Міністерство освіти і науки України
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Факультет систем управління літальних апаратів

Кафедра систем управління літальних апаратів

Пояснювальна записка

до дипломної роботи
магістра
(освітньо-кваліфікаційний рівень)

на тему «Розробка та дослідження мобільного додатку аутентифікації користувача через голосове повідомлення»

ХАІ.301.362.22О.151. 173.039 ПЗ

Виконав: студент 2 курсу, групи 362

Галузь знань 15 «Автоматизація та приладобудування»

Спеціальність

151 «Автоматизація та комп'ютерно- інтегровані технології»

Освітня програма
«Інженерія мобільних додатків»

Сосюра Б.В.
(прізвище та ініціали студента)

Керівник проф., д.т.н. Барсов В.І
(прізвище та ініціали)

Рецензент Рибочкіна Л. М.
(прізвище та ініціали)

Харків – 2022

Міністерство освіти і науки України
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Факультет систем управління літальних апаратів
Кафедра систем управління літальних апаратів (№301)
Рівень вищої освіти___магістр_____
Галузь знань _____15 «Автоматизація та приладобудування»_____
Спеціальність ____«Автоматизація та комп'ютерно- інтегровані технології» _____
Освітня програма__ Інженерія мобільних додатків _____

ЗАТВЕРДЖУЮ
Завідуючий кафедрою 301
к.т.н., с.н.с., доцент

_____ К. ДЕРГАЧОВ
"____" _____2022 р

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

_____Сосюра Богдану Вадимовичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи «Розробка та дослідження мобільного додатку аутентифікації користувача через голосове повідомлення» керівник роботи проф., д.т.н. Барсов В.І.
затверджені наказом вищого навчального закладу від _17.11. 2022 року № _1602-уч
2. Строк подання студентом роботи: 09 грудня 2022 року
3. Вихідні дані до роботи: Розробка програмного засобу розпізнавання голосу і голосових команд. Організація даних: файлова з прямим доступом . Форма діалогу: мобільний додаток. Перелік використовуваних програмних засобів: ОС Microsoft Windows 10, інтегроване середовище програмування Android Studio. Технічне забезпечення: мобільний пристрій.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)_
1. Аналіз тематичної області. 2. Опис та аналіз підходів по розробці засобів розпізнавання. 3. Проектування та розробка програмного засобу. 4. Розробка інструкції по експлуатації мобільного пристрою. 5. Аналіз результатів отриманих ПЗ. 6. Економічна частина.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
10 слайдів формату А4: Аналіз проблем розпізнавання вербальної інформації, Опис алгоритму розпізнавання голосової команди, Опис алгоритму оцінки якості голосових команд, Опис алгоритму перевірки диктора, Моделювання алгоритму перевірки диктора, Створення дизайну програми для аутентифікації голосових команд, Вимо-

ги для програмного забезпечення, Інтерфейс користувача, Аналіз результатів, отриманих за допомогою програмного забезпечення, Економічна частина.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	проф., д.т.н. Барсов В.І.	20.09 2022	30.09 2022
2	проф., д.т.н. Барсов В.І.	30.09.2022	10.10 .2022
3	проф., д.т.н. Барсов В.І.	5.10.2022	25.10.2022
4	проф., д.т.н. Барсов В.І.	10.10.2022	10.11.2022
5	проф., д.т.н. Барсов В.І.	20.10.2022	30.11.2022
6	к.е.н., доц. Попов О.С.	1.12.2022	10.12.2022

Нормоконтроль _____ ____В.І Барсов «9» 12.2022 р.

7. Дата видачі завдання_____19.09.2022_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1.	Початок переддипломної практики	12.09.2022	
2.	Формулювання теми роботи. Розробка технічного завдання	19.09.2021	
3.	Математичний опис системи управління. Аналіз і синтез системи управління. Проведення експериментальних досліджень	28.10.2022	Залік з переддипломної практики
4.	Конструкторська частина роботи. Дослідницька частина роботи. Експериментально-практична частина. Економічне обґрунтування розробки. Розробка питань охорони праці і безпеки в надзвичайних ситуаціях	28.11.2022	
5.	Оформлення розрахунково-пояснювальної записки і графічного матеріалу	09.12.2022	
6.	Попередній захист роботи. Рецензування роботи	16.12.2022	
7.	Захист роботи	20.12.2022	

Студент _____ __Сосюра Б.В.___

Керівник роботи _____ Барсов В.І.__

Міністерство освіти і науки України
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Кафедра систем управління літальних апаратів (№301)

«ЗАТВЕРДЖУЮ»
Завідуючий кафедрою 301
к.т.н., с.н.с., доцент
_____ К. ДЕРГАЧОВ
«__» _____2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на дипломне проектування
Сосюри Богдана Вадимовича

1 Тема роботи: «Розробка та дослідження мобільного додатку аутентифікації користувача через голосове повідомлення»
затверджена наказом по університету від «17» 11 2022 р. № 1602 уч .
2 Строк здачі студентом закінченої роботи « 09 » грудня 2022 р.
3 Область застосування розробки: додаток може бути використаний для авторизації на персональних пристроях за допомогою голосу._____
4 Початкові дані для розроблювальної системи
4.1 Призначення і мета створення системи: розробка мобільного додатку аутентифікації користувача через розпізнавання голосу._____
4.2 Загальні відомості мобільний додаток має ідентифікувати користувача використовуючи методу розпізнавання за ключовими словами на основі перетворення піддіапазону з використанням алгоритму динамічних спотворень_____
5 Технічні вимоги до каналів системи управління
5.1 Питання, що підлягають розробці: розробка мобільного додатку, який по своїй швидкості та процентів правильної ідентифікації буде краще ніж аналоги.
5.2 Вимоги до структури й функціонування системи: система має функціонувати на операційній системі андроїд версії 4.2 та вище._
5.3 Вимоги до показників якості системи : коефіцієнт помилкових прийомів (FAR) і коефіцієнт помилкових відхилень (FRR) маю бути меньші 5%
5.4 Вимоги до конфігурації обчислювальної техніки: комп'ютері з частотою процесора 1,60 ГГц і кількістю ОЗУ 1 Гб) та мобільний пристрій на операційній системі андроїд версії 4.2 та вище__
6 Умови експлуатації системи
6.1 Вимоги до програмної та інформаційної сумісності: повинні бути сумістні
6.2 Вимоги до захисту інформації від несанкціонованого доступу:_система повина бути захищена
6.3 Вимоги до зовнішніх збурень: система повинна вміти фільтрувати зовнішні шуми
7 Характер роботи системи (безперервної, циклічний, одноразового дії):

циклічний _____
8 Додаткові функції, реалізовані системою (сигналізація про несправності, реєстрація необхідної інформації, самоконтроль самої системи і т.ін.): реєстрація необхідної інформації_____
9 Обсяг виконуваних розроблювачем робіт
9.1 Етапи проведення роботи: 1. Аналіз тематичної області. 2. Опис та аналіз підходів по розробці засобів розпізнавання. 3. Проектування та розробка програмного засобу. 4. Розробка інструкції по експлуатації мобільного пристрою. 5. Аналіз результатів отриманих ПЗ. 6. Економічна частина
9.2 Обсяг розробки по кожному етапу: 1. Стан проблеми(13 ст.), 2. Аналіз існуючих підходів(29 ст.), 3.Конструкторсько-дослідницька частина(10 ст.), 4. Розробка інструкції по експлуатації мобільного пристрою (2 ст.), Експеременталько-дослідницька частина(2 ст.), 6.Економічна частина(15 ст.)
9.3 Вимоги до чисельності й кваліфікації персоналу:
визначаються в процесі проектування _____
10 Вимоги до захисту інформації й надійності:
визначаються в процесі проектування _____
11 Порядок контролю й приймання системи:
визначаються в процесі проектування _____
12 Дослідницька частина:
Дослідження програмних засобів по ідентифікації користувача через голосове повідомлення_____
13 Експериментально-практична частина: Інтеграційне тестування програмного забезпечення по ідентифікації користувача через голосове повідомлення___
14 Економічна частина
14.1 Розробити (розрахувати, одержати): розрахувати собівартість розробки мобільного додатку____
14.2 Умови і вимоги: одиничне виробництво, розрахунок проводиться за статтями калькуляції
14.3 Очікуваний результат: повна собівартість виробу грн. з урахуванням усіх витрат_____
15 Перелік графічних матеріалів: 10 слайдів формату А4: Аналіз проблем розпізнавання вербальної інформації, Опис алгоритму розпізнавання голосової команди, Опис алгоритму оцінки якості голосових команд, Опис алгоритму перевірки диктора, Моделювання алгоритму перевірки диктора, Створення дизайну програми для аутентифікації голосових команд, Вимоги для програмного забезпечення, Інтерфейс користувача, Аналіз результатів, отриманих за допомогою програмного забезпечення, Економічна частина.

Керівник проектування	Прийняв до виконання
___Барсов В.І.	___Сосюра Б.В._
« 12 » . 09 . 2022 р.	« 12 ». 09. 2022 р.

Погоджено з питань:

проектування
_____В.Г. Джулгаков_____
(П.І.Б.)
_____
« 12 » 09.2022 р.

дослідницької частини
___В.І. Барсов_____
(П.І.Б.)
_____
« 12 ».09. 2022 р.

економіки
_____О.С. Попов_____
(П.І.Б.)
_____
« 15 » .11. 2022 р

РЕФЕРАТ

Дипломна робота містить: 94 сторінки, 29 рисунків, 12 таблиць, 1 додаток, 17 джерел.

АЛГОРИТМ DTW, АУТЕНТИФІКАЦІЯ, БІОМЕТРІЯ, МОВНА МОДЕЛЬ, РОЗПІЗНАВАННЯ ГОЛОСУ, ФІЛЬТРАЦІЯ ШУМУ, ANDROID.

Об’єкт розробки – мобільний додаток аутентифікації користувача через розпізнавання голосу.

Мета атестаційної роботи – розробка мобільного додатку аутентифікації користувача через розпізнавання голосу.

Методи розробки – язик програмування Java, JavaScript.

Результати атестаційної роботи – мобільний додаток аутентифікаціЇ користувача через розпізнавання голосу.

Область застосування – додаток може бути використаний для авторизації на персональних пристроях за допомогою голосу.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

FDS — це метод фонетичного декодування слів.

ARM - це мікропроцесорна архітектура.

VoiceXML — це інтерактивна мова розмітки.

ГГц - гігагерц.

ГБ - гігабайт.

FAR- коефіцієнт помилкових прийомів

FRR- коефіцієнт помилкових відхилень

DTW -Dynamic Time Warping

HMM -Hidden Markov Model

НЧ – низькочастотний сигнал

SWOT- матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities)

ВСТУП

Найважливішим засобом спілкування є мова. Цей процес, у свою чергу, передбачає створення, сприйняття та розуміння певних мовних конструктів.

У світі інформаційних технологій голосове керування створює новий спосіб взаємодії з функціями різних пристроїв. Ця опція зручна в деяких ситуаціях, наприклад, коли ви не хочете шукати інформацію на своєму смартфоні або коли вам потрібно отримати доступ до певної функції пристрою на відстані (наприклад, якщо вам потрібно зробити фото). Для створення такої технології необхідно було вирішити таке завдання, як розпізнавання мови.

В останні роки біометричні технології все частіше використовуються для ідентифікації людей. Вони використовуються в системах контролю доступу, при здійсненні фінансових операцій, при запиті конфіденційної інформації по телефону, в управлінні різними пристроями, в криміналістиці і т. д. Використання біометричних технологій в цих сферах має ряд істотних переваг перед традиційними методами ідентифікації. (наприклад, за допомогою пароля). Серед таких переваг, перш за все, висока надійність ідентифікації та зручність використання для людини.

Як ідентифікаційні параметри в біометричних технологіях використовуються фізіологічні та поведінкові характеристики людини. До таких ознак належать відбитки пальців, голос, райдужна оболонка ока, обличчя людини, почерк тощо.

В даний час найпоширенішими біометричними ознаками людини є відбитки пальців і райдужна оболонка ока. У той же час голос не так широко використовується, хоча він має ряд суттєвих переваг, наприклад, легкість зняття біометричного параметра (потрібен тільки стандартний мікрофон), а також зручність використання.

Зараз у світі існує багато компаній, які займаються розробкою систем розпізнавання голосу. У цій сфері є певні успіхи (ймовірність помилки ідентифікації 1-3%). Однак існуючі рішення мають ряд недоліків.

Алгоритми досить складні та вимагають великих обчислювальних ресурсів, що обмежує сферу їх застосування лише високопродуктивними комп'ютерами (час ідентифікації 3-5 секунд при довжині фрази 3 секунди на комп'ютері з частотою процесора 1,60 ГГц і кількістю ОЗУ 1 Гб).

Крім того, всі системи не мають можливості адаптувати алгоритми до різних умов використання (рівень шуму, голосові характеристики конкретної людини, поріг помилки тощо).

Крім того, жоден розробник не надає коштів на тестування розробленої ним системи голосової ідентифікації, а специфіка умов застосування може істотно вплинути на якість алгоритму.

Більшість алгоритмів не враховують текстовий зміст вимовленої фрази (фонематичний компонент), підкреслюючи лише індивідуальні характеристики голосу, що значно знижує достовірність ідентифікації.

Враховуючи вищевикладене, ставиться завдання розробити нову модель ідентифікації голосового повідомлення за фонематичною складовою та індивідуальними особливостями голосу, позбавлену зазначених недоліків, а також набір програм, що реалізують цю модель та дозволяють її реалізувати. підлягає перевірці.

Метою даного проекту є розробка математичної моделі ідентифікації голосового повідомлення на основі фонематичного компоненту та індивідуальних особливостей голосу та розробка комплексу програм, що реалізують цю модель. Цільовою групою користувачів цього програмного забезпечення є системи контролю доступу, при проведенні фінансових операцій, при запиті конфіденційної інформації по телефону, при управлінні різними пристроями, у криміналістиці тощо.

Виходячи з поставлених цілей в роботі вирішуються наступні завдання:

– аналіз математичних методів, які можуть бути використані для вирішення задачі ідентифікації голосових повідомлень;

– розробка математичної моделі ідентифікації голосових повідомлень за фонематичним компонентом та індивідуальними особливостями голосу;

– програмна реалізація розробленої моделі ідентифікації голосових повідомлень;

– оцінка впливу значень різних параметрів (параметрів моделі, якими вона коригується) розробленої моделі на якість ідентифікації;

– оцінка впливу різних вимовлених фраз на якість ідентифікації.

Методи дослідження були запозичені з наступних напрямків:

– цифрова обробка сигналів;

– коливання і хвилі;

– математичний аналіз;

– математичне моделювання;

– Чисельні методи;

– Теорія ймовірностей і математична статистика;

– теорія мови програмування;

– теорія побудови баз даних.

Науковою новизною атестаційної роботи є результати, отримані під час вирішення завдань:

– модель ідентифікації вокал повідомлення на фонематичний компонент та індивідуальні особливості голосу;

– спосіб розбиття голосового повідомлення на фонеми;

– спосіб обробки фонем для їх порівняння;

– метод матриця аналіз порівняння фонема голосові повідомлення;

Розроблений метод може бути використаний для голосової ідентифікації в різних областях.

1 АНАЛІЗ ТЕМАТИЧНОЇ ОБЛАСТІ ТА ПОСТАНОВКИ ПРОБЛЕМИ

1.1 Проблема розпізнавання вербальної інформації

Розпізнавання мовлення — це процес перетворення мовного сигналу в цифрову інформацію (наприклад, текстові дані).

Кожна людина має свої особливі голосові якості, які визначаються індивідуальною будовою її голосового апарату. Слухаючи розмову, людина може підсвідомо ідентифікувати голоси інших людей, але розвиток автоматичного дискримінатора мовлення пов'язаний із значними труднощами. Завдання розпізнавання людського голосу полягає у виділенні людської мови з вхідного аудіопотоку, її класифікації та розпізнаванні. При цьому зазвичай вирішуються дві підзадачі: розпізнавання та верифікація мовця. Щоб подолати ці підпроблеми, можна визначити метод розрахунку ступеня подібності між зразком і еталонними сигналами. Ступінь подібності між еталонним зразком і тестовим зразком можна розрахувати за допомогою визначеної міри відстані або за допомогою ймовірнісних критеріїв [2].

Алгоритм ідентифікація спікерти можеш також визначити як залежить від тексту і самостійний текст. Якщо алгоритм ідентифікації мови є текстозалежним, він може використовувати як готові фрази, так і фрази, згенеровані системою розпізнавання. Текстовонезалежні системи необхідні для довільної обробки мови.

З 1980-х років найсерйознішою проблемою розпізнавання мовлення була наявність перешкод. Системи ефективно працюють за ідеальних умов запису, але в той же час не можуть впоратися з фоновим шумом, таким як звук із сусідньої кімнати. Штучні звуки відрізнити цілком можливо, але людський голос, який ми повинні розпізнати, важко відрізнити від голосу того, хто розмовляє поруч. Проблема завадостійкості ще не вирішена [3].

При створенні системи розпізнавання мовлення слід вибрати, який рівень абстракції адекватний для поставленої задачі, які параметри звукової хвилі будуть

розпізнавати та розпізнавати ці параметри. Розглянемо основні відмінності в структурі та процесі роботи різних систем розпізнавання мови.

Більшість існуючих механізмів можна розділити на чотири основні модулі:

– збирач даних - включає прийом вхідного сигналу та попередню обробку, яка може включати автоматичне регулювання посилення, виявлення присутності/відсутності мовлення та визначення інтонації кінця фрази. Модуль також включає вибір мовного сегмента з вхідного мовного сигналу.

– Екстрактор - виконує частотний аналіз сигналу. Потік акустофонетичних даних розбивається на короткі кадри, або вектори, тривалістю близько 10 мс. Як правило, для кожного кадру, який використовує швидке перетворення Фур'є, визначається ряд параметрів. Крім того, багато систем використовують інші замість або разом з цими характеристиками, наприклад спектральні характеристики, а також перші та другі похідні спектральних характеристик [4].

– компаратор - виконує акустичне порівняння: кожен кадр або вектор порівнюється з доступними акусто-фонетичними зразками, що зберігаються в спеціальній базі даних. При цьому можна порівнювати окремі фонеми, слова і навіть фрази. При невеликій кількості слів, які використовує диктор, можна очікувати більшої надійності та швидкості розпізнавання цілих слів, але зі збільшенням словникового запасу швидкість різко падає, а розпізнавання окремих фонем стає оптимальним.

– інтерпретатор вирішує задачу динамічного програмування, щоб знайти найкращий поділ «алфавітного» потоку, отриманого від компаратора, на слова та фрази. Залежно від обсягу використовуваного словника та застосовних правил синтаксису використовуються різні стратегії пошуку та перевірки.

Системи розпізнавання мовлення можна класифікувати відповідно до:

– розподіл (системи диктування, системи команд);

– тип мови (цілісна чи окрема мова);

– розмір словника (обмежений набір слів, великий словник);

– диктор (диктор-залежні та диктор-незалежні системи);

– механізм роботи (найпростіші (кореляційні) детектори, експертні системи з різними методами створення та обробки бази знань, імовірнісні мережеві моделі прийняття рішень, у тому числі нейронні мережі);

– використовуваний алгоритм (нейронні мережі, приховані марковські моделі, динамічне програмування);

– тип структурної одиниці (фрази, слова, фонеми, дифони, алофони);

– принцип виділення структурних одиниць (розпізнавання образів, виділення лексичних елементів).

На рисунку 1.1 наведено класифікацію систем розпізнавання мовлення.

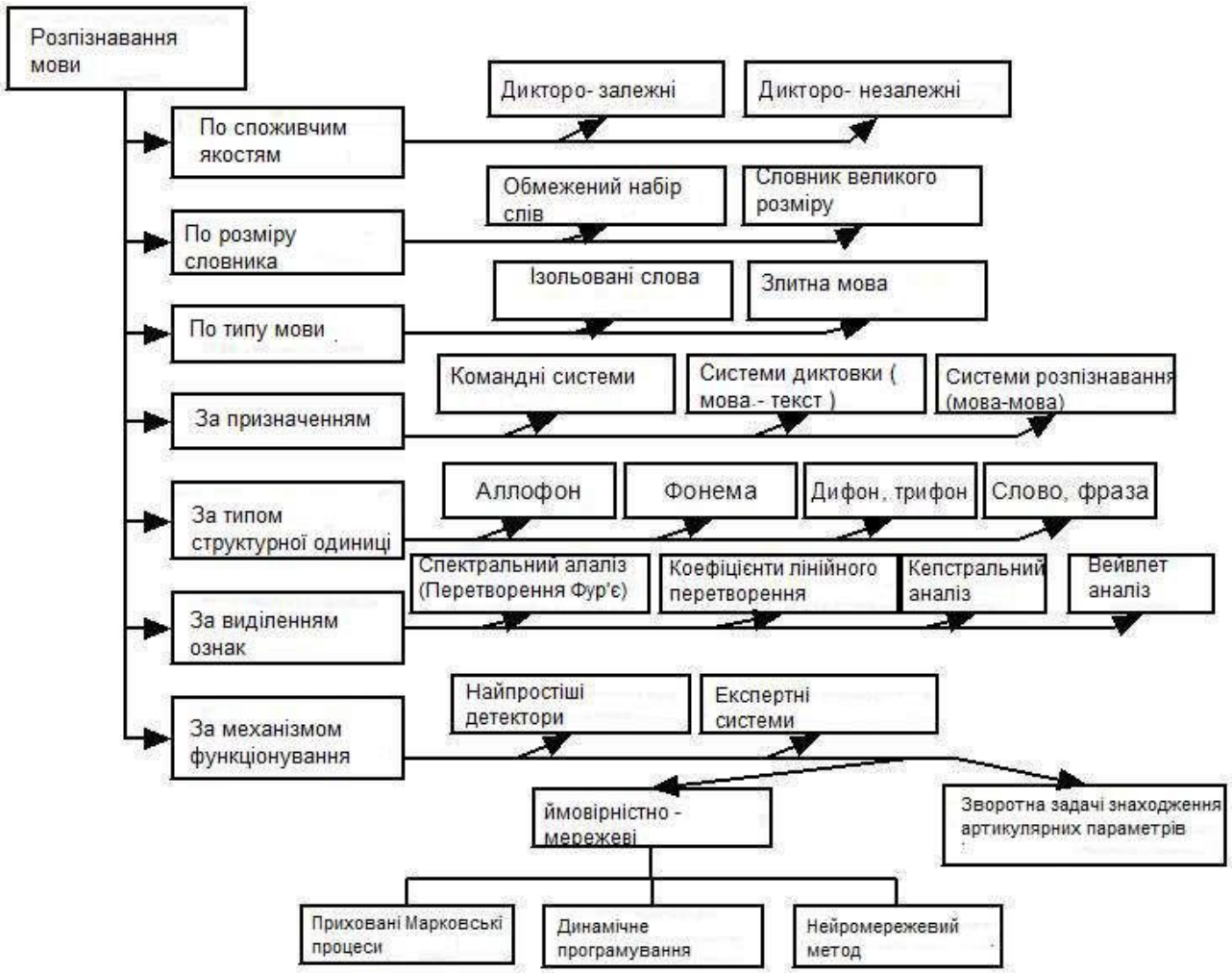

Рисунок 1.1 – Класифікація систем розпізнавання мовлення

За типом структурного підрозділу. При аналізі мови за основну одиницю можна вибрати окремі слова або частини вимовлених слів, наприклад фонеми, дифон або трифон, алофон. Залежно від виділеної структурної частини змінюється структура, універсальність і складність словника.

За вибором персонажів. Сама послідовність вимірювань тиску звукової хвилі є надмірно надлишковою для систем розпізнавання звуку та містить багато зайвої інформації, яка є непотрібною або навіть шкідливою для розпізнавання. Таким чином, щоб представити мовний сигнал, необхідно виділити з нього всі параметри, які адекватно представляють цей сигнал для розпізнавання.

За механізмом функціонування. У сучасних системах широко використовуються різноманітні підходи до механізму роботи систем розпізнавання. Імовірнісний підхід полягає в тому, що мовний сигнал ділиться на певні частини (кадри або за фонетичною ознакою) і робиться ймовірнісне судження щодо того, який елемент розпізнаного словника є цією частиною та/або всім вхідним сигналом. Підхід, заснований на розв’язанні зворотної задачі звукового синтезу, базується на тому, що характер руху артикуляторів мовного тракту визначається вхідним сигналом, а вимовлені фонеми – за допомогою спеціальний словник.

Значення голосової технології для біометрії було доведено знову і знову. Проте лише якісне впровадження систем автоматичного розпізнавання мовців здатне реально впровадити такі технології на практиці. Подібні системи вже існують. Вони використовуються в системах безпеки, банківських технологіях, електронній комерції, практиці правоохоронних органів.

Використання систем розпізнавання мовців є найбільш природним і економічним способом вирішення проблем несанкціонованого доступу до комп'ютера або систем передачі інформації, а також проблем багаторівневого контролю доступу до мереж або інформаційних ресурсів.

Системи розпізнавання мовців можуть вирішувати два завдання: ідентифікувати особу із заданого обмеженого списку осіб (ідентифікація особи) або підтвердити особу мовця (верифікація особи). Ідентифікація і

верифікація голосової особистості є напрямками розвитку технології обробки мовлення.

В даний час розпізнавання мови зводиться до вирішення трьох типів завдань: -

розпізнавання окремо вимовлених слів (використовуваних для мови)

комп'ютерне керування);

– злите розпізнавання мовлення (призначене для перетворення природної людської мови в текст);

– ідентифікація шаблону мовлення (використовується з метою безпеки).

У процесі реєстрації користувача запам'ятовуються характеристики його голосу і т.зв модель мовлення. Під час тестування запропонований зразок мовлення порівнюється з моделлю мовлення користувача, яка запам’яталася, а також з моделлю «самозванця», складеною з голосів багатьох інших людей. Якщо результат порівняння позитивний для першого випадку і негативний для другого, тест вважається успішним.

На рисунку 1.2 показано напрями розвитку технології розпізнавання мовців.

Рисунок 1.2 – Тенденції розвитку технології розпізнавання мовців

1.2 Огляд підходів і технологій голосової ідентифікації

У цьому розділі розглядаються основні існуючі рішення завдання автоматичної ідентифікації диктора за голосом. Незважаючи на те, що методи багато в чому відрізняються, загалом можна виділити такі основні етапи, властиві ко-

жному з цих методів:

1. Вилучення ознак із вхідного мовного сигналу.

2. Побудова моделі (шаблону) диктора на основі отриманих на попередній крок векторів ознак.

Процес визначення диктора, зареєстрованого в системі, вхідному мовному сигналу у всіх аналізованих методах полягає в пошуку найбільш підходящої збереженої моделі на основі будь-яких критеріїв.

1.2.1 Dynamic Time Warping

Dynamic Time Warping (DTW) – метод динамічного програмування, що дозволяє знайти близькість між двома послідовності вимірів за деякий проміжок часу. У загальному випадку ці послідовності можуть бути різної довжини, і вимірювання можуть проводитись з різною швидкістю [14].

Як збережена модель у цьому методі виступає послідовність векторів ознак вхідного мовного сигналу навчальної вибірки $Q=\{q_1,\ldots,q_n\}$. Нехай $C=\{c_1,\ldots,c_m\}$– послідовність векторів ознак вхідного мовного сигналу тестової вибірки. Також вводяться поняття матриці вирівнювання двох послідовностей M_{m*n}, в позиції (i,j) якої міститься значення вирівнювання між елементами c_i і q_j послідовностей C і Q відповідно, і набору індексів суміжних елементів цієї матриці $W=\{w_1,\ldots,w_T\}$, Що визначає відповідність між елементами зіставних послідовностей. При цьому елементи набору мають задовольняти такі умови:

1. $w_1=(1,1)$, $w_T=(m,n)$

2. Якщо $w_{t-1}=(a',b')$, то $w_t = (a,b)$, де $a-a' \leq 1$, $b-b' \leq 1$

Метою алгоритму DTW є знаходження такого набору W, що відповідає умовам 1 і 2, при якому сумарне спотворення послідовності C відносно послідовності Q було б мінімальним, тобто:

$$DTW(Q,C) = \min\left\{\frac{1}{T}\sqrt{\sum_{t=1}^{T} M(w_t)}\right\} \quad (1.1)$$

Значення цього виразу і визначатиме міру близькості послідовностей та . Для знаходження значення застосовується метод динамічного програмування, де на кожному кроці обчислюється значення за такою формулою:

$$M(i,j) = d(i,j) + \min\{M(i-1,j-1), M(i-1,j), M(i,j-1)\} \quad (1.2)$$

При цьому M(0,0)=0 , а всі інші елементи стовпця та рядки з індексом 0 ініціалізуються значенням ∞. d(i,j) визначає евклідово відстань між елементами c_i та q_j . Результатом роботи алгоритму є значення DTW(Q,C)=M(m,n).

Для визначення диктора обчислюється значення DTW(Q_i ,C), для кожного збережений шаблон. Значення i, у якому досягається мінімум, визначає номер диктора, зразок голосу якого найближчий до зразком вхідного мовного сигналу. Основною перевагою алгоритму DTW є простота реалізації. Тим не менш, даний алгоритм не застосовується для вирішення задачі текстонезалежної ідентифікації диктора.

1.2.2 Hidden Markov Model

Hidden Markov Model (HMM) – статистична модель, яка може використовуватися для вирішення задачі класифікації прихованих параметрів на основі спостережуваних. HMM являє собою кінцевий автомат, якому переходи між станами здійснюються з деякою ймовірністю, і задано стартовий стан, з якого починається процес. Через дискретні моменти часу може здійснюватися перехід у нові стану. При цьому кожному прихованому стану із заданою ймовірністю відповідає стан, що спостерігається. Крім того, поточний стан автомата залежить тільки від кінцевого числа попередніх, а закон зміни станів не змінюється у часі [15]. У цій роботі розглядається випадок, коли поточний стан залежить від попереднього (модель першого порядку).

Формально HMM визначається такими параметрами:

- безліч прихованих станів Q={q_0,…,q_n}, де q_0– початковий стан q_{end} – кінцевий стан;
- безліч спостережень O={o_1,…, o_m};
- вихідний розподіл станів π = {π_i}, 1 ≤ I ≤ N, яке визначає можливість розпочати роботу у стані i;
- матриця ймовірностей переходів між прихованими станами A_{NxN}:

$$A(i,j) = a_{ij} = P(q_i, q_j),\ 1 \le i,j \le N;$$

- матриця ймовірностей спостережень B_{NxM}: B(i,j)=b_{ij}=P(o_j|q_i), 1 ≤ i ≤ N,

  1 ≤ j ≤ M.

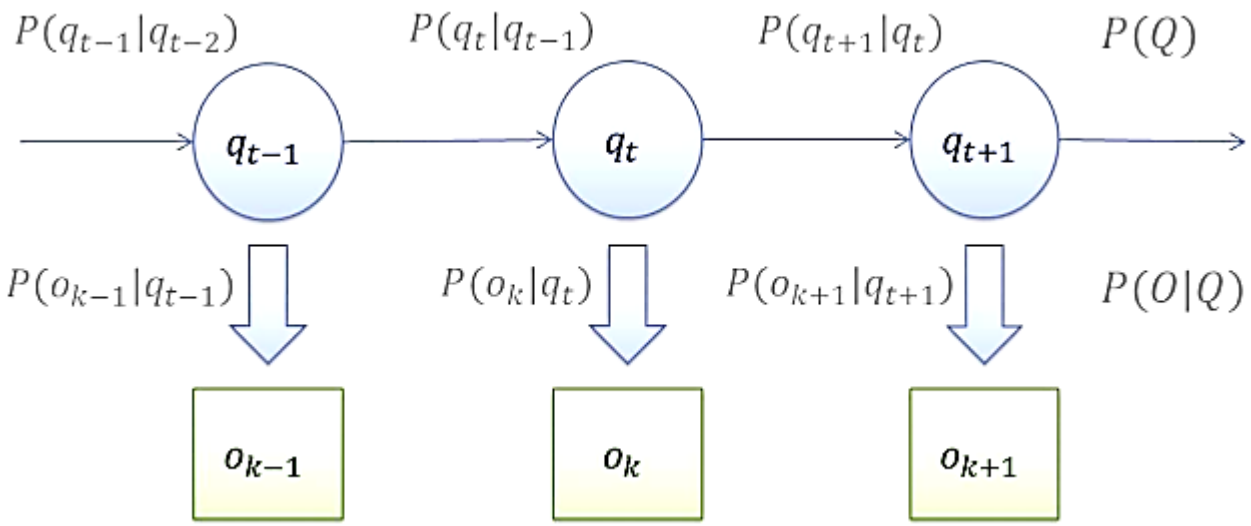

Рисунок 1.3 - Hidden Markov Model

Виділяються три основні завдання, які вирішуються за допомогою HMM:

Завдання 1. Обчислення ймовірності послідовності спостережень.

Потрібно визначити можливість появи заданої послідовності спостережень. Для цього використовується наступний алгоритм:

$$\alpha_0(i) = \alpha_{0i}, 1 \le i \le N \qquad (1.3)$$

$$\alpha_j(r) = \sum_{i=1}^{N} \alpha_i(r-1) a_{ij} b_{jt}, 1 \le j \le N, 1 \le r \le R, \qquad (1.4)$$

$\alpha_j(r)$- ймовірність того, що на r-му етапі модель виявиться в стані j. Тоді ймовірність визначається за формулою:

$$P(O|A,B) = \sum_{i=1}^{N} \alpha_i(R) \quad (1.5)$$

Завдання 2. Знаходження найбільш правдоподібної послідовності прихованих станів для послідовності, що спостерігається.

Потрібно знайти найбільш правдоподібну послідовність прихованих станів $Q=\{q_1,\ldots,q_R\}$ для заданої послідовності спостережень $O=\{o_1,\ldots,o_R\}$, при якій досягається maxP(O|Q) . Це завдання вирішується з допомогою алгоритму, подібного до алгоритму з попереднього пункту з лише різницею, що у кожному кроці запам'ятовується стан і, у якому $\alpha_j(r)$ набуває найбільшого значення. У результаті вибирається 12 послідовність станів, на яку приймає найбільше значення. Цей алгоритм називається алгоритмом Вітербі [16].

Завдання 3. Навчання параметрів моделі за заданою послідовності спостережень та безлічі прихованих станів.

Потрібно обчислити для заданої моделі матриці та . Для вирішення даної задачі застосовується алгоритм Баума-Велша. Для завдання розпізнавання диктора прихованими станами є вектори ознак мовного сигналу з навчальної вибірки, як спостережень – вектори ознак мовного сигналу із тестової вибірки. У В якості збереженої моделі тут виступають матриці A та B. HMM досить прості у розумінні, мають досить високу точність розпізнавання, але, як і DTW, застосовуються в основному для завдань текстозалежної ідентифікації диктора.

1.2.3 Vector Quantization

Завдання векторного квантування з кодовими векторами $W=\{w_1,\ldots,w_n\}$ для послідовності вхідних векторів $C=\{c_1,\ldots,c_m\}$ ставиться як завдання мінімізації спотворення під час заміщення кожного вектора із Q відповідним кодовим вектором.

Моделью диктора в даному методі є безліч кодових. векторів, що отримується з вхідної послідовності векторів ознак мовного сигналу. Для побудови цієї множини вихідна послідовність векторів ознак розбивається на L кластерів, і як кодові вектори беруться їхні центри.

Процес визначення диктора за вхідним мовним сигналом відбувається наступним чином. Для кожного тестового вектора c_i визначаються k найближчих кодових векторів. Нехай k_{ij} - кількість векторів, що належать диктору S_j серед знайдених найближчих.

Тоді ймовірність того, що вектор c_i належить диктору S_j, визначається формулою:

$$P(S_j|c_i) = \frac{k_{ij}}{k} \tag{1.6}$$

Таким чином, послідовність тестових векторів може бути класифікована за правилом:

$$S = \underset{1 \le j \le N}{\operatorname{argmax}} \prod_{i=1}^{L} P(S_j|c_i) \tag{1.7}$$

Для згладжування похибки вимірювання, пов'язаної з близькими до нуля ймовірностями, з урахуванням постійності числа, частіше використовують правило:

$$S = \underset{1 \le j \le N}{\operatorname{argmax}} \sum_{i=1}^{L} P(S_j|c_i) = \underset{1 \le j \le N}{\operatorname{argmax}} \sum_{i=1}^{L} k_{ij} \tag{1.8}$$

Метод векторного квантування простий у реалізації, застосуємо до завдання текстонезалежної ідентифікації диктора, проте не завжди дає високу точність розпізнавання

1.2.4 Gaussian Mixture Model

Модель гаусових сумішей є зваженою сумою M компонент і може бути описана виразом:

$$P(\bar{x}|\lambda) = \sum_{i=1}^{M} p_i b_i(\bar{x}), \tag{1.9}$$

Де $\bar{x}$ - D-вимірний вектор випадкових величин, p_i, $1 \le i \le M$ – ваги компонентів моделі, $b_i(\bar{x})$, $1 \le i \le M$ – функції щільності розподілу складових моделі:

$$b_i(\bar{x}) = \frac{1}{(2\pi)^{D/2}|\sum_i|^{1/2}} exp\left\{-\frac{1}{2}(\bar{x}-\bar{\mu}_i)^T \sum_i^{-1}(\bar{x}-\bar{\mu}_i)\right\}, \tag{1.10}$$

где $\bar{\mu}$ – вектор математического ожидания и $\sum_i$ – ковариационная матрица. При умові:

$$\sum_{i=1}^{M} p_i = 1 \tag{1.11}$$

Повністю модель гаусової суміші визначається векторами математичного очікування, коварійними матрицями та вагами сумішей для кожного компонента моделі:

$$\lambda = \left\{p_i, \bar{\mu}_i, \sum_i\right\}, \; i = 1, \dots M \tag{1.12}$$

При використанні цього методу кожен диктор представляється моделлю гаусових сумішей λ.

Для побудови моделі диктора необхідно оцінити її параметри, які найкраще відповідають розподілу векторів ознак навчального висловлювання. Найбільш популярним та широко методом розв'язання цього завдання є метод оцінки максимальної правдоподібності. Метою оцінки максимального правдоподібності є знаходження параметрів моделі, які максимізують правдоподібність цієї моделі, при заданих даних.

Для послідовності навчальних векторів $X=\{\bar{x}_1,\dots,\bar{x}_T\}$ правдоподібність моделі гаусових сумішей може бути записана у вигляді:

$$P(X|\lambda) = \prod_{t=1}^{T} P(\overline{x_t}|\lambda) \quad (1.13)$$

Цей вираз є нелінійною функцією від параметрів λ, і її безпосереднє обчислення неможливе, тому зазвичай для оцінки параметрів застосовується EM алгоритм [5].

Нехай $S=\{S_1,\ldots,S_N\}$ – група дикторів, які представлені набором моделей гаусових сумішей $\lambda_1,\ldots, \lambda_N$.

При ідентифікації диктора потрібно знайти модель, яка має найбільше значення апостеріорної ймовірності для заданого висловлювання:

$$S = \underset{1\le k\le N}{\operatorname{argmax}} P(\lambda_k|X) = \underset{1\le k\le N}{\operatorname{argmax}} \frac{P(X|\lambda_k)P(\lambda_k)}{P(X)} = \underset{1\le k\le N}{\operatorname{argmax}} P(X|\lambda_k) \quad (1.14)$$

Використовуючи логарифм та незалежність між спостереженнями, система ідентифікації диктора у результаті обчислює:

$$S = \underset{1\le k\le N}{\operatorname{argmax}} \sum_{t=1}^{T} \log P(\overline{x_t}|\,\lambda_k) \quad (1.15)$$

Моделі гаусових сумішей є ефективним алгоритмом. високою точністю розпізнавання. Водночас виникає низка проблем, пов'язаних з вибором числа компонентів моделі та ініціалізацією її початкових властивостей.

1.3 Постановка завдання розробки методу аутентифікації користувача голосом

Завданням, яке вирішується в рамках дипломної роботи, є розробка мобільного додатку аутентифікації користувача за допомогою голосового повідомлення. Цільовою групою користувачів такого програмного засобу є системи розділення доступу при здійсненні фінансових операцій, при аутентифікації на персональних пристроях тощо.

Фахівцям з розпізнавання голосу важливо забезпечити максимальний відсоток правильної верифікації з мінімальними зусиллями користувача програ-

ми. Для цього використовується модуль шумозаглушення та розділення корисних сигналів, акустичні та мовні моделі. Для розробки методу буде розглянуто алгоритм динамічних спотворень. Також необхідно знайти спосіб удосконалити цей алгоритм.

Розроблений метод голосової ідентифікації повинен виконувати наступні завдання:

– оцінка якості мовного сигналу. На цьому етапі визначається рівень спотворень і спотворень;

– оцінка інформації про частини мови, дієслівні форми та статистичні зв’язки між словами;

– голосова ідентифікація;

– проаналізувати результати тестування програмного засобу та визначити, наскільки вихідні дані програми відповідають критеріям ефективності поставленого в рамках роботи завдання.

Введення – це повідомлення голосової пошти максимальною тривалістю п’ять секунд, достатньо одного слова. На виході з'явиться повідомлення про результат ідентифікації - ідентифікований / не ідентифікований.

Результатом сертифікаційної роботи стане математичний метод, за допомогою якого можна ідентифікувати диктора за ключовим словом.

Для аналізу повноти реалізації програмного функціоналу виконайте тести розпізнавання голосу. Будуть виділені голосові сигнали кількох людей. Запис мовлення здійснювався в монорежимі за допомогою вбудованого в комп’ютер мікрофона з частотою дискретизації 16 кГц. Критеріями для проходження тесту є коефіцієнт помилкових прийомів (FAR) і коефіцієнт помилкових відхилень (FRR):

$$FAR = \frac{N_{FAR+}}{N_{FAR\ all}}, \tag{1.16}$$

де FAR – частка помилкових підтвердження користувача; N_{FAR+} – кількість успішних автентифікацій зловмисника; $N_{FAR\ all}$ – загальна кількість спроб пройти автентифікацію зловмисником.

$$FRR = \frac{N_{FRR-}}{N_{FRR\ all}}, \tag{1.17}$$

де FRR – частка помилкових відмов користувачеві; N_{FRR}− – кількість відмов у автентифікації користувача; N_{FARall} – загальна кількість спроб пройти автентифікацію користувачем.

Одним із критеріїв продуктивності системи може бути наступний підхід: чим менше значення FRR, тим краще система для однакових значень FAR Зараз багато компаній (Agnitio, Nuance, Voice Security Systems) розробляють голосові біометричні системи. Більшість передових систем мають 1–3% ймовірність помилки ідентифікації, але ці зміни мають ряд недоліків.

Слід звернути увагу на такі дослідження:

– Agnitio (FAR: 2,1%, FRR: 1,5%);
– Нюанс (FAR: 2,6%, FRR: 1,8%);
– Системи захисту голосу (FAR: 1,1%, FRR: 2,2%);
– VoiceTrust(FAR: 2,5%, FRR: 2,8%);

Слід розуміти, що методика розрахована на повсякденне використання, тобто учасник тестування є користувачем мобільного пристрою або ПК, тому така ситуація накладає низку істотних обмежень.

Перш за все, користувач дуже чутливий до швидкості доступу, тобто немає можливості тривалих тестів і складних процедур.

По-друге, найчастіше кількість користувачів мобільних пристроїв дуже обмежена. Тому актуальність ЗПД дещо слабша, ніж за інших умов.

2 ОПИС ТА АНАЛІЗ ПІДХОДІВ ДО РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ РОЗПІЗНАВАННЯ ГОЛОСОВИХ КОМАНД

2.1 Опис алгоритму розпізнавання голосової команди

На рис. 2.1 зображено загальний алгоритм розпізнавання голосових команд.

Рисунок 2.1 – Загальний алгоритм розпізнавання голосової команди

Цей загальний алгоритм застосовується до всіх інструментів розпізнавання мовлення. Попереднє виправлення, фільтрування та зменшення шуму не є важливими етапами розпізнавання мовлення.

Розглянемо систему верифікації мовців докладніше. Перевірка ідентифікатора голосу визначає, чи хтось є тим, за кого себе видає. Користувач, раніше зареєстрований в системі, називає свій ідентифікатор, яким є реєстраційний номер, пароль або фраза. У разі розпізнавання тексту пароль відомий системі і «пропонує» користувачеві його вимовити. Виглядає погано для пароля, і людина говорить його в мікрофон. Для розпізнавання, незалежного від тексту, промовлений пароль користувача не відповідає контрольному слову, що означає, що користувач може вимовити будь-яке слово чи фразу як пароль. Система перевірки приймає мовний сигнал, обробляє його і вирішує прийняти чи відхилити запропонований ідентифікатор користувача.

Система може повідомити користувача про недостатню відповідність його голосу існуючому стандарту та попросити прокоментувати додаткову інформацію для прийняття остаточного рішення.

Користувач промовляє запропонований системою номер у мікрофон, щоб система могла перевірити, чи відповідає його голос стандарту, який зберігається в базі даних системи. Як правило, існує компроміс між точністю розпізнавання голосу та розміром вибірки мовлення, тобто чим довша вибірка мовлення, тим вища точність розпізнавання. Крім вашого голосу до мікрофона може доноситися луна і сторонні звуки.

Схема взаємодії людини з системою голосової перевірки ідентифікації наведена на рис. 2.2

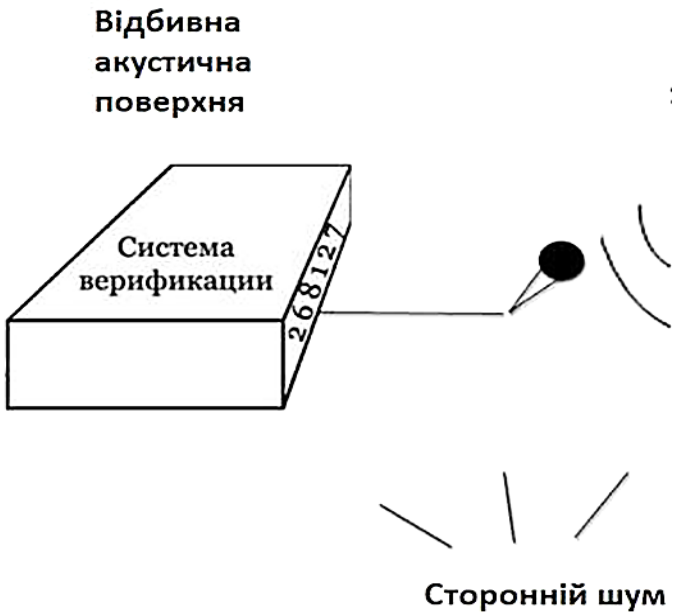

Рисунок 2.2 – Діаграма взаємодії людини з системою перевірки голосової ідентифікації

Є ряд факторів, які можуть сприяти виникненню помилок перевірки та ідентифікація, наприклад:

– неправильне проголошення або прочитання слова чи фрази пароля;

– емоційний стан викладача (стрес, застосування сили для вимови пароля тощо);

– складне акустичне середовище (шум, перешкоди, радіохвилі тощо);

– різні канали зв'язку (використання різних мікрофонів при реєстрації та верифікації диктора);

– простудні захворювання;

– природні зміни голосу.

Деякі з них можна усунути, наприклад, використовуючи кращі мікрофони. На рис 2.3 показано загальний алгоритм перевірки мовця.

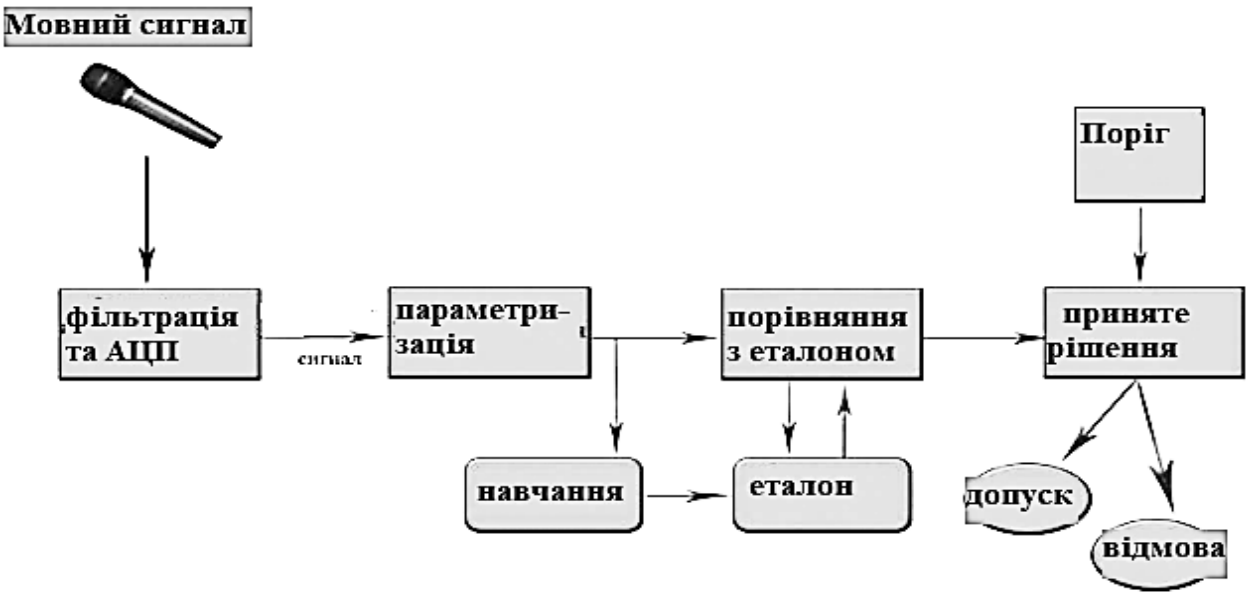

Рисунок 2.3 – Загальний алгоритм перевірки мовця

Під час реєстрації новий користувач вводить свій ідентифікатор, а потім кілька разів повторює ключове слово або фразу, створюючи контрольні показники. Кількість ключових фраз може бути різною для кожного користувача, або вона може бути постійною для всіх користувачів.

Щоб комп’ютер обробив мовний сигнал, звукова хвиля перетворюється в аналоговий, а потім у цифровий.

На етапі виділення голосових характеристик мовний сигнал розбивається на окремі звукові кадри, які потім перетворюються в цифрову модель. Такі моделі називаються

«Голосові відбитки». Щойно отриманий «голосовий відбиток» порівнюється із заздалегідь визначеним зразком. Для розпізнавання особистості мовця найважливішими є найбільш виразні відмінні риси голосу, що дозволить системі з ви-

сокою точністю розпізнавати голос кожного конкретного користувача.

Зрештою, система вирішує, дозволити або заборонити доступ користувача, залежно від того, чи збігається його голос із встановленим стандартом. Якщо система неправильно зіставляє поданий голос зі стандартом, виникає помилка «хибне прийняття» (FA). Якщо система не розпізнала біометричну ознаку, яка відповідає наявному в ній ідеалу, це називається помилкою «хибної відмови» (FR). Помилка підробленого доступу спричиняє порушення системи безпеки, а помилка підробленої відмови знижує зручність використання системи, іноді не розпізнаючи особу з першого разу. Спроба знизити ймовірність появи однієї помилки призводить до більш частого виникнення іншої, тому в залежності від вимог до системи вибирається певний компроміс, тобто встановлюється поріг прийняття рішення.

Процес перевірки голосової ідентичності складається з 5 етапів: отримання мовного сигналу, параметризація, тобто вибір характеристик голосу, порівняння отриманого зразка голосу з заздалегідь визначеним стандартом, прийняття рішень «прийняти / відхилити», навчання або оновлення еталонна модель.

2.2 Опис алгоритму оцінки якості голосових команд

Обробка мовних сигналів — галузь науки, у якій виконуються фільтрація та придушення шумів, посилення, розділення інформаційного потоку, виділення інформації, кодування, стиснення та пошук мовних сигналів. Воно поширилося в усіх напрямках мовних технологій.

Чіткість голосу - відносна кількість правильно сприйнятих елементів мови (звуків, складів, слів, фраз), виражена у відсотках від загальної кількості переданих елементів.

Якість мовлення – параметр, що характеризує суб'єктивну оцінку звучання мовлення в досліджуваній системі передачі мовлення.

Нормальна швидкість мовлення – це мовлення зі швидкістю, при якій се-

редня тривалість контрольної фрази становить 2,4 секунди.

Прискорений темп - виголошення мови зі швидкістю, при якій середня тривалість контрольної фрази становить 1,5 - 6 с.

Розпізнавання голосу мовця – це здатність слухача ідентифікувати звук голосу з конкретною людиною, яка була раніше відома слухачеві.

Смислова зрозумілість є показником ступеня правильності інформаційного відтворення змісту мовлення.

Інтегральна якість – це показник, що характеризує загальне враження слухача про мову, що сприймається [6].

Напрям обробки мовних команд в системах голосового управління включає наступні завдання:

– реєстрація;
– початкова корекція;
– фільтрація і шумозаглушення;
– сегментація на кадри;
– сегментація «сигнал/пауза»;
– сегментація «тонна / НІ»;
– визначення інформаційних параметрів.

Фільтрація та придушення шумів – етап обробки мовних команд, що дозволяє підвищити розбірливість і зменшити частку шуму, викликаного як акустичними, так і технологічними причинами.

Шум - це невпорядковані коливання різної фізичної природи, що відрізняються складністю часових і спектральних структур. Що стосується мовних сигналів, то шум – це сукупність аперіодичних звуків різної інтенсивності та частоти, які змінюють інформаційні параметри сигналу.

Шум від взаємодії з корисним мовним сигналом поділяється на адитивний і мультиплікативний. До корисного сигналу додається додатковий шум, який створює невелику похибку. Мультиплікативні шуми множаться на корисний сигнал і вносять найбільшу похибку - вони можуть змінити інформаційні параметри голосових команд.

Рисунок 2.4 – Мовні технології

У найзагальнішому вигляді поєднання сигналу і шуму виглядає так:

$$S\,(t) = (kc\,(t) + k\text{sh}\,(t)) * e\,(t) + n\,(t), \qquad (2.1)$$

де kc (t) – коефіцієнт, що характеризує корисний мовний сигнал; ksh (t) – коефіцієнт, що характеризує мультиплікативний шум; e(t) – це додатковий шум

Акустичні мовні сигнали часто спотворюються додатковими перешкодами, що значно знижує ефективність систем перевірки диктора. Загалом ці додаткові збурення можна розділити на дві великі групи: стаціонарні, присутні по всьому сигналу (наприклад, добре відомий білий і рожевий шум), і нестаціонарні, короткочасні, що виникають в окремих ділянках сигналу.

При наявності перешкод другої групи вхідні сигнали рідко спотворюються повністю. Злегка спотворені ділянки сигналу чергуються з областями, сильно спотвореними різними типами імпульсного шуму: обрізання, коротка тривалість електронаведення, перевантаження і т. д. Саме ці нестаціонарні збурення і спотворення мають найбільший негативний вплив. Тому, використовуючи детектори, здатні з високою ймовірністю виявляти цей тип перешкод і спотворень на етапі попередньої обробки (з метою їх подальшого придушення або виключення з аналізу), якість систем обробки мови може бути значно покращена [7] .

Співвідношення потужності сигналу і шуму. Це відношення називається «відношення сигнал/шум» і відіграє важливу роль у задачі фільтрації та придушення шуму. Відношення сигнал/шум виражається в логарифмічних безрозмірних одиницях - децибелах (дБ, дБ):

$$N = 10lgIc\ /\ I\text{sh}, \tag{2.2}$$

де Ic, Ish. - потужність сигналу та шум.

Рисунок 2.5 - Класифікація шуму в мовних сигналах

На рисунку 2.5 показано класифікацію шуму в мовних сигналах. За своїм походженням шуми в мовних командах можна розділити на фізіологічні та антропоген-

ні. Перший тип шуму складається з набору звуків різної інтенсивності і частоти в хаотичних поєднаннях з корисними мовними сигналами.

Генез цього виду шуму безпосередньо пов'язаний з мовленнєвими розладами (порушеннями окремих або комплексних органів артикуляції відділу мовного апарату). Наука, що вивчає порушення мовлення, їх подолання та попередження шляхом корекційного навчання, називається логопедією. Шуми, пов'язані з порушенням мови, включають велику кількість звуків, форма і структура яких безпосередньо пов'язані з типом порушення мови:

– порушення швидкістьі ритм язик сигнали (браділла, тахілалія, спотикання, заїкання);
– порушення голосу (афонія, дисфонія, ринофонія);
– втрата мовних сигналів (афазія).

До антропогенних шумів в орієнтовній інтерпретації відносяться всі інші види звуків, крім фізіологічних. Назва «антропогенні» походить від зв'язку з людиною, іншими словами, це звуки, які виходять від людини і є результатом її дій. Їх також називають промисловими або виробничими шумами. Джерелами антропогенного шуму є транспорт: автомобілі, потяги та літаки, промислові підприємства, будівельно-ремонтні роботи, побутова та офісна техніка тощо.

При постійних параметрах всі шуми можна розділити на стаціонарні та нестаціонарні. Стаціонарний шум — це шум, що характеризується незмінністю середніх параметрів: інтенсивності (потужності), розподілу інтенсивності в спектрі (спектральної щільності), автокореляційної функції. Класичною моделлю стаціонарного шуму є білий шум, спектральні компоненти якого рівномірно розподілені по всьому діапазону задіяних частот.

Основні поширені збої та спотворення, про які йдеться в цій статті, це клацання, перевантаження, короткі звукові сигнали та обрізки.

Існують певні труднощі з виявленням клацань, оскільки короткі сплески, які людина сприймає як «клацання», можуть суттєво відрізнятися як за часом, так і за частотою (малюнок 2.6 показує типи клацань, 1 — це коротке, високочастотне «класичне» клацання; 2 - низькочастотне клацання; 3 - клацання з короткими ко-

ливаннями; 4 - довге клацання з шумовим заповненням).

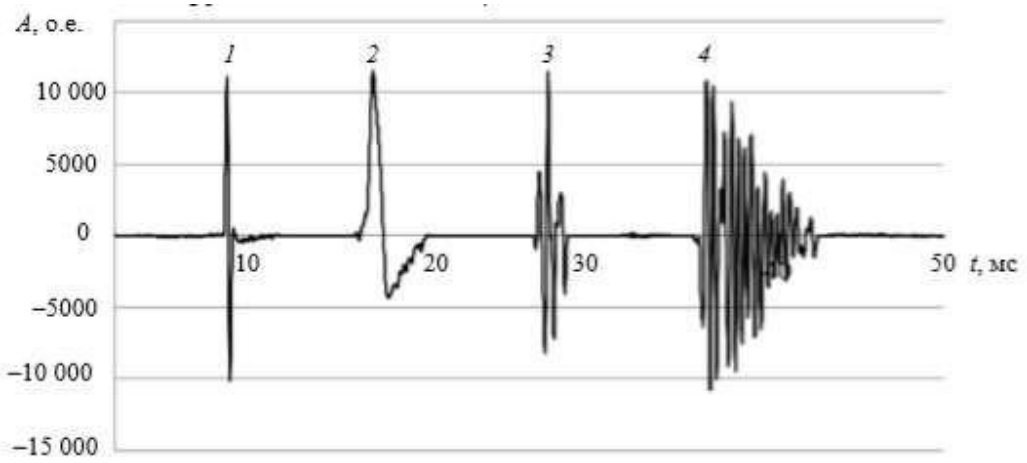

Рисунок 2.6 - Типи кліків

Наприклад, таким чином добре розпізнається короткий високочастотний клацання. Аналізований сигнал x (i), де i - індекс дискретного часу, спочатку пропускається через високочастотний (HF) фільтр з частотою зрізу близько 2-4 кГц. Потім розраховується перша різниця d (i) = y (i) - y (i - 1), де y (i) - сигнал на виході фільтра, потім його абсолютне значення порівнюється з граничним значенням.

На жаль, цей спосіб не працює на низькочастотних (НЧ) клацаннях (крива 2), тому що, по-перше, основна частина їх енергії зосереджена в області низьких частот і «зрізається» фільтром високих частот, а по-друге, значення d (i) клацання цього типу і мовні сигнали відрізняються незначно.

Результати досліджень різних алгоритмів, заснованих на методах лінійного прогнозування та моделях авторегресії, показали їх високу обчислювальну складність, тому був розроблений більш простий алгоритм виявлення різних типів кліків.

Розроблений алгоритм включає наступні кроки:

а) Довжина вікна аналізу (t0, t3) вибирається таким чином, щоб виконувалася умова t3 - t0 = KLc, де Lc - розрахункова тривалість кліка, а K - коефіцієнт масштабування від 10 до 100. .

б) Вікно розділене на три частини, довжина центральної частини вибирається пропорційно очікуваній довжині клацання, t1 - t0 = t3 - t2.

с) Початкове значення Vc, яке далі буде порівнюватися з граничним значенням, розраховується як:

$$V_c\,(t_{center}) = \frac{2(t_1 - t_0)}{t_2 - t_1} \frac{\sum_{t=t_1}^{t_2} x^2(t)}{\sum_{t=t_0}^{t_1} x^2(t) + \sum_{t=t_2}^{t_3} x^2(t)}\ , \qquad (2.3)$$

де x (t) – аналізований сигнал; центр t = 0,5 (t3 + t0) є центром інтервалу [t3, t0].

Неважко зрозуміти, що Vc в (2.3) – це відношення потужностей сигналу на різних ділянках, нормоване таким чином, що у випадку стаціонарного сигналу (наприклад, білого кольору шум) Vc = 1. Для мовних сигналів отримані значення Vc коливаються від нуля до кількох одиниць.

Значення Vc> 8 вказує на наявність клацання (строго кажучи, конкретна межа залежить від обраної прийнятної ймовірності помилкової тривоги та розміру вікна аналізу та визначається експериментально).

На рисунку 2.7 зображено розроблений алгоритм виявлення клацань (суцільна крива — відрізок аналізованого сигналу з клацанням, пунктирна лінія — вихідне значення алгоритму (помножене на 1000 для відображення на одному графіку з сигналом);

$t0$ - $t3$ – тимчасові мітки меж вікна аналізу).

Зрозуміло, що довжина інтервалу t2-t1 в ідеалі повинна відповідати часу виявлення клацання, чого в реальних умовах досягти важко. У проведених експериментах було встановлено, що якщо це значення знаходиться в діапазоні декількох довжин клацання, результати детектора також цілком прийнятні. В іншому випадку, при значній апріорній невизначеності щодо тривалості очікуваних клацань, необхідно виконати пошук.

В результаті моделювання отримано такі часові параметри детектора: t2 - t1 інтервал 5 мс; t1 - t0 і t3 - t2– 60 мс. З цими значеннями були отримані хороші результати при виявленні типових клацань реальних мовних сигналів. Зверніть увагу, що при виявленні високочастотних коротких клацань корисно попередньо

відфільтрувати високочастотний фільтр з частотою зрізу 2-4 кГц.

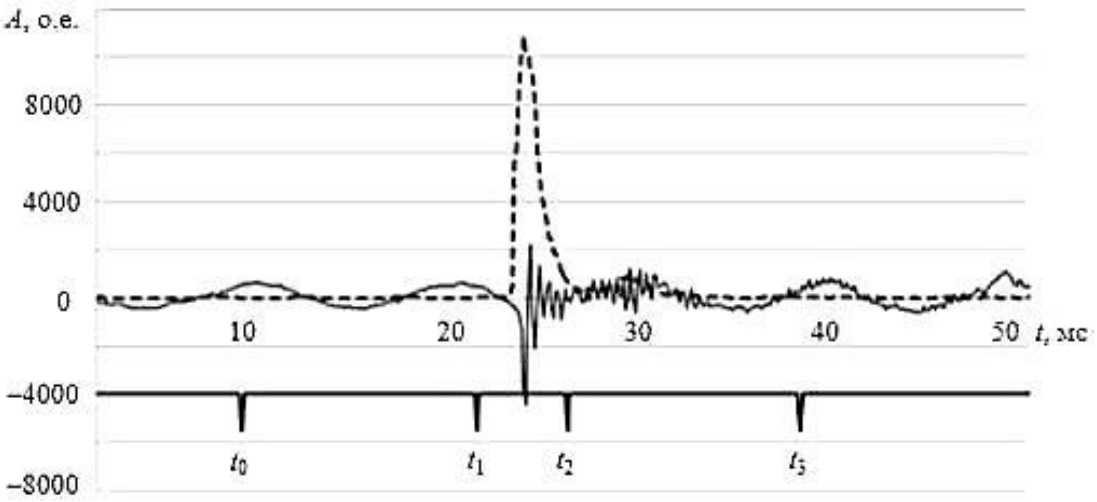

Рисунок 2.7 – Алгоритм виявлення клацань

Короткі (1-2 відліки) стрибки сигналу, імпульси або серії однотипних імпульсів великої амплітуди, викликані зміною знака сигналу під час т.зв. «Цілочисельні переповнення» називаються перевантаженнями. Причини перевантажень наступні. На практиці найбільш часто використовуваним типом квантування при перекладі звукових сигналів в цифрову форму є 16-розрядне квантування. У цьому типі квантування кожен сигнал є двобайтовим цілим числом у форматі "signed short int" (стандарт ANSI), що означає, що амплітуда підрахунку коливається від -32 768 до 32 767. У той же час обробка сигналу може виконуватися, наприклад, у режимах «довгий», «плаваючий» або «подвійний». При цьому, якщо отримане після обробки число виходить за межі діапазону [-32 768 32 767], то при його простому перетворенні в тип "

число -32769 - в 32 767 і т.д.

Загальні вирази для результату можна записати так:

$$\text{якщо } (x > 32\,767), \text{ то } y = (x \bmod 32767) - 32\,768),$$

$$\text{якщо } (x < -32768), \text{ то } y = -(|x| \bmod 32\,768) + 32768, \qquad (2.4)$$

де x — число для перетворення, y — результат перетворення, mod — операція обчислення за модулем.

На слух одноразове перевантаження сприймається як високочастотне клацання, а серія таких клацань як різке, гучне клацання значно погіршує як розбірливість мови, так і продуктивність систем обробки мови.

На малюнку 2.8 показаний типовий приклад перевантаження, яке виникло під час перетворення значення подвійного формату (час перевантаження 6,68 мс, значення x = 56 981) у багатобайтове коротке ціле зі знаком.

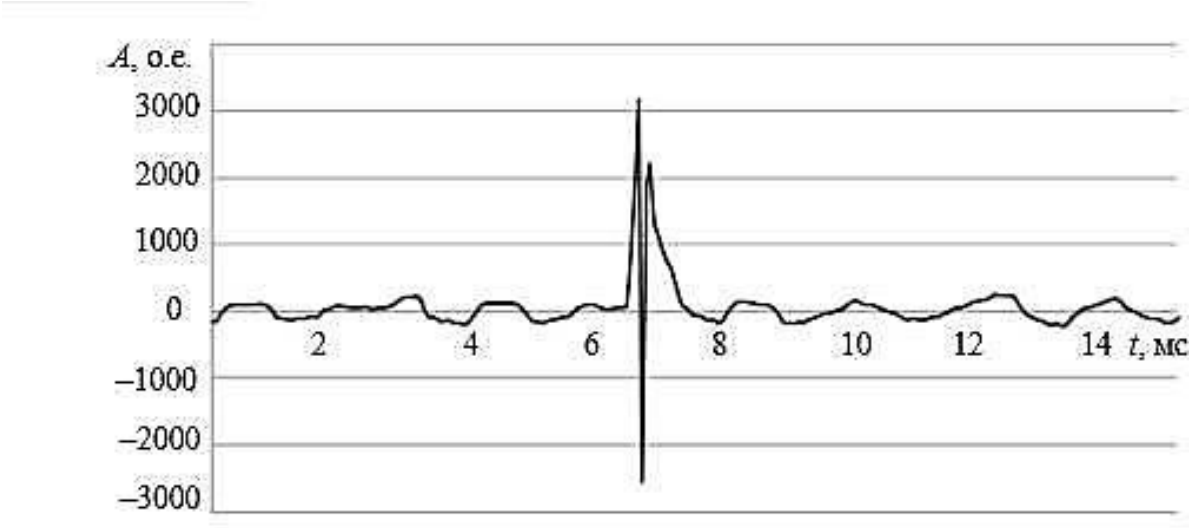

Рисунок 2.8 - Приклад перевантаження

Одиночне перевантаження (на відміну від серії) може бути успішно виявлено за допомогою високочастотного детектора клацань. Однак, використовуючи першу відмінність (яка раніше була описана як неефективна при виявленні клацань LF другого типу), можна створити алгоритм, який виявляє як одиничні, так і багаторазові перевантаження. Справа в тому, що «перевертання» знака викликає сильні різкі стрибки амплітуди за один відлік, часто сумірні з динамічним діапазоном сигналу. У цьому випадку коефіцієнт розраховується наступним чином:

$$d(i) = \frac{|x(i) - x(i-1)|}{A_{max} - A_{min}}. \quad (2.5)$$

де Amax і Amin — максимальна і мінімальна амплітуди сигналу, розраховані для всієї вибірки. Теоретично 0 ≤ d (i) ≤ 1, але чистою мовою, без перевантажень, значення d (i) зазвичай набагато менше одиниці. Алгоритм виявлення перевант-

таження:

a) Вибрано порогове значення Td, наприклад 0,7.

b) Його максимальні значення max A і мінімальні min A обчислюються для всієї вибірки сигналу.

c) Для кожного показання сигналу x (i), i = 1, N - 1 (N повна довжина сигналу), коефіцієнт d (i) розраховується за формулою (2.4). Виконується порівняння d (i) із попередньо вибраним порогом, і у випадку d (i)> Td приймається рішення про наявність перевантаження.

Короткі тональні сигнали - це добре відомі телефонні сигнали, які зазвичай складаються з однієї або двох гармонік тривалістю близько однієї секунди. Характерною особливістю таких сигналів є високий рівень і стабільність частоти гармонік. Таким чином, переважна більшість алгоритмів визначення тону використовують аналіз спектрів потужності (або модулів спектру потужності) сигналів.

Слід зазначити, що тональні сигнали без зовнішнього шуму або змішані з шумовим шумом низької потужності також можуть бути успішно виявлені детектором кліпів на основі аналізу гістограми.

Алгоритм виявлення:

a) Вибрано значення M - довжина сегмента сигналу для розрахунку спектра потужності.

b) Для кожного сегмента сигналу довжиною M розраховується модуль миттєвого спектра потужності S (m), де m = 0, M / 2 - дискретна частота.

c) Для всіх $m = 0, M / 2 =$ – спектральний максимум Smax.

d) Розраховується пороговий рівень$Ts = Ts0Smax$.

f) Для всіх m = 0, M / 2, цільове значення Ks - кількість розраховується спектральні показання вище рівня Ts, тобто: $Ks = \sum^{M/2}_{m=0} k_s$, де

$$k_s = \begin{cases} 1 \; if \; S(m) \geq T_s \\ 0 \; if \; S(m) < T_s \end{cases} \qquad (2.6)$$

e) Проводиться порівняння: якщо $3 \leq Ks$, то приймається рішення про наявність тональної складової в досліджуваному фрагменті сигналу. Алгоритм оцінки сталос-

ті амплітуди спектральних піків використовує той факт, що амплітуда тональної складової незначно змінюється в сусідніх сегментах сигналу. Цей алгоритм порівнює максимуми модуля двох спектрів потужності сусідні сегменти сигналу $maxS_j$ і $maxS_{j+1}$ (де j індекс сегмента) і обчислюються відносна різниця:

$$D_S = \frac{S_{max}^{j+1} - S_{max}^{j}}{S_{max}^{j}} \quad (2.7)$$

Порівняння значення Ds із заданим порогом Td дає очікуваний результат: якщо Ds <Td, то приймається рішення про наявність тональної складової в j-му фрагменті сигналу.

Вирізка - спотворення форми сигнал Що відбувається в перевантаження підсилювача і коли вихідна напруга підсилювача виходить за межі динамічного діапазону. На осцилографі обмеження зазвичай відображається як обмеження амплітуди сигналу. На слух кліпсування сприймається як поява надмірної звучності, «металевого» звуку і може значно знизити якість обробки мови. Алгоритм виявлення кліпінгу на основі аналізу гістограми сигналу наведено в [8]. У бібліотеці використовуються наведені вище алгоритми виявлення шуму Sphinx4.

2.3 Опис алгоритму перевірки диктора

В даний час використовуються різні алгоритми створення таких систем. Наприклад, алгоритм Dynamic Time Warping використовується в текстовозалежних системах. Цей алгоритм використовується для розпізнавання мови, коли дві різні людини вимовили одну фразу і це необхідно з'ясувати, хто саме.Для цього комп'ютер порівнює дві «картки» голосів, відображені на синусоїдальних графіках. Для порівняння достатньо лише двох-трьох слів.

На рисунку 2.9 показаний синусоїдальний графік порівняння двох голосів.

Але бувають випадки, коли ці діаграми дуже схожі у людей. Щоб порівняти ці голоси, комп’ютер «перекосить» часову шкалу одного чи обох графіків для кращого узгодження. Помилки виникають через те, що порівнювані послідовності мають різну довжину, і точка в одному рядку буде трохи вище або трохи нижче

відповідної точки в другому рядку. Отже, алгоритму важко знайти видиме вирівнювання двох рядків. Однак він добре розпізнає окремі слова з обмеженого словникового запасу. Це простий і відкритий для вдосконалення алгоритм, який підходить для використання в телефонах, автомобільних комп'ютерах і системах безпеки.

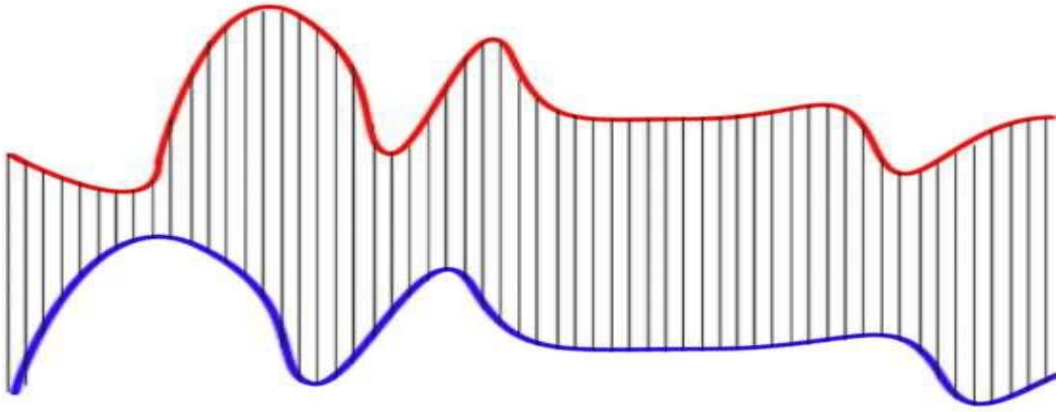

Рисунок 2.9 – Діаграма синуса порівняння двох голосів

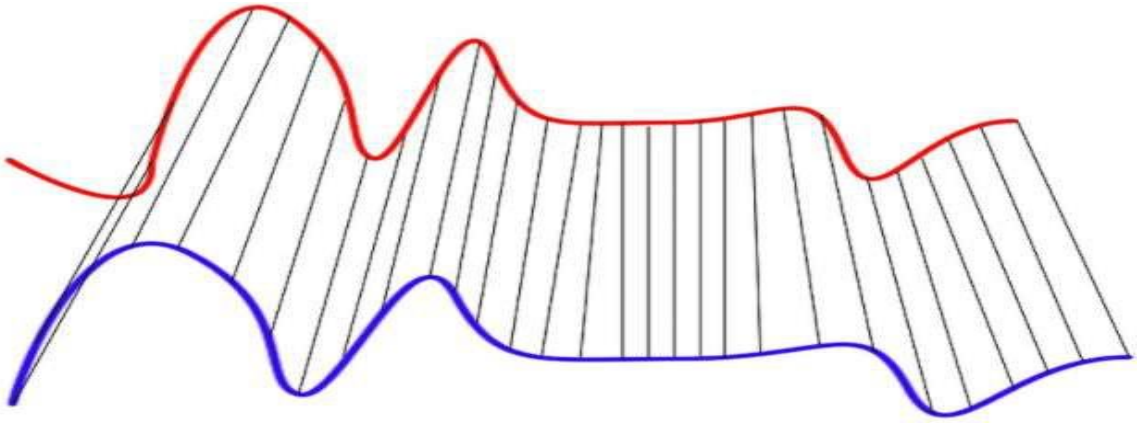

Рисунок 2.10 – Синусоїдальний графік із деформованою шкалою часу

На рис. 2.10 показана синусоїдальна діаграма з деформованою шкалою часу.

Взаємодія між людиною та побутовою технікою значно розшириться, якщо керувати машиною звичайним голосом у реальному часі. Цієї мети можна досягти шляхом вирішення такої задачі: необхідно розпізнати конкретне ключове слово в природному мовному потоці одного диктора. Наявність ключ-слова дає змогу створити Умовний попереджувальний сигнал, який при подальшому застосуванні може викликати певні дії. Ключове слово можна заздалегідь ввести в словник.

Корпорація Google, яка пропонує лінгвістичне введення під час пошуку в Інтернеті, досягла хороших успіхів у вирішенні подібних проблем, але деталі технології невідомі, тому існує потреба в подальшому розвитку технології розпізнавання мовлення. Відповідно до класифікації теорії розпізнавання, сформульоване завдання відноситься до типу розпізнавання, залежного від мовця, з обмеженим словниковим запасом. Для розпізнавання мовлення акустичний (мовної) сигнал за допомогою сприймають (мікрофона) і оцифровують (діскретізіруется) пристроїв і машинної обробки фіксується і перетворюється в цифрову форму В результаті дискретизації безперервний (аналоговий) сигнал перетворюється в послідовність чисел.

Першим кроком у розпізнаванні мовлення є вибір різних функцій. Потім за допомогою певної стратегії навчання створюються шаблони, з якими в майбутньому будуть порівнюватися невідомі частини мовного сигналу. Традиційні моделі органу слуху людини припускають можливість аналізу звукової енергії в залежності від діапазону частот.

Математично таку гіпотезу зручно описувати за допомогою апарату підзонної матриці. Визначення енергії в заданому діапазоні частот виконується наступним методом. Вісь частот, нормована на частоту дискретизації, ділиться на R рівних інтервалів:

$$v_r = v_0 / R, \quad r = 1 \ldots (R - 1), \tag{2.8}$$

де R — кількість рівних інтервалів, на які поділена вісь частот.

Енергія в кожному частотному діапазоні оцінюється за виразом:

$$Pr = \vec{x^T} \cdot Ar \cdot \vec{x}, \tag{2,9}$$

де x – сегмент аналізованого сигналу тривалістю N відліків; T – знак транспозиції;

Ar це матриця піддіапазонів з елементами форми:

$$A_r(i,k) = \begin{cases} \frac{\sin(v_r(i-k)) - \sin(v_{r-1}(i-k))}{\pi(i-k)}, & \text{при } i \neq k \\ \frac{(v_r - v_{r-1})}{\pi}, & \text{при } i \neq k \end{cases}, \tag{2.10}$$

де v_r – верхня межа частотного діапазону r;

$$i = 1 \dots N$$

$$k = 1 \dots N \qquad (2.11)$$

Матриці піддіапазонів мають лише кілька ненульових власних значень. Ця обставина прискорює обчислення квадратичної форми за формулою:

$$P_r = \sum_{k=1}^{J} \lambda_k^r \, (\vec{x}\vec{Q}_k^r)^2 \,, \qquad (2.12)$$

де λ_k^r – власні значення матриці , їх кількість J, власні значення розташовані в порядку зростання з індексом більшим за J нерелевантний $\lambda_{j+k}^r \approx 0$, для практичних розрахунків J = 8; $(\vec{x}\vec{Q}_k^r)$ - скалярний добуток вектора аналізованого сигналу на власний вектор $\vec{Q}_k^r$ матриці Ar, що відповідає власному значенню λ_k^r.

Також слід зазначити, що формула (2.12) відповідає нейронній мережі з двома розташованими в порядку зростання шарами. На рис. 2.11, отримана оцінка енергії в r-му діапазоні частот на виході другого шару, вагові коефіцієнти λ_k^r, k = 1 ... K, які є власними значеннями від першого до K-го піддіапазонних масивів. Ар. Між першим і другим шарами є функція активації, яка відповідає операції зведення в квадрат. Перший шар складається з нейронів, в яких вагові коефіцієнти q_{ki}^r, $k = 1 \dots K, i = 1 \dots N$ є елементами власних векторів $\vec{Q}_k^r$ матриці A_r.

Отримані оцінки енергії в частотних діапазонах являють собою повний набір ознак з точки зору адитивності.

$$\|\vec{x}\|^2 = \sum_{r=1}^{R} \vec{x}^T \cdot A_r \cdot \vec{x} \,, \qquad (2.13)$$

Для аналізу вихідного сигналу необхідно вибрати кількість читань N. Довжина вікна аналізу не повинна бути занадто довгою, інакше виникнуть ефекти накладання фонем. Велика довжина вікна аналізу також збільшує кількість основних операцій (додавання та множення), необхідних для подальших перетворень вихідного сигналу. Мала довжина вікна аналізу погіршує частотну роздільну здатність тестованого сигналу. З цих умов довжину вікна аналізу N слід вибирати в діапазоні від 60 до 160 відліків. На рис. 2.11 показана нейронна мережа для оцінки енергії в частотному діапазоні.

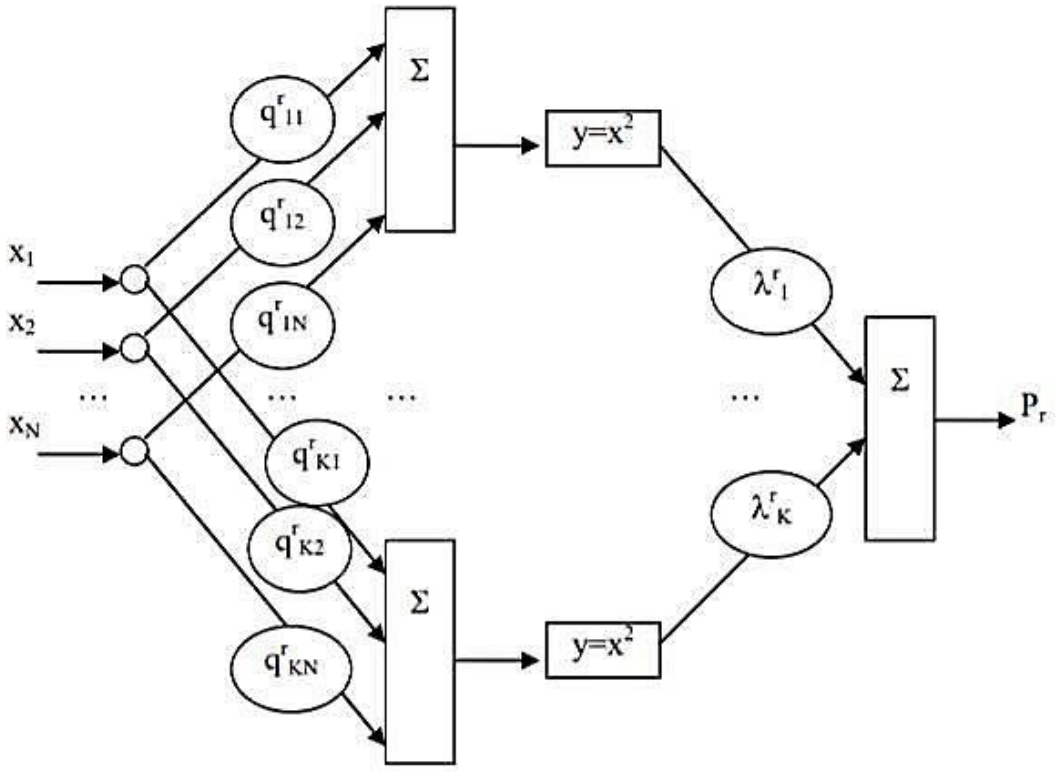

Рисунок 2.11 - Нейронна мережа для оцінки енергії в діапазоні частот

Для експерименту були взяті матриці піддіапазонів зі значеннями N = 60 і R = 15. Кількість власних значень, відмінних від нуля, приймається як J = 8. Відліки були отримані з аудіофайлу, що містить мовний сигнал, що становить вхідний сигнал $\vec{x}$

$$\vec{x}_n = [x_{n-N+1} \;\; x_{n-N+2} \dots x_{n-1}\, x_n]^T, n > N, \qquad (2.14)$$

де n - контрольне число, дискретне.

Видається раціональним дослідження сигналу, зміщуючи вікно аналізу на один градус. Тоді значення оцінки енергії можуть бути записані з тим же позначенням моменту часу, що і для вхідного сигналу.

$$P_r = \sum_{k=1}^{J} \lambda_k^r (\vec{x}\vec{Q}_k^r)^2 \,, r = 1 \dots R \,, \qquad (2.15)$$

Оцінки енергії повинні бути усереднені за деякий інтервал часу [n - Ts, n], n> Ts.

$$\beta_r(n) = \frac{1}{Ts}\sum_{k=n-Ts}^{n} P_r(k)\ , r = 1 \dots R\ , \qquad (2.16)$$

Величина інтервалу Ts повинна бути співмірна довжині сегмента мовного сигналу, на якому він нерухомий. Практичні спостереження показують, що кількість фонем за одну секунду не перевищує 25. При частоті дискретизації 8000 Гц необхідно в середньому 320 відліків на фонему. Враховуючи середні квадратичні відхилення від середнього для однієї фонеми, доцільно вибрати Ts від 100 до 200 відліків, оскільки точність усереднення знижується зі зменшенням кількості членів. Усереднення ковзного середнього є фільтром CIC, який є дуже економічним з точки зору складності обчислень.

$$\beta_r(n) = \frac{1}{Ts}\big(P_r(n) + \beta_r(n-1) - P_r(n-Ts)\big), r = 1 \dots R\ . \qquad (2.17)$$

В експерименті було прийнято Ts = 200.

Після усереднення стає можливим аналізувати оцінки енергії не для всіх відліків, а лише після кожного відліку Ts / 2.

$$\beta 1_r(n1) = \beta_r(n), n = \frac{Ts}{2} n1\ , r = 1 \dots R\ , n1 = 1,2,3 \qquad (2.18)$$

Для найпростішого етапу навчання було створено мобільний додаток на мові Java, який передавав дані до комп’ютера в якому відбудовувались графіки в середовищі Simulink . Моделі передали аудіофайл, який містив слово «п'ять», сказане кілька разів. Результат обробки за формулами (2.14) - (2.18) наведено на рис. 2.12.

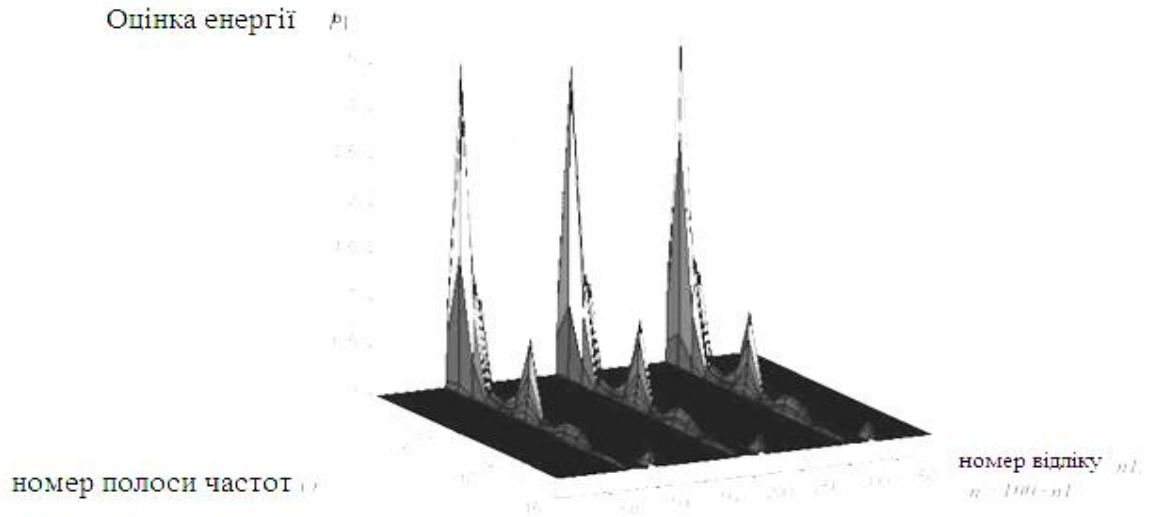

Рисунок 2.12 - Оцінка енергетики слова «п'ять» після досліду

Ви вручну вибрали все слово, малюнок 2.13, і зберегли його для подальшого використання як шаблон. Шаблон являє собою Pt-матрицю з кількістю рядків, що дорівнює кількості аналізованих частотних діапазонів. кількість колонок відповідає кількості вибраних показань .

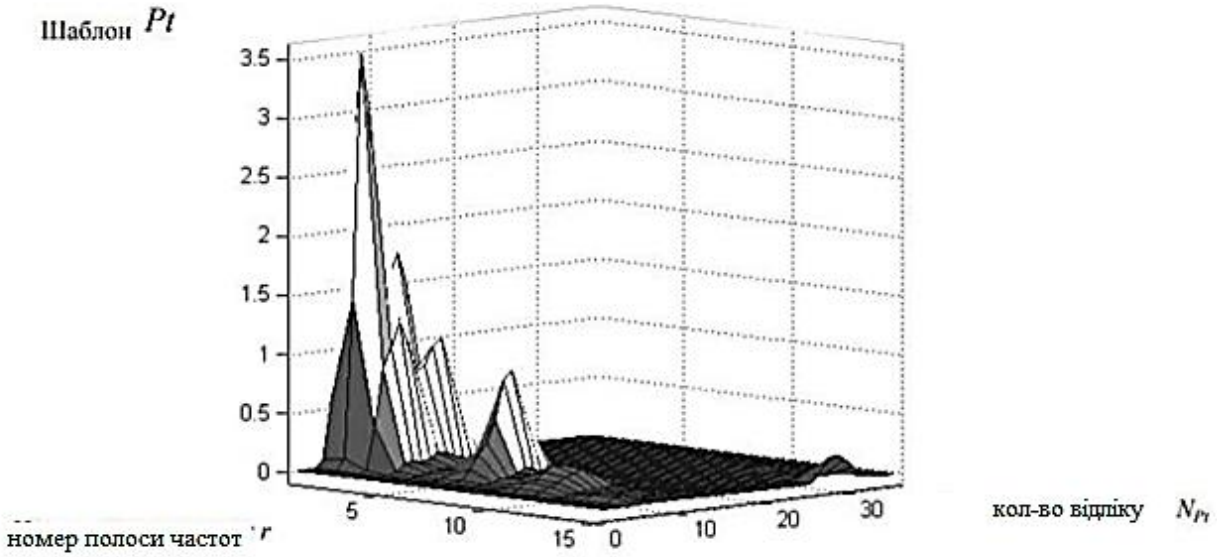

Рисунок 2.13 - Шаблон для слова "п'ять"

Для вирішення проблеми розпізнавання необхідно порівняти сегмент мовного сигналу, із записаною формулою Pt в словнику. Основна складність порівняння

— варіативність слова, як за довжиною, так і за звучанням. Тому необхідно порівняти дві матриці між собою, кількість рядків у матрицях однакова і відповідає числу розглянутих частотних інтервалів R, а кількість стовпців різна.

При такому підході ці матриці являють собою послідовності з різною кількістю елементів, які є стовпцями [9].

Для вимірювання подібності двох послідовностей, які мають різну кількість елементів, часто використовується алгоритм динамічного викривлення часу. Порівняння відбувається за принципом «найкращого підходу». Відповідно до цього алгоритму матриця D повинна бути спочатку складена з відстанями між елементами послідовності. При визначенні відстані пропонується використовувати міри відносної різниці:

$$D(i,k) = \frac{\sum_{r=1}^{R}|\beta 1(r,i) - Pt(r,k)|}{\sum_{r=1}^{R}|\beta 1(r,i) + Pt(r,k)|}, i = 1 \dots N1, k = 1 \dots N_{Pt}, \qquad (2.19)$$

Потім створюється матриця C із кумулятивною відстанню між окремими елементами двох послідовностей:

$$C\ (i,k) = D\ (i,l) + \text{хв}\ (C\ (i,k\text{ - }1), C\ (i\text{ - }1,k), C\ (i\text{ - }1,k\text{ - }1)), \qquad (2{,}20)$$

де мінімум береться між трьома сусідніми елементами;

$$C\ (1{,}1) = 0, C\ (1,k) = \infty, C\ (i,1) = \infty;$$
$$k = 2 \dots N \qquad (2{,}21)$$

У розв'язуваній задачі опорним є стовпець зі значеннями енергетичних оцінок. Тоді в матриці C шлях від точки (01) до точки (1,1) відновлюється за принципом руху в напрямку з найменшим значенням:

$$(is, ks) = arg\ \text{хв}\ (C\ (i,k\text{ - }1), C\ (i\text{ - }1,k), C\ (i\text{ - }1,k\text{ - }1)), \qquad (2{,}22)$$

де s – номер кроку на шляху від (N1, NPt) до (1,1). Загальна кількість кроків S НЕ є постійною, кроки виконуються, доки обидва покажчики і і k не зміняться від своїх максимальних значень 01 і до 1.

Потім обчислюється загальна різниця SH між двома послідовностями. В класичному варіанті алгоритму:

$$SH = \sum_{s=1}^{S} D(i_s, k_s), \quad (2.23)$$

ця робота пропонує:

$$SH = \text{максимум}\ (D\ (is, ks)), s = 1 \dots S. \quad (2,24)$$

При порівнянні невідомої послідовності з шаблоном рішення про відповідність шаблону приймається після порівняння SH із заздалегідь визначеним порогом.

Для експерименту один диктор продиктував п’ять аудіофайлів однакового змісту: промовлені цифри в порядку: «1», «2», «3», «4», «5», «6», «7», « 8", "9" і "0". Мовний сигнал з аудіофайлів порівнювався з попередньо збереженим шаблоном слова "five".

Оцінки енергії n на кроці n1 були сформовані в матрицю з кількістю стовпців N1 = 50, оскільки кількість стовпців у шаблоні виявилася 36.

Матриця невідомого сигналу береться для більш ніж у показань шаблон (50> 36), враховуючи можливу повільну вимову шуканого слова. Результат моделювання на рис. 2.14 для мовного сигналу з файлу № 2 і на рис. 2.15 - з файлу № 4.

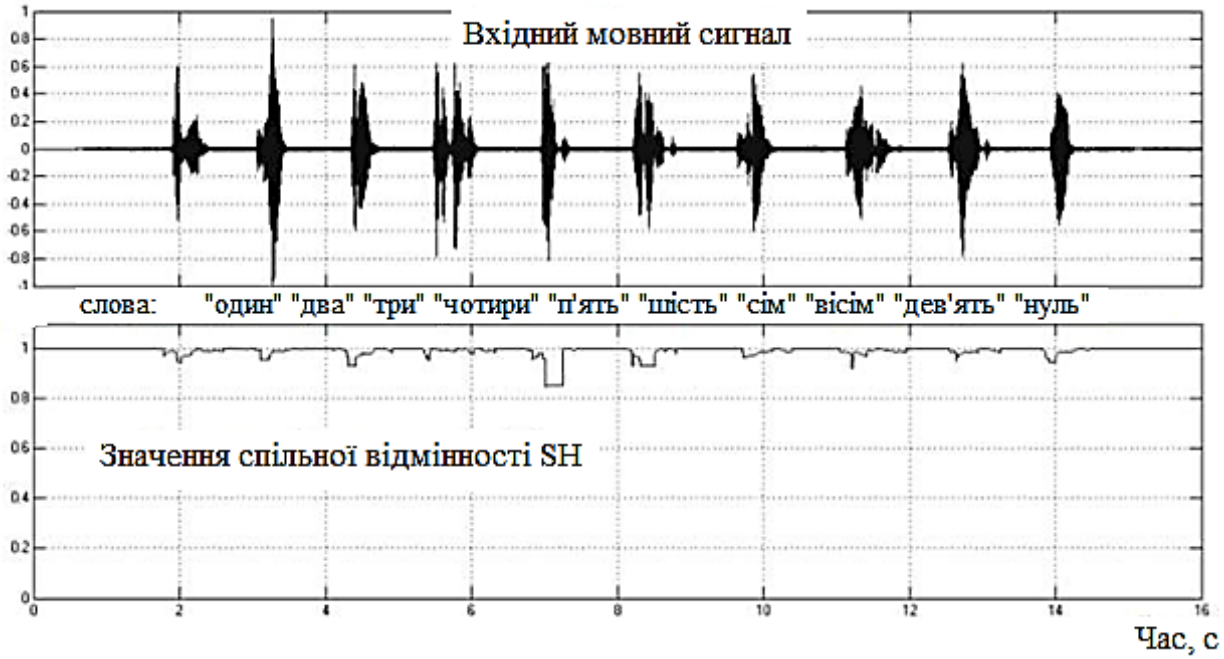

Рисунок 2.14 – Графіки вхідного сигналу та величини різниці від шаблону після застосування відносної міри. Вхідний сигнал з файлу №2

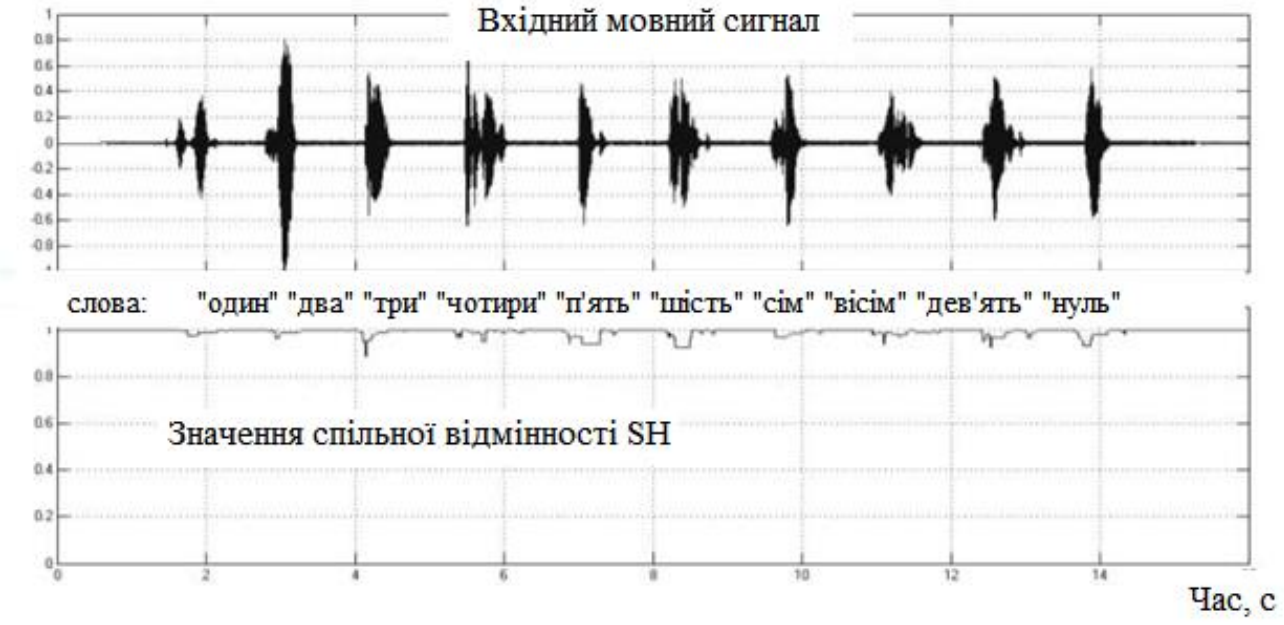

Рисунок 2.15 – Графіки вхідного сигналу та величини різниці від шаблону після застосування відносної міри. Вхідний сигнал з файлу №4

На рис. На малюнках 2.14 показана чітка відповідь, а на рис. 2.15 відповідь на слово «шість» виглядає більш переконливо, хоча зі словом «п'ять» відмінність від шаблону менша, ніж з рештою слів.

Для підвищення чіткості при аналізі різниці між шаблоном і вхідним сигналом

можна використовувати нелінійне перетворення значень оцінки енергії. Відповідно до емпіричного психофізіологічного закону Вебера-Фехнера, сила відчуття пропорційна логарифму інтенсивності подразника. Інший вчений, Стівенсон, запропонував використовувати степеневу функцію для опису залежності сили відчуття від величини подразнення.

В експериментах Стівенсона для різних відчуттів використовується індекс статичної функції від 3,5 до 0,67.

Для вирішення проблеми розпізнавання може бути використана нелінійність вилучення квадратного кореня зі значень оцінок енергії, яка буде схожа на статичний закон Стівенсона, але не внесе значних обчислювальних труднощів.

Нові оцінки для нелінійностей:

$$W(n1) = \sqrt{\beta 1(n1)},$$
$$W_{Pt} = \sqrt{Pt}. \tag{2.25}$$

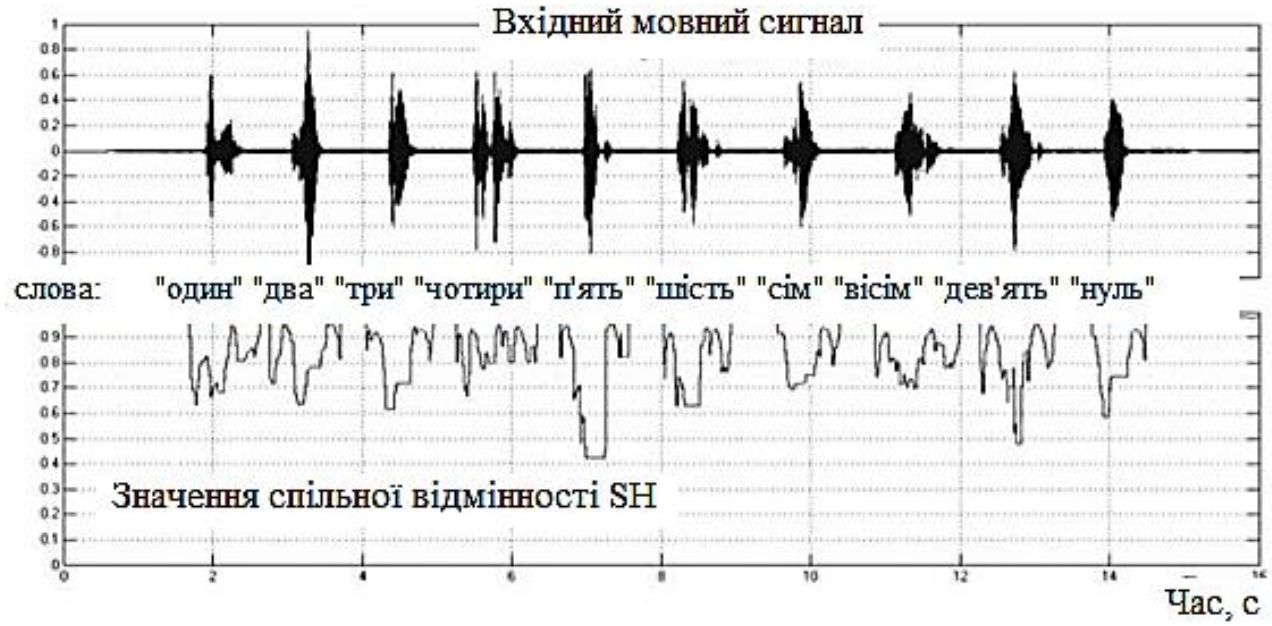

Рисунок 2.16 – Графіки вхідного сигналу та величини різниці від шаблону після застосування відносної міри та нелінійності. Вхідний сигнал з файлу №2

Результат моделювання при наявності нелінійностей на рис. 2.17 і 2.18.

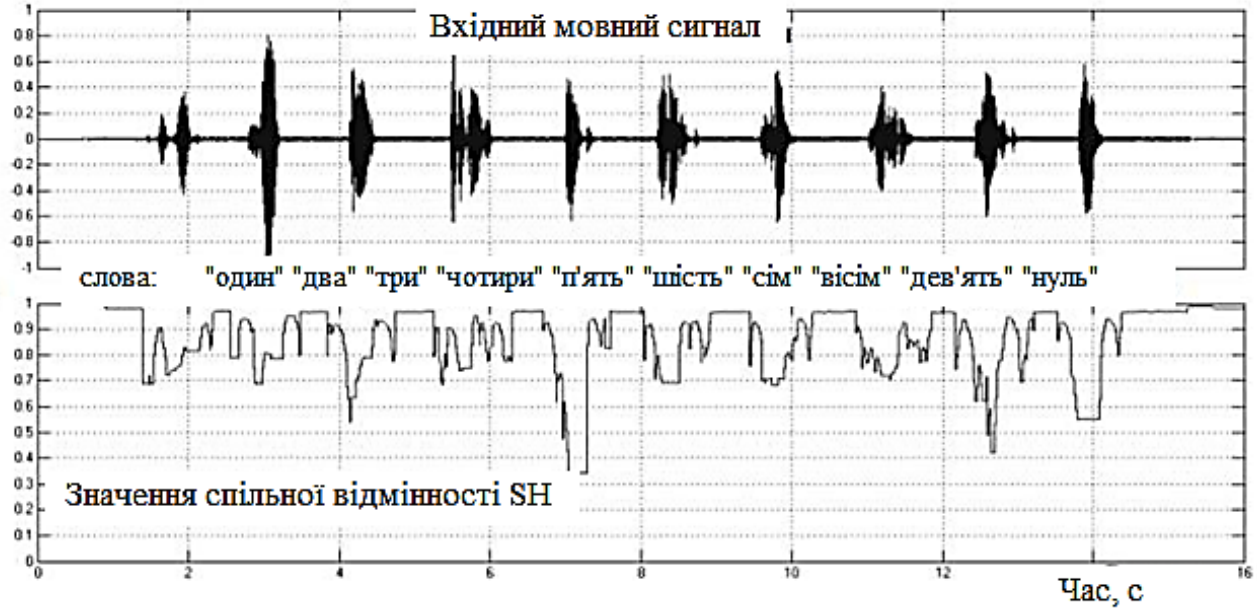

Рисунок 2.17 – Графіки вхідного сигналу та величини різниці від шаблону після застосування відносної міри та нелінійності. Вхідний сигнал з файлу №4

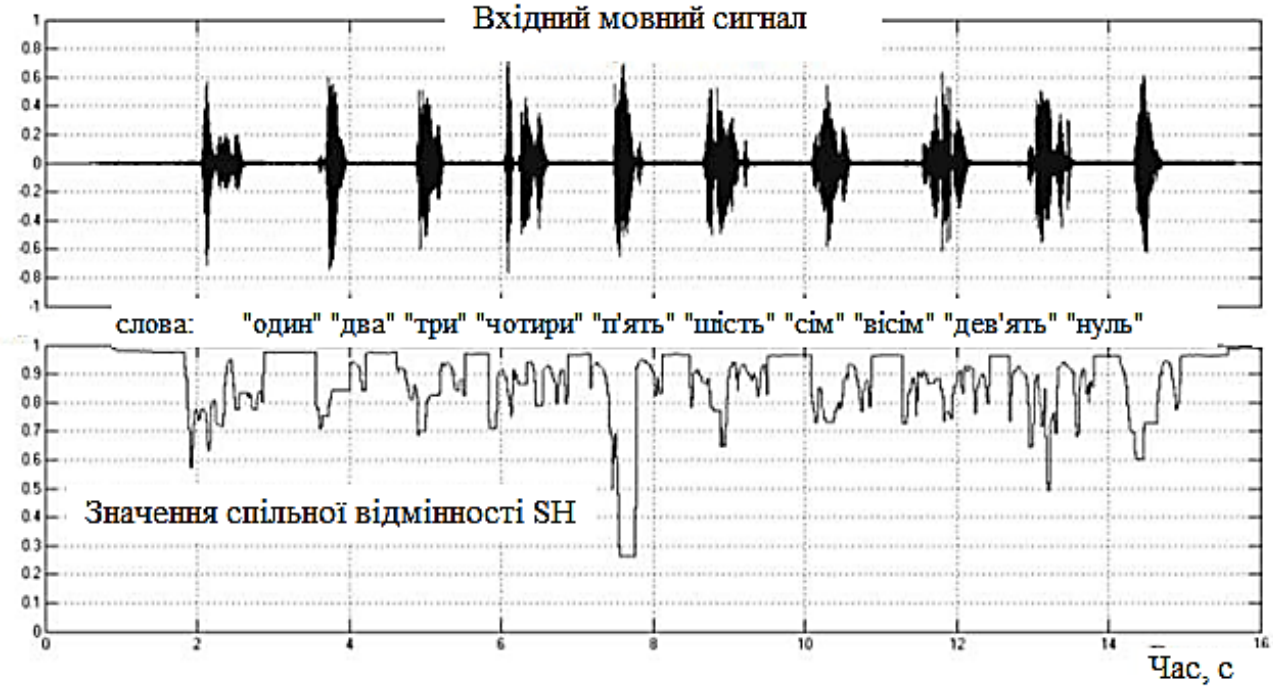

Рисунок 2.18 – Графіки вхідного сигналу та величини різниці від шаблону після застосування відносної міри та нелінійності. Вхідний сигнал з файлу №1

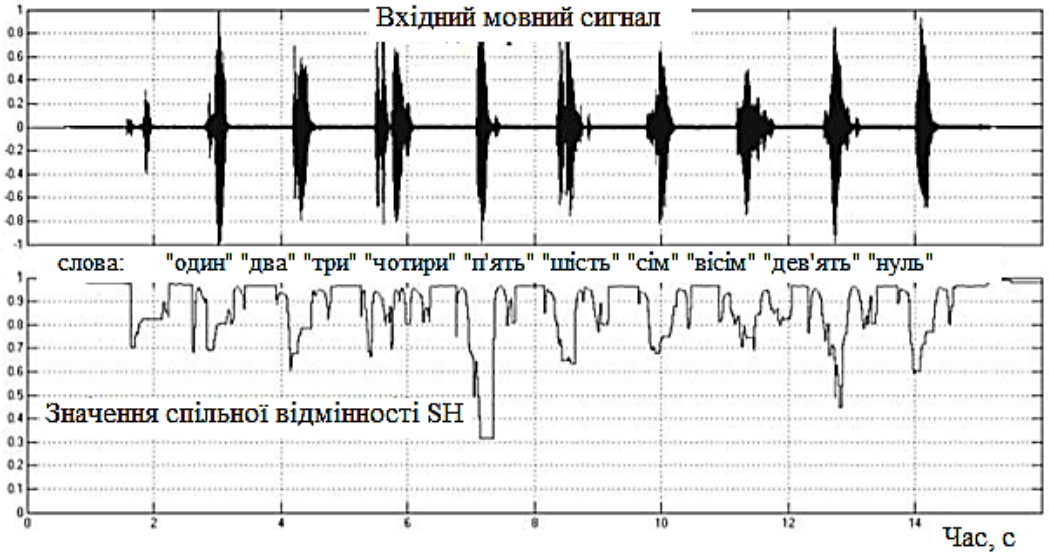

Рисунок 2.19 – Графіки вхідного сигналу та величини різниці від шаблону після застосування відносної міри та нелінійності. Вхідний сигнал з файлу №3

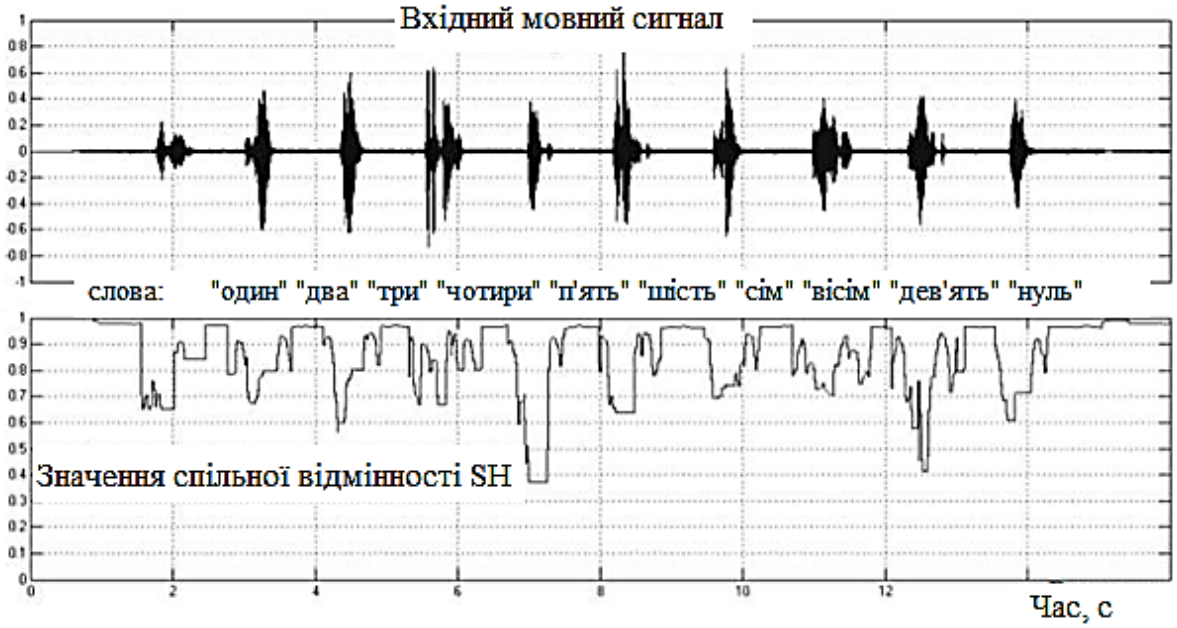

Рисунок 2.20 – Графіки вхідного сигналу та величини різниці від шаблону після застосування відносної міри та нелінійності. Вхідний сигнал з файлу №5

Використання нелінійності дозволило збільшити роздільну здатність алгоритму. Мінімум до слова «п'ятірка» сильніше виділяється серед інших слова та графіки попередніх експериментів. Для наочності на рис. 2.18-2.20 наведено графіки мовних сигналів з файлів №1, №3, №5. Поведінка діаграм однакова.

Порогове значення має бути встановлено з урахуванням різноманітності вимови ключового слова. Для цього доступні звукові файли відбиралися вручну та зберігалися у змінних матрицях = 1,2,3,4,5 з енергетичними оцінками для слова «п’ять». Потім матриці порівнювали кожну з кожним алгоритмом динамічного спотворення, використовуючи відносну міру між стовпчиками та нелінійне перетворення оцінок енергії шляхом вилучення квадратного кореня.

$$SH_{ij} = DTW\ (\sqrt{\beta 1}_i, \sqrt{\beta 1}_{\text{j}},) \qquad (2{,}26)$$

де i, j = 1,2,3,4,5 – індекси файлів, з яких взято слово «п’ять», та в матриці ознак для слова «п’ять», елементи матриці обчислюються за формулою (4) з подальшим використанням середньої змінної (2.17); кількість рядків у матриці дорівнює кількості частотних інтервалів, кількість стовпців відповідає звуковій тривалості слова після проріджения (2.18), заданої вручну; є функцією алгоритму динамічного скручування, описаного формулами (2.19) - (2.22), (2.24).

На відміну від безперервного мовного сигналу, немає необхідності переміщати вікно аналізу, відзначаючи величину різниці в кожен момент. На цьому етапі встановлюються матриці, що описують слово «п'ять». Таким чином, результатом порівняння є одне число, а не графік у часі. Результати парних порівнянь показують, наскільки по-різному вимовляється слово «п'ять» одним мовцем.

3 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ

Розроблений у рамках роботи веб-додаток – це мобільний додаток для аутентифікації користувача за допомогою голосового повідомлення.

Додаток повинен мати логічно декомпозовану структуру. Інтерфейс користувача програми повинен бути простим і зрозумілим.

Для розробки мобільного додатку для розпізнавання та виконання голосових команд необхідно підготувати це програмне забезпечення. Також важливо вибрати апаратне та програмне забезпечення, яке відповідатиме вашим потребам у розробці програмного забезпечення за мінімальних витрат.

3.1 Створення дизайну програми для аутентифікації голосових команд

Для доменно-орієнтованого проектування методологія IDEF0 була використана для створення функціональної декомпозиції процесів.

IDEF0 (Function Modeling) — це методологія функціонального моделювання та графічного опису процесів, призначена для формалізації та опису бізнес-процесів. Особливістю IDEF0 є його акцент на ієрархічному представленні об’єктів, що полегшує розуміння теми. В IDEF0 враховуються логічні зв'язки між роботами, відображаються сигнали управління. Опис виглядає як «чорна скринька» з входами, виходами, елементами керування та механізмом, який поступово допрацьовується до необхідного рівня. Ця модель є однією з найпрогресивніших моделей, її використовують при організації багатьох проектів.

DFD (data flow diagram) - діаграми потоку даних. Це назва методології графічного структурного аналізу, яка описує джерела даних і одержувачів поза системою, логічні функції, потоки даних і сховища даних, до яких здійснюється доступ. Ddin є одним з основних інструментів для структурного аналізу та проектування інформаційних систем, які існували до широкого використання UML.

На рисунках 3.1-3.2 показано контекстну діаграму та діаграми декомпозиції процесу.

Рисунок 3.1 – Контекстна діаграма

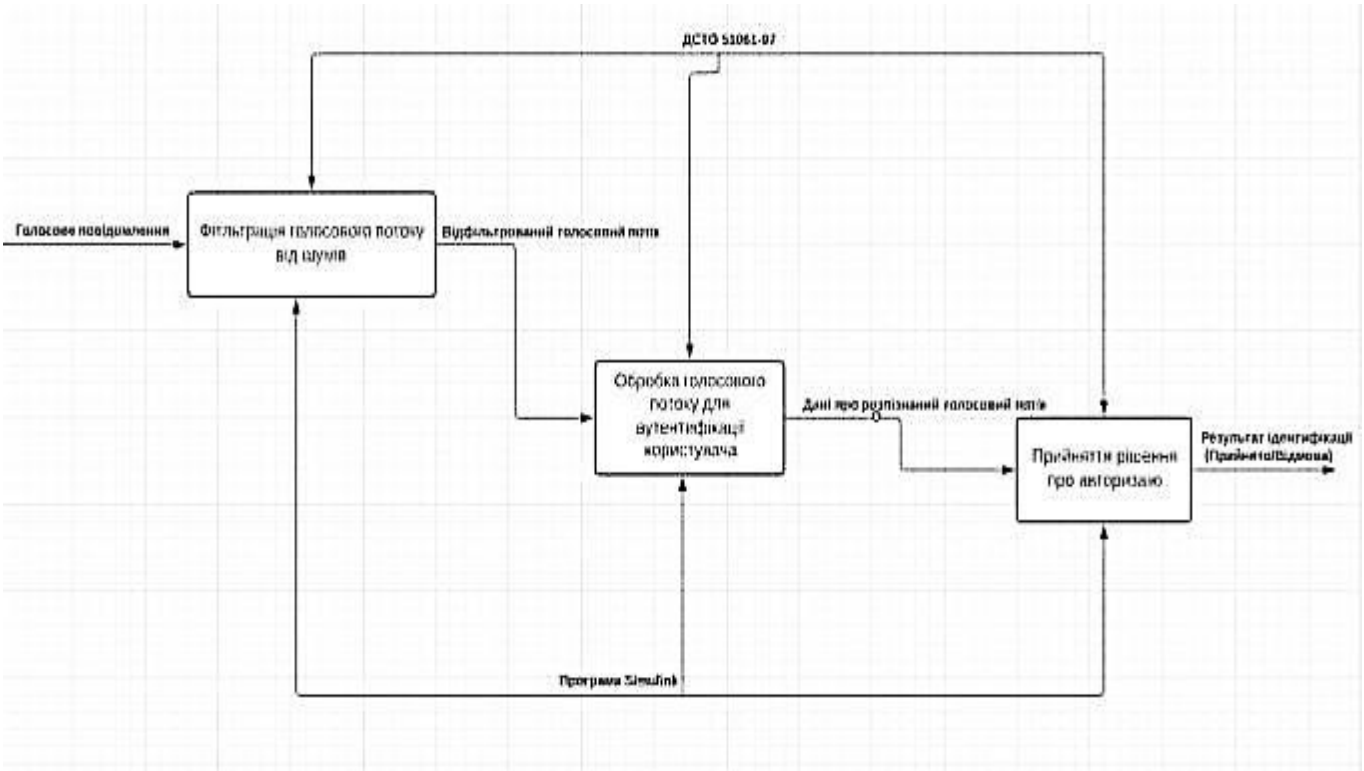

Рисунок 3.2 – Декомпозиція основного бізнес-процесу На малюнку 3.3 показана діаграма DFD.

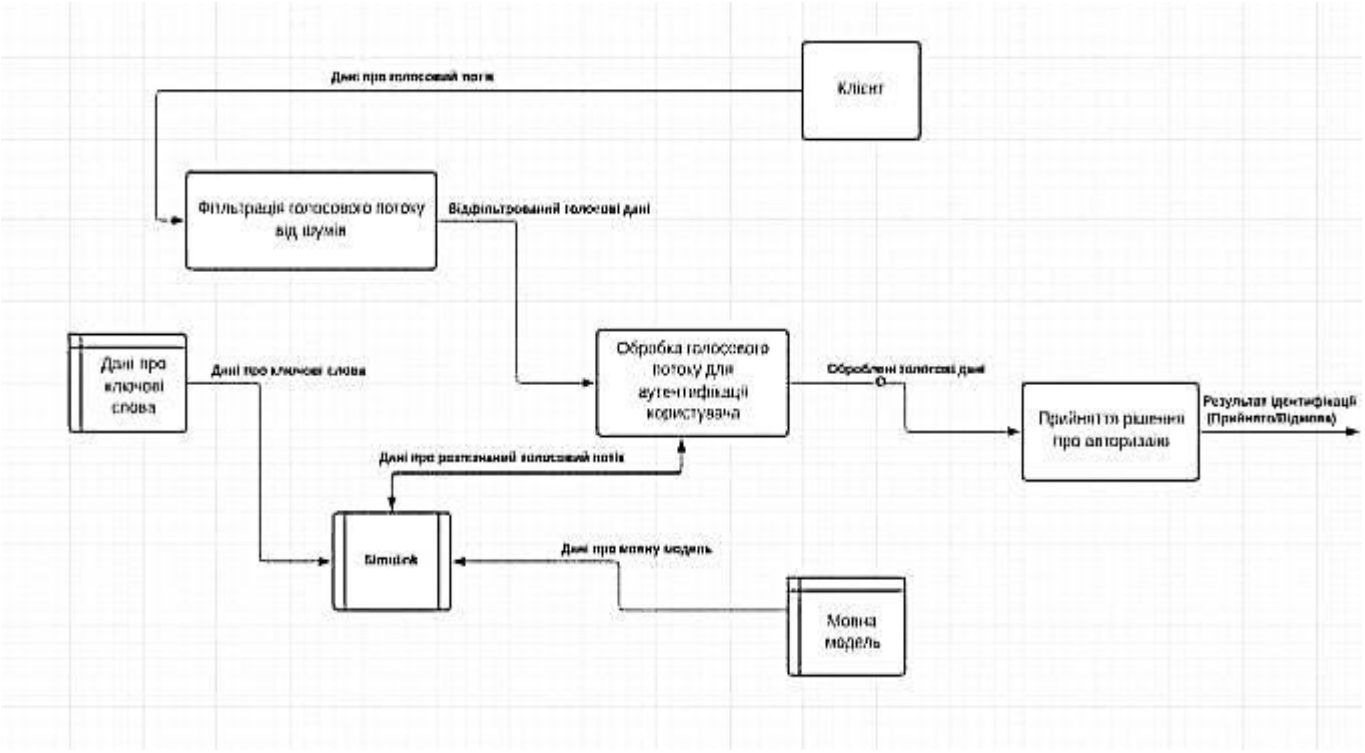

Рисунок 3.3 - Діаграма DFD

UML — це мова графічного опису для об'єктно-орієнтованого моделювання в області розробки програмного забезпечення, для моделювання бізнес-процесів, проектування систем і відображення організаційних структур.

UML — це мова загального призначення з відкритим стандартом, яка використовує графічні нотації для створення абстрактної моделі системи, яка називається моделлю UML. UML було створено для визначення, візуалізації, проектування та документування переважно програмних систем. UML не є мовою програмування, але генерація коду можлива на основі моделей UML.

Щоб описати, як програмний інструмент, який ви створюєте, використовуватиметься на концептуальному рівні, слід створити схему випадку.

Прецедентом є функціональність системи, яка дозволяє користувачеві отримати якийсь значущий, відчутний і вимірний для нього результат. Кожен екземпляр відповідає окремій службі, що надається моделюваною системою у відповідь на запит користувача, тобто визначає, як використовується система. Варіанти використання найчастіше використовуються для визначення зовнішніх вимог до розробленої системи або для визначення функціональної поведінки вже існуючої системи. Крім того, варіанти використання неявно описують типові способи вза-

ємодії користувача з системою, які дозволяють коректно працювати з сервісами, що надаються системою.

Мобільний додаток, що розробляється, призначений для запису голосу та аутентифікації голосу. З додатком взаємодіятиме один актор.

Діаграма параметрів використання системи, показана на малюнку 3.4, базується на вимогах до програмного засобу.

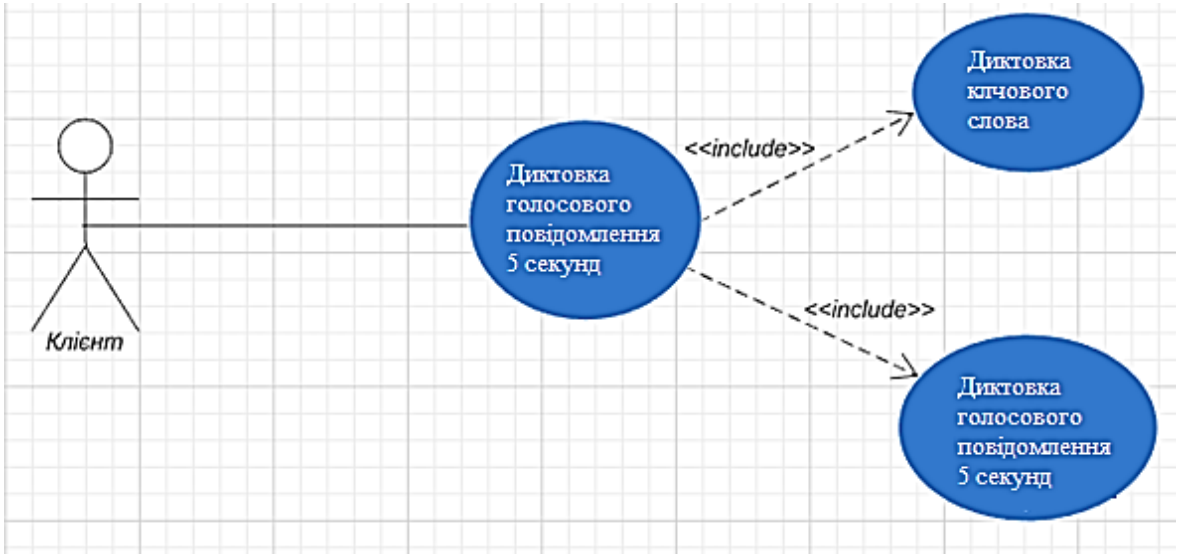

Рисунок 3.4 – Схема варіантів використання системи

Щоб написати програмний код, потрібно створити діаграму класів, яка допоможе визначити архітектуру програмного забезпечення.

Діаграма класів — це структурна діаграма мови моделювання UML, яка показує загальну структуру ієрархії класів системи, їх взаємодію, атрибути (поля), методи, інтерфейси та їхні зв’язки.

На рисунку 3.5 показана діаграма класів, яка використовується для написання коду інструменту.

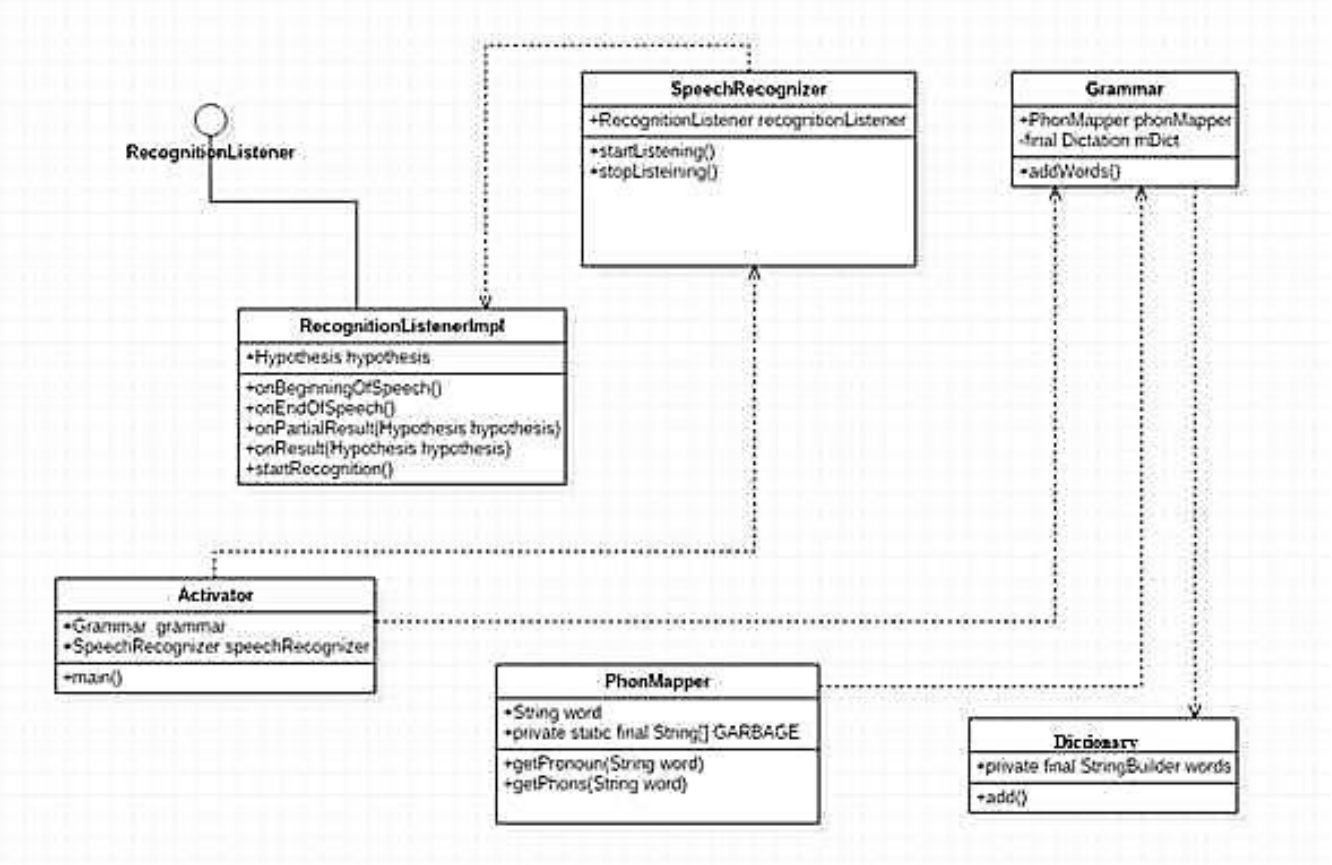

Рисунок 3.5 – Діаграма класів

3.2 Вибір шаблонів оформлення

Впровадження залежностей (DI) або реалізація залежностей — це механізм, який дозволяє використовувати слабозв’язані об’єкти у вашій програмі. Такі об’єкти пов’язані разом через абстракції, наприклад через інтерфейси, що робить загальну систему більш гнучкою, адаптивною та розширюваною [10].

Часто для установки залежностей в подібних системах використовуються спеціальні контейнери - контейнери IoC (Inversion of Control). Такі контейнери служать свого роду фабриками, які встановлюють зв'язки між абстракціями і конкретними об'єктами і, як правило, контролюють створення цих об'єктів.

Інверсія керування (IoC) — це принцип розробки програмного забезпечення, який передає керування об’єктами або частинами програми до контейнера або платформи. Найчастіше ми використовуємо його в контексті об'єктно-орієнтованого програмування.

На відміну від традиційного програмування, де наш код користувача викликає бібліотеку, під час використання IoC розробник передає контроль над потоком програми та викликає наш код користувача до зовнішнього контейнера. Це дозволяє зосередитися на розробці бізнес-логіки, а сам контейнер подбає про моменти створення та видалення об’єктів, перенаправлення викликів API до відповідних сервісів, надсилання повідомлень про помилки тощо. Для цього в структурах використовуються абстракції з додаткова вбудована поведінка. Щоб додати власну поведінку, вам потрібно розширити класи рамки або додати власні класи.

Model-View-ViewModel (MVVM) — це шаблон проектування архітектури програми, який використовується для розділення моделі та її представлення, щоб їх можна було змінювати незалежно одне від одного. Наприклад, розробник задає логіку роботи з даними, а дизайнер працює з інтерфейсом користувача.

MVVM зручно використовувати замість класичного MVC і тому подібного в тих випадках, коли платформа розробки має прив’язку даних. На відміну від MVVM, у шаблонах проектування MVC / MVP зміни в інтерфейсі користувача не впливають безпосередньо на модель, але спочатку проходять через контролер або презентатор [11].

Спостерігач — це шаблон поведінкового проектування, який створює механізм підписки, який дозволяє одному об’єкту відстежувати та реагувати на події в інших об’єктах. Він визначає зв’язок «один-до-багатьох» між об’єктами, так що коли стан одного об’єкта змінюється, всі залежні об’єкти отримують сповіщення про подію. Класи, на події яких підписані інші класи, називаються суб’єктами, а класи, на які підписуються, називаються спостерігачами [11].

Шаблони проектування Dependency Injection, MVVM і Observer будуть використані для написання кваліфікаційної роботи, щоб забезпечити правильну роботу зовнішніх бібліотек і фреймворків, правильну взаємодію користувача з інтерфейсом програми, а також зв’язок інтерфейсу користувача з даними внутрішнього коду програми та бази даних.

3.3 Вибір і обґрунтування мовних і програмних засобів

Сьогодні існує багато мов програмування для розробки мобільних додатків. Ми розглянемо Rust, Java і Kotlin.

Rust — це скомпільована багатопарадигмальна мова програмування загального призначення, яка поєднує парадигми функціонального та процедурного програмування з об’єктно-орієнтованою системою. Управління пам'яттю здійснюється через механізм «володіння» за допомогою афінних типів, що дозволяє обійтися без системи збирання сміття під час виконання програми. Rust гарантує безпечну роботу з пам'яттю завдяки вбудованій у компілятор системі перевірки статичних посилань (borrow checker) [12].

Ключові мовні пріоритети: безпека, швидкість і паралелізм. Rust підходить для системного програмування, зокрема, вважається перспективною мовою для створення ядер операційних систем. За швидкістю та можливостями Rust можна порівняти з C ++ / Si, але він пропонує велику безпеку при роботі з пам’яттю, що забезпечується вбудованою перевіркою посилань на мову.

Rust підтримує кілька платформ і може використовуватися для створення мобільних програм для Android, iOS, Windows, macOS, Linux і багатьох варіантів Unix. Rust підходить для створення нативних веб-додатків, а також операційних систем, компонентів браузера та ігрових движків .

Java є однією з найпоширеніших і популярних мов програмування. Java була розроблена як мова програмування загального призначення, яку можна використовувати для різноманітних завдань. І поки Java пройшла довгий шлях, випущено багато різних версій.

Java перетворилася з універсальної мови на цілу платформу та екосистему, яка поєднує різні технології, що використовуються для багатьох завдань: від створення настільних програм до написання великих порталів і веб-сервісів.

Подібна архітектура забезпечує крос-платформну сумісність і портативність апаратного забезпечення для програм Java, тому схожі програми можуть працювати на різних платформах без перекомпіляції – Windows, Linux, Mac OS, Android тощо.

Кожна платформа може мати власну реалізацію JVM, але кожна може виконувати той самий код.

Ще одна ключова особливість Java полягає в тому, що вона підтримує автоматичне видалення сміття. Це означає, що вам не доведеться вручну вивантажувати раніше використані об'єкти, як у C++, тому що збирач сміття робить це автоматично .

Kotlin — це статична об’єктно-орієнтована мова програмування, яка працює на віртуальній машині Java і розроблена JetBrains. Автори поставили собі за мету створити мову, більш лаконічну та безпечнішу за Java та простішу за Scala. Спрощення порівняно зі Scala також призвело до швидшої компіляції та кращої підтримки мови в IDE. Мова повністю сумісна з Java, що дозволяє розробникам Java поступово переходити на її використання; зокрема, мова також вбудована в Android, що дозволяє існуючій програмі Android реалізувати нові функції в Kotlin без повного переписування програми .

Ключовою особливістю Kotlin є те, що код спочатку перекладається на спеціальний байт-код, сумісний із різними платформами. Потім цей байт-код виконується JVM (віртуальна машина Java).

Kotlin є об'єктно-орієнтованою мовою, тому він підтримує поліморфізм, успадкування та статичну типізацію. Об’єктно-орієнтований підхід дозволяє вирішити завдання побудови великих, але в той же час гнучких, масштабованих і розширюваних мобільних додатків .

Для розробки мобільного додатку буде використовуватися мова Java. Завдяки своїй гнучкості, універсальності та доступним фреймворкам можна швидко та елегантно реалізувати інтерфейс клієнта та взаємодіяти з серверними сервісами.

Також слід зазначити, що для коректної роботи створеного додатку та бази даних буде використовуватися мова SQL для написання коректних запитів на взаємодію між ними.

На сьогодні існує лише кілька варіантів середовищ розробки мобільних додатків через специфічну архітектуру проекту, а також необхідність підтримки віртуального пристрою для тестування та налагодження. Серед таких середовищ було обрано три найяскравіших приклади за функціональністю – Eclipse IDE, Intelij Idea та

Android Studio.

Eclipse IDE — це відкрита та безкоштовна IDE із модульною архітектурою. Він написаний на Java і вважається найпопулярнішим Java IDE, працює на всіх основних платформах, включаючи Android, Linux, Windows, Mac OS тощо. І має потужні функції, які можна використовувати для реалізації повноцінних проектів. Eclipse підтримує Git, Apache Maven, Gradle тощо.

Eclipse відомий широкою підтримкою плагінів. Він також дозволяє створювати плагіни в середовищі розробки плагінів (PDE). Eclipse також має стандартний набір віджетів, або SWT, який використовується для доступу та використання елементів GUI з операційної системи, на якій створено додаток .

IntelliJ IDEA розроблено компанією JetBrains, раніше відомою як IntelliJ. Перша версія була випущена в 2001 році і включала такі функції, як покращена навігація по коду та покращені можливості рефакторингу коду, що зробило її дуже популярною. У 2010 році його визнали найкращим інструментом розробки на основі Java, випередивши NetBeans, Eclipse і JDeveloper.

Середовище розробки з відкритим кодом для Android, випущене Google у 2014 році, також базується на IntelliJ IDEA. IDE підтримує багато інших мов програмування, таких як Python, Lua і Scala.

Основною причиною, чому його вважають одним із найкращих інструментів розробки на основі Java, є його допоміжні функції, які роблять IntelliJ IDEA простим у використанні та значно допомагають вам писати високоякісний код. Він також має розширені функції перевірки помилок, які дозволяють швидше та легше перевіряти розроблений код [12].

Android Studio — інтегроване середовище розробки (IDE) для платформи Android, представлене 16 травня 2013 року на конференції Google I/O Еллі Пауерс, менеджером із продуктів корпорації Google .

Android Studio замінив плагін ADT для платформи Eclipse. Середовище побудовано на основі коду продукту IntelliJ IDEA Community Edition, розробленого JetBrains. Android Studio розроблено за відкритою моделлю розробки та поширюється за ліцензією Apache 2.0. Середовище надає інструменти для створення додат-

ків не тільки для смартфонів і планшетів, а й для пристроїв на базі Wear OS, телевізорів (Android TV), Google Glass і автомобільних інформаційно-розважальних систем (Android Auto).

Для додатків, спочатку розроблених з використанням Eclipse і плагіна ADT, було підготовлено інструмент для автоматичного імпорту існуючого проекту в Android Studio.

Середовище розробки адаптовано для виконання типових завдань, які вирішуються в процесі створення додатків для платформи Android. Серед них інструменти, що полегшують тестування програм на сумісність з різними версіями платформи та інструменти для проектування додатків, які працюють на пристроях з екранами різної роздільної здатності (планшети, смартфони, ноутбуки, годинники, окуляри тощо). На додаток до можливостей, присутніх в IntelliJ IDEA, деякі додаткові функції були реалізовані в Android Studio, такі як нова уніфікована підсистема для створення, тестування та розгортання програм.

Для розробки програми було обрано Android Studio, оскільки IDE має широкий набір функцій підтримки, які прискорюють розробку, допомагають контролювати якість коду та можливі помилки. Крім того, Android Studio містить багато плагінів, які дозволяють легше інтегрувати із зовнішніми джерелами даних (базами даних, зовнішніми службами кешу, інтерфейсом та іншими мікросервісами), а також має всі переваги своїх попередників, що є значною перевагою при розробці клієнта-сервер.

4. ІНСТРУКЦІЯ ПО ЕКСПЛУАТАЦІЇ МОБІЛЬНОГО ПРИСТРОЮ

4.1 Опис інтеграційних тестів

Інтеграційне тестування — одна з фаз тестування програмного забезпечення, на якій окремі модулі програмного забезпечення об’єднуються та тестуються в групі. Як правило, інтеграційне тестування виконується після тестування компонентів і до тестування системи.

Інтеграційне тестування приймає як вхідні модулі, які пройшли модульне тестування, групує їх у більші збірки, виконує тести, визначені в плані тестування для цих збірок, і представляє їх як вихідні та вхідні дані для подальшого тестування системи.

Метою інтеграційних випробувань є перевірка відповідності спроектованих вузлів функціональним вимогам, вимогам приймання та надійності. Тестування сконструйованих таким чином одиниць - комбінації, набору або групи модулів - відбувається через їхній інтерфейс, за допомогою тестування "чорного ящика".

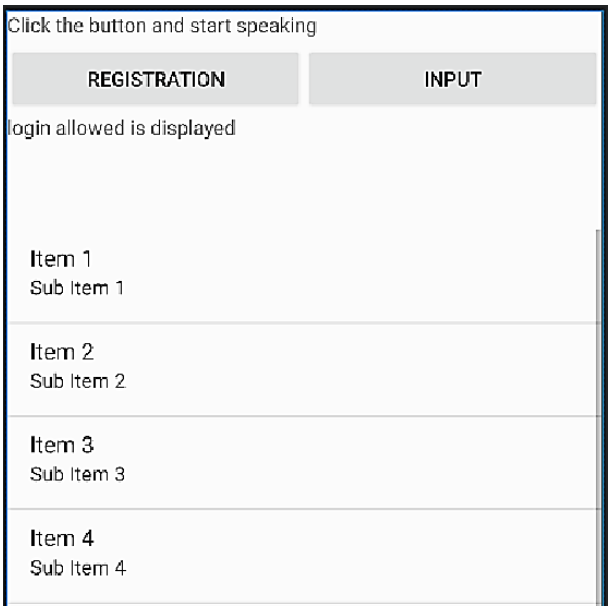

Рисунок 4.1 – Головна сторінка додатка

4.2 Інтерфейс користувача

Щоб зрозуміти, як працює програма, користувачеві часто потрібно спочатку переглянути документацію, включаючи великі комерційні програми .

Тому, підсумовуючи виконану роботу, опишемо покроково для користувача процес взаємодії з додатком.

Після відкриття програми користувач переходить на сторінку аутентифікації.

Зверху є дві кнопки : «Реєстрація» та «Вхід», користувач вибирає потрібний пункт залежно від наявності того чи був заповнений список попередньо.

Список є ключовими словами на які буде опиратися програма при ідентифікації користувача. Якщо користувач вже зареєстрований, він натискає кнопку «Війти» - та каже ключове слово.

Результат розпізнавання відображається на екрані під кнопками (Рис. 4.1)

5 АНАЛІЗ РЕЗУЛЬТАТІВ, ОТРИМАНИХ ЗА ДОПОМОГОЮ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Аналіз результатів

Розроблено програмний засіб для тестування алгоритму ідентифікації диктора, входом якого є голосовий сигнал, а виходом – результат ідентифікації – «Ідентифіковано / Помилка».

Тести проводилися на десяти користувачах. У програмі заздалегідь створено десять мовних моделей користувача, які необхідно визначити. Ключовим було обрано слово «п'ять». Користувачі говорили числа від одного до десяти, програма мала витягти це слово з голосового потоку, створити вхідну модель голосу та порівняти її з моделями голосу в базі даних. Кожен користувач намагався увійти по десять разів.

H_{FRR} - кількість відмов в аутентифікації користувача,

H_{FAR+} -кількість успішних аутентифікацій зловмисника,

$H_{FAR\,all}$ - кількість намагань аутентифікації зловмисником.

Для цінності FAR необхідно зібрати статистику. Результати наведені в таблиці 5.1.

Таблиця 5.1 - Результати випробувань

Номер моделі	Кількість успішних автентифікації порушника	Кількість помилок автентифікації користувача
1	4	2
2	5	3
3	3	4
4	3	4
5	4	2

Продовження таблиці 5.1

6	4	3
7	3	2
8	5	3
9	4	3
10	4	3

В середньому FAR = 3,9 і FRR = 2,9.

5.2 Порівняння отриманих результатів із аналогами

Таблиця 5.2 - Порівняння отриманих результатів з аналогами

Ім'я	*FAR*	*FRR*	*кількість людей*
Agnitio	2,1%	1,5%	24
Нюанс	2,6%	1,8%	33
Голосові системи	1,1%	2,3%	14
VoiceTrust	2,5%	2,8%	25

Дослідження було присвячено оцінці якості автентифікації мовця за допомогою методу динамічного перетворення часу. Було проведено ряд експериментів. Для цього були створені мовні моделі за допомогою програми і порівняні з голосовими потоками, представленими на вході.

Після тестування десяти моделей загальний висновок полягав у тому, що зі збільшенням кількості успішних автентифікацій зловмисника кількість невдалих автентифікацій користувачів зменшується. Це послаблює рівень безпеки, але спрощує використання програмного забезпечення. Тому цей спосіб більше підходить для авторизації на мобільних пристроях або персональних комп'ютерах, він досить безпечний в повсякденному житті і коли користувач не хоче витрачати багато часу на авторизацію.

6 . ЕКОНОМІЧНА ЧАСТИНА

6.1 Аналіз ситуації на ринку

Для початку проаналізуємо зміст мобільного додатка, її можливі напрямки застосування, чим запропонована ідея відрізняється від існуючих аналогів, а також основні вигоди, які може отримати користувач товару. Результати аналізу представлені у таблиці 6.1.

Таблиця 6.1. Опис ідеї проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Автоматична аутентифікація користувача за голосовим повідомленням	1. Мобільні додатки	Можливість аутентифікувати користувача за голосовим повідомленням
	2. Веб-сайти	Автоматизація процесу аутентифікації користувача за голосовим повідомленням

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів- конкурентів. Для цього спочатку проводиться аналіз попиту (таблиця 6.2).

Таблиця 6.2. Попередня характеристика потенційного ринку проекту

Показники стану ринку	Характеристика
Загальна потреба в продукції	Необхідна, але офіційних запитів на розробку немає(брак фінансування)
Можливі річні обсяги випуску в натуральних показниках	До 100 копій
Ціна одиниці продукції	350$
Річні обсяги випуску в вартісних показниках	3000 – 5000$
Динаміка ринку (якісна оцінка)	Зростає
Наявність обмежень для входу	Отримання доступу до правдивої статистичної інформації.
Показники стану ринку	Характеристика
Специфічні вимоги до стандартизації та сертифікації	Для ПЗ відсутні. Для коректної роботи - використання стандартів ISO 9126 та ISO 25010
Середня норма рентабельності в галузі (або по ринку)	87%

За попереднім оцінювання ринок не здається достатньо привабливим для входження. Але при проведенні аналізу поточного стану ринку виявлено підвищений інтерес до сфери голосової аутентифікації .

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів- конкурентів.

Надалі визначаються потенційні групи клієнтів, їх характеристики, та формується орієнтовний перелік вимог до товару для кожної групи (таблиця 6.3).

Таблиця 6.3. Характеристика потенційних клієнтів проекту

Потреба, що формує ринок	Цільова аудиторія	Особливості поведінки споживачів	Вимоги споживачів до товару
Автоматизація процесу аутентифікації користувача за голосовим повідомленням	Розробники мобільних додатків	Розробники займаються написанням програм, які не завжди: відповідають стандартам (за стильовою характеристикою, наприклад), не завжди достатньо оптимізовані, що впливає на подальше життя створених програмних продуктів – виникають проблеми, недоліки та конфлікти. Тривале вирішення проблем несумісності або критичних помилок ПЗ.	– доступна ціна; – зручність і простота використання; – мобільність
	Користувачі мобільних додатків	Основною метою створити програмний продукт та випустити його на ринок, власники програмних розробок націлені на основні завдання такі, як: швидше створити ПП і якомога вигідніше продати (більше копій, вища ціна). Тому важливо мати й можливість контролю.	– зручність І Простота використання

Таблиця 6.4. Фактори загроз

Фактор	Зміст загрози	Можлива реакція компанії
Поява конкурентів	Можлива поява конкурентів, які спроможуться створити більш якісний продукт. Можлива поява більш дешевих продуктів	Зменшення ціни з підвищенням якості при цьому, розробка удосконалень, розширення асортименту (додавання нових можливостей, нового функціоналу) збільшення точності розпізнавання, підвищення швидкості розпізнавання, зменшення тренувальної вибірки зображень
Зміни тенденцій ринку	Можлива ситуація, в якій з’явиться більш досконала програмна система від конкурентів, які значно довше на ринку.	Але можливості вирішення найпростіші - розробка нових сучасних необхідних удосконалень, тобто додання або заміна старого функціоналу на можливості розрахунку нових параметрів
Зниження репутації компанії	Можлива ситуація, коли конкуренти спроможуться на більший попит	Зміна партнерів, заключення нових контрактів, проведення рекламних та промо-акцій
Економічний спад	Відсутність попиту на товар компанії через економічну складову	Збільшення обсягів продажів, зменшення ціни; зміна цільової аудиторії

Таблиця 6.5. Фактори можливостей

Фактор	Зміст загрози	Можлива реакція компанії
Невелика кількість конкурентів	На ринку на сьогоднішній день дуже не значна кількість конкурентів, їх програмні продукти в переважній більшості вузько спеціалізовані	Розповсюджувати створений продукт, розвивати його можливості
Відповідні тенденцій ринку	ІТ-ринок на сьогоднішній день потребує, а відповідно і надає всі можливості для впровадження систем, які надаватимуть користувачам можливість розпізнавання інформаційних вкидань	Розповсюджувати створений продукт, розвивати його можливості
Можливість побудови власної репутації	Новий «гравець» на ринку має всі можливості для побудови власної репутації з «чистого листка»	Пошук замовників, можливих покупців створеного продукту, розширення бази замовників. Зарекомендувати себе, як надійну компанію. Можливо на вигідних умовах співпраці

Таблиця 6.6. Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	У чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії)
Тип конкуренції	Чиста Залежить від кількості конкурентів та якості надання ними послуг у порівнянні з послугами компанії	Покращення власного продукту через зниження ціни та підвищення якості
За рівнем конкурентної боротьби	Локальна Конкуренція на вітчизняному ринку	На вітчизняному ринку конкурентів не виявлено, а тому компанія має можливість встановлення власної бажаної ціни, та наробляти клієнтську базу. Перспектива – вихід на міжнародний рівень
За галузевою ознакою	Внутрішньогалузева Продукт націлений лише на конкретну сферу діяльності	Немає можливостей та сенсу розширювати функціонал за межі ІТ-сфери, але існує багато варіантів розвиватись всередині неї
Конкуренція за видами товарів	Марки-конкуренти Створений товар може мати конкурентів, які пропонують аналогічний товар	Зниження ціни, розширення функціональних, встановлення в державних закладах охорони здоров’я (задля популяризації методу)
За характером конкурентних переваг	Цінова Важливо за скільки продається товар, та скільки з нього прибутку	Можливе підвищення ціни на нові розробки, зниження на старі версії для заохочення покупців у порівнянні з цінами конкурентів
За інтенсивністю	Марочна Можуть з’являтись конкуренти	На ринку цільової аудиторії поки що конкурентів не виявлено. Але при виході на міжнародний ринок потрібно рекламувати кращий функціонал створеного продукту, встановлювати конкурентоспроможні ціни, та доводити свою надійність

Таблиця 6.7. Обґрунтування факторів конкурентоспроможності

Фактор конкурентоспроможності	Обґрунтування
Невелика кількість конкурентів на ринку	На вітчизняному ринку, на який для старту націлена розроблена система, конкурентів немає
Доступність створеного продукту (програмно)	Немає жорстких системних вимог, програма буде працювати навіть на застарілих ПК
Легкість і простота Використання	Зручний зрозумілий інтерфейс, створені довідка та інструкція для користувача
Відсутня потреба у постійному Супроводі	Не потребує супроводу спеціалістів і постійних доробок з боку розробника
Підключення до мережі Інтернет	Потрібен доступ до Інтернету, як у всіх аналогах
Додаткові компоненти	Немає необхідності встановлення додаткових компонентів, на відміну від деяких аналогів, які не працюють без АПК

6.2 SWOT-аналіз

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) (таблиця 6.8) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін.

На основі SWOT-аналізу розробляються альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок.

Визначені альтернативи аналізуються з точки зору строків та ймовірності отримання ресурсів (таблиця 6.9).

Таблиця 6.8. SWOT-аналіз проекту

Сильні сторони (S): – невелика кількість працівників; – молодий і перспективний колектив; – гнучка політика керівництва; – інноваційні технології	Слабкі сторони (W): – недостатньо оборотних коштів; – відсутність репутації компанії;
Можливості (O): – додаткові послуги; – вихід на нові ринки; – розширення клієнтської бази; – співробітництво з іншими компаніями	Загрози (T): – поява нових конкурентів; – зміни тенденцій попиту; – зниження репутації компанії; – економічний спад

Таблиця 6.9. Альтернативи ринкового впровадження стартап-проекту

Альтернатива ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
Вихід на нові ринки	Пошук інвесторів	1-6 місяців
Розширення виробничої лінії	Пошук інвесторів	Після виходу на ринок основного продукту, до 6 місяців

6.3 Розрахунок витрат на розробку та розрахунок собівартості

Спочатку потрібно вивести на основний ринок розроблену систему, а вже потім шукати можливості розширення програмного функціоналу для користувачів.

Кошторис витрат – це зведений план усіх витрат підприємства на плановий період виробничо-фінансової діяльності .

Кошторис на розробку бекенд частини веб-платформи передбачає такі основні витрати:

1) на основну заробітну плату;
2) на додаткову заробітну плату;
3) нарахування на заробітну плату;
4) амортизація обладнання, техніки та приміщень, що використовувались в процесі розробки;
5) на оренду;
6) витрати на матеріали, що були використані на розробку рішення;
7) на комплектуючі;
8) на силову електроенергію;
9) інші витрати.

Основна заробітна плата бекенд розробників розраховується за формулою 6.1.

$$З_о = \frac{М}{Т_р} * t, \quad (6.1)$$

де М – місячний посадовий оклад конкретного бекенд розробника, грн.;

Тр – число робочих днів у місяці, дні (21…23);

t – число днів роботи розробника.

Над програмним додатком працювали один бекенд розробник та науковий керівник. Місячний оклад бекенд розробника складає близько 15000 грн., наукового керівника, що займає посаду доцента кафедри – близько 9500 грн.

Робота з проектування, написання коду, тестування велась близько двох місяців. Науковий керівник надавав консультації один раз в тиждень.

Тоді основна заробітна плата бекенд розробника $З_{оР}$ та керівника $З_{оК}$ складає:

$$З_{оР} = \frac{15000}{22} * 44 = 30000 \text{ грн}$$

$$З_{оК} = \frac{9500}{22} * 9 = 3886 \text{ грн}$$

Кошторис витрат на основну заробітну плату наведено у таблиці 6.10.

Таблиця 6.10 – Основна заробітна плата працівників

Назва посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.	Примітка
Інженер-програміст	15000	681,8	44	30000	
Науковий керівник	9500	431,8	9	3886	Консультація раз на тиждень
Всього				33886	

Додаткова заробітна плата працівників розраховується як відсоток від основної заробітної плати у розмірі 10-12% (формула 6.2). Тоді додаткова заробітна плата складає:

$$З_Д = \frac{З_о * 10...12\%}{100\%}, \quad (6.2)$$

$$З_Д = \frac{33886 * 10}{100} = 3388{,}6 \text{ грн}$$

До статті нарахування включають відрахування на державне (обов'язкове) соціальне страхування у вигляді єдиного соціального внеску. Відрахування на соціальні заходи здійснюється від суми всіх витрат на оплату праці робітників, зайнятих безпосередньо виробництвом продукції .

Нормативи, за якими здійснюється відрахування, встановлюються на державному рівні відповідно до класів професійного ризику виробництва. Для видання програмного забезпечення ЄСВ становить 22% (формула 6.3).

$$В_{СЗ} = \frac{22\% * (З_о + З_Д)}{100\%}, \quad (6.3)$$

$$В_{СЗ} = \frac{22 * (33886 + 3388{,}6)}{100} = 8200 \text{ грн}$$

Амортизаційні відрахування по кожному пункту розраховуються прямолінійним методом (формула 5.4).

$$А = \frac{Ц}{Т_К} * \frac{Т}{12} \quad (6.4)$$

де Ц – балансова вартість пункту, що входить до підрахунку, грн.;

Тк – термін корисного використання обладнання (для ЕОМ від 2-х років);

Т – термін використання (місяці).

У роботі використовувались: один персональний комп'ютер з операційною системою Windows 10. Підставивши відповідні значення у формулу 4 отримаємо амортизаційні відрахування.

$$А = \frac{10000}{6} * \frac{2}{12} = 277{,}8 \text{ грн}$$

Витрати на матеріали визначаються за формулою 6.5. У зв'язку з особливостями збуту розробленого програмного забезпечення витрати на носії з інсталятором програмного забезпечення відсутні. До уваги беруться витратні канцелярські матеріали.

$$М = \sum_i^n Н_i \, Ц_i К_i - \sum_i^n В_i \, Ц_{Bi} \quad (6.5)$$

де $Н_i$ – витрати матеріалу і-го найменування;

$Ц_i$ – вартість матеріалу і-го найменування;

$К_i$ – коефіцієнт транспортних витрат;

$В_i$ – маса відходів матеріалу і-го найменування;

$Ц_{Bi}$ – ціна відходів матеріалу і-го найменування.

Підставивши відповідні значення у формулу 6.5, отримаємо величину витрат на матеріали:

$$М = 80 * 1 + 1{,}1 + 110 * 1 * 1{,}1 + 5 * 4 * 1{,}1 = 231 \text{ грн}$$

Витрати на матеріали зведено до таблиці 6.11.

Таблиця 6.11 – Витратні матеріали

Назва	Ціна, грн.	Кіль-кість, шт.	Коефіцієнт транспортних витрат	Вартість витраченого матеріалу, грн.
Блок паперу А4	80	1	1,1	88
Тонер чорний	110	1		121
Ручка кулькова синя	5	4		22
Всього				231

Загальні витрати є сумою всіх попередніх статей витрат.

$$В = З_о + З_Д + В_{СЗ} + А + М \quad (6.6)$$

$$В = 33886 + 3388{,}6 + 8200 + 277{,}8 + 231 = 45983 \text{ грн}$$

Отже, кошторис витрат на розробку бекенд частини програмного продукту становить 45983 грн.

Програмний продукт є специфічним товаром, що відрізняється від продуктів звичайного матеріального виробництва. Ця специфіка виявляється у формуванні витрат на виробництво та збут програмних продуктів, а не на виробництво і відтворення.

Програмний продукт буде реалізовуватись через мережу Інтернет, тому немає необхідності розраховувати вартість матеріального носія з програмним продуктом. Також до витрат необхідно включити витрати на інтелектуальну власність

$$I_В = К * I_р \quad (6.7)$$

де $I_р$ – кошти, які буде отримано від продажу однієї копії програмного продукту;

К – коефіцієнт, який враховує відповідні нарахування на заробітну плату, (k =

1,22, 22% ЄСВ).

Кошти, які планується отримувати від продажу кожної копії бекенд частини програмного продукту, становлять 5000 грн. Тоді витрати на інтелектуальну власність становлять:

$$I_B = 1{,}22 * 5000 = 6100 \text{ грн}$$

Інші витрати включають витрати пов'язані зі збутом програмного забезпечення і дорівнюють вартості реєстрації акаунту у сервісі дистрибуції програмного забезпечення Windows Store і становить 20% від ціни на продукт. Результати калькулювання всіх статей виробничої собівартості занесені до таблиці 6.12.

Таблиця 6.12 – Собівартість програмного продукту

Стаття калькуляції	Витрати, грн.
Інтелектуальна власність	6100
Інші витрати	40
Всього	6140

Розрахуємо вартість реалізації бекенд частини програмного продукту. Нижню межу ціни реалізації товару розраховують за формулою:

$$Ц_{НМ} = S_B(1 + \frac{Р}{100})(1 + \frac{\alpha_{ПДВ}}{100}) \quad (6.8)$$

де S_B – виробнича собівартість продукту, грн.;

Р – норматив рентабельності, % (Р=30…60%);

$\alpha_{ПДВ}$ – ставка податку на додану вартість, % (20%).

Отже, нижня межа ціни становить:

$$Ц_{НМ} = 6140\left(1 + \frac{50}{100}\right)\left(1 + \frac{20}{100}\right) = 11052 \text{ грн}$$

Верхню межу ціни можна розрахувати за формулою:

$$Ц_{ВМ} = Ц_{НМ} * К_{ВЯ} \quad (6.9)$$

де $К_{ВЯ}$ – відносний коефіцієнт якості ($К_{ВЯ}$. = 1.36).

Підставимо відповідні значення у формулу 5.9:

$$Ц_{ВМ} = 11052 * 1{,}36 = 15030 \text{ грн}$$

Тоді ціну реалізації можна прийняти у розмірі 15030 грн.

Аналітично критичний обсяг виробництва продукту можна визначити за формулою 5.10.

$$Q_K = \frac{ПВ}{Ц_{НМ} - ЗМ} \quad (6.10)$$

де ПВ – постійні витрати, їх прирівнюємо до витрат на розробку;

ЗМ – змінні витрати або виробничі витрати .

$$Q_K = \frac{45983}{11052 - 6140} = 9{,}3 \approx 10 \text{ копій}$$

Отже, для того, щоб досягти точки беззбитковості, необхідно продати 10 копій

ВИСНОВКИ

Розпізнавання мови широко використовується в сучасних інформаційних технологіях.

Під час дипломної роботи було проаналізовано поставлене завдання та розроблено програмний засіб ідентифікації мовлення за ключовим словом. Цільовою групою користувачів такого програмного засобу є системи розділення доступу, при авторизації на персональному пристрої, при управлінні різними пристроями тощо.

У результаті аналізу особливостей різних підходів до розробки програмних засобів ідентифікації мовлення встановлено, що найбільш вигідним у даному випадку є використання методу розпізнавання за ключовими словами на основі перетворення піддіапазону з використанням алгоритму динамічних спотворень, тобто було змінено для підвищення продуктивності.

Функціональність методу була перевірена за допомогою тестового програмного забезпечення, яке було створено в середовищі Android Studio, на мові програмування Java. У додатку було сформовано та протестовано десять мовних моделей користувача. Проаналізовано процес запуску тестового набору та його результати.

Після тестування був зроблений загальний висновок що при зростанні кількості успішник автонтифікацій зловмисника зменшується кількість відмов у автонтифікації користувача. Це послабляє рівень безпеки, однак спрощує користування програмним засобом. Він є достатньо безпечний у повсякденному використанні і коли користувач не хочу витрачати багато часу на авторизацію, що і виділяє його серед конкурентів.

Було розраховано економічну частину поставленої задачі, і для того щоб почати виходити у прибуток після розробки повноцінного мобільного додатку потрібно буде розпродати 10 копій готового продукту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Болл, Р. М. Довідник з біометрії / Р. М. Болл, Дж. Х. Коннелл, Н. К. Рата; переклад з англійської Н. Е. Агапова. - М .: Техносфера, 2007. - 352 с 13. Фролов А. В. Синтез і розпізнавання мови. Сучасні рішення / Х. В. Фролов. - М.: Зв'язок, 2003.- 216 с.

2. Алейник С.В., Матвєєв Ю.М., Раєв А.М. Спосіб оцінки рівня відсікання мовного сигналу // Науч.техн. Газетні інформаційні технології, механіка та оптика. 2012. № 3 (79). С. 79 - 83.

3. Бєлих І.Н., Капустін А.В., Козлов А.В., Лоханова А.І., Матвєєв Ю.Н., Пеховський Т.С., Симончик К.К., Шулипа А.К. Система оголошення голосів для NIST SRE 2010 // IT and applications. 2012. Т. 6,

№ 1. С. 91-98.

4. S. Cho, C. Han, D. H. Han, and H.-I. Kim. Web-based keystroke dynamics identity verification using neural network // Journal of Organizational Computing and Electronic Commerce. 2000. №10 (4). P. 295-307.

5. D.-T. Lin. Computer-access authentication with neural network based keystroke identity verification // International Conference Neural Networks. 1997. №1. P. 174-178.

6. M. Obaidat, Sadoun B. Verification of computer users using keystroke dynamics // in Systems, Man, and Cybernetics, Part B: Cybernetics. 1997. №27(2). P. 261- 269.

7. Obaidat, Macchiarolo D. An online neural network system for computer access security // in Industrial Electronics, IEEE Transactions. 1993. № 40(2). P. 235-242. Kawalerowicz, M. and Berntson, C. Continuous Integration in .NET. — Manning, 2011. — 303 p.

8. Nechiporenko A.S., Gubarenko E.V., Gubarenko M.S. (2019) Authentication of users of mobile devices by their motor reactions. Telecommunications and Radio Engineering, 78 (11). pp. 987-1003. DOI: 10.1615/TelecomRadEng.v78.i11.60

9. Priority directions of science and technology development Abstracts of I International Scientific and Practical Conference Kyiv, Ukraine 27-29 September 2020, p. 207.

10. Chernomorec A.A. Prohorenko E.I., Goloshhapova A.A., 2009. About properties of subband matrices eigenvectors. Nauchnye vedomosti BelGU. Istoriya. Politologiya. Ekonomika. Informatika. [Belgorod State University Scientific Bulletin. History Political science Economics Information technologies], p. 122-128.

11. Selina Chu and Eamonn Keogh and David Hart and Michael Pazzani Iterative Deepening Dynamic Time Warping for Time Series, In Proc 2 nd SIAM International Conference on Data Mining, 2002.

12. Ann Chotirat and Ratanamahatana Eamonn and Keogh Everything you know about Dynamic Time Warping is Wrong, The 31st Annual International Symposium on Forecasting, 2004.

13. Georgios N. Banavas and Sue Denham and Michael J. Denham Fast Nonlinear Deterministic Forecasting Of Segmented Stock Indices Using Pattern Matching And Embedding Techniques, Society for Computational Economics, 2000.

14. John Aach and George M. Church Aligning Gene Expression Time Series With Time Warping Algorithms, 2001.

15. Ing-Jr Ding, Chih-Ta Yen, Yen-Ming Hsu. Developments of Machine Learning Schemes for Dynamic Time-Wrapping-Based Speech Recognition // Mathematical Problems in Engineering. 2013.

16. Daniel Ramage. Hidden Markov Models Fundamentals // CS229 Section Notes. 2007.

17. Campbell W., Campbell J., Reynolds D., Jones D., Leek T. Phonetic speaker recognition with support vector machines // Advances in Neural Information Processing Systems. 2004. Vol.16. MIT Press, Cambridge, MA

ДОДАТОК А

Список мобільних додатків

VoiceRecognition.java

```
package com.example.diplom;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.speech.RecognizerIntent;
import android.speech.SpeechRecognizer;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import java.util.ArrayList;

public class VoiceRecognition extends Activity implements OnClickListener {

        public TextView mText;
        public ListView mList;
        public SpeechRecognizer sr;
        public static final String TAG = "VoiceTest";

        private static final int REQUEST_CODE = 1234;

        @Override
        public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button buttonList = (Button) findViewById(R.id.buttonList);
        Button buttonIntent = (Button) findViewById(R.id.buttonIntent);

        buttonList.setOnClickListener(this);
        buttonIntent.setOnClickListener(this);

        mText = (TextView) findViewById(R.id.textView1);
        mList = (ListView) findViewById(R.id.list);

        initSpeechRecognizer();
    }

    public void onClick(View v) {
        if (v.getId() == R.id.buttonList) {
            Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
            // Intent intent = new Intent(RecognizerIntent.ACTION_WEB_SEARCH);

        intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);

        //intent.putExtra(RecognizerIntent.EXTRA_CALLING_PACKAGE,"com.tkjelectronics.voice");
            //intent.putExtra(RecognizerIntent.EXTRA_MAX_RESULTS,5);
            sr.startListening(intent);
        }
```

```
                if (v.getId() == R.id.buttonIntent) {
                    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
                //Intent intent = new Intent(RecognizerIntent.ACTION_WEB_SEARCH);

intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
                    intent.putExtra(RecognizerIntent.EXTRA_PROMPT, "Voice recognition Demo...");
                    startActivityForResult(intent, REQUEST_CODE);
                }
            }

            /**
     * Handle the results from the voice recognition activity.
     */
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
       if (requestCode == REQUEST_CODE && resultCode == RESULT_OK) {
          // Populate the wordsList with the String values the recognition engine thought it heard
          ArrayList<String> matches = data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
          float[] value = data.getFloatArrayExtra(RecognizerIntent.EXTRA_CONFIDENCE_SCORES);
          mText.setText("Results: " + String.valueOf(matches.size()));

          if(value != null) { // EXTRA_CONFIDENCE_SCORES wasn't added until API level 14
              String[] combined = new String[matches.size()];
```

```
            for(int i = 0; i < matches.size(); i++) // The size of the data and value is the same
                combined[i] = matches.get(i).toString() + "\nScore: " + Float.toString(value[i]);
            mList.setAdapter(new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1,combined));
        } else
            mList.setAdapter(new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1,matches));
      }
      super.onActivityResult(requestCode, resultCode, data);
  }

  private void initSpeechRecognizer() {
            if(sr == null) {
                sr = SpeechRecognizer.createSpeechRecognizer(this);
                if (!SpeechRecognizer.isRecognitionAvailable(getApplicationContext())) {
                    Toast.makeText(getApplicationContext(),"Speech Recognition is not available",Toast.LENGTH_LONG).show();
                    finish();
                }
                sr.setRecognitionListener(new VoiceRecognitionListener(this));

            }
  }
  @Override
  protected void onResume() {
            super.onResume();
            initSpeechRecognizer();
  }
```

```
    @Override
                protected void onDestroy() {
                if (sr != null) {
                sr.stopListening();
                sr.cancel();
                sr.destroy();
        }
                super.onDestroy();
    }
}
```

VoiceRecognitionListener.java

```
package com.example.diplom;

import java.util.ArrayList;

import android.os.Bundle;
import android.speech.RecognitionListener;
import android.speech.SpeechRecognizer;
import android.widget.ArrayAdapter;

class VoiceRecognitionListener implements RecognitionListener {
                VoiceRecognition mVoiceRecognition;
                //private static final String TAG = "VoiceRecognitionListener";

                public VoiceRecognitionListener(VoiceRecognition instance) {
                    mVoiceRecognition = instance;
```

```
        }
        public void onResults(Bundle data) {
            //Log.d(TAG, "onResults " + data);
            ArrayList<String> matches =
data.getStringArrayList(SpeechRecognizer.RESULTS_RECOGNITION);
            float[] value =
data.getFloatArray(SpeechRecognizer.CONFIDENCE_SCORES);
            mVoiceRecognition.mText.setText("Results: " +
String.valueOf(matches.size()));

            if(value != null) { // CONFIDENCE_SCORES wasn't added until
API level 14
        String[] combined = new String[matches.size()];
      for(int i = 0; i < matches.size(); i++) // The size of the data and value is the
same
        combined[i] = matches.get(i).toString() + "\nScore: " +
Float.toString(value[i]);
      mVoiceRecognition.mList.setAdapter(new
ArrayAdapter<String>(mVoiceRecognition,
android.R.layout.simple_list_item_1,combined));
    } else
        mVoiceRecognition.mList.setAdapter(new
ArrayAdapter<String>(mVoiceRecognition,
android.R.layout.simple_list_item_1,matches));
        }

        public void onBeginningOfSpeech() {
            //Log.d(TAG, "onBeginningOfSpeech");
            mVoiceRecognition.mText.setText("Sounding good!");}
        public void onBufferReceived(byte[] buffer) {
            //Log.d(TAG, "onBufferReceived");}
```

```
            public void onEndOfSpeech() {
                //Log.d(TAG, "onEndofSpeech");
                mVoiceRecognition.mText.setText("Waiting for result...");}
            public void onError(int error) {
                //Log.d(TAG, "error " + error);
                mVoiceRecognition.mText.setText("error " + error);}
            public void onEvent(int eventType, Bundle params) {
                //Log.d(TAG, "onEvent " + eventType);}
            public void onPartialResults(Bundle partialResults) {
                //Log.d(TAG, "onPartialResults");}
            public void onReadyForSpeech(Bundle params) {
                //Log.d(TAG, "onReadyForSpeech");}
            public void onRmsChanged(float rmsdB) {
                //Log.d(TAG, "onRmsChanged");}
}
```

Main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical" >

   <TextView
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:paddingBottom="4dip"
      android:text="@string/buttonText" />

   <LinearLayout
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <Button
            android:id="@+id/buttonList"
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="Registration" />

        <Button
            android:id="@+id/buttonIntent"
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="Input" />
    </LinearLayout>

    <TextView
        android:id="@+id/textView1"
        android:layout_width="match_parent"
        android:layout_height="82dp"
        android:text="login allowed is displayed"/>

    <ListView
        android:id="@+id/list"
        android:layout_width="match_parent"
        android:layout_height="112dp"
        android:layout_weight="1" />

</LinearLayout>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.diplom"
    android:versionCode="1"
    android:versionName="1.0">

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="30" />

    <uses-permission android:name="android.permission.RECORD_AUDIO" />

    <application android:label="VoiceRecognitionDemo"
        android:allowBackup="true" >
        <activity android:name="com.example.diplom.VoiceRecognition"
            android:label="@string/app_name"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
             </intent-filter>
        </activity>
    </application>
</manifest>
```