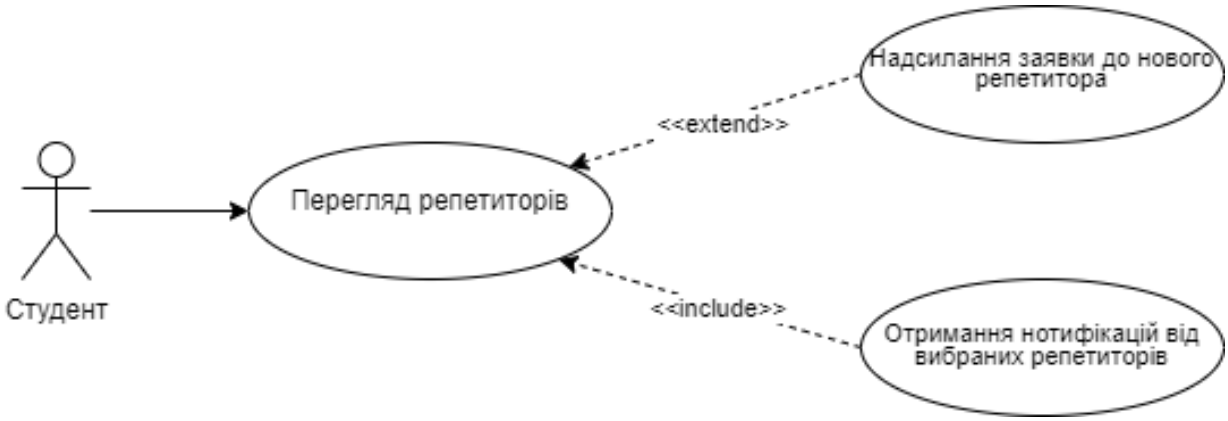


Рис. 3.4. Перегляд студентом репетиторів

Прецедент 4: «Перегляд студентом репетиторів» (рис. 3.7).

Передумова: студент авторизований, репетитор зареєстрований.

1. Студент на сторінці перегляду репетиторів
2. Студент вибирає одного, і надсилає запит на отримання затвердження від репетитора.
3. Студент отримує повідомлення про затвердження/відхилення на сайті, або в телеграмі (якщо налаштував).

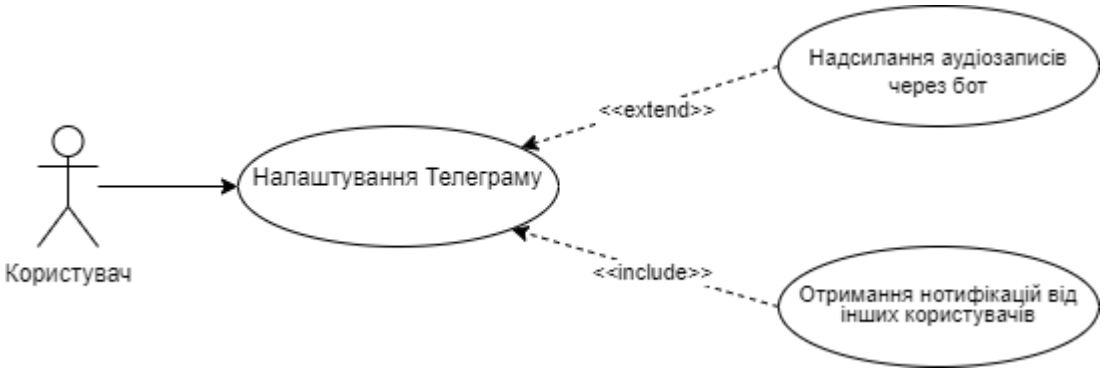


Рис. 3.5. Налаштування Телеграму користувачем

Прецедент 5: «Налаштування користувачем Телеграму» (рис. 3.8).

Передумова: користувач авторизований.

1. Користувач на сторінці налаштування Телеграму.
2. Користувач за посиланням переходить до чату із ботом.
3. Користувач починає роботу із ботом за допомогою ключового слова “/start”.
4. Відтепер, якщо користувач – це студент, він має можливість записувати аудіо, та отримувати його оцінку в Телеграмі.

Прецеденти дозволяють описати функціональні вимоги (табл. 3.4).

Таблиця 3.4

Функціональні вимоги до вебдодатку

Код вимоги	Зміст вимоги	Атрибути	
		Пріоритет	Складність
F1	Реєстрація користувачів за допомогою електронної адреси, нікнейму та паролю	Високий	Середня
F2	Відновлення пароля за допомогою листа на електрону адресу користувача	Середня	Середня
F3	Авторизація користувачів за допомогою електронної адреси або нікнейму та паролю	Високий	Середня
F4	Перегляд існуючих аудіозаписів авторизованим користувачем	Високий	Низька

Продовження табл. 3.4

F5	Налаштування телеграм-боту користувачем для отримання повідомлень	Середній	Низький
F6	Перегляд оцінки репетитора на аудіо студентом	Середній	Низька
F7	Перегляд інформації про специфіку вимови кожного акценту користувачем	Низький	Низька
F8	Запис студента до репетитора	Середній	Середня
F9	Затвердження студента репетитором	Середній	Середня
F10	Перегляд репетитором списку записів своїх студентів	Висока	Середня
F11	Додавання оцінки до аудіозапису репетитором	Низький	Середня

3.6.2. *Структурна організація програмного забезпечення*

Оскільки програмним забезпеченням даної дисертації виступає система для оцінювання англомовних акцентів користувача за допомогою алгоритмів машинного навчання, то в основі цієї системи має знаходитися саме модуль, що аналізує аудіозаписи та оцінює за ними акцент, а обгорнутий цей модуль у вебдодаток, який побудований на основі клієнт-

серверної архітектури. Таким чином можна розбити систему на три модуля:

- нейронна мережа;
- клієнт;
- сервер.

Клієнтська частина додатку

В другому розділі дисертації було вибрано фреймворк для написання клієнтської частини додатку: React у зв'язці із легковажним фреймворком Next.js для покращеної роботи із навігацією в додатку. Інтеграція Next.js дозволяє інтуїтивно створювати нові сторінки по принципу, що створений файл із назвою *filename* у папці */pages/filename* буде відображати компонент, який знаходиться в цьому файлі за посиланням */filename*. React, натомість, дозволяє розділити код за компонентами, що потім використовуються сторінками, а вбудована в React технологія JSX, що є синтаксичним розширенням над Javascript [18], дозволяє оперувати HTML елементами використовуючи повну потужність Javascript. В дисертації натомість використовується мова Typescript, яка наслідує цю особливість у вигляді TSX, але додає усю потужність статичної типізації.

Для того, щоб полегшити розробку UI та не обмежуватися стандартними компонентами, було інтегровано бібліотека Chakra UI. Оскільки в системі пропонується реєстрація, авторизація, а також створення нових аудіозаписів, для полегшеної роботи із формами завантаження на сервер, в проєкт було інтегровано бібліотеку для роботи з формами в React – Formik.

Формування запитів до сервера відбувалося за допомогою GraphQL клієнту urql. GraphQL запити можна розділити на два типи: запит (query) та зміна (mutation). Для кожного запиту було створено відповідний .graphql файл, який містив код запиту написаний мовою GraphQL. За допомогою команди генерування Typescript коду

yarn graphql-codegen --config ./path/to/config.yml [19]

а також файлу конфігурації, який описує за яким посиланням знаходиться GraphQL схема (зазвичай це посилання на сервер) та які файли необхідно використовувати в генерації (всі файли із розширенням .graphql), GraphQL код кожного файлу, а також відповідні типи даних, які в них використовувалися, генеруються в загальний файл. Таким чином, згенеровані функції запитів, можна використовувати у вигляді іменних хуків (React Hooks) для кожного запиту та імпортувати їх будь-де на клієнті. urql за замовчуванням не підтримує нормалізоване кешування, але надає пакет Graphcache для роботи з кешем. Ця технологія дозволяє керувати кешем додатку та відповідно оновлювати його в разі змін, таким чином, зменшуючи кількість оновлень сторінки.

Таким чином клієнтська частина займається відображенням UI, компоненти якого динамічно оновлюються за необхідністю, а також формуванням запитів до серверу та обробкою відповідей від нього.

Серверна частина додатку

Для побудови серверної частини додатку було обрано фреймворк для побудови комплексних додатків Node.js. Бібліотека express та ApoloServer дозволили створити звичайний GraphQL сервер. Використання ApolloServer вмотивовано його підтримкою GraphQL клієнту.

Для поєднання серверу із відповідною БД було використано ORM typeorm, в основі налаштувань якої лежить підключення до PostgreSQL. Оскільки в додатку зумовлено використання сесій, було використано відповідну бібліотеку для створення сесій express-session. Сесія була зв’язана із сховищем RedisStore на основі бази даних Redis.

Для безпечного зберігання персональних даних користувача необхідно зберігати їх в БД у хешованому вигляді, тому використано бібліотеку argon2, що дозволяє хешувати дані та перевіряти їх валідність без отримання початкового значення.

Серверна частина оперує усією логікою взаємодії із БД та займається підготовкою даних до відправлення клієнту.

Нейронна мережа

Даний компонент програми не є інтегрованим із клієнт-серверною архітектурою, а виступає у якості автономної програми, ціллю якої є тренування ваг для використання їх в аналізі аудіозаписів на серверній частині. Програма для нейронної мережі написана за допомогою мови програмування Python, а також використовує допоміжні бібліотеки для роботи із великою кількістю комплексних математичних даних.

Вище згадані модулі формують архітектуру системи наступним чином (рис. 3.9).

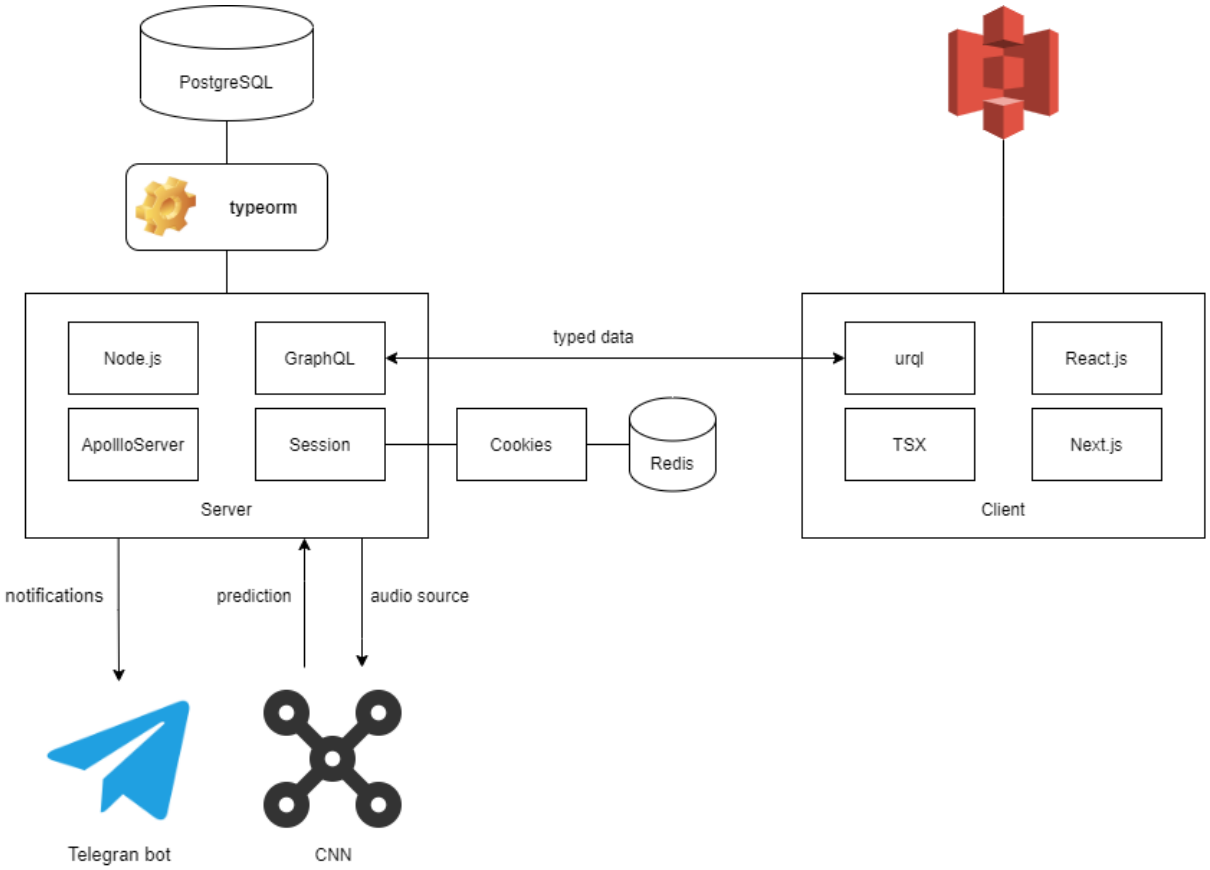


Рис. 3.6. Архітектура системи

Структура БД

Опираючись на вище згадані функціональні вимоги, визначено сутності для зберігання у БД системи.

1. Користувач (User):
    - Ім’я користувача (Username);
    - Електронна пошта (Email);
    - Пароль (Password);
    - Роль (Role) – користувач може зареєструватися в системі як студент або репетитор;
    - Репутація (Reputation) – показник активності користувача в системі;
    - Підтверджено (Approved) – флаг для отримання студентом підтверджень від репетитора.

2. Аудіозапис (Audio):
    - Назва (Name);
    - Посилання (Link) – посилання на аудіозапис;
    - Акцент (Accent);
    - Ідентифікатор цілі (goalId) – зовнішній ключ, що посилається на ціль;
3. Акцент (Accent):
    - Назва (Name);
    - Процент (Percentage) – відсоток в аудіо;
    - Ідентифікатор аудіо (audioId) – зовнішній ключ, що посилається на аудіозапис;
4. Ціль (Goal):
    - Акцент (Accent);
    - Опис (Description) – відсоток в аудіо;

- Ідентифікатор автора (creatorId) – зовнішній ключ, що посилається на автора цілі;

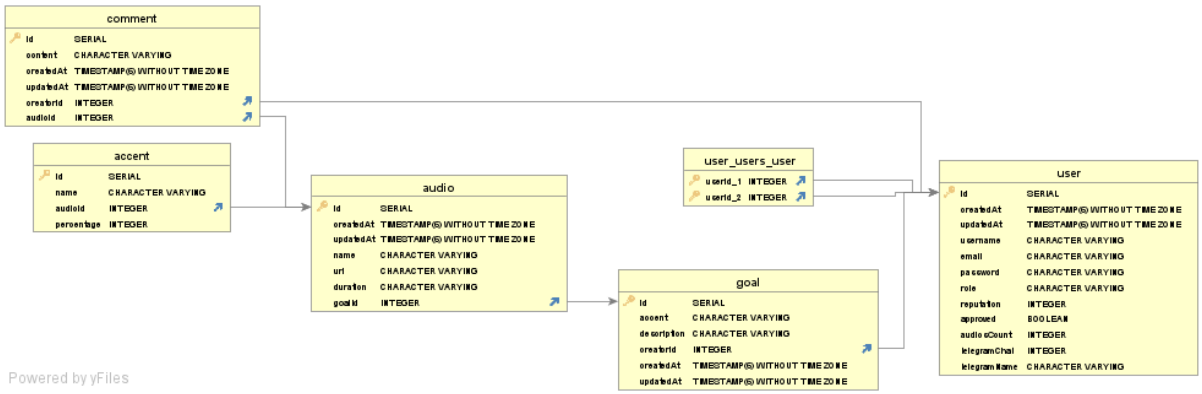


Рис. 3.7. Структура бази даних

4. РЕЗУЛЬТАТИ ТА ЇХ АНАЛІЗ

Задля найкращої репрезентації роботи розробленого методу на реалізованій системі для оцінки англомовних акцентів було почергово реалізовано кожний метод видобування ознак із аудіо файлів у вигляді функцій, по кожній для MFFC, LPC, PLP та DWT відповідно. Наступним наводяться результати тренування однієї нейронної мережі і на одному наборі даних для кожного методу окремо задля отримання об’єктивних відсотків точностей передбачень. Для цього необхідно описати процес класифікації вже проаналізованих даних, а саме за допомогою згорткової нейронної мережі (CNN).

4.1. CNN

Згорткова нейронна мережа (Convolutional Neural Network або CNN) – це тип нейронної мережі, що використовує математичну операцію згортки для аналізу візуальних зображень. Оскільки дані у вигляді MFCC (коефіцієнти мел-частотної кепстральної перетворення) можуть бути представлені як зображення, їх також можна використовувати для аналізу через CNN. Операція згортки є легкою для реалізації та інтуїтивно зрозумілою. Вона використовує матрицю фільтрів, яка проходиться по вхідному зображенню з певним кроком, для знаходження закономірностей та граней в зображенні. Після знаходження особливостей фільтр оцінюється через операцію згортки, яка є поелементним добутком та сумою між двома матрицями (рис. 4.4) [21].

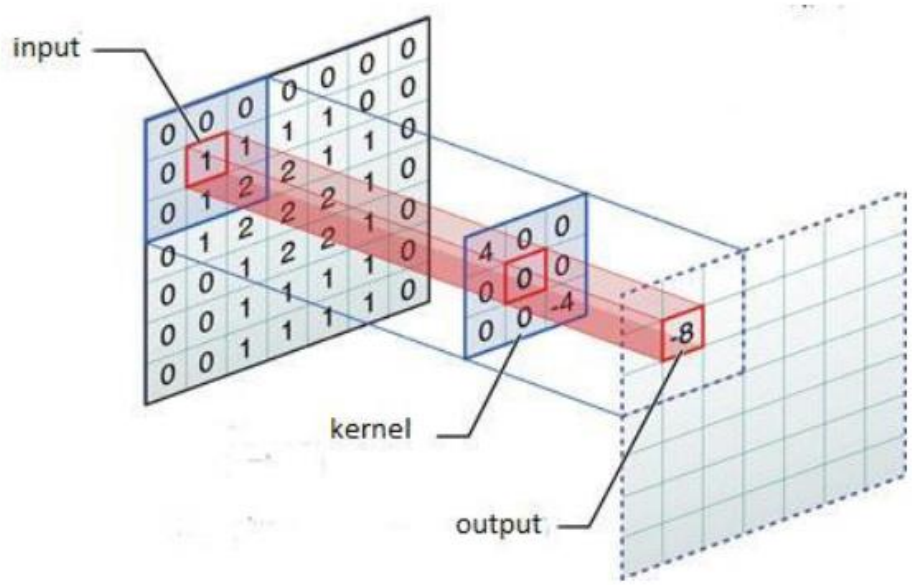


Рис. 4.1. Операція згортки

Операція згортки між вхідним зображенням та фільтром повертає значення, яке вказує на наявність чи відсутність певної особливості. Якщо особливість присутня, значення буде великим, в іншому випадку – мале або навіть 0.

Фільтр повинен бути підданий нелінійному відображенню, щоб мережа могла вивчити значення, які потім використовуватимуться фільтром для знаходження особливостей. Значення з операції згортки підсумовуються з упередженістю, яка відповідає константі у лінійній функції, і далі проходять через нелінійну функцію активації.

У найновіших нейронних мережах з використанням загальних архітектур, таких як CNN, застосовується функція активації ReLU (Rectified Linear Unit, або ReLU), яка повертає лише позитивні значення на виході та 0 в разі негативних значень [23].

$$\mathrm{ReLU}(x) = \max(0.0, x). \tag{4.4}$$

Наступним у нейронній мережі процес зменшення вибірки (downsampling) є необхідною операцією. Це зумовлено тим, що при операції згортки на декількох шарах вихідне зображення значно зменшується порівняно з вхідним, але при великих розмірах даних це може

призвести до витрати значної кількості часових ресурсів. Функція максимізаційного агрегування (max pooling) використовується для зменшення споживання мережею пам'яті та прискорення роботи нейронної мережі. Однією з переваг цієї операції є те, що вона змушує мережу зосередитися на декількох нейронах, замість усіх, що зменшує ризик перенавчання та поліпшує узагальнення.

У функції максимізаційного агрегування вікно з фіксованим розміром проходить по матриці вхідного зображення з певним кроком. На кожному кроці в вихідній матриці залишається максимальне значення серед тих, що знаходяться в межах даного вікна. Цей процес дозволяє ефективно зменшити розмір матриці та зберегти найважливішу інформацію.

$$n_{out} = \left\lfloor \frac{n_{in} - f}{s} \right\rfloor + 1. \tag{4.5}$$

де f – розмір вікна, n_{in} – розмірність вхідного зображення, s – крок.

Після проходження кількох шарів, що включають операції згортки та зменшення вибірки, вихідне зображення необхідно перетворити у вектор-особливість, який потім використовується в повнозв’язних шарах мережі. Це досягається шляхом сплющення вхідного зображення $n \times m$ у вектор розміром $n*m \times 1$. Потім вектор проходить через декілька щільних шарів мережі, на кожному з яких він множиться на ваги цього шару, підсумовується з упередженням та проходить нелінійну функцію активації. Для задачі класифікації вихідний шар мережі має надавати вірогідності для кожного класу, тому останній щільний шар має таку ж кількість нейронів, скільки класів у задачі (в даному випадку - 8 акцентів, отже 8 класів). Вихідний результат цього шару проходить функцію активації Softmax, яка нормалізує вихідний вектор і дає вірогідності для кожного класу [24].

$$\mathrm{f}(x_j) = \frac{e^{x_j}}{\sum_i e^{x_j}}, \tag{4.6}$$

де x – кожен елемент виходу останнього шару.

Для визначення точності передбачень нейронної мережі можна використати функцію втрат (функцію помилки). Зазвичай для багатокласових передбачень використовують категоричну функцію перехресної ентропії (Categorical Cross-Entropy Loss function), яка обчислюється за допомогою формули:

$$H(y, \hat{y}) = \sum_i y_i \log \hat{y}_i, \tag{4.7}$$

де $\hat{y}$ – передбачення, y – справжній результат.

Було вирішено використовувати згорткову нейронну мережу, оскільки її реалізація не є складною, а також вона демонструє хороші результати при роботі з зображеннями.

4.2. Порівняння результатів методів

У процесі навчання побудованої нейронної мережі використовувалось 80% від загального тензору розміром (11633, 50, 87). Швидкість навчання складала 0.01, а розмір партії становив 32 штуки. Навчання проводилося протягом двох епох, загальний час навчання становив 6 годин 22 хвилини 24 секунди. (рис. 4.2).

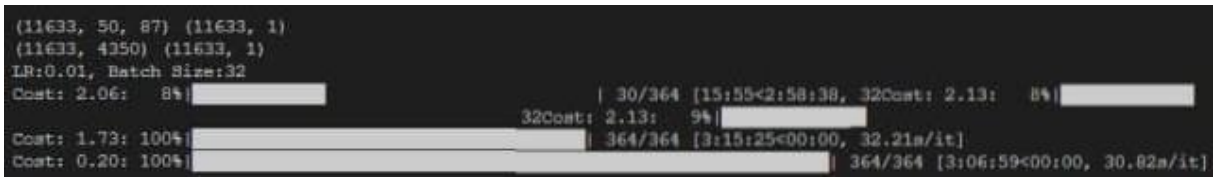


Рис. 4.2. Результат тренування мережі

Нейронна мережа змогла досягнути точності передбачень методом MFCC на рівні 89.07%, під час прогону тестових даних розміром (2909, 50, 87) з вилученою міткою акценту (рис. 4.3).

Рис. 4.3. Результат точності передбачень

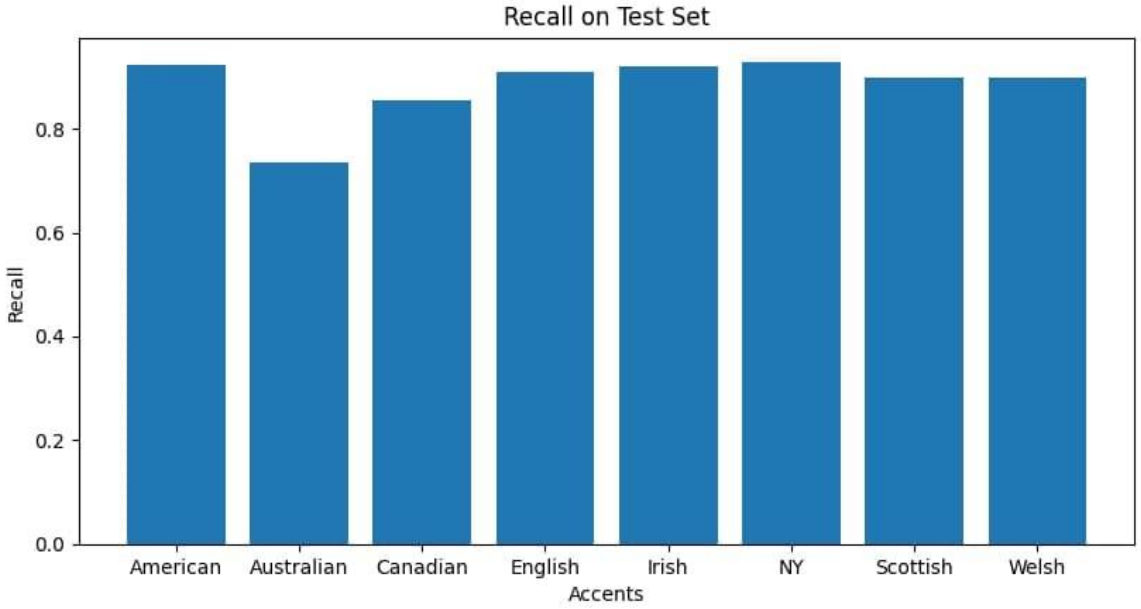


Рис. 4.4. Результат точності передбачень за акцентом (MFCC)

Для отримання найбільш точних даних при оцінці кращого із методів, було вирішено використати тензор відповідної до MFCC величини. Отже, для подальших методів був використаний тензор розміром (11633, 50, 87). Результати навчання мережі на різних методах можна спостерігати на рис. 4.5 – 4.8. Точність передбачень у методів PLP, LPC, DWT становить 84.52%, 81.12% та 88.38% відповідно. Натомість, результат комбінованого методу становить 94,28% точності передбачень і тим самим він перевищив передбачення кожного із методів. Отже було досягнуто мети роботи, так як покращення передбачень в середньому по акцентах становить приблизно 5% в порівнянні із найкращим із методів – MFCC.

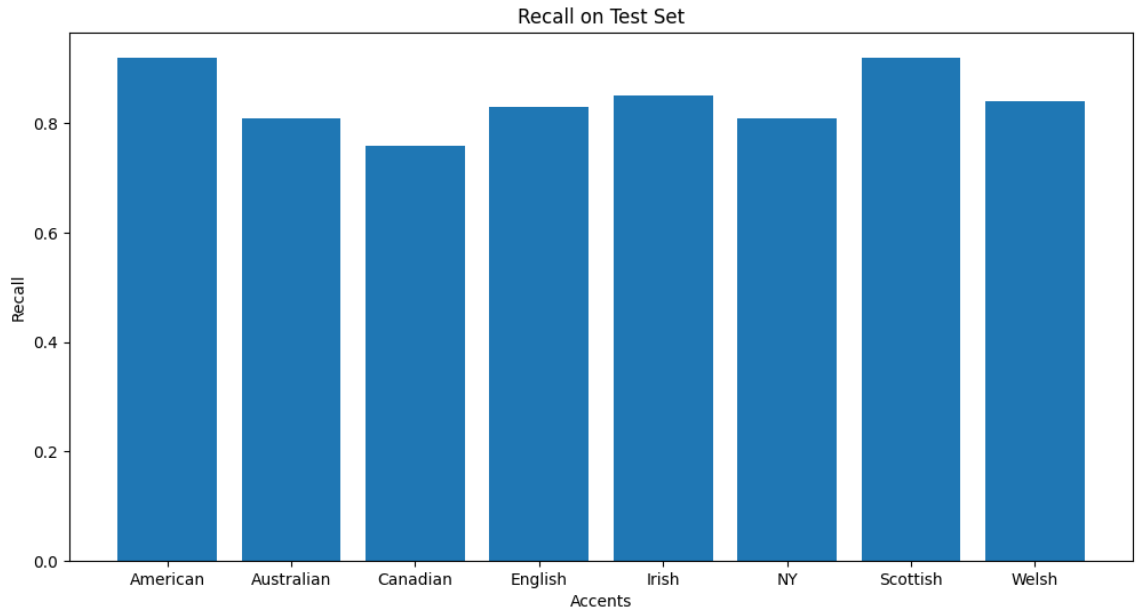


Рис. 4.5. Результат точності передбачень за акцентом методу PLP

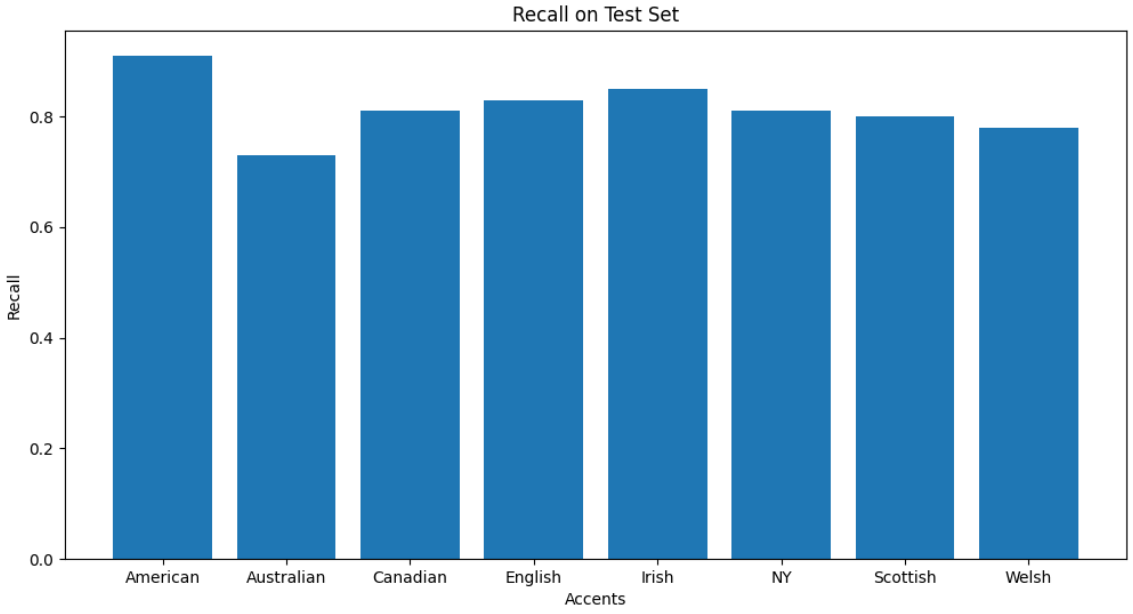


Рис. 4.6. Результат точності передбачень за акцентом методу LPC

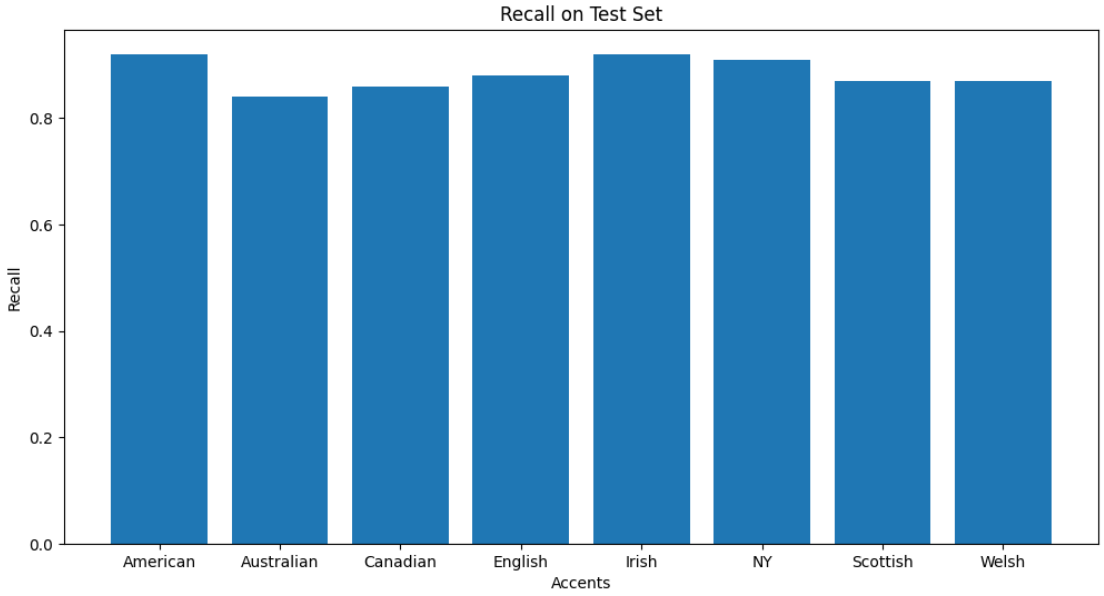


Рис. 4.7. Результат точності передбачень за акцентом методу DWT

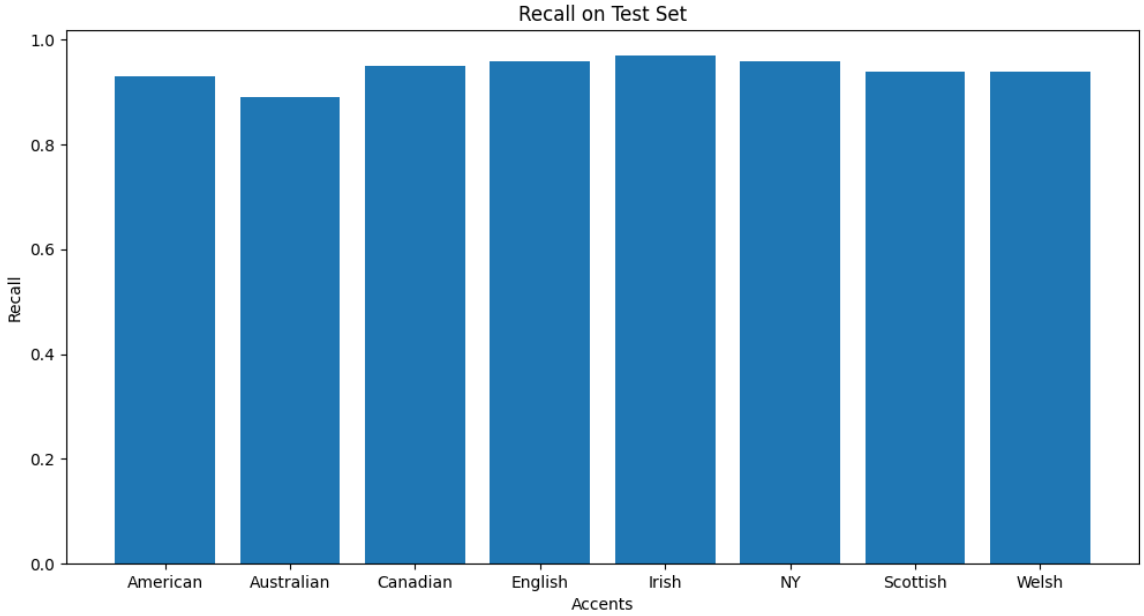


Рис. 4.8. Результат точності передбачень за акцентом комбінованого методу

ВИСНОВКИ

За результатами дослідження, проведеного в даній магістерській дисертації, можна зробити наступні висновки:

1. Проведено дослідження існуючих підходів до аналізу аудіоданих та виокремлення ключових ознак. Детально проаналізовано основні відомі методи аналізу аудіоданих, а саме MFCC, LPC, PLP та DWT.
2. Проаналізовано переваги та недоліки існуючих методів за визначеними критеріями та зроблено висновки, щодо доцільності використання даних методів. Проаналізовано які з методів гарно доповнюють один одного, а саме було припущено, що використання двох методів підряд до набору даних позитивно зіграє роль у підвищенні точності аналізу акцентів, тим самим посприяє в створенні узагальненого метода для оцінки англомовних акцентів.
3. Розроблено комбінований метод аналізу аудіоданих з урахуванням проведених досліджень. Представлено алгоритм роботи комбінованого методу аналізу аудіоданих.
4. Розроблено програмне забезпечення, що реалізує запропонований комбінований метод реферування текстових даних. Було реалізовано 5 методів за допомогою мови програмування Python, а також за допомогою розробленого програмного забезпечення була 5 раз натренована згорткова нейронна мережа, щоб на практиці отримати результати порівняння методів.
5. Визначено напрями подальшого дослідження. До них відносяться: додавання нових акцентів, а також розширення датасету для покращеного тренування даних і вищої точності оцінки акцентів в подальшому.

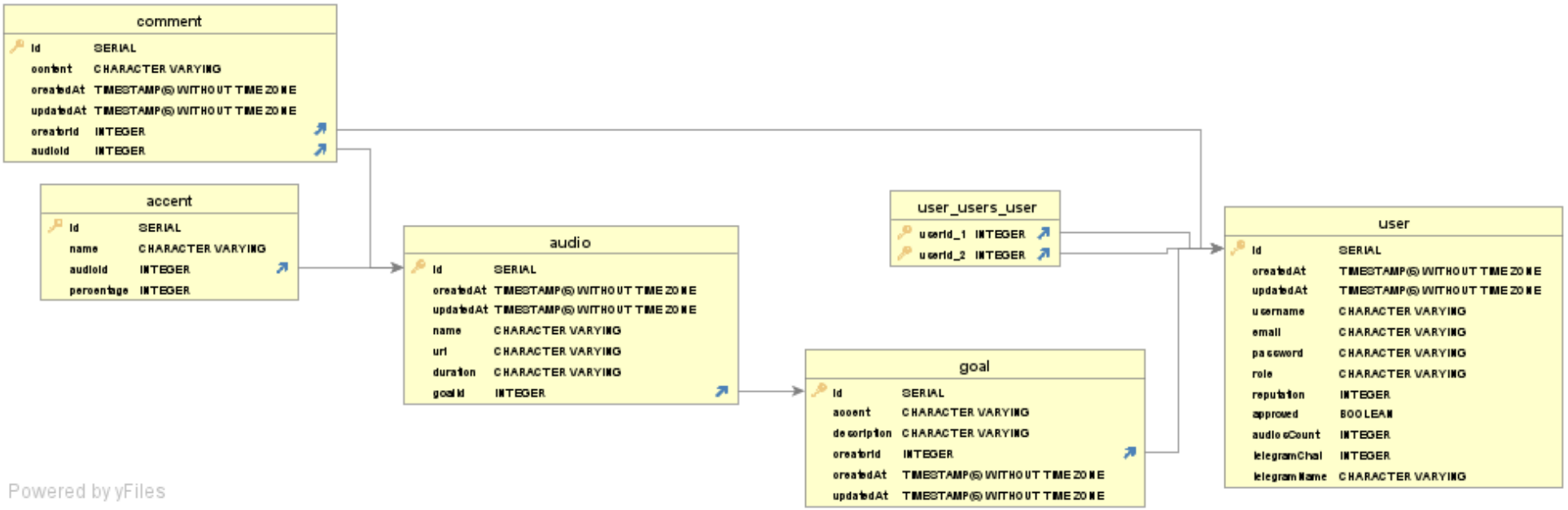
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. MFCC [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Mel-frequency_cepstrum
2. LPC [Електронний ресурс] – Режим доступ до ресурсу: https://en.wikipedia.org/wiki/Linear_predictive_coding
3. IFT [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Fourier_inversion_theorem
4. PLP [Електронний ресурс] – Режим доступу до ресурсу: https://jonathan-hui.medium.com/speech-recognition-feature-extraction-mfcc-plp-5455f5a69dd9
5. DWT [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Discrete_wavelet_transform
6. C++ [Електронний ресурс] – Режим доступу до ресурсу: https://www.cplusplus.com/info/description/
7. Welcome to Python [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Python_(programming_language)
8. Мова прорграмування Javascript [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/JavaScript
9. Мова програмування Typescript [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/TypeScript
10. Мова програмування Koltin [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Kotlin_(programming_language)
11. Переваги та недоліки Angular JS [Електронний ресурс] – Режим доступу до ресурсу: https://stfalcon.com/ru/blog/post/why-use-angularjs-for-webapps
12. React docs [Електронний ресурс] – Режим доступу до ресурсу: https://reactjs.org/
13. Vue js web framework [Електронний ресурс] – Режим доступу до

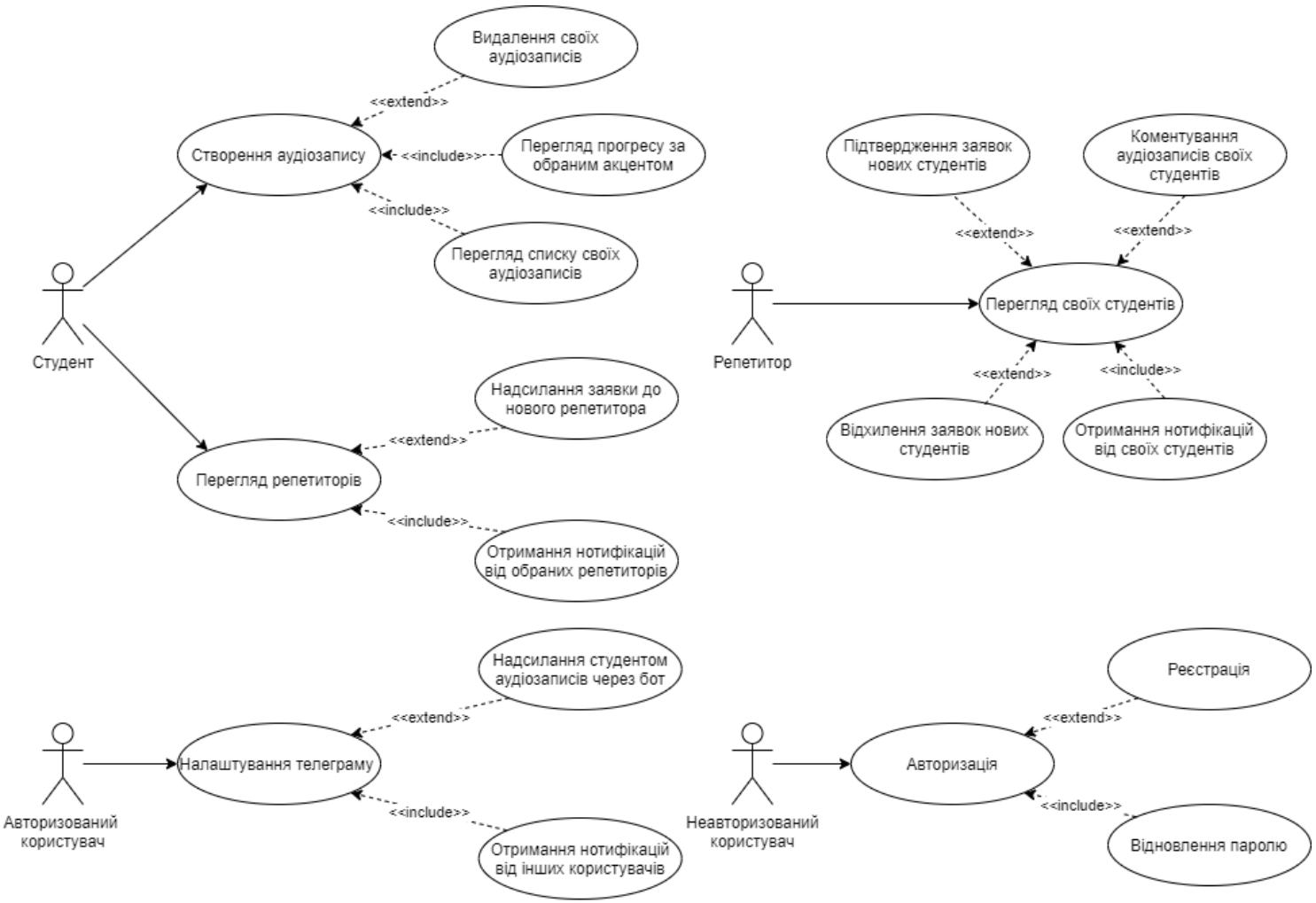
ресурсу: https://en.wikipedia.org/wiki/Vue.js
14. GraphQL docs [Електронний ресурс] – Режим доступу до ресурсу: https://graphql.org/
15. SQLite [Електронний ресурс] – Режим доступу до ресурсу: https://www.sqlite.org/index.html
16. Welcome to PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: http://www.postgresql.org/
17. Wiegers, K. Software Requirements [Текст] / Karl Wiegers. – 2nd edition. – New York : Microsoft, 2003. – 544 p.
18. Deep Learning Approach to Accent Classification [Електронний ресурс] – Режим доступу до ресурсу: http://cs229.stanford.edu/proj2017/final-reports/5244230.pdf
19. Mel Frequency Cepstral Coefficient (MFCC) tutorial [Електронний ресурс] – Режим доступу до ресурсу: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/
20. Приклад роботи операції згортки [Електронний ресурс] – Режим доступу до ресурсу: http://technodocbox.com/3D_Graphics/70716176-Deep-neural-networks-applications-in-handwriting-recognition.html
21. Bias term [Електронний ресурс] – Режим доступу до ресурсу: https://www.pico.net/kb/the-role-of-bias-in-neural-networks#:~:text=Bias%20allows%20you%20to%20shift,transposed%20by%20the%20constant%20value.
22. ReLU activation function [Електронний ресурс] – Режим доступу до ресурсу: https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/#:~:text=for%20Output%20Layers-,Activation%20Functions,a%20layer%20of%20the%20network.
23. Softmax activation function [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Softmax

ДОДАТКИ

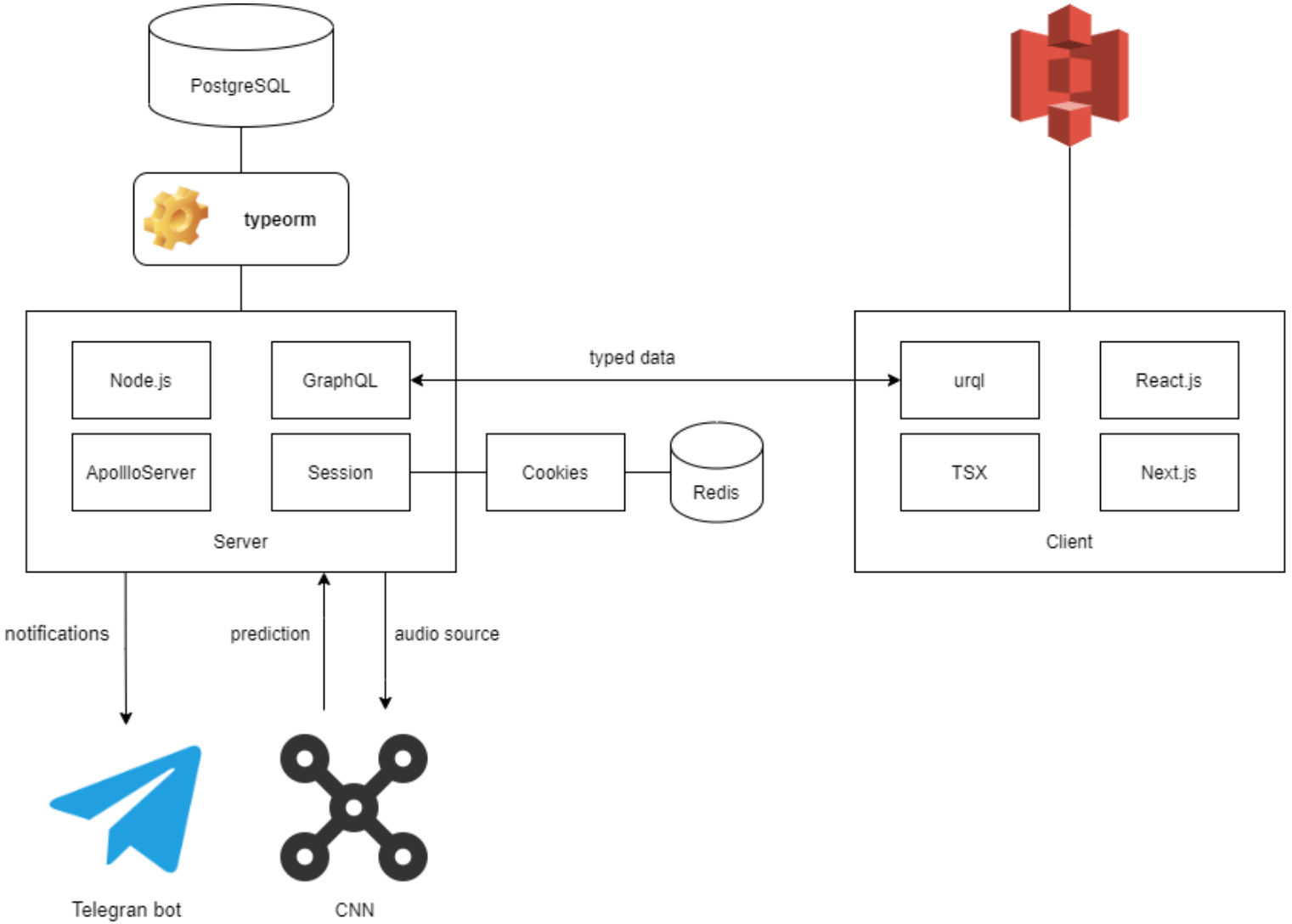
Додаток 1
Копії графічних матеріалів



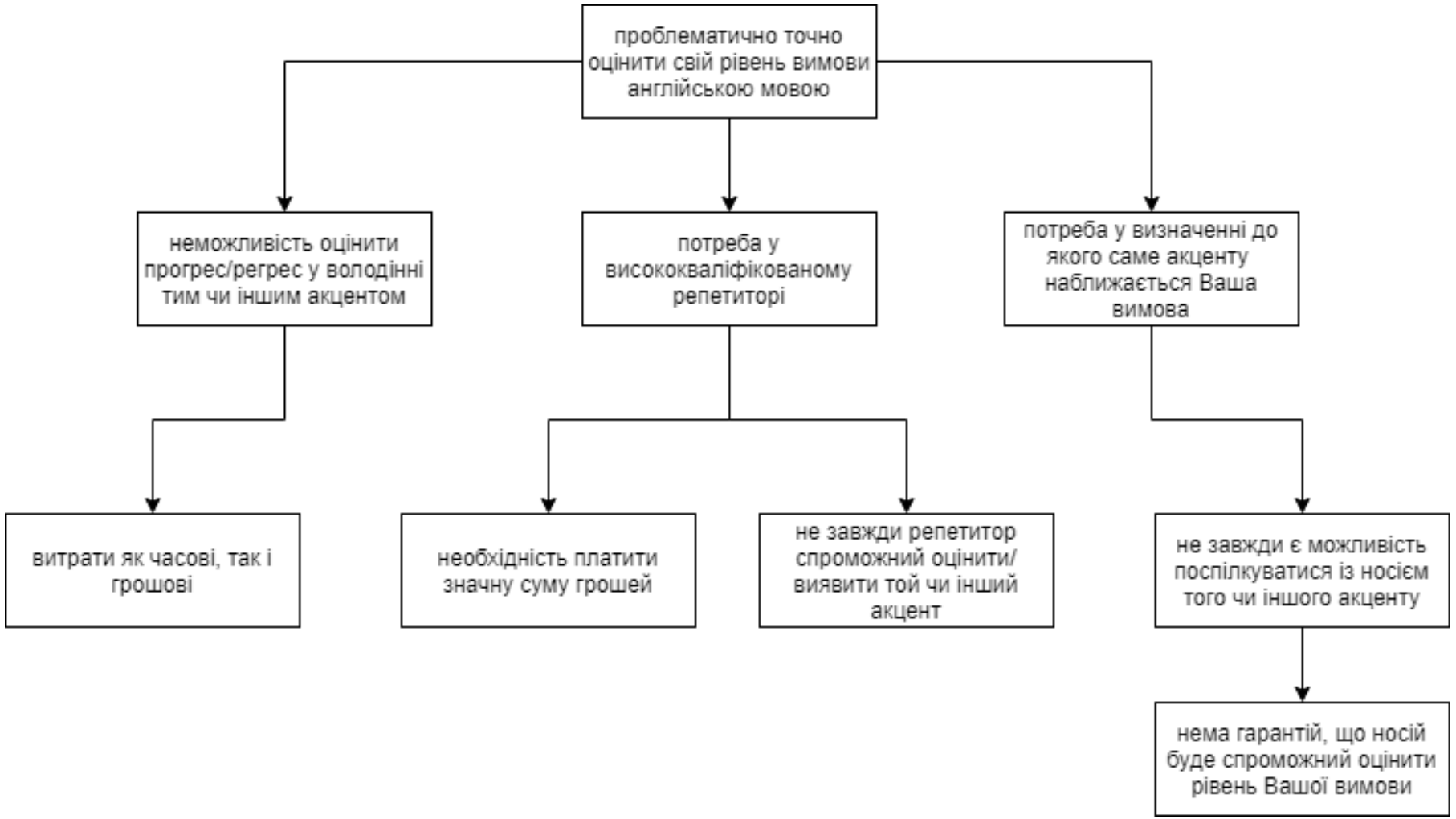
Манохін Андрій КП-11мн



Манохін Андрій КП-11мн



Манохін Андрій КП-11мн



Манохін Андрій КП-11мн

Додаток 2
Лістинг програми

```
import { Audio } from "../entities/Audio";
import "dotenv-safe/config";
import {
  Arg,
  Ctx,
  Field,
  InputType,
  Int,
  Mutation,
  ObjectType,
  Query,
  Resolver,
  UseMiddleware,
} from "type-graphql";
import { MyContext } from "src/types";
import { isAuth } from "../middleware/isAuth";
import { getConnection } from "typeorm";
import { User } from "../entities/User";
import { spawn } from "child_process";

@InputType()
class AudioInput {
  @Field()
  url: string;

  @Field()
  name: string;

  @Field()
  duration: string;
}

@ObjectType()
class PaginatedAudios {
  @Field(() => [Audio])
  audios: Audio[];

  @Field()
  hasmore: boolean;
}

@Resolver(Audio)
export class AudioResolver {
  @Query(() => PaginatedAudios)
  async audios(
    @Arg("limit", () => Int) limit: number,
    @Arg("cursor", () => String, { nullable: true }) cursor: string,
    @Ctx() { req }: MyContext
  ): Promise<PaginatedAudios> {
    const realLimit = Math.min(50, limit);
    const qb = getConnection()
      .getRepository(Audio)
      .createQueryBuilder("a")
      .innerJoinAndSelect("a.creator", "u", 'u.id = a."creatorId"')
      .where('a."creatorId" = :id', { id: req.session.userId })
      .orderBy('a."createdAt"', "DESC")
      .limit(realLimit + 1);
    if (cursor) {
      qb.where('a."createdAt" < :cursor', {
        cursor: new Date(parseInt(cursor)),
      });
    }
```

```
    const audios = await qb.getMany();
    console.log("PagAudios: ", audios);
    return {
      audios: audios.slice(0, realLimit),
      hasmore: audios.length === realLimit + 1,
    };
  }

  @Query(() => Audio, { nullable: true })
  audio(@Arg("id") id: number): Promise<Audio | undefined> {
    return Audio.findOne(id);
  }

  runAsyncAccentDeterminer(url: string) {
    console.log("RUNNING!");
    const python_cmd = spawn("python", ["D:\\diploma\\CNN\\diploma-
cnn\\predict_by_data.py", "D:\\diploma\\CNN\\diploma-cnn\\cnn\\params.pkl",
url]);
    python_cmd.stdout.on('data', (data) => {
      console.log(`stdout: ${data}`);
    });
  }

  @Mutation(() => Audio)
  @UseMiddleware(isAuth)
  async createAudio(
    @Arg("audioInput", () => AudioInput) audioInput: AudioInput,
    @Ctx() { req }: MyContext
  ): Promise<Audio> {
    const { name, url, duration } = audioInput;
    let userId;
    if (req.session?.userId) {
      userId = req.session.userId;
    }
    console.log("userId", userId);
    const user = await User.findOne(userId);
    if (user) {
      user.audiosCount++;
      user.reputation += 10;
      User.update(userId, user);
      this.runAsyncAccentDeterminer(url);
    }
    return Audio.create({
      name: name,
      url: url,
      duration: duration,
      accent: "undefined",
      creatorId: userId,
    }).save();
  }

  @Mutation(() => Int)
  async deleteAudio(@Arg("id", () => Int) id: number): Promise<number> {
    await Audio.delete(id);
    return id;
  }
}
```

```
import { User } from "../entities/User";
import { MyContext } from "src/types";
import {
  Arg,
  Field,
  Mutation,
  Resolver,
  Ctx,
  Query,
  ObjectType,
  Int,
  UseMiddleware,
} from "type-graphql";
import argon2 from "argon2";
import { COOKIE_NAME, FORGET_PASSWORD_PREFIX } from "../constants";
import { sendEmail } from "../utils/sendEmail";
import { UsernamePasswordInput } from "./UsernamePasswordInput";
import { validateRegister } from "../utils/validateregister";
import { v4 } from "uuid";
import { getConnection } from "typeorm";
import { isAuth } from "../middleware/isAuth";

@ObjectType()
class FieldError {
  @Field()
  field: string;

  @Field()
  message: string;
}

@ObjectType()
class PaginatedUsers {
  @Field(() => [User])
  users: User[];

  @Field()
  hasmore: boolean;
}

@ObjectType()
class UserResponse {
  @Field(() => [FieldError], { nullable: true })
  errors?: FieldError[];

  @Field(() => User, { nullable: true })
  user?: User;
}

@Resolver(User)
export class UserResolver {

  @Query(() => PaginatedUsers)
  @UseMiddleware(isAuth)
  async getUsers(
    @Arg("limit", () => Int) limit: number,
    @Arg("cursor", () => String, { nullable: true }) cursor: string,
    @Ctx() { req }: MyContext
  ): Promise<PaginatedUsers> {
    const realLimit = Math.min(50, limit);
    const realLimitPlus1 = realLimit + 1;
    let users: User[] = await getConnection().query(`
      select * from public.user u1
      inner join user_users_user uu
```

```
      on (u1."id" = uu."userId_1" or u1."id" = uu."userId_2")
      inner join public.user u2
      on (u2."id" = uu."userId_1" or u2."id" = uu."userId_2")
      where u1."id" = $1
      ${cursor ? 'and where u2."createdAt" < :cursor' : ''}
      order by u2."createdAt" desc
      limit $2
    `, [req.session.userId, realLimitPlus1]);
    users = users.filter(user => user.id !== req.session.userId);
    console.log("USER: ", users);
    return {
      users: users.slice(0, realLimit),
      hasmore: users.length === realLimit + 1,
    };
  }

  @Query(() => [User])
  allUsers() {
    return User.find({});
  }

  @Mutation(() => Boolean)
  async forgotPassword(
    @Arg("email") email: string,
    @Ctx() { redis }: MyContext
  ) {
    const user = await User.findOne({ where: { email } });
    if (!user) {
      return true;
    }

    const token = v4();
    await redis.set(
      FORGET_PASSWORD_PREFIX + token,
      user.id,
      "ex",
      1000 * 60 * 60 * 24 * 3
    ); // 3 days

    await sendEmail(
      email,
      `<a href="http://localhost:3000/change-password/${token}">reset
password</a>`
    );

    return true;
  }

  @Mutation(() => UserResponse)
  async changePassword(
    @Arg("newPassword") newPassword: string,
    @Arg("token") token: string,
    @Ctx() { redis, req }: MyContext
  ) {
    if (newPassword.length <= 2) {
      return {
        errors: [
          {
            field: "newPassword",
            message: "length must be greater than 2",
          },
        ],
      };
    }
```

```
  const key = FORGET_PASSWORD_PREFIX + token;
  const userId = await redis.get(key);
  if (!userId) {
    return {
      errors: [
        {
          field: "token",
          message: "token expired",
        },
      ],
    };
  }

  const id = parseInt(userId);
  const user = await User.findOne(id);
  if (!user) {
    return {
      errors: [
        {
          field: "token",
          message: "user no longer exists",
        },
      ],
    };
  }

  await User.update(
    { id },
    {
      password: await argon2.hash(newPassword),
    }
  );

  await redis.del(key);

  req.session.userId = user.id;

  return { user };
}

@Query(() => User, { nullable: true })
me(@Ctx() { req }: MyContext) {
  const id = req.session.userId;
  console.log(req.session.userId);
  if (!id) {
    return null;
  }
  return User.findOne(id);
}

@Mutation(() => UserResponse)
async register(
  @Arg("options") options: UsernamePasswordInput,
  @Ctx() { req }: MyContext
): Promise<UserResponse> {
  const errors = validateRegister(options);
  if (errors) {
    return { errors };
  }
  const hashedPassword = await argon2.hash(options.password);
  let user;
  try {
    user = await User.create({
      username: options.username,
```

```
      email: options.email,
      password: hashedPassword,
      role: options.role,
    }).save();
  } catch (err) {
    console.log("error: ", err);
    if (err.code === "23505" || err.detail.includes("already exists")) {
      return {
        errors: [
          {
            field: "username",
            message: "username already exists",
          },
        ],
      };
    } else {
      console.log(err);
    }
  }

  req.session.userId = user?.id;

  return { user };
}

@Mutation(() => UserResponse)
async login(
  @Arg("usernameOrEmail") usernameOrEmail: string,
  @Arg("password") password: string,
  @Ctx() { req }: MyContext
): Promise<UserResponse> {
  const user = await User.findOne(
    usernameOrEmail.includes("@")
      ? { where: { email: usernameOrEmail } }
      : { where: { username: usernameOrEmail } }
  );
  if (!user) {
    return {
      errors: [
        {
          field: "usernameOrEmail",
          message: "username doesn't exist",
        },
      ],
    };
  }
  const valid = await argon2.verify(user.password, password);
  if (!valid) {
    return {
      errors: [
        {
          field: "password",
          message: "wrong password",
        },
      ],
    };
  }

  req.session.userId = user.id;

  console.log("session: ", req.session.userId);

  return { user };
}
```

```
  @Mutation(() => Boolean)
  logout(@Ctx() { req, res }: MyContext) {
    return new Promise((resolve) =>
      req.session.destroy((err) => {
        res.clearCookie(COOKIE_NAME);
        if (err) {
          console.log(err);
          resolve(false);
          return;
        }
        resolve(true);
      })
    );
  }

  @Mutation(() => Boolean)
  subscribe(@Ctx() { req }: MyContext, @Arg("username") username: String) {
    return new Promise(async (resolve) => {
      const coach = await getConnection()
        .getRepository(User)
        .createQueryBuilder("u")
        .leftJoinAndSelect("u.users", "user")
        .where('u."username" = :username', {username})
        .getOne();
      const student = await User.findOne(req.session.userId);
      console.log(coach?.users)
      if (!student) {
        console.log("Not authenticated!");
        resolve(false);
      } else if (!coach) {
        console.log("Coach not found!");
        resolve(false);
      } else {
        if (coach.users && coach.users.length > 0)
coach.users.push(student);
        else coach.users = [student];
        student.approved = false;
        await getConnection().manager.save([student, coach]);
        resolve(true);
      }
    });
  }

  @Mutation(() => Boolean)
  @UseMiddleware(isAuth)
  approve(@Ctx() { req }: MyContext, @Arg("username") username: String) {
    return new Promise(async (resolve) => {
      const student = await User.findOne({ where: { username } });
      const coach = await getConnection()
        .getRepository(User)
        .createQueryBuilder("u")
        .leftJoinAndSelect("u.users", "user")
        .where('u."id" = :id', {id: req.session.userId})
        .getOne();
      if (!student) {
        console.log("User not found!");
        resolve(false);
      } else if (
        coach?.users.find((u: User) => u.id === student.id) &&
        !student.approved
      ) {
        student.approved = true;
        await getConnection().manager.save(student);
```

```
      }
      resolve(true);
    });
  }

  @Mutation(() => Boolean)
  @UseMiddleware(isAuth)
  discard(
    @Ctx() { req }: MyContext,
    @Arg("username", () => String) username: String
  ) {
    return new Promise(async (resolve) => {
      const coach = await getConnection()
        .getRepository(User)
        .createQueryBuilder("u")
        .leftJoinAndSelect("u.users", "user")
        .where('u."id" = :id', {id: req.session.userId})
        .getOne();
      if (coach?.role === "student") {
        console.log("Unpredicted behaviour");
        resolve(false);
      } else {
        coach?.users.forEach((el, idx) => {
          if (el.username === username) coach.users.splice(idx, 1);
        });
      }
      resolve(true);
    });
  }
}
```

Додаток 3
Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

Метод виокремлення ознак із аудіо для розпізнавання англомовних акцентів

Доповідач: Манохін Андрій Віталійович

Науковий Керівник: к.т.н., ст. викладач кафедри ПЗКС, Рибачок Наталія Антонівна

Київ – 2023

АКТУАЛЬНІСТЬ ДОСЛІДЖЕННЯ

- Відсутність генералізації у сучасних популярних методів
- Внаслідок цієї проблеми, погано оцінюються акценти на не натренованих даних

Об’єкт дослідження: процес виокремлення інформації із аудіоданих записаних англомовними користувачами

Предмет дослідження: метод виокремлення інформації із аудіоданих записаних англомовними користувачами

Наукове дослідження: розробка методу для виокремлення ознак для подальшої класифікації англомовних акцентів за аудіозаписом
Мета дослідження: підвищити ефективність класифікації англомовних акцетів в середньому на 5% шляхом виокремлення більш детальних даних із аудіозаписів

Окремі завдання

1. Дослідити існуючі методи аналізу аудіозаписів для виокремлення важливих даних.
2. На основі результатів дослідження розробити власний метод видобування необхідних даних із аудіофайлів, що призведуть до кращих результатів класифікації методами машинного навчання.
3. Розробити програмне забезпечення, що реалізує даний метод.
4. Проаналізувати ефективність розробленого методу, шляхом порівняння результатів класифікації даних, отриманих за його допомогою і за допомогою вже існуючих методів.

НАЙБІЛЬШ ВІДОМІ РІШЕННЯ ЦІЄЇ НАУКОВОЇ ПРОБЛЕМИ

Розглянемо наступні методи ефективного виокремлення даних із аудіозапису:

- Лінійне передбачуване кодування (Linear predictive coding (LPC) / Fumitada Itakura)[1];
- Перцептивний лінійно-передбачуваний аналіз мовлення (Perceptual linear predictive (PLP) analysis of speech / Hermansky, Hynek)[2];
- Мелчастотні кепстральні коефіцієнти (Mel Frequency Cepstral Coefficients (MFCC) / Paul
- Mermelstein)[3];
- Дискретне вейвлет-перетворення (discrete wavelet transform (DWT) / Donald B. Percival, Debashis Mondal)[4];

ПОРІВНЯННЯ ІСНУЮЧИХ РІШЕНЬ

	Що моделюється	Швидкість обчислень	Шумостійкість	Чутливість до квантування/додатков их шумів	Надійність	Зафіксована частота
Мелчастотні кепстральні коефіцієнти (MFCC)	Слухова система людини	Висока	Середня	Середня	Висока	Низька
Лінійні коефіцієнти передбачення (LPC)	Вокальний тракт людини	Висока	Висока	Висока	Висока	Низька
Дискретне вейвлет-перетворення (DWT)	-	Висока	Середня	Середня	Середня	Низька & Висока
Перцептивний лінійно-передбачуваний аналіз мовлення (PLP)	Слухова система людини	Середня	Середня	Середня	Середня	Низька & Середня

Гіпотези

- Застосування багатошарової згорткової мережі (CNN) до результатів вилучення ознак кожного метода таким чином дозволить дослідити ефективність кожного з них.
- Кожен метод має як свої недоліки, так і переваги, що можна використати на користь комбінованого методу.

Використання підходу DWT

DWT - перетворення, яке розкладає даний сигнал на кілька наборів, де кожен набір являє собою часовий ряд коефіцієнтів, що описують тимчасову еволюцію сигналу у відповідній смузі частот.

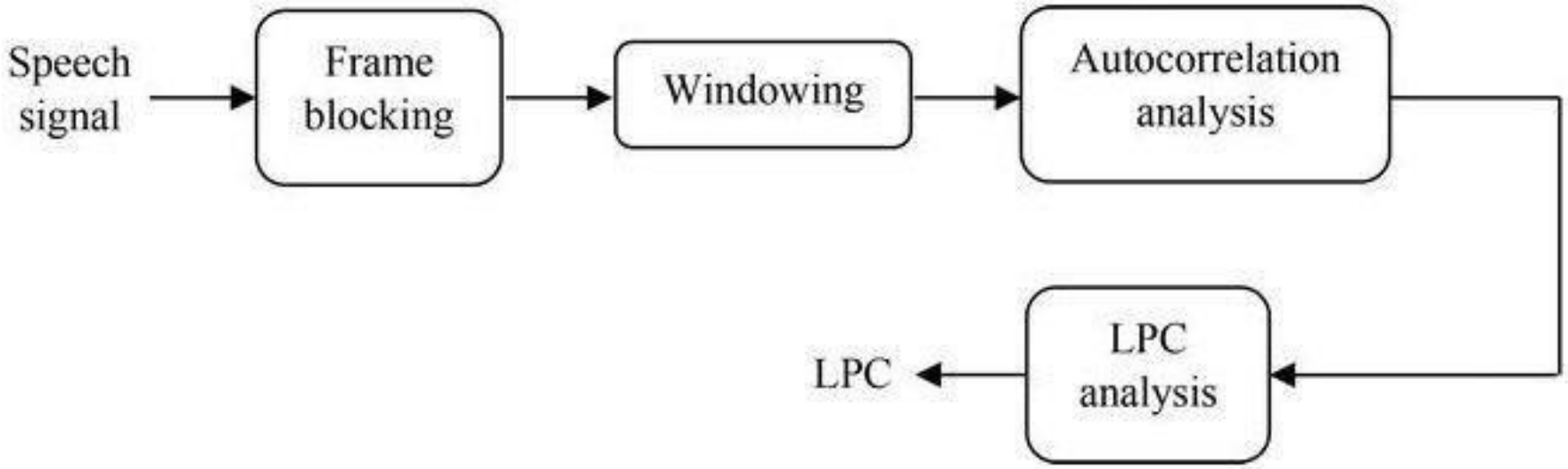
Теорія DWT вимагає двох наборів пов'язаних функцій, які називаються функцією масштабування та вейвлет-функцією, заданою як

$$\phi(t) = \sum_{n=0}^{N-1} h[n]\sqrt{2}\phi(2t-n) \qquad \psi(t) = \sum_{n=0}^{N-1} g[n]\sqrt{2}\phi(2t-n),$$

де ϕ(t) — функція масштабування, ψ(t) — вейвлет-функція, h[n] — імпульсна характеристика фільтра нижніх частот, а g[n] — імпульсна характеристика фільтра високих частот.

Використання підходу LPC

Лінійні коефіцієнти передбачення (LPC) імітують голосовий тракт людини і надають надійну характеристику мовлення. Метод оцінює мовленнєвий сигнал шляхом апроксимації формант, позбавляється його ефектів від мовного сигналу та оцінює концентрацію та частоту залишку.



Мотивація

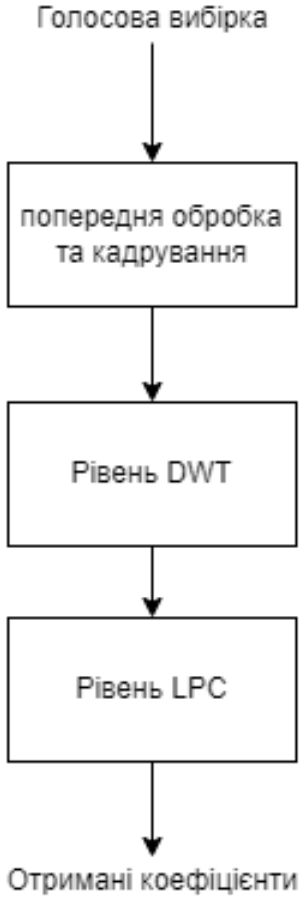
Отримані коефіцієнти LP можуть вловити високочастотні піки, присутні в мовному сигналі, а також не можуть точно проаналізувати локалізовані події, які натомість може аналізувати вейвлет-перетворення (DWT). Однак LPC здатні краще розрізняти слова, за допомогою моделювання голосового тракту. DWT здатний моделювати деталі глухої звукової частини мови, на відміну від LPC.

Отже, ми можемо застосувати техніку LPC до кожного піддіапазонного сигналу після вейвлет-декомпозиції, що дає комбіновані переваги LPC та WT.

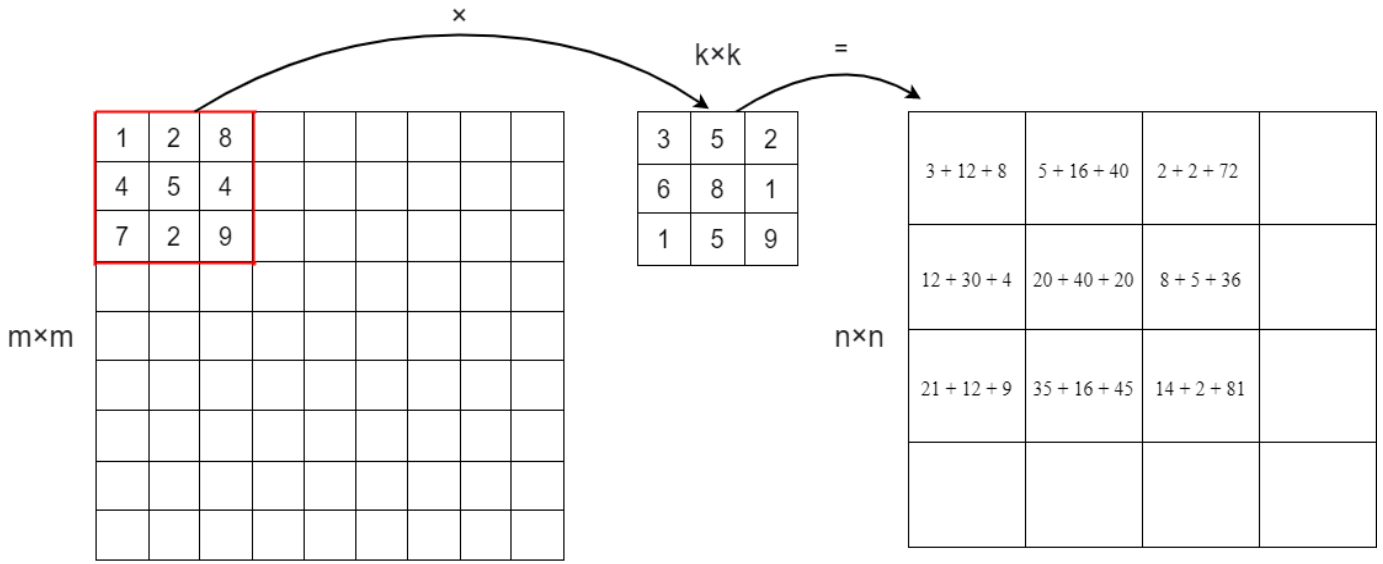
Поєднання

Декомпозиція DWT попередньо оброблених і віконних кадрів мовлення виконується за допомогою вейвлет-фільтрів.

Фактичні вейвлет-коефіцієнти зберігають інформацію про час, отже, характеристики LPC оцінюються за коефіцієнтами DWT у часовій області.



Згорткова нейронна мережа



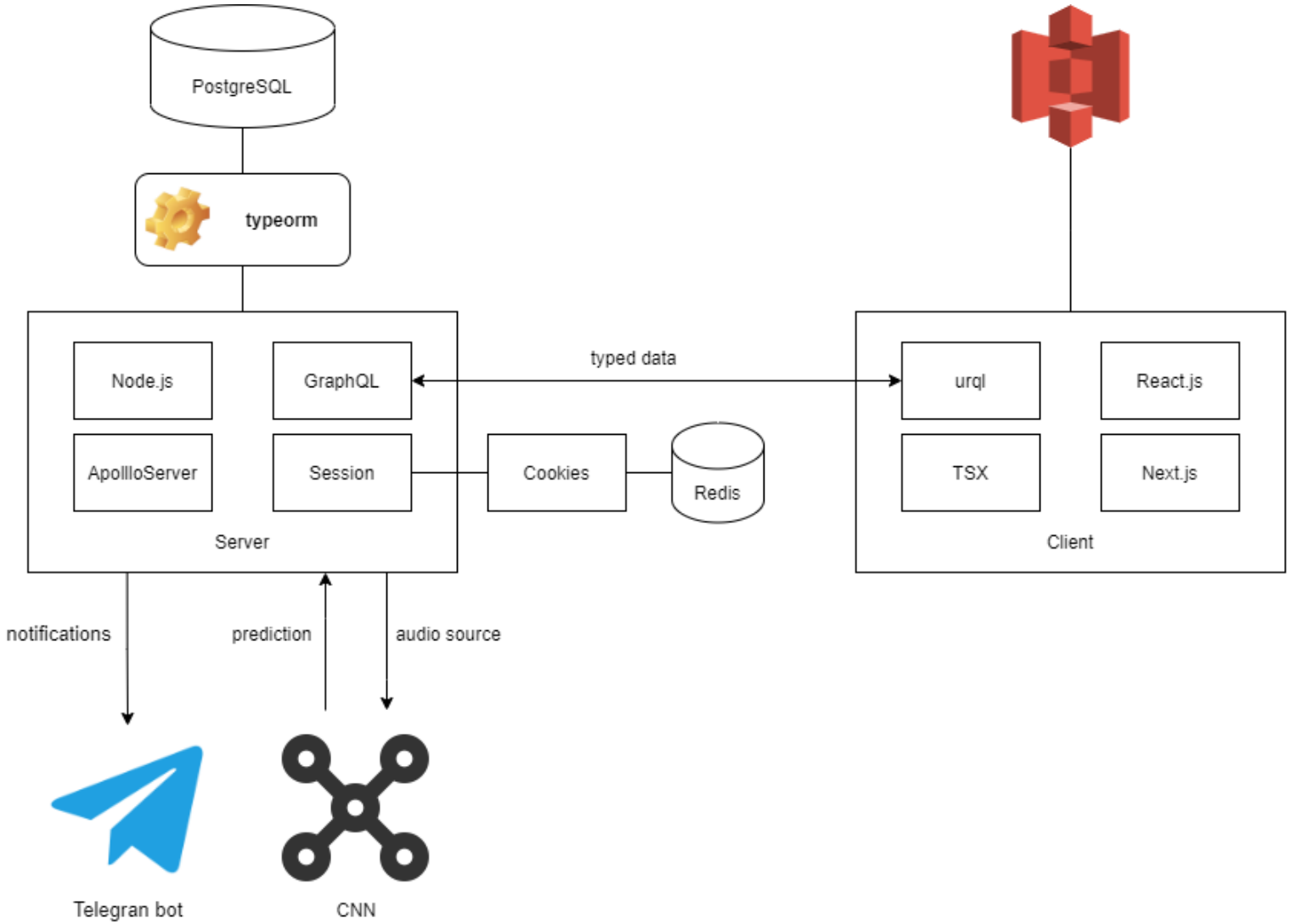
$$n = \left\lfloor \frac{m-k}{s} \right\rfloor + 1,$$

n – розмір вихідної матриці,

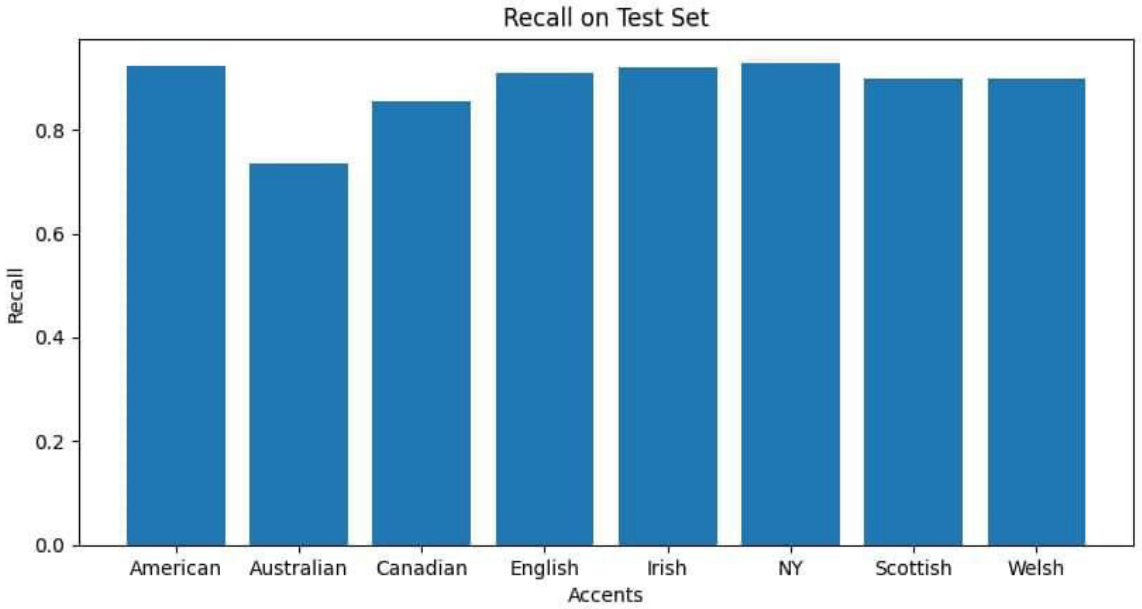
m – розмір вхідної матриці,

k – розмір матриці фільтру,
s – крок просування вікна.

СТРУКТУРА ВЕБСЕРВІСУ

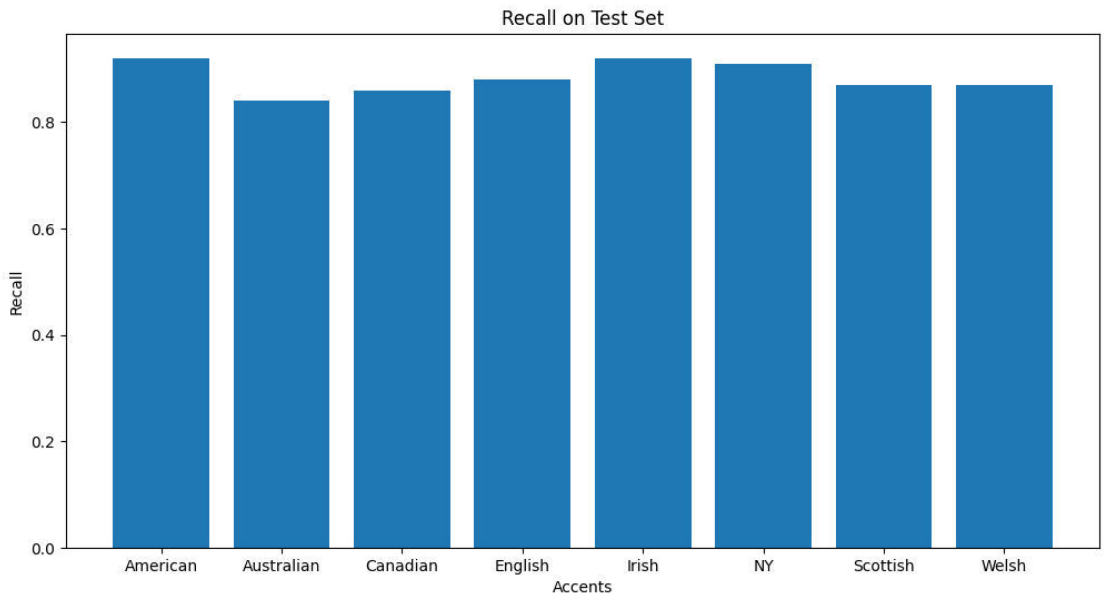


Результати



Передбачення MFCC

Точність - 89.07%



Передбачення комбінованого методу

Точність – 94.28%

НАУКОВА НОВИЗНА

Уперше запропоновано метод розпізнавання англомовних акцентів, який відрізняється від існуючих унікальним порядком виконання процедур оброблення мовленнєвого сигналу, а саме, застосуванням алгоритму LPC після оброблення сигналу методом DWT, що дозволяє підвищити точність передбачень англомовних акцентів у середньому на 5%

ПРАКТИЧНЕ ЗНАЧЕННЯ

Практичне значення одержаних результатів полягає у тому, що запропонований метод розпізнавання англомовних акцентів дозволяє підвищити якість програмного оброблення мовленнєвого сигналу, що у свою чергу підвищує якість відповідного програмного продукту.

ВИСНОВКИ ТА ПОДАЛЬША РОБОТА

1. Було розглянуто складові частини поставленого завдання, виокремлено етапи, які необхідні для реалізації.
2. У результаті аналізу наукової літератури за обраною тематикою було виокремлено найбільш відомі методи ефективного виокремлення даних із аудіозапису та досліджено їх переваги та недоліки.
3. Встановлено, що найбільш популярні методи мають свої недоліки, та погано підходять для узагальнення.
4. Запропоновано гіпотези, які виправляють проаналізовані недоліки знайдених методів. А саме: сформульовано ідею щодо комбінування переваг методів.
5. Була представлена схема комбінування двух непоганих методів, які добре перекривають недоліки один одного
6. В подальшому можна розширити датасет на якому тренуються задля більш точної оцінки акцентів на практиці, а також додати нові акценти.

АПРОБАЦІЯ РОБОТИ

XV наукова конференція магістрантів та аспірантів “Прикладна математика та комп’ютинг” ПМК-2022

Дякую за увагу!