МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

другого (магістерського) рівня вищої освіти

на тему: **«Система розпізнавання голосових даних із використанням алгоритмів машинного навчання»**

Виконав: студент групи Іт-62

Спеціальності 126 «Інформаційні системи та технології»
(шифр і назва)
Владика Назар Михайлович
(Прізвище та ініціали)

Керівник: к.т.н., в.о. доцента Падюка Р.І.
(Прізвище та ініціали)

Рецензент:
(Прізвище та ініціали)

ДУБЛЯНИ-2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Другий (магістерський) рівень вищої освіти
Спеціальність 126 «Інформаційні системи та технології»

«ЗАТВЕРДЖУЮ»

Завідувач кафедри_____

д.т.н., проф. А.М. Тригуба
«____»_____2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу студенту

Владиці Назару Михайловичу

1. Тема роботи: «Розробка системи розпізнавання голосових даних із використанням алгоритмів машинного навчання»

Керівник роботи: Падюка Роман Іванович, в.о. доцента
затверджені наказом по університету від 28 квітня 2023 року № 133/к-с.
2. Строк подання студентом роботи 10.01.2024 р.
3. Вихідні дані до роботи: *набір голосових даних; алгоритми машинного навчання; методика дослідження моделей машинного навчання.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)
Вступ.
1. Аналіз стану розпізнавання голосових даних та завдання кваліфікаційної роботи.
2. Особливості вирішення задач класифікації та вибір методів машинного навчання для розпізнавання голосових даних
3. Результати розробки системи розпізнавання голосових даних
4. Охорона праці та безпека у надзвичайних ситуаціях.
5. Визначення економічної ефективності запропонованої системи.
Висновки та пропозиції.

Список використаної літератури.

5. Перелік ілюстраційного матеріалу (з точним зазначенням обов'язкових слайдів): *Мета проектування; актуальність теми роботи; практична цінність; дерево проблем та дерево цілей; технології і інструменти які використовуються для реалізації системи; робота рекурентних нейронних мереж; праклад роботи системи; висновки.*

6.Консультанти з розділів:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1, 2, 3, 5	*Падюка Р.І., в.о.доцента кафедри інформаційних технологій*		
4	*Городецький І.М., доцент кафедри управління проектами та безпеки виробництва*		

7. Дата видачі завдання 01 травня 2023 р.

Календарний план

№ з/п	Назва етапів кваліфікаційної роботи	Терміни виконання етапів роботи	Примітка
1	*Написання першого розділу*	01.05.-30.05.23	
2	*Виконання другого розділу та аркушівілюстраційного матеріалу до нього*	01.06.-30.06.23	
3.	*Виконання третього розділу та аркушівілюстраційного матеріалу до нього*	01.09.-30.09.23	
4.	*Написання розділу «Охорона праці та безпека унадзвичайних ситуаціях»*	01.10.-30.10.23	
5.	*Оцінення ефективності запропонованої системи*	01.11.-30.11.23	
6.	*Завершення оформлення розрахунково-пояснювальноїзаписки та аркушів ілюстраційного матеріалу*	01.12.-30.12.23	
7.	*Завершення роботи вцілому*	01.01.-10.01.24	

Студент _____Владика Н.М.
(підпис)

Керівник роботи _____Падюка Р.І.
(підпис)

УДК 004.9 : 631.1

Розробка системи розпізнавання голосових даних із використанням алгоритмів машинного навчання.

Владика Н.М. Кафедра інформаційних технологій – Дубляни, ЛНУП, 2024. Кваліфікаційна робота: 75 с. текст. част., 38 рисунків, 7 таблиць, 13 арк.ілюстраційного матеріалу, 23 джерел і 1 додаток.

Кваліфікаційна робота присвячена проектуванню та розробці системи розпізнавання емоцій людини з використанням штучних нейронних мереж.

У вступній частині обгрунтовано актуальність дослідження, визначенно мету, об'єкт, придмет, завдання, методи наукового дослідження кваліфікаційної роботи та описано теоретичну та практичну цінність роботи

В першому розділі описано об’єкт дослідження проектованої системи та здійснено постановку задачі на бакалаврську роботу. А також здійснено огляд літературних джерел та проаналізовано наявні підходи до вирішення проблеми.

У другому розділі проведено системний аналіз розроблюваної системи та побудовано дерева проблеми та цілей. Та описано алгоритми роботи системи у вигляді різних діаграм.

У третьому розділі описано основні моменти реалізації модулів та інтерфейсу системи, що реалізується. Детально було проведено тестування якості розробленої системи, адже саме від коректності її роботи і залежить те, чи будуть нею користуватись в майбутньому.

У четвертому розділі описано охорону праці та безпеку у надзвичайних ситуаціях.

У п’ятому розділі зроблено розрахунок вартості системи та проведено оцінку економічної ефективності системи.

У висновках підсумовані результати теоретичного та практичного дослідження кваліфікаційної роботи.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Перш ніж слова були написані, люди навчилися їх вимовляти. В повсякденному житті людина може говорити 150 слів за хвилину, але зазвичай може вводити лише 40 слів протягом того ж часу. Клавіатура та миша все ще залишаються найпопулярнішими способами взаємодії з комп'ютером.

Люди все більше звикли передавати інформацію словами, жестами та виразами обличчя. Завжди зручніше питати пошуковик голосом, ніж вводити текст, чи це ви на вулиці чи в автомобілі.

Системи розпізнавання мови вже увійшли в наше щоденне життя, дозволяючи комп'ютерам та додаткам слухати голос користувача і перетворювати його в текст, що підвищує продуктивність.

Конвертація розмови ніколи не є абсолютно точною при перетворенні на текст. Це новий концепт для програмістів, які звикли працювати з детермінованими системами. Це створює кілька проблем, специфічних для голосових технологій. Однією з найбільш складних проблем сьогодні є розпізнавання мови.

Проблема полягає в технічній складності, а також її включенні в наше повсякденне життя. Хоча сучасні цифрові асистенти здатні розуміти англійську, вони не є інтерфейсами розмови, які очікують постачальники технологій. Крім того, обмежена кількість кінцевих товарів обмежує розпізнавання мови.

Враховуючи вищевикладене, зроблено висновок, що тема кваліфікаційної роботи «**Система розпізнавання голосових даних із використанням алгоритмів машинного навчання**» є актуальною на момент написання даної роботи.

Метою: цієї кваліфікаційної роботи є розробка системи для розпізнавання голосових команд із застосуванням машинного навчання, що дозволяє точніше розпізнавати голос у поданому файлі з форматом .wav або при записі аудіофайлів навіть у шумному середовищі з неякісними мікрофонами.

Щоб досягти мети кваліфікаційної роботи, необхідно виконати наступні **завдання**:

1) Вивчити та проаналізувати діючі системи та методики розпізнавання звукових даних;

2) На основі аналізу виявити недоліки досліджуваних систем і методів;

3) Провести моделювання архітектури потенційних рішень виявлених труднощів у системах розпізнавання звукових даних;

3) Розробити систему яка відповідатиме завданню;

4) Зробити висновки та запропонувати рішення для вирішення або зменшення негативного впливу на систему проблем, виявлених шляхом дослідження.

Об’єкт дослідження: Процеси розпізнавання звукового сигналу (природної мови) за допомогою шутчних нейронних мереж.

Предмет дослідження: Методи і алгоритми нейронних мереж для розпізнавання природної мови людини.

Практична цінність: отриманих в роботі результатів полягає в тому, що розроблений модифікований спосіб дозволяє покращити процес розпізнавання голосових команд. Крім того, запропонована система може бути використана у різних сферах для полегшення роботи з комп’ютером. Розроблена в роботі програмна реалізація системи для розпізнавання голосових команд може бути використана для голосового введення на різних пристроях, також для автоматичного створення субтитрів для відео.

РОЗДІЛ 1
АНАЛІЗ СТАНУ ТЕОРІЇ ТА ПРАКТИКИ СТВОРЕННЯ ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Актуальність задачі розпізнавання та класифікації звукових даних

Коливальний рух частинок у середовищі, який хвилеподібно поширюється в середовищах, таких як рідини, тверді тіла або гази, і сприймається слуховим апаратом, називають звуком. Термін "звук" також відноситься до вібрацій, відчуття яких люди та тварин [2]. У цьому контексті це порушення одного з вищезазначених параметрів, яке поширюється з певною частотою. Ці рухи можуть бути виявлені слуховим апаратом людини в певному частотному діапазоні. Велика більшість слухових апаратів тварин може відчути звукові вібрації в набагато ширшому частотному діапазоні. Загалом цей термін вказує на процес поширення вібрацій у середовищі з різними фізичними характеристиками, при якому використовується пружна сила для повернення вихідного розташування частинок, які були збуджені до стану спокою. Хвильові збурення, визначені звуком, об'єктивно реальні та існують незалежно від того, як їх відчуває будь-яка жива істота.

Акустика - це галузь науки, яка вивчає закони сприйняття, формування та передачі звукових хвиль в різних середовищах.

Майже всі природні явища супроводжуються шумом, які сприймаються та розпізнаються слуховими апаратами тварин та людей і служать засобом комунікації та орієнтації в просторі [3].

Звуки, які мають конкретне значення в певному діапазоні, зазвичай неприємні або небажані, наприклад, спів птахів, ліра та інші гармонійні звуки, згідно з характеристиками сприйняття звукових вібрацій людським вухом, розділяються на гармонійні (приємні) звуки, такі як спів птахів, ліра та інші гармонійні звуки, а також звуки, які мають певний зміст в певному діапазоні, зазвичай неприємні або небажані, що, як правило, означає шум.

Шум, часто відомий як акустичний шум, - це коливання частинок оточуючого середовища, які людське вухо сприймає як небажані сигнали. З точки зору акустики, шум визначається як нестабільні або непередбачувані акустичні коливання з випадковими змінами амплітуди та частоти [4].

Загалом шуми класифікуються за джерелом їх походження, а саме [5]:

1) Шуми, що виникають у газоподібному середовищі, називаються аеродинамічними;

2) шуми, що виникають у рідкому середовищі, називаються гідродинамічними;

3) шуми, що виникають у результаті впливу змінних магнітних сил, що викликають небажане збудження електромеханічних елементів в приладах, називаються електромагнітними;

4) шуми, що виникають в результаті ударів у з'єднаннях деталей, коливань поверхні називають поверхневими.

Діапазон частот шумів також класифікується за типами:

1) Низькі частоти до 400 Гц;

2) середня частота від 400 до 1000 герц;

3) висока частота понад 1000 герц.

У вузькоспеціалізованих дисциплінах, таких як електроніка та акустика, існують концепції кольорів шуму, згідно з якими шумовому сигналу приписується певний колір на основі його статистичних і спектральних якостей.

Спектральна щільність, яка описує розподіл потужності за частотою, є однією з них. Часто спектри шуму в акустиці поділяють на чотири кольори: білий шум, рожевий шум, червоний (коричневий) шум і сірий шум. Іноді виявляють інші варіації [5].

Білий шум – це безперервний шум з рівномірно розподіленими спектральними компонентами по всьому діапазону частот (рисунок 1.1). З іншого боку, білий шум можна визначити як будь-який шум з однаковою (або майже однаковою) спектральною щільністю у всьому діапазоні частот.

Біле світло, яке охоплює електромагнітні хвилі всіх частот у видимому спектрі електромагнітного випромінювання дало йому таку назву [6].

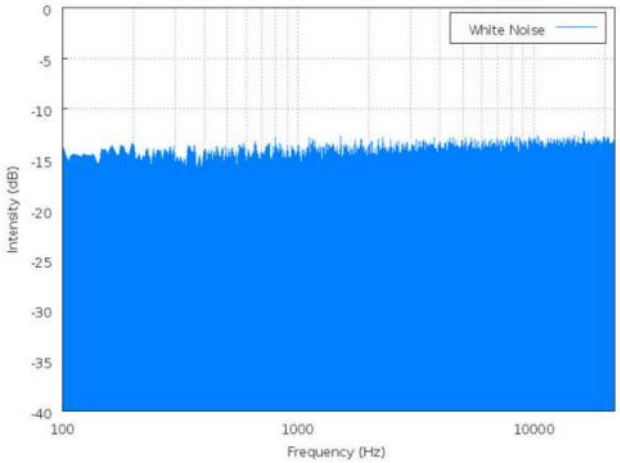

Рисунок 1.1 – Спектр білого шуму [6]

Спектральна щільність рожевого шуму змінюється з частотою f за правилом 1/f (рисунок 1.2). Будь-який шум, спектральна щільність якого падає зі зменшенням частоти, називається рожевим шумом [7].

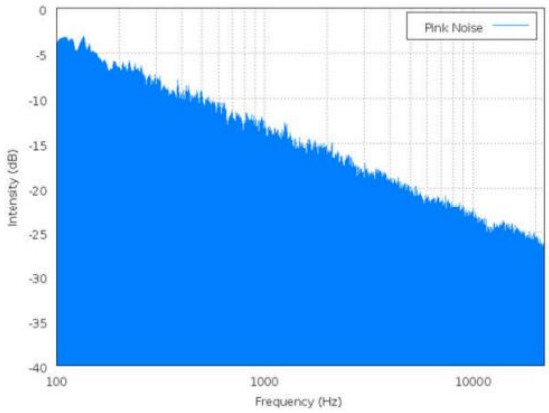

Рисунок 1.2 – Спектр рожевого шуму [6]

Броунівський шум (червоний шум) – шум іноді називають броунівський шум, оскільки звук змінюється випадковим чином від однієї секунди до наступної. Червоний шум має спектральну щільність, пропорційну 1/f2, де f – частота (рисунок 1.3) [8].

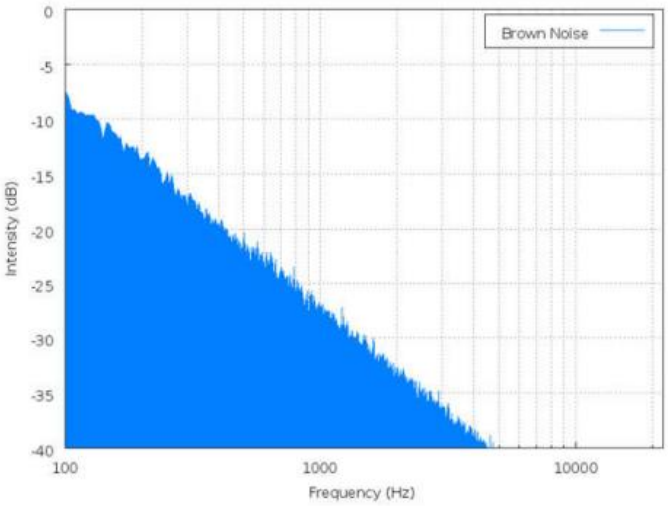

Рисунок 1.3 – Спектр червоного шуму

Сірий шум – це шумовий сигнал, який відповідає акустичній кривій з постійною гучністю на всіх частотах, тобто має однакову гучність для людського вуха на всіх частотах (рисунок 1.4) [9].

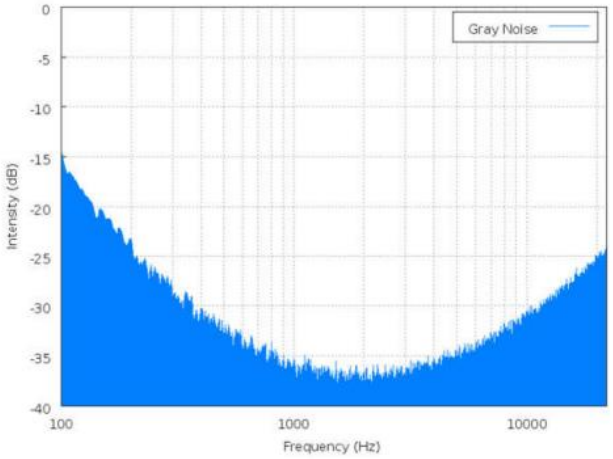

Рисунок 1.4 – Спектр сірого шуму

Хоча розпізнавання звуку та людська мова все більше і більше інтегруються в наше повсякденне життя, вони також стають все простішими, що значно полегшує спілкування за допомогою голосового введення, пошук інформації за допомогою звукових шаблонів, керування широким спектром пристроїв, і навіть розважатися з ними. На жаль, точне розпізнавання людської мови – це не тільки трудомістка операція, але й вимагає достатньо чіткого та чистого людського голосу, чого важко досягти з огляду на постійний шквал шуму.

1.2 Постановка задачі дослідження

Кінцеві користувачі систем розпізнавання аудіоданих і голосового введення, з іншого боку, очікують високої точності розпізнавання команд і швидкого реагування системи.

Проте поєднання вищеописаних явищ та шуму, а також кожного з них окремо, істотно ускладнює процес виявлення людської мови. З цього можна зробити висновок, що тема «***Система розпізнавання голосових даних із використанням алгоритмів машинного навчання***» зараз є важливою.

Результати показують, що робота зі створення штучних нейронних мереж для ідентифікації голосових команд, які б покращили існуючі системи розпізнавання мовлення в цілому, зараз є актуальною. Одним з рішень для подолання цієї проблеми є застосування методів шумозаглушення на основі штучних нейронних мереж для побудови методу розпізнавання голосових команд, що покращить точність розпізнавання голосових команд поточними системами.

Як наслідок оцінки актуальності створення методик та порівняння існуючих еквівалентів, може виникнути питання про те, чого не вистачає нинішнім способам і системам і що можна зробити для підвищення ефективності розпізнавання команди. Тому що втручатися в поточні алгоритми ASR важко, а розробити власні майже неможливо через необхідність величезної кількості даних для навчання та такої ж можливості запропонувати навчальні системи.

Отже, метою цього дослідження є надання методики, яка використовує сучасні рішення ASR для підвищення точності їх роботи.

1.3 Огляд існуючих підходів рішення задачі

Автоматичне розпізнавання мовлення (ASR) використовує різні методи для трансформації голосового сигналу на текст. Декілька типів ASR включають виокремлене слово, безперервне мовлення, комбіноване слово та спонтанне мовлення.

Виокремлене слово передбачає, що висловлювання містить лише окремі слова та фрази з мовчазними інтервалами. Безперервне мовлення дозволяє говорити безперервно, а ASR розпізнає зміст мовлення. Комбіноване слово

враховує різні твердження з мінімальними паузами. Спонтанне мовлення є природним мовленням, не підготовленим спеціально для вивчення.

Класифікація звуків може ґрунтуватися на обструктивному (з та без інтерференції) або гучному/приглушеному процесі. При цьому голосні та приголосні розрізняються за наявності інтерференції та вираженою або приглушеною вибуховою силою.

У цьому розділі описано кілька методів розпізнавання голосу, включаючи багато фундаментальних підходів до розпізнавання мовлення:

- акустично-фонетичний метод;
- метод розпізнавання образів;
- метод штучного інтелекту;
- метод нейронної мережі;
- метод прихованої моделі Маркова (HMM) та моделі гаусової суміші (GMM).

1.3.1 Акустично-фонетичний підхід

Дослідники використовують акустико-фонетичний підхід (рисунок 1.5), також відомий як акустичне моделювання, для розуміння фонетичних норм та їх застосування в системах розпізнавання мовлення (ASR). Цей метод є ключовим у сучасних статистичних ASR та фокусується на вивченні фонетичних одиниць та їх взаємодії в різних мовленнєвих контекстах.

Етапи акустико-фонетичного методу в ASR включають збір голосових даних, екстракцію характеристик, машинне навчання, тренування моделі та розпізнавання голосу. Цей підхід дозволяє дослідникам вивчати природу мовних сигналів та створювати ефективні системи розпізнавання для різних мов та акцентів.

Акустико-фонетичний метод особливо корисний для категоризації акценту, виявлення мовленнєвої активності та вибору голосової мелодії. Він дозволяє дослідникам аналізувати фонетичні одиниці, такі як алофони та морфеми, для підвищення точності ASR.

Рисунок 1.5 - Схема акустично-фонетичного методу

1.3.2 Підхід розпізнавання образів

Техніка розпізнавання образів включає два основні етапи: вивчення шаблонів і порівняння шаблонів. У цьому підході використовується математична основа та послідовне представлення лінгвістичних шаблонів для ефективного порівняння.

Акустико-фонетичний метод є ключовим у цьому підході і зосереджується на вивченні фонетичних одиниць та їх взаємодій в різних контекстах. Застосування ймовірнісних моделей, таких як HMM, SVM, DTW, VQ, дозволяє ефективно вирішувати завдання розпізнавання голосу.

1) Векторне квантування (VQ) - Цей підхід використовує циркулюючі прототипи векторів для моделювання функцій ймовірності. Він здатний стискувати дані, використовувати їх для виправлення втрат і кластеризації, а також для розпізнавання шаблонів.

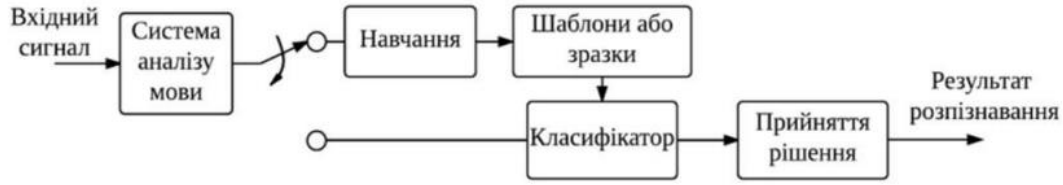

Рисунок 1.6 - Схема підходу розпізнаванням образів

2) Метод використання опорних векторів (SVM). Це одна з найпотужніших технологій розпізнавання образів. SVM використовує лінійні і нелінійні гіперплощини для класифікації даних. Важливо враховувати, що для використання в реальному часі, дані повинні бути перетворені у вектори фіксованої довжини. Використання функції максимальної відповідності допомагає узагальнити класифікатор і контролювати його складність.

1.3.3 Підхід штучного інтелекту (підхід на основі знань)

Цей метод представляє собою синтез акустично-фонетичної техніки та підходу до розпізнавання образів. Штучний інтелект націлено автоматизувати процес розпізнавання, інспіруючись тим, як людина використовує свій інтелект для сприйняття, аналізу та прийняття рішень на основі вимірюваних акустичних характеристик.

У цьому підході широко використовуються висококваліфіковані системи. Експертне знання про варіації мовлення ручно кодується в системі за допомогою методів, що базуються на знаннях. Це важливо для точного моделювання мовленнєвих змін; однак здобуття та ефективне використання такого експертного знання виявляється складним завданням. У результаті цей метод визнано недосяжним, і замість нього використовується техніка автоматизованого навчання.

1.3.4 Підхід ANN

Техніка Підхід ANN (Artificial Neural Network) в розпізнаванні мовлення базується на імітації структури та функціонування нейронних мереж у людському мозку. ANN використовується для розпізнавання патернів у великій кількості вхідних даних та для навчання моделі розпізнавати вказані образи.

Основні характеристики підходу ANN:

Штучні нейрони: Мережа складається з штучних нейронів, які приймають вхідні сигнали, обробляють їх та передають результати на вихід.

Шари нейронів: ANN має різні шари нейронів, такі як вхідний, прихований та вихідний. Вхідний шар отримує сигнали, приховані шари обробляють дані, а вихідний шар генерує результати.

Ваги та зв'язки: Кожен зв'язок між нейронами має вагу, яка визначає його вплив на передачу сигналу. Ваги навчаються під час тренування мережі.

Функції активації: Нейрони використовують функції активації для моделювання нелінійних залежностей між вхідними та вихідними даними.

Навчання: ANN навчається на великій кількості даних, адаптуючи ваги зв'язків для вірного розпізнавання патернів.

Підхід ANN в розпізнаванні мовлення використовується для розв'язання завдань класифікації та регресії. Проте, для успішного застосування цього підходу потрібна велика кількість даних для тренування та налаштування параметрів мережі.

1.3.5 Підхід на основі HMM

Підхід на основі HMM (Hidden Markov Model) в розпізнаванні мовлення є стохастичною стратегією, що використовуємо для моделювання непомітних (прихованих) станів системи, які генерують спостереження. Такий підхід широко застосовується в акустичному розпізнаванні мовлення.

Основні характеристики підходу на основі HMM:

Приховані стани: Модель мовлення розбивається на послідовність прихованих станів, які не можна спостерігати безпосередньо. Кожен стан асоційований зі звуковими патернами в мовленні.

Спостереження: Кожен стан генерує певний вихід, який можна спостерігати. У випадку розпізнавання мовлення, це може бути звуковий вектор або фонема.

Модель переходів: Визначає ймовірності переходу від одного прихованого стану до іншого. Ці ймовірності визначаються під час тренування моделі на великій кількості даних.

Модель емісії: Визначає ймовірності генерації конкретного спостереження кожним прихованим станом.

Алгоритм Вітербі: Використовується для пошуку найбільш ймовірного шляху прихованих станів, що відповідає спостереженню.

Підхід на основі HMM дозволяє ефективно моделювати часові залежності та нелінійні зв'язки в акустичних даних. Він застосовується для розпізнавання фраз, слів та фонем в мовленні. Тренування моделей HMM вимагає великої кількості акустичних даних для досягнення високої точності розпізнавання.

1.4 Архітектура штучних нейронних мереж для розпізнавання голосових даних

Штучні нейронні мережі (ШНМ або ANN) (рис. 1.7) — це комп'ютерні системи, що моделюються за аналогією з біологічними нейронними мережами, які складають мозок тварин. Ці системи вивчають завдання, покращуючи свою продуктивність на прикладах і, в більшості випадків, без спеціального програмування завдань. Наприклад, вони можуть навчитися розпізнавати фотографії зображень, на яких зображені кішки, вивчаючи приклади із позначками "кішка" та "не кішка", а потім використовувати отримані знання для ідентифікації кішок на інших зображеннях. Це досягається без попередніх знань про конкретні риси кішок, таких як шерсть, хвост, вуса. Штучні нейрони складаються з взаємоз'єднаних вузлів, утворюючи Штучну Нейронну Систему (аналогічну біологічним нейронам у мозку тварин). Кожне з'єднання між штучними нейронами, аналогічно синапсу, може передавати сигнали від одного до іншого. Сигнали обробляються штучним нейроном, який їх отримує, і подаються до інших штучних нейронів, з якими він пов'язаний. У більшості реалізацій ШНМ сигнал на межі штучного нейрона представлений дійсним числом, а вихід кожного штучного нейрона генерується за допомогою нелінійної функції його входів.

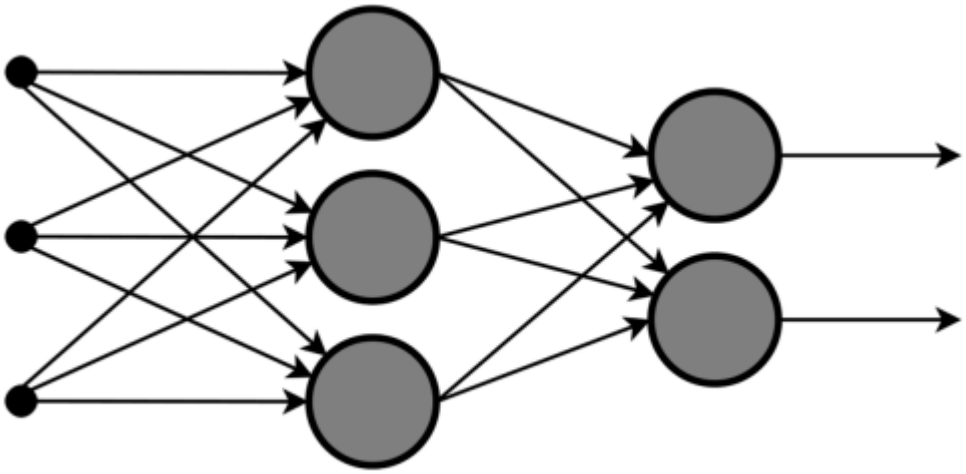

Рисунок 1.7 – Загальна архітектура ШНМ (ANN)

Вага часто змінюється під час тренування для штучних нейронів і їхніх зв’язків. Вони регулюють інтенсивність сигналів передачі, збільшуючи або зменшуючи їх. Штучні нейрони можуть мати поріг, при якому сигнал передається лише в разі, якщо сукупний сигнал перевищує цей поріг. Зазвичай використовуються шари штучних нейронів. (рис. 1.8).

Різні рівні можуть здійснювати різні види модифікацій введення. Сигнали переміщуються від першого (вхідного) шару до кінцевого (вихідного) шару, можливо, після кількох проходів через проміжні шари.

Основна мета підходу ANN полягала в тому, щоб вирішувати проблеми так само, як і людський мозок. З часом акцент перемістився на відповідність певним розумовим здібностям, що призвело до біологічних аналогій. Розпізнавання образів в комп'ютерному зорі, розпізнавання голосу, машинний переклад, фільтрація соціальних мереж, гра в настільні та відеоігри, а також медична діагностика – це лише деякі з завдань, для вирішення яких використовуються штучні нейронні мережі. [15].

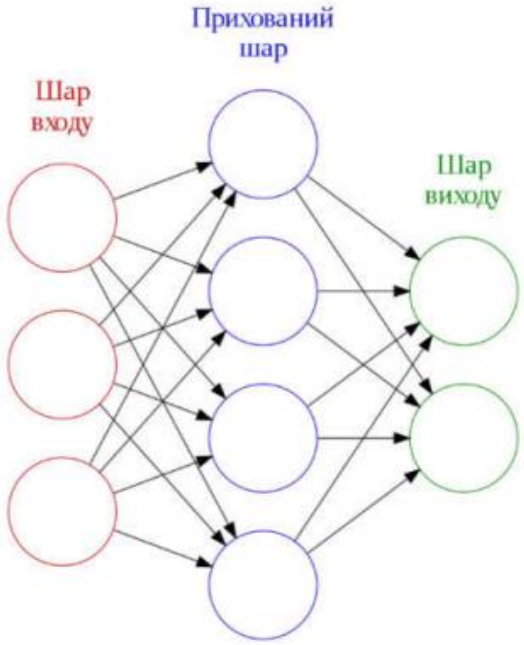

Рисунок 1.8 — ANN (ШНМ) з кількома шарами

Звичайні нейронні мережі та періодичні нейронні мережі є двома основними типами штучних нейронних мереж, які використовуються в ASR.

RBF, яка стає все більш популярною нейронною мережею з широким спектром застосувань, є головним конкурентом багатошарового персептрона.

Мережі RBF були мотивовані традиційними способами категоризації статистичних моделей.

Вхідний вектор X використовується в RFB для введення всіх фундаментальних радіальних функцій, кожна зі своїм набором параметрів.

Вихід задає лінійну комбінацію вихідних даних фундаментальних радіальних функцій (рисунок 1.9).

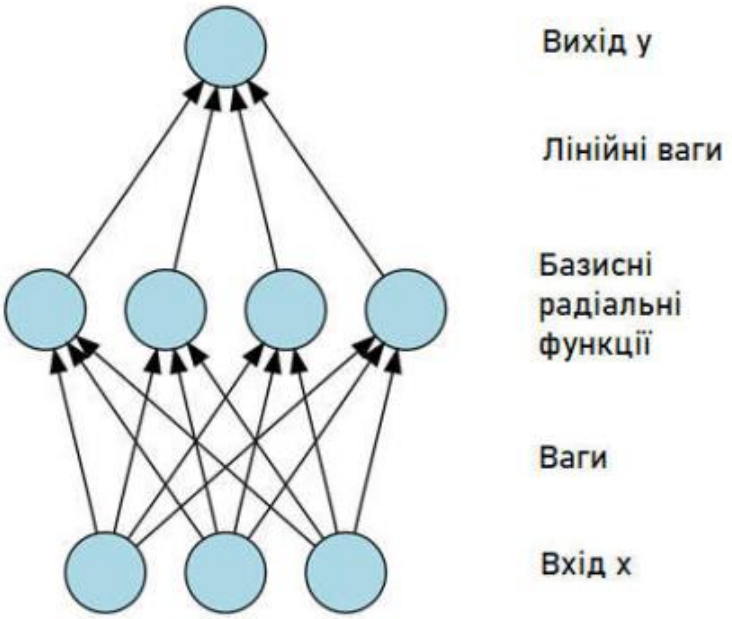

Рисунок 1.9 – Архітектура мережі RBF

RBF (рисунок 1.9) зазвичай складаються з трьох шарів: вхідного шару, прихованого шару і лінійного вихідного шару. Для представлення вхідних даних може використовуватися вектор дійсних значень $x \in R^n$. Отже, вихід мережі є скалярною функцією вхідного вектора, φ: $R^n \rightarrow R$ у вигляді [16]:

$$\varphi\,(x) = \sum_{i=1}^{N} a_i\, p\,(||x - c_i||)$$

де N – кількість нейронів прихованого шару, c_i – центральний вектор нейрона i та a_i – вага нейрона i в лінійному виході нейронів.

Радіальні базисні функції — це функції, які визначаються виключно від відстані від центру вектора і мають радіальну симетрию відносно цього вектора. У їхній найпростішій формі кожен нейрон прихованого шару пов’язаний з усіма вхідними сигналами.

Евклідова відстань часто використовується як норма (хоча відстань Махаланобіса краще підходить в більшості випадків), а гауссовий розподіл зазвичай використовується як радіальна базисна функція [16]:

$$p\left(\left||x - c_i|\right|\right) = \exp\left[-\beta||x - c_i||^2\right]$$

У тому сенсі, що зміна параметрів одного нейрона має лише незначний вплив на вхідні значення, віддалені від центру нейрона, гауссові базисні функції є близькими до центрального вектора [16].

$$\lim_{||x||\to\infty} p\left(\left||x - c_i|\right|\right) = 0$$

1.4.1 Рекурентні нейронні мережі

Рекурентні нейронні мережі (RNN) представляють собою тип штучних нейронних мереж, які мають циклічну структуру, де кожен вузол пов'язаний з іншими. Ці мережі ідеально підходять для завдань, пов'язаних з обробкою послідовностей, таких як розпізнавання голосу. Основна особливість RNN полягає в тому, що вони мають внутрішню пам'ять, яка дозволяє їм працювати з послідовностями вхідних даних.

У зв'язку з циклічною структурою, RNN можуть працювати зі вхідними послідовностями різної довжини, що робить їх ефективними для завдань, де важко визначити сталий розмір вхідних даних. Також важливо відзначити, що рекурентні зв'язки дозволяють мережі враховувати контекст та взаємодіяти зі змінами во часі.

Багатошаровий персептрон (MLP) із зворотніми зв'язками може бути розглянутий як один тип RNN. Такі мережі мають додаткові циклі, що дозволяє їм взаємодіяти та обробляти інформацію в часі. Основна відмінність у тому, що MLP може мати більше оперативної пам'яті завдяки нелінійним функціям активації та більш складній структурі. На рисунку 1.11 зображена архітектура рекурентних мереж RNN.

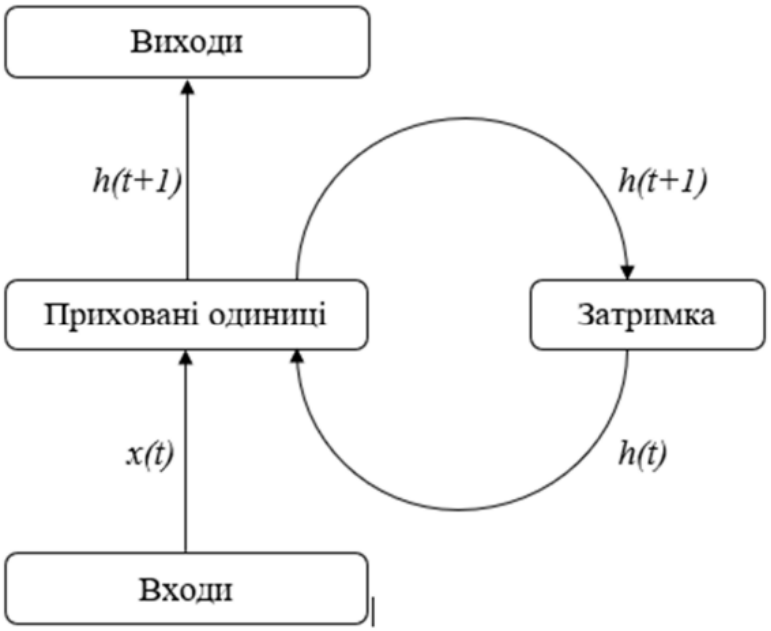

Рисунок 1.10 – Архітектура RNN [18]

Присутні вхідні та вихідні шари, а також прихований шар з повноцінними повторюваними зв’язками. Навіть базові проекти можуть отримати користь від навчання RNN.

На рисунку 1.11 показана найпростіша версія повністю рекурентної нейронної мережі, а саме MLP з попереднім набором прихованих одиничних активацій $h(t)$, які повертаються в мережу разом із входом $h(t+1)$. Активації повинні оновлюватися на кожному кроці часу, щоб час вибірки t.

Для штучних систем шкала часу може відповідати роботі фактичних нейронів, і можна використовувати будь-яку кількість кроку часу, достатню для цієї роботи. Щоб відкласти активацію до наступного кроку часу, потрібно буде ввести одиницю затримки.

РОЗДІЛ 2
РЕЗУЛЬТАТИ ПРОЕКТУВАННЯ (МОДЕЛЮВАННЯ) ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ РОЗПІЗНАВАННЯ ГОЛОСОВИХ ДАНИХ

Після проведення характеристики об’єкту дослідження та встановлення завдання програмного застосунку, доцільно визначити спосіб виконання поставлених завдань. Зрозуміло, що існує декілька методів досягнення поставленої мети, але потрібно вибрати той, який буде найкраще підходити для вирішення задачі. Власне, для цього і потрібний системний аналіз.

Системний аналіз — це науковий метод пізнання, що являє собою послідовність дій з установлення структурних зв'язків між змінними або елементами системи, яка досліджується. Він застосовується для визначення проблем і цілей системи, що досліджується. Також допомагає визначити альтернативні шляхи подолання проблем чи досягнення поставлених цілей. Тому для вивчення проблем та цілей програмного застосунку, будемо використовуватимемо такі методи системного аналізу як дерево проблем та дерево цілей.

2.1. Побудова дерева проблеми

Основний методом аналізу проблем у системному аналізі - дерево проблем. Це графічно зображена ієрархічна структура проблематики системи, яка отримана шляхом поділу загальної проблематики на основний тип проблеми (стовбур), інші присутні типи (гілки), підтипи (відгалуження) і власне проблеми (листи).

Цей метод застосовується для отримання стійкої структури проблематики. Дерево проблеми системи, що досліджується у даній роботі, зображено на Рисунку 3.1. Воно допоможе краще зрозуміти перешкоди, які нас спіткають на шляху до реалізації нашого програмного застосунку.

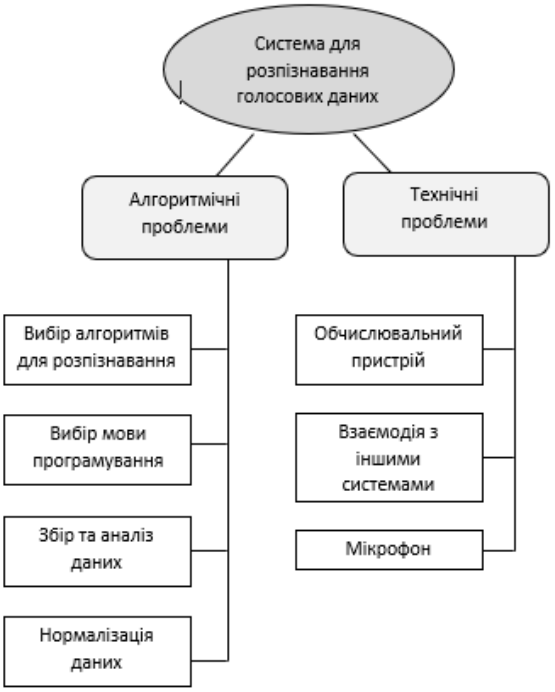

Рисунок 2.1. - Дерево цілей

Проаналізувавши проблему, бачимо, що в корені знаходиться потреба у створенні Системи для розпізнавання голосових даних.

Основними 2 головні проблемами є:

- Алгоритмічні проблеми
- Технічні проблеми

В свою чергу, алгоритмічні проблеми поділено на 4 задачі:

- Вибір мови програмування

В залежності від вибору мови програмування та апаратної платформи залежить реалізація нашої системи. Тому нам потрібно обрати мову, яка має велику кількість необхідних бібліотек. Також варто враховувати що розроблювана система повинна відкриватись на всіх популярних операційних системах.

- Збір та аналіз даних

Оскільки дане система не є новою на ринку, а лише покращення вже створених систем. То необхідно проаналізувати вже готові моделі, які ми можемо використати у наші системі.

- Нормалізація даних

Щоб найкраще розпізнати дані, вони повинні бути правильно підготовлені. Система повинна розпізнавати не тільки дані які записані в спеціальних приміщеннях на потужну техніку. Тому для цього ми повинні проводити обробку даних: визначення шумів та їх видалення, а також регулювання гучності.

- Вибір алгоритмів для розпізнавання

Оскільки розпізнавання посмішки це завдання, яке потребує навчання системи, нам потрібно знайти ефективний та швидкий спосіб вивести системи на найвищий рівень продуктивності у співвідношенні до витрат часу на це навчання. Саме тому ефективний алгоритм навчання системи може значно знизити час розробки системи і підвищити ефективність її роботи.

Технічні проблеми поділені на 3 задачі:

- Обчислювальний пристрій

В залежності від вибору обчислювального пристрою залежить те, як буде працювати наша система. Все залежить від того, з якими об’ємами даних ми будемо працювати. Вирішення даної проблеми дозволить нам зробити компактну систему з максимальною швидкодією.

- Мікрофон

Хоч ми і будемо проводити нормалізацію даних, все ж від вибору мікрофона залежить якість голосових даних. Чим краща кінцева якість даних, тим краще система зможе їх розпізнати. Тому вибір якісного мікрофону є важливим аспектом для успіху даного проекту.

- Взаємодія з іншими системами

Як говорилось раніше, у розроблюваній системі ми будемо використовувати уже готові модулі. Тому необхідно організувати їх взаємодію. Для цього потрібно буде використовувати API цих самих систем.

2.2. Побудова дерева цілей

Після побудови дерева проблем, потрібно сформувати дерево цілей, основною метою якого є графічне представлення взаємо-зв’язків між цілями.

На вершині дерева цілей зображується головна мета, яка є орієнтиром. Нижче головної мети розміщуються цілі першого рівня, які є способами досягнення головної мети.

Потрібно визначити всі цілі, які ми плануємо реалізувати в майбутній системі. В сукупності всі цілі повинні зумовлювати досягнення початкової цілі. Відповідно до вимог до системи, дерево цілей можна зобразити наступним чином:

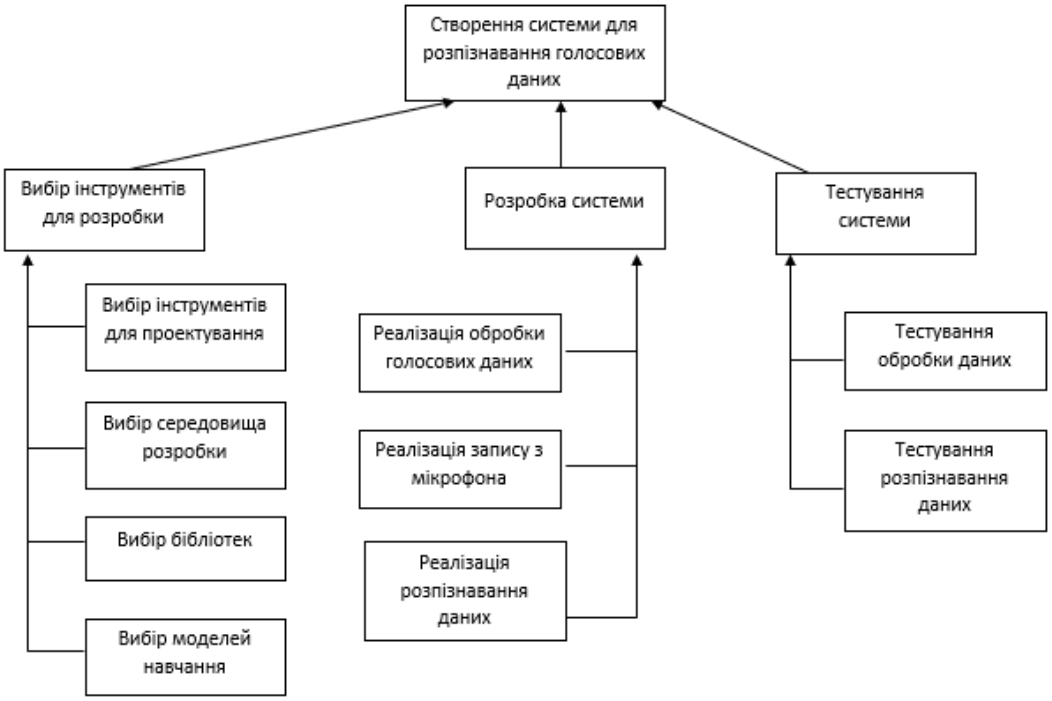

Рисунок 2.2. - Дерево цілей

Основною метою є створення системи для розпізнавання голосових даних, яка даватиме змогу користувачеві обрати файл з голосовими даними, обробити ці дані, розпізнати їх та записати у файл.

Для досягнення головної мети потрібно виконати три цілі.

Перша ціль – це вибір інструментів для розробки. Її можна розділити на такі заходи:

- вибір інструментів для проектування системи;
- вибір середовища розробки;
- вибір бібліотек.
- Вибір моделей навчання

Друга ціль – це розробка системи. Цю ціль можна розбити на наступні менші етапи:

- реалізація обробки голосових даних;
- реалізація запису з мікрофона.
- реалізація розпізнавання даних

Останньою ціллю є тестування системи. Це відіграє важливу роль, тому що потрібно впевнитися, що всі функції працюють коректно та відповідають встановленим вимогам. Можна розділити на такі складові:

- тестування обробки даних;
- тестування розпізнавання даних.

2.3. Аналіз і вибір методів, алгоритмів та засобів розв'язання задачі

2.3.1 Мова програмування Python

Для побудови взаємодії між розділами системи та їх об'єктами, такими як програмне забезпечення, хмарні нейронні мережі та апаратне забезпечення, яке використовує нейронні мережі для виявлення та ідентифікації різних шумних інструкцій, була використана мова сценаріїв Python. Оскільки база даних бібліотек і мікросервісів Python велика і містить необхідні для читання з мікрофона та різноманітні звукові карти, файли чи інші пристрої голосового введення для продовження роботи з ними, а також велику кількість бібліотек і підтримуваних API, у на додаток до попередньо вибраного, для подальшого покращення вибраного методу розпізнавання мови, такого як бібліотеки PyAudio, Google Speech API, апаратне забезпечення для розробки та mo.

Це ідеальний суперник для системних вимог командної обробки, оскільки завдяки своїй потужності та низькому бар'єру входу на Python, а також його інструменти та методології.

Запит користувача API розпізнавання мовлення Апаратне програмне забезпечення Демонстраційне програмне забезпечення Програмне забезпечення для майбутнього розвитку запропонованого підходу Python — це мова програмування, яка дозволяє писати сценарії. Завдяки своїй адаптивності він підходить для широкого спектру робіт практично на всіх платформах, від

серверної ОС і Android до, але не обмежуючись ними, систем керування мобільними пристроями.

Він використовується для створення веб-додатків, веб-сайтів, плагінів та інших веб-продуктів, а також додатків для персональних комп'ютерів, смартфонів та інших мобільних пристроїв і систем керування, розробки ігор, програмування чат-ботів, і на даний момент є одним з найбільш потужні мови машинного навчання, великих даних і науки про дані.

Python є інтерпретованою мовою програмування, що означає, що сценарій, згенерований посібником, буде текстовим файлом з іншим розширенням, зокрема «.py», до того, як він буде ініціалізований та запущений інтерпретатором. Його можна використовувати для створення програмного забезпечення практично на будь-якому пристрої чи платформі, і це логічна мова, яку легко зрозуміти завдяки своїй лаконічністю та розумній архітектурі.

Розробка програмного забезпечення Python у багато разів швидша, ніж C, Java або інші мови через відсутність шаблонного коду.

2.3.2 PyCharm IDE

PyCharm — це повнофункціональна, спеціалізована та адаптована IDE для програмування на Python, якщо не найкраща. Він пропонує кілька варіантів для економії часу, допомагаючи виконувати звичайні справи. PyCharm — це найпопулярніша і часто використовувана IDE для розробки додатків і програмування на Python. PyCharm, на відміну від VisualStudio, є багатоплатформною IDE, створеною JetBrains, яка зосереджена на Python. У таких компаніях, як Twitter, Facebook, Amazon і Pinterest, PyCharm є стандартною IDE для програмування на Python.

Для розробки та створення додатків PyCharm IDE (рисунок 2.3) включає редактор і компілятор. Він поєднує в собі різноманітні елементи, які полегшують розробку програмного забезпечення.

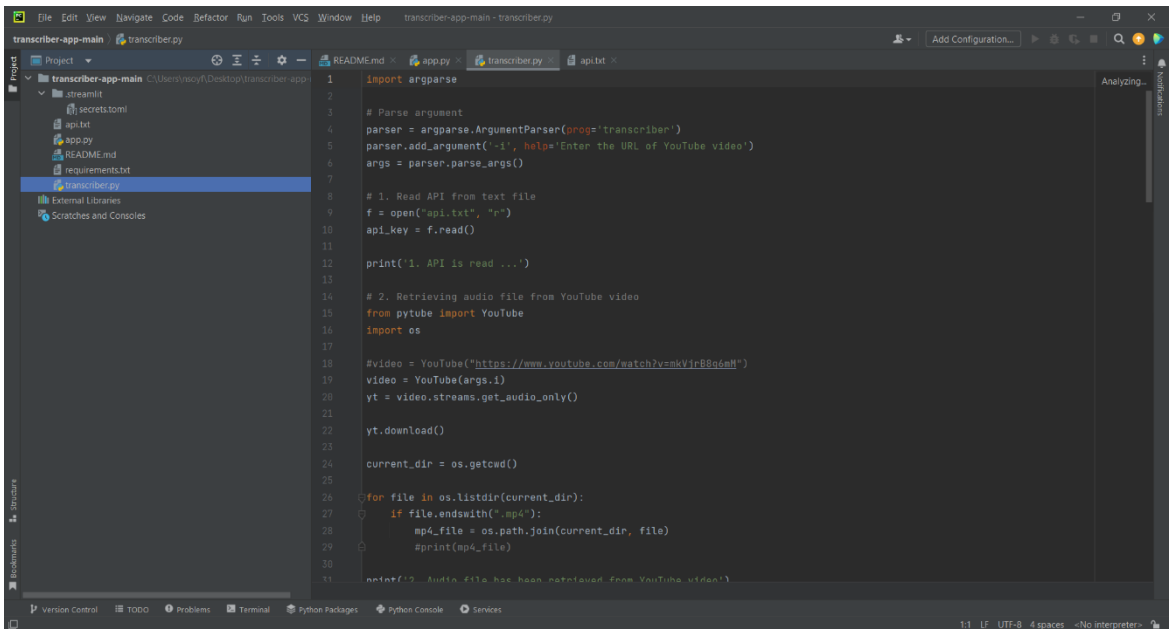

Рисунок 2.3 - PyCharm IDE

Використання IDE значно полегшує процес розробки та програмування. Він розшифровує написаний код і пропонує ключові слова для включення. Оскільки PyCharm IDE показує класи та методи різними кольорами, їх легко відрізнити.

Нижче наведено основні причини розробки за допомогою IDE PyCharm:

- IDE складається з вікна текстового редактора, у якому ви можете вводити комп'ютерний код із підтримкою автозаповнення, завершення та редагування коду, зазначеного раніше, що допомагає стиснути код, покращити читабельність та легше виявляти проблеми.

- Навігація по коду: PyCharm надає певні комбінації клавіш для доступу до файлу, класу або функції, що значно спрощує зміну коду в різних місцях у списку програм, особливо дещо «віддалених» фрагментів програмного коду та знаходження їх наявності.

- Рефакторинг: вищезгадані функції редактора значно полегшують внесення змін до проекту, змінення коду для нових потреб або роботу над застарілими проектами.

- Багато мов забезпечують підтримку синтаксису. Незважаючи на те, що PyCharm був розроблений спеціально для Python, він підтримує синтаксис широкого спектру мов програмування, включаючи JavaScript, CoffeeScript,

TypeScript, Cython, SQL, HTML/CSS, мови шаблонів, AngularJS, Node.js та інші, і допоможе вам встановити плагін, якщо ви не підтримуєте потрібну мову за замовчуванням.

- IDE також містить вікно редактора проекту, де ви можете переглянути загальну структуру проекту, а також усі пов'язані файли з створеного програмного проекту.

- Є можливість використовувати вбудований компілятор для надання різних вхідних даних та оцінки продуктивності та ефективності створеного коду безпосередньо в IDE, дивлячись на вихідні дані у вікні виводу компілятора.

- У вікні виведення IDE відображає попередження та ідеї щодо видалення чи виправлення помилок чи невідповідного форматування коду.

2.4. Блок-схеми алгоритмів

На основі проведених досліджень, зваживши переваги та недоліки було обрано алгоритмічний підхід. Його перевагами є простота, зрозумілість, а головне –легкість навчання.

Опис алгоритму: На початку виконання програми користувач має змогу обрати спосіб розпізнавання даних: дані зчитуються з наперед підготованого *.wav файлу, або з мікрофона. Далі перевіряється наявність сигналу: у випадку з файлом, чи розпізнало фрейми; а у випадку з мікрофоном, чи є підключено мікрофон. У випадку помилки виводиться відповідне повідомлення і прохання повторити спробу. Якщо сигнал розпізнало, переходимо до наступного пункту: Обробка звуку: потрібно провести нормалізацію, перевірити на наявність шумів і очистити від них. Запит до GSR API за допомогою якого буде проводитись ASR розпізнавання тексту. Після виконання розпізнавання, текст виводиться на екран і записується у файл.

Завдяки виконаним дослідженням, було побудовано алгоритми функціонування програми, які повинні істотно допомогти при розробці програмного рішення розглянутої проблеми.

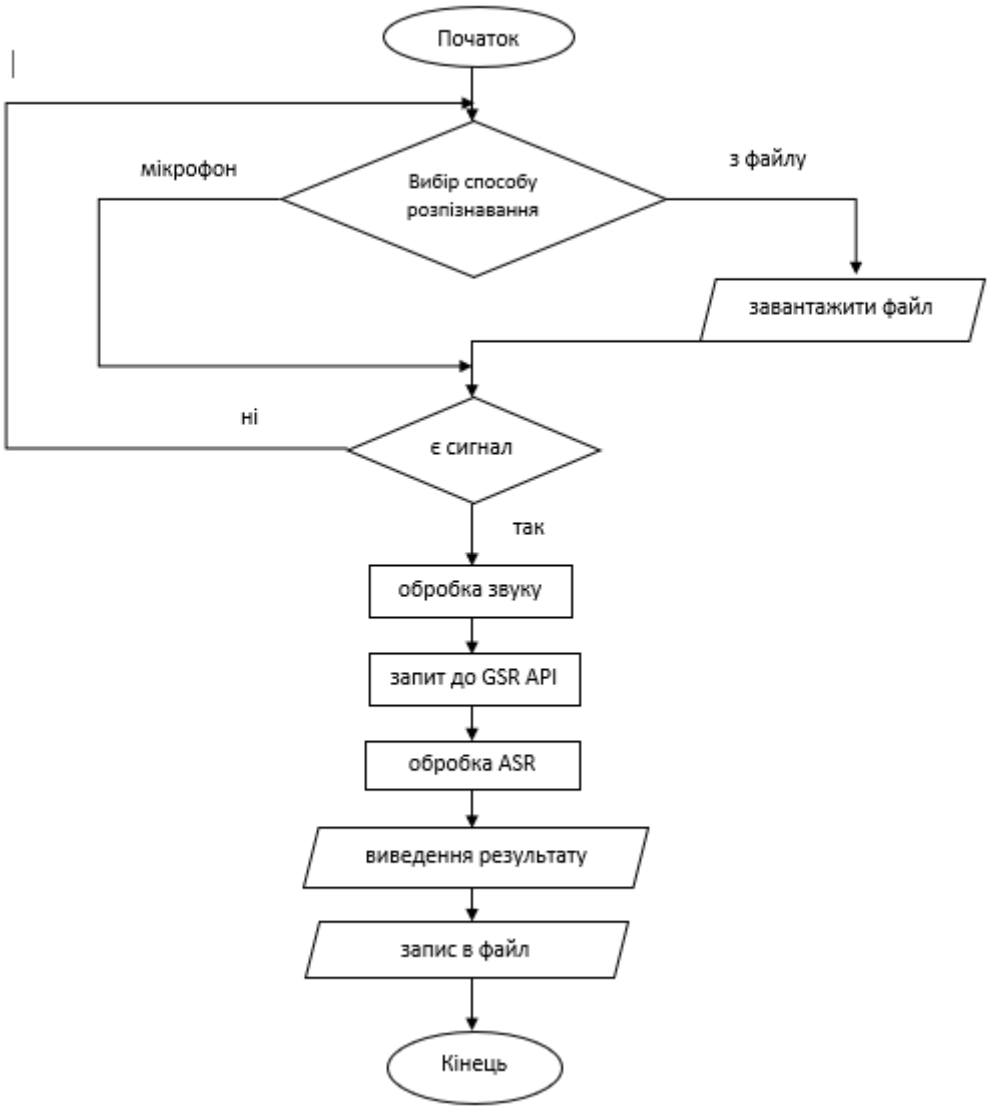

Рисунок 2.4. - Блок-схема алгоритму виконання програми

2.5. Побудова моделей бізнес-процесів за допомогою діаграм IDEF0

IDEF0 – стандарт функціонального моделювання, який призначений для формалізації і опису бізнес процесів. Особливістю цієї методології є наголос на ієрархічному представленні об’єктів, що дозволяє полегшити розуміння предметної області та проектованої системи.

Опис системи в даному стандарті проводиться за допомогою програми BpWin. Контекстна діаграма є діаграмою найвищого рівня, що описує загальний процес системи, тобто її основну функцію та взаємодію з зовнішнім середовищем. Контекстна діаграми для проектованої системи зображена на Рисунку 2.5. Основною функцією даної інформаційної системи є розпізнавання голосових даних.

Рисунок 2.5. - Контекстна діаграма

Проведемо декомпозицію 1-го рівня даної діаграми (Рисунок 2.6.). На даному рівні головна функція описується виконанням певних під функцій, розкриваючи суть основного процесу системи. Це допоможе зрозуміти що відбувається всередині системи.

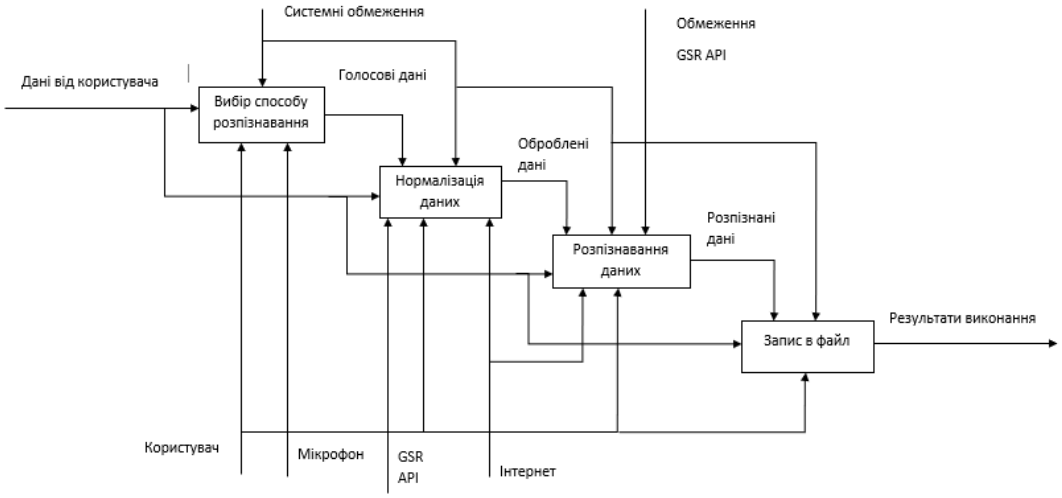

Рисунок 2.6. - Діаграма декомпозиції першого рівня

Отже, розпізнавання голосових даних програмним застосунком відбувається послідовним виконанням таких функцій: вибір способу розпізнавання, нормалізація даних, розпізнавання даних, запис у файл. Кожна з функцій отримує певний вхід і в результаті виконання видає вихід. Наприклад, для нормалізації даних вхідними даними є попередньо обрані голосові дані, і в результаті виконання даної роботи отримуємо оброблені дані.

Для кращого опису системи декомпозиції 1-го рівня недостатньо. Бажано проводити декомпозицію для кожної з майбутніх робіт, тільки таким шляхом можна досягнути максимальної деталізації системи. Тож проведемо декомпозицію другого рівня для завдання Вибір способу розпізнавання (Рис. 2.7) та Запис у файл (Рис. 2.8).

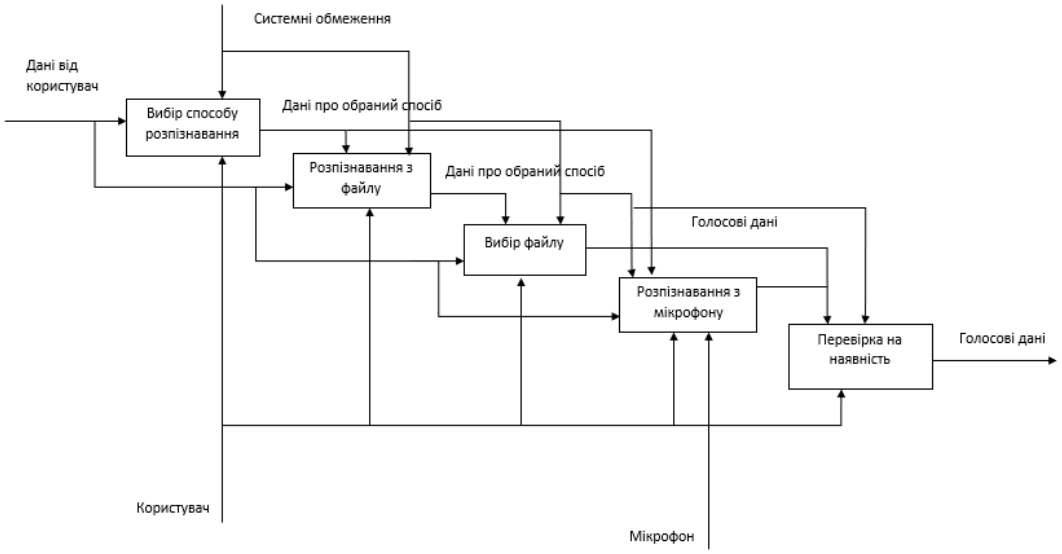

Рисунок 2.7. - Діаграма декомпозиції другого рівня для функції «Вибір способу розпізнавання»

Як бачимо на діаграмі, розпізнавання даних ми можемо проводити двома способами: розпізнавати з наперед підготованого файлу, або ж продиктувати в мікрофон. У першому випадку нам необхідно вказати файл, з якого необхідно зчитати дані. Тоді зробити перевірку: якщо вибір з файлу, то чи є там наявність відповідних даних. У випадку з мікрофоном, чи є підключений хоча б один мікрофон.

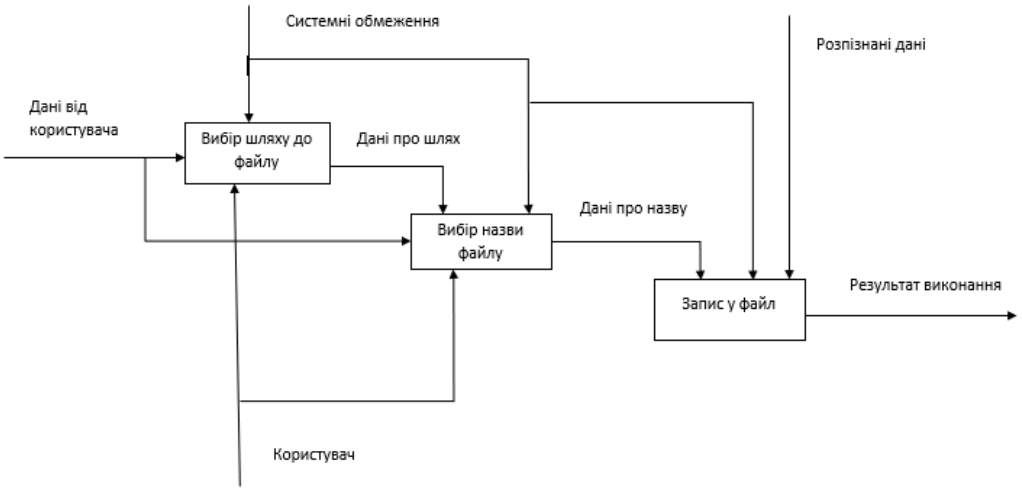

Рисунок 2.8. - Діаграма декомпозиції другого рівня для функції «Запис у файл»

Останнім етапом роботи ІС є запис у файл. Як видно з діаграми, щоб виконати запис даних у файл, спочатку потрібно обрати шлях до файлу, та вказати назву даного файлу. Тоді програма після розпізнавання, запише дані які були розпізнані у цей файл.

2.6. Відображення потоків даних за допомогою DFD діаграми

Після побудови контекстної діаграми та діаграми декомпозицій стало зрозуміліше, які процеси відбуваються в системі. Однак слід описати процеси обробки інформації та документообігу в системі у вигляді діаграми потоків даних для кращого розуміння того, які потоки використовуються системою.

Основним методом моделювання функціональних вимог проектованої системи є діаграма потоків даних, також відома як діаграма потоків даних, або DFD. Вимоги представлені ієрархією процесів, пов’язаних із потоками даних. Діаграми потоків даних показують, як кожен процес перетворює свої вхідні дані у вихідні, і показують, як ці процеси пов’язані один з одним. DFD -діаграми є корисним доповненням до моделі IDEF0 для опису обробки даних і документообігу.

Будуємо DFD-діаграму для покращення відображення передачі даних у майбутньому сервісі.

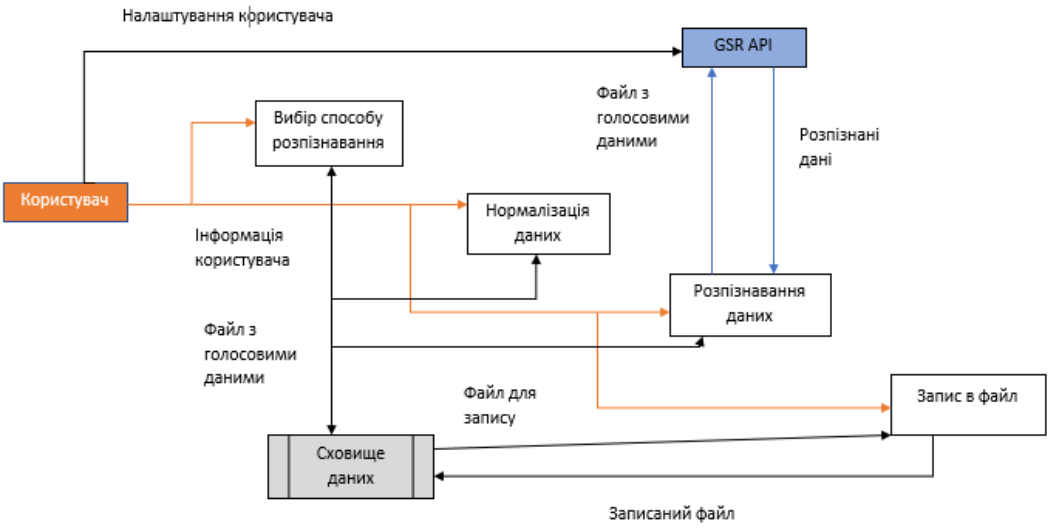

Рисунок 2.9. - Діаграма потоків даних

Отже, наша система містить дві зовнішні сутності. Зовнішньою сутністю зображаю входи та виходи системи. Зовнішня сутність «Користувач» з усіма процесами системи, надаючи необхідні дані та налаштування. У цьому випадку сховище вхідних даних – це виділена фізична пам'ять пристрою, де зберігаються усі потрібні вхідні дані у вигляді змінних та файлів. При розпізнаванні даних ми використовуємо сутність «GSR API», в яку передається *.wav файл з голосовими даними, а отримуємо розпізнані дані. Ці дані записуємо в файл і передаємо у сховище даних.

2.7. Побудова UML діаграм

Описуючи даний програмний застосунок у вигляді вище розгорнутих діаграм, ми можемо краще зрозуміти всі процеси, що відбуваються в системі. Але ще одним важливим аспектом опису ІС є опис системи з боку користувача, тобто опис випадків використання системи, функціонал, що буде надаватися користувачу з боку системи, та опис окремих функціональних блоків системи, таких як класи, модулі чи підсистеми. Саме для цього використовують UML.

У парадигмі об’єктно-орієнтованого програмування використовується мова моделювання UML. Вона є важливою складовою стандартного процесу розробки програмного забезпечення. UML-модель — це абстрактна система-модель, створена за допомогою відкритого стандарту, відомого як UML. Була розроблена для документування, проектування, візуалізації та визначення програмних систем. У UML є 14 видів діаграм, але ми використаємо лише два

для опису нашої системи: діаграму прецедентів або випадків використання, діаграму послідовності.

2.7.1. Діаграма прецедентів

В UML (use-case diagrams) діаграма прецедентів показує зв’язки між прецедентами та акторами системи.

Діаграми прецедентів, також відомі як діаграми варіантів використання, описують функції, які будуть доступні користувачам системи, яка проектується. Створюються за допомогою взаємодії між акторами та прецедентами. Набір усіх прецедентів діаграми фактично визначає функціональні вимоги, за допомогою яких може бути розроблено технічне завдання.

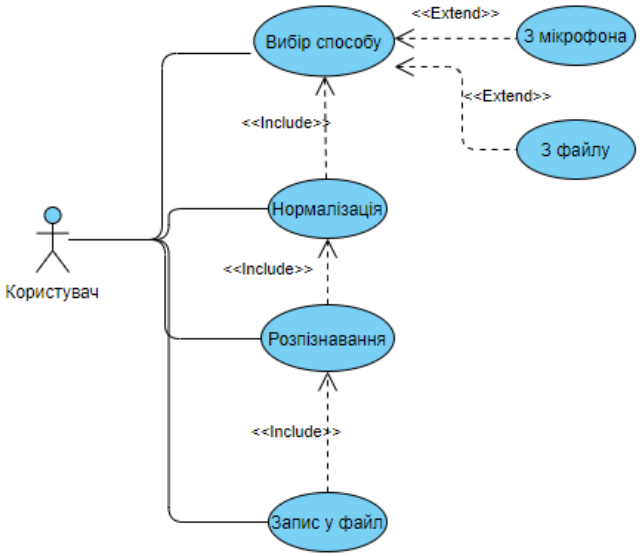

Рисунок 2.10. - Діаграма прецедентів

Як бачимо, на даній діаграмі вказаний увесь необхідний функціонал системи з боку його використання користувачем у тій послідовності, в якій це можна здійснити. Користувачу необхідно обрати один з запропонованих способів, тоді ці дані обробляються, розпізнаються і записуються у файл.

2.7.2. Діаграма послідовностей

Діаграма послідовності (sequencediagram) - діаграма, на якій показані взаємодії об'єктів, впорядковані за часом їх прояву. За допомогою діаграми послідовності можна представити взаємодію елементів моделі як своєрідний часовий графік "життя" всієї сукупності об'єктів, пов'язаних між собою для

реалізації варіанту використання програмної системи, досягнення бізнес-мети або виконання якої-небудь задачі

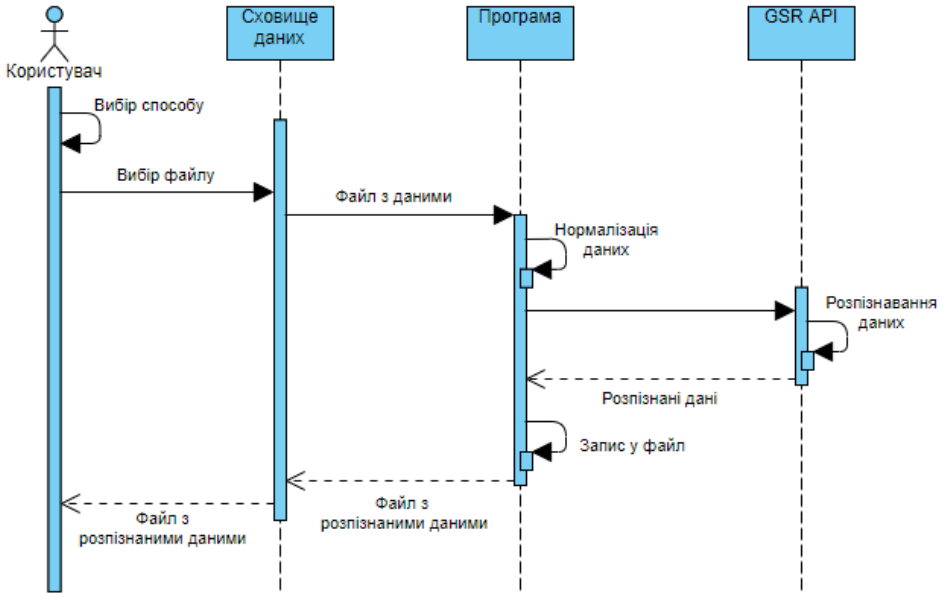

Рисунок 2.11. - Діаграма послідовностей

На даній діаграмі у вигляді вертикальних лінійно показано різні процеси або об'єкти, що існують водночас. Надіслані повідомлення зображуються у вигляді горизонтальних ліній, в порядку відправлення, а відповіді на повідомлення – у вигляді пунктирних ліній.

В даному випадку всі об’єкти існують від початку роботи системи. Користувач обирає спосіб розпізнавання та файл з даними у сховищі даних. Ці дані у програмі обробляються і передаються на розпізнавання у GSR API. Розпізнані дані повертаються в програму та записуються у файл. Записаний файл передається користувачеві.

2.8. Загальна структура програмного проекту

Основна мета методу — зменшити кількість помилок ідентифікації в шумному середовищі, а також розробити загалом зручний інтерфейс для демонстрації або подальшого використання (у формі скрипту Python).

Створений скрипт повинен:

- Підвищити точність розпізнавання ASR;

- Переконатись, що інструкції виконуються правильно.

Початкові дані для проведеннядослідження:

- Google ASR - Методи та функції розпізнавання мовлення ANN

- GSR API - методи та функції, які "з'єднують" ASR на сервері зі скриптом Python, надаючи йому доступ до конкретних можливостей Google ASR.

- Python Script - функції, класи та методи, які передають команди за допомогою тестування ASR та забезпечують їх виконання в середовищі ОС за допомогою GSR API.

- Librosa Audio - Рішення на основі ANN для видалення шуму з введення голосової послідовності.

- Операційна система Windows - середовище для обробки отриманих інструкцій.

Зокрема, користувач передає голосовий запит від системи введення ОС до сценарію (з якого сторонні шуми були усунені), який потім асоціюється з GSR API, який запитує обробку отриманого голосового повідомлення в ASR, розміщеному на серверах Google. Дані обробляються і повиртаються назад в скрипт. Де вони записуються в файл з операційної системи.

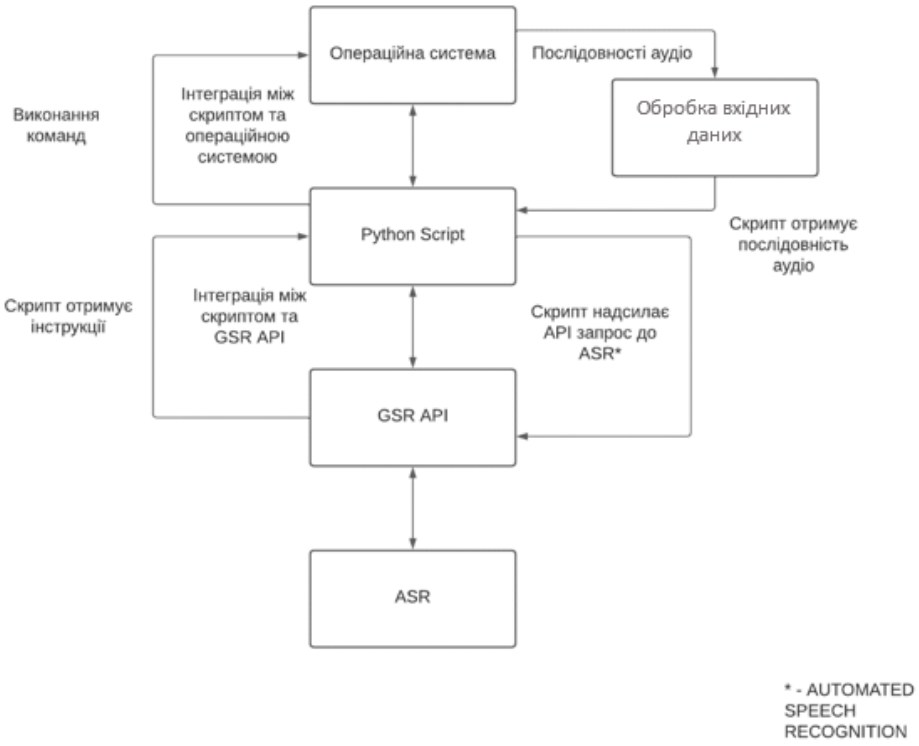

Рисунок 2.12 - Загальна архітектура системи

2.9. Опис використаних сторонніх бібліотек та модулів

2.9.1 Google Speech API

GSR API – це технологія, яка широко використовується в різних дисциплінах і мовах. Це компанія, що належить Google. Це дозволяє використовувати голосовий пошук на основі теорії розпізнавання мовлення. Мобільні телефони та ноутбуки з підтримкою мовлення включають цю технологію. Google почав інтегрувати голосові технології в свою пошукову систему влітку 2011 року. (Google Search).

Лише Google Chrome підтримує цю технологію на ПК. Смартфони під управлінням Android та iOS також сумісні.

Голосовий пошук Google спочатку дозволяв лише короткі запити до 35-40 слів. Щоб надіслати запит, мікрофон потрібно ввімкнути та вимкнути. Ця можливість все ще доступна в рядку пошуку Google; просто торкніться мікрофона, щоб почати. Однак голосовий пошук Google став мовним введенням, коли Chrome представив можливість постійного виявлення мовлення в лютому 2013 року. Доступ до API був доступний з травня 2014 року. Багато відомих служб підтримують голосовий пошук Google, зокрема Google, YouTube, Yahoo, DuckDuckGo, Бінг, Вольфрам | Альфа, Вікіпедія та інші. Ви можете додати власні пошукові системи, якщо хочете. Для сайтів, які використовують форми пошуку HTML5, є навіть додаток, який додає кнопку для надсилання голосової інформації. Робота з додатками вимагає використання мікрофона. Щоб скористатися голосовим пошуком Google, надішліть запит POST на адресу, яка містить аудіодані у форматі.flac або.spx. Тоді будь-яка програма повинна мати можливість ідентифікувати файли WAVE.

GSR API порівнянний з Nuance Dragon Mobile SDK з точки зору функціональності, однак немає обмежень щодо кількості запитів, які можна робити щодня. Щоб виявити ключове слово «Окей, Google», інженери Google використовують глибоку нейронну мережу.

Синхронне розпізнавання (REST і gRPC) доставляє аудіодані до GSR API, який розпізнає їх, а потім дає результати, коли все аудіо буде оброблено. Запити

на синхронізоване розпізнавання обмежені 1 хвилиною або менше аудіоданих. Асинхронне розпізнавання (REST і gRPC) доставляє аудіодані до GSR API, ініціюючи тривалу процедуру. Ви можете регулярно опитувати результати розпізнавання, використовуючи цю дію. Ви можете виявити аудіодані будь-якої тривалості до 480 хвилин за допомогою асинхронних запитів. Розпізнавання в режимі реального часу (тільки gRPC) розпізнає аудіодані з двонаправленого потоку gRPC в режимі реального часу. Потокові запити призначені для програм, які потребують розпізнавання в режимі реального часу, наприклад, для запису живого звуку з мікрофона. Під час запису аудіо розпізнавання потоку дає проміжні результати, що дає змогу показати результат, наприклад, поки користувач все ще говорить. GSR API працює з широким діапазоном мов і діалектів.

Необхідно вказати відповідну ідентифікацію в полі коду мови запиту, щоб вибрати мову аудіо (та національний або регіональний діалект). Google Speech API дійсно корисний. Завдяки величезній обчислювальної потужності це дуже легко та швидко, а перевага полягає в тому, що немає обмежень на кількість запитів щодня.

GSR API має ряд переваг перед іншими ASR, включаючи чудову точність розпізнавання (через великі бібліотеки баз даних і потужність комп’ютера) і швидкість розпізнавання мовлення.

2.9.2 Librosa Audio

Librosa була створена з наміром обробляти короткі аудіо фрагменти, як правило, тривалістю не більше кількох хвилин. Хоча це справедливо для більшості популярної музики (обсяг диктував мету), це погана відповідь для багатьох інших типів аудіоданих, особливо тих, що зустрічаються в біоакустиці та екологічній акустиці. За таких обставин аудіосигнали зазвичай витримують багато годин, якщо не днів чи тижнів. Тут виникає питання про те, чи можна обробляти тривалі аудіофайли за допомогою librosa.

Перш ніж зануритися в особливості обробки великих файлів, важливо спочатку зрозуміти модель даних librosa загалом. Замість того, щоб створити

більш організовану модель об’єктів, розробники librosa від самого початку покладалися лише на типи даних numpy.

Як наслідок, переміщення даних з librosa до інших програм на Python є легким. Наявність власного об’єктного інтерфейсу для аудіо та функцій, навіть якщо це спрощує певні міркування щодо дизайну, було б перешкодою. Проблема з величезними аудіофайлами пов’язана з обмеженнями numpy, такими як numpy.ndarray: і об’єктні, і процедурні методи мають переваги та недоліки. Процедурна техніка має ряд переваг, але один недолік, успадкований від numpy, полягає в тому, що всі вхідні дані повинні бути створені до створення melspec. Нарешті, тип numpy.ndarray має таке обмеження, яке створюється контейнером для безперервних розділів пам’яті фіксованої довжини з послідовними основними типами даних (наприклад, int або float). Це підходить для коротких записів, які є найбільш поширеними в librosa: записи часто зберігаються в пам’яті та відомі наперед.

Однак ndarray не є життєздатним типом контейнера, коли аудіо для аналізу триває годину або довше або транслюється з пристрою введення. Розробники знали про це, але вирішили оптимізувати роботу для регулярного використання, а не боротися з наслідками. Цю проблему можна було б вирішити різними способами, якби Librosa отримав об’єктивний інтерфейс.

Наприклад, буфер Audio.get (час = ДЕЯКЕ ЧИСЛО, тривалість = ДЕЯКЕ ЧИСЛО) абстрагує логіку індексації часу таким чином, що дані завантажуються лише тоді, коли це необхідно.

Необхідно використовувти внутрішній стан об’єкта, щоб зберегти буфер у пам’яті, не завантажуючи повний запис пам’яті, і запропонуйте інтерфейс для пошуку певного місця в часі в сигналі.

Різні бібліотеки реалізують такі рішення і роблять це успішно, але вони ускладнюють інтерфейс і можуть обмежувати взаємодію [20].

Librosa підтримує такі функції:

— Декодування Вітербі: тимчасово згладжує передбачення кадрів

— Попередні налаштування: дозволяє налаштувати параметри librosa за замовчуванням, такі як частота дискретизації, кількість вибірок між кадрами та кількість вибірок на кадр у STFT-подібних аналізах, за допомогою попередньо встановленого пакета. Вони корисні, коли вам потрібно змінити ці змінні відповідно до програми, але робити це кожен раз, коли ви використовуєте функцію, громіздко.

Попередні налаштування дозволяють легко змінювати все відразу, включаючи зміну параметрів за замовчуванням і охоплення модуля та всіх викликів функцій.

- Синхронізація музики з використанням динамічного спотворення часу (DTW): ця функція використовує DTW для синхронізації музики з librosa.

Наприклад, два записи «Sir Duke» Стіві Уандера, які виконують початкові такти відомої духової партії.

Перший запис триває приблизно 7 секунд, а другий запис триває приблизно 5 секунд через різницю в темпі. DTW можна використовувати для виявлення збігу між двома записами (рис. 2.13).

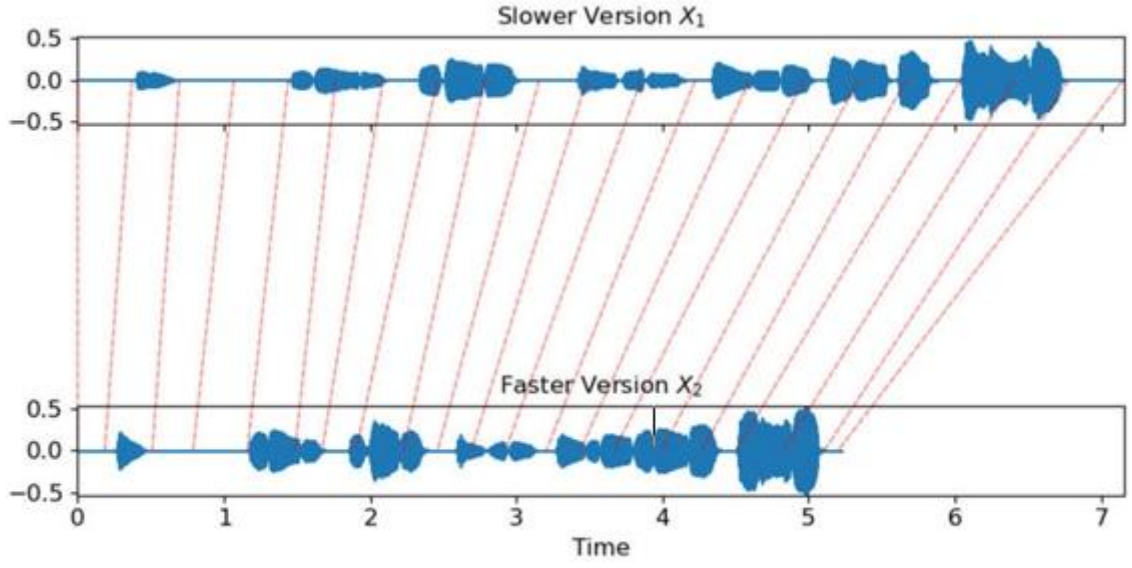

Рисунок 2.13 - Візуалізація використання алгоритмів попередніх налаштувань

- Vocal separation: ця функція відповідає за відокремлення вокалу від допоміжних інструментів за допомогою простого підходу.

РОЗДІЛ 3

РЕЗУЛЬТАТИ ВИРІШЕННЯ ЗАДАЧІ

3.1. Розробка та опис програмних модулів

Код програми було винесено в декілька модулів, адже в такому випадку можна легко змінювати та читати код. Коротко опишемо основні модулі:

- модуль для розпізнавання з файлу;
- модуль для розпізнавання з мікрофону;
- модуль, який відповідає за обробку звуку;
- модуль для запису в файл.

3.2. Розробка та опис інтерфейсу користувача

На рисунку 3.1 зображено розроблена секція меню веб-додатку для взаємодії з можливостями розпізнавання:

1. Користувач може обрати варіант розпізнавання даних з файлу (формат .wav або .mp3), завантажити наявний на комп’ютері файл та розпізнати голосові дані.

2. Користувач може обрати варіант розпізнавання даних з мікрофону (необхідно промовити текст, програма виконує розпізнавання та зберігає отриманий результат до текстового файлу).

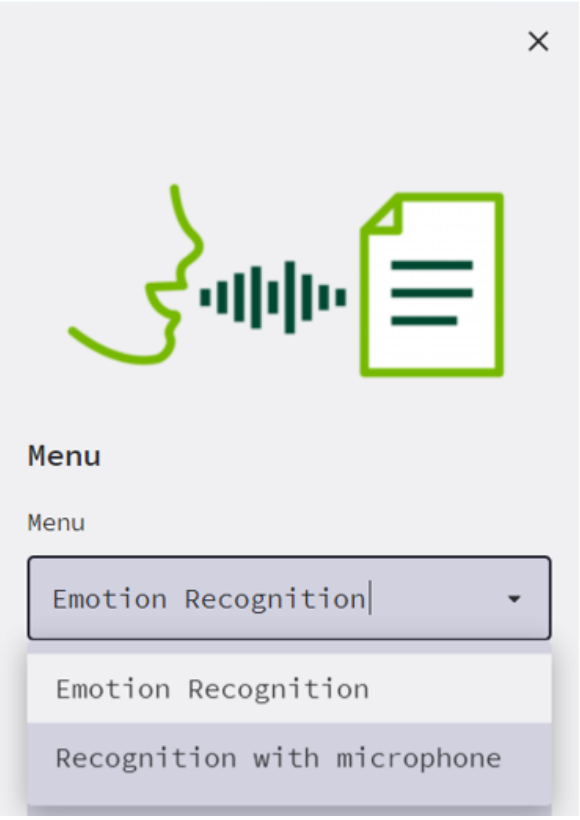

Рисунок 3.1 – Розроблене меню веб-додатку

На рисунку 3.2 зображений перший варіант використання додатку – розпізнавання голосових даних користувача з мікрофону.

Користуваціві потрібно нажати кнопку “Start” тоді промовити бажану фразу, які ви хочете розпізнати. Після завершення нажати “Stop”.

Рисунок 3.2 – Розпізнавання даних з мікрофону

Протестуємо розпізнавання даних з мікрофону з використанням AssemblyAI API. Результати тестування представимо на рисунку 3.3.

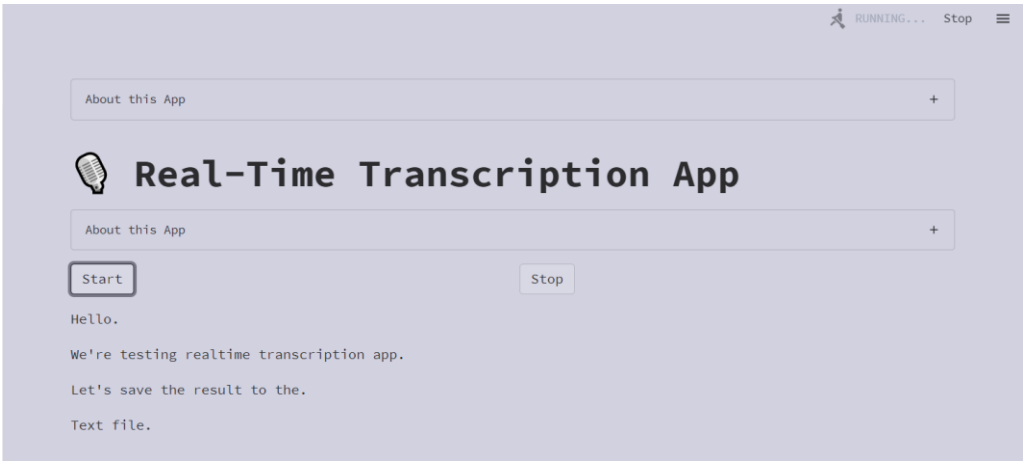

Рисунок 3.3 – Запис голосових даних з мікрофону

В результаті голосові дані були успішно розпізнані та відформатовані. Результат збережено у текстовий файл.

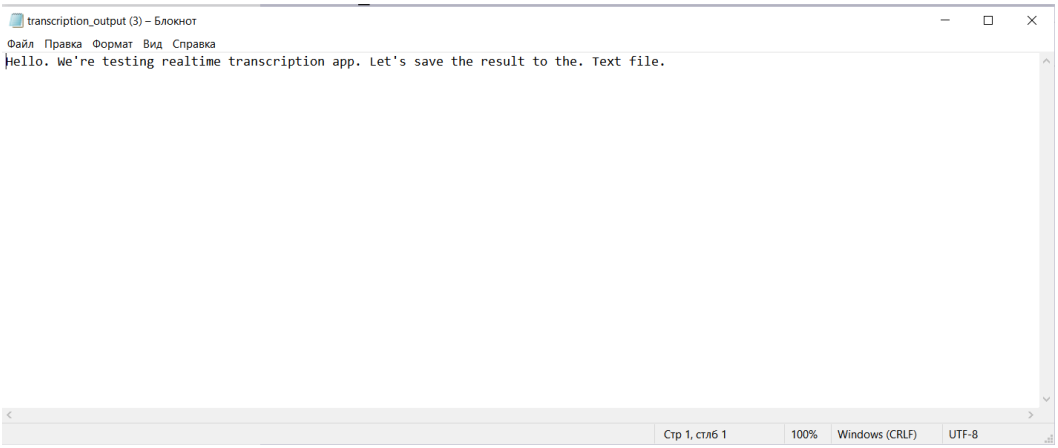

Рисунок 3.4 – Результат збереженого текстового файлу

Під час розробки програмної реалізації були об’єднані дві моделі: попередньо навчений DenseNet для mel-спектрограм і CNN для mfcc.

Набори даних, які були використані для навчання нейронної мережі:

1) Crowd-sourced Emotional Mutimodal Actors Dataset (Crema-D)

2) Ryerson Audio-Visual Database of Emotional Speech and Song (RAVdess)

На рисунку 3.5 зображена структура розробленої нейромережі для розпізнавання голосових даних з використанням машинного навчання.

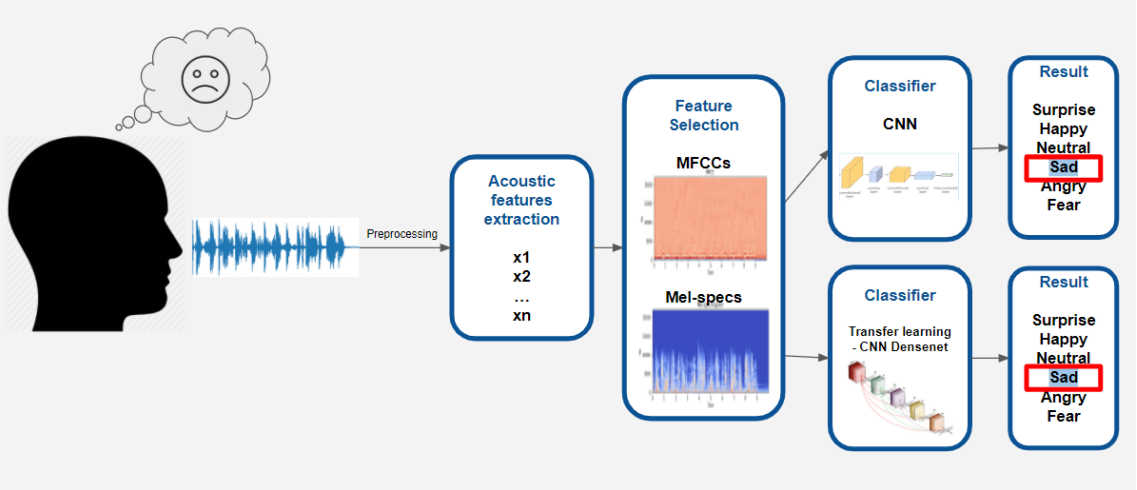

Рисунок 3.5 – Структура попередньої обробки даних для розробленої нейронної мережі

На рисунках 3.6 – 3.7 зображені результати тестування завантаження файлу з розширенням .wav (baker_arduous.wav), який має наступну аудіо-доріжку:‘This has been a long and arduous process for everyone involved on both sides. Thank you and good evening.’.

Рисунок 3.6 – Результати тестування розпізнавання тексту з аудіо-доріжки №1

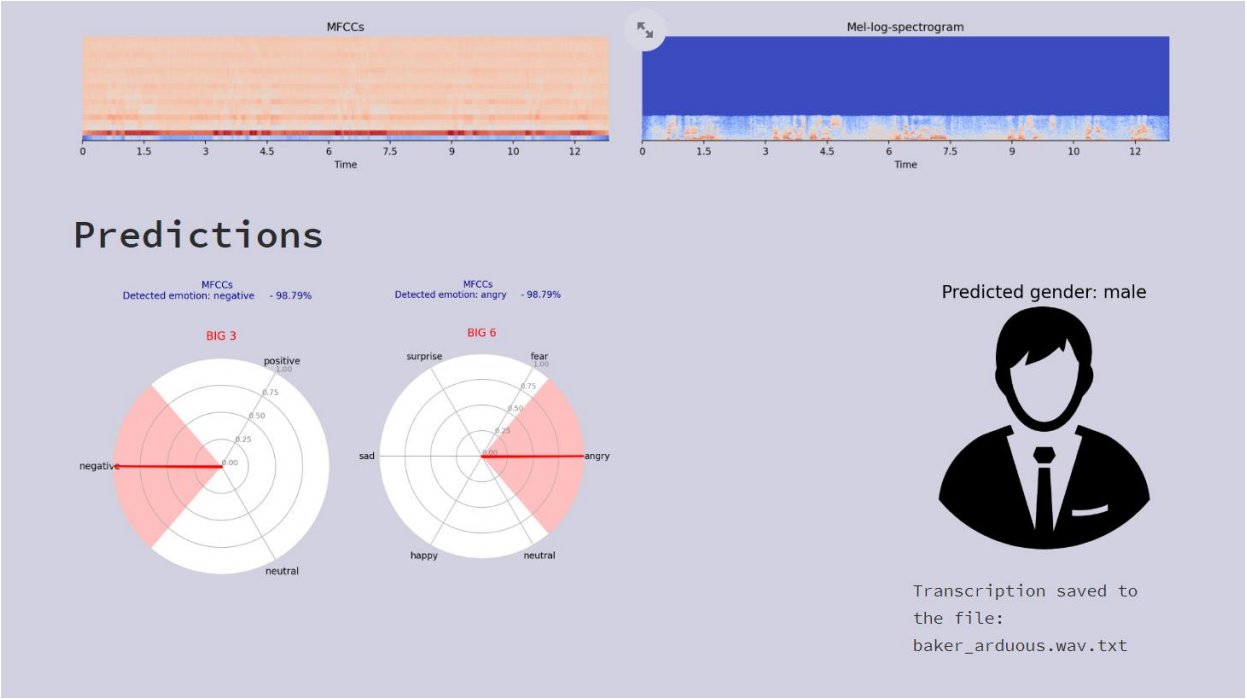

Рисунок 3.7 – Результати класифікування даних з аудіо-доріжки

Як можемо побачити розпізнавання пройшло успішно. Також нейронна мережа успішно класифікувала дані, розпізнавши що це говорить чоловік і його вимова є негативною і агрисивною. Результати розпізнаного тексту також були збереженні у текстовий файл.

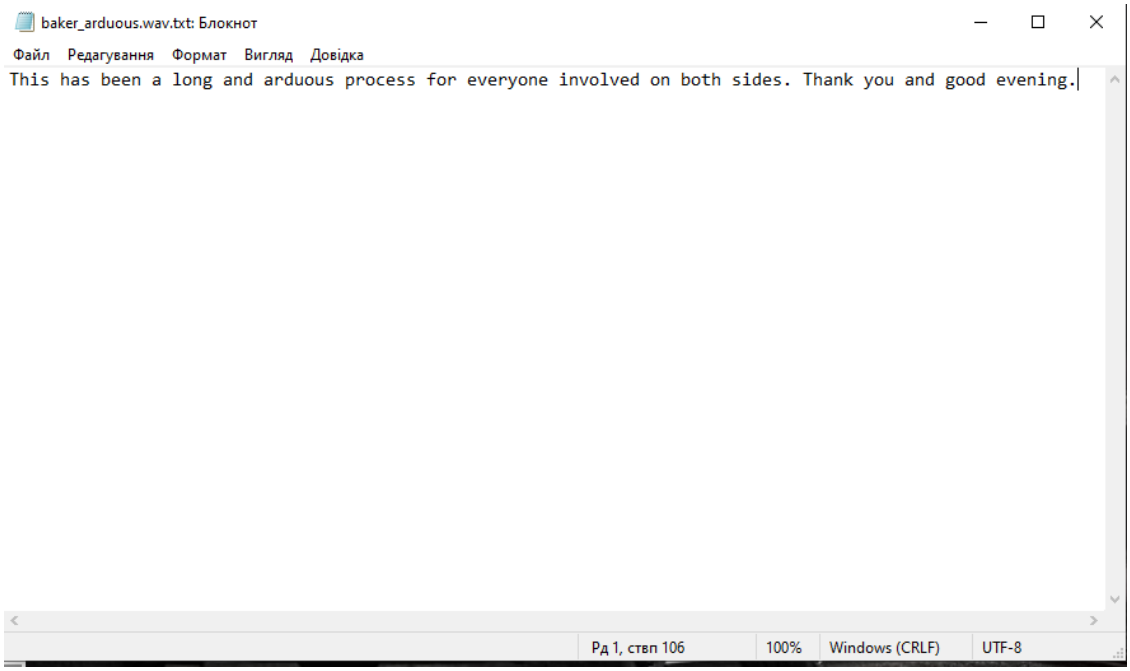

Рисунок 3.8 – Результат збереженого текстового файлу

3.3. Опис альтернативних підходів, які розглядалися під час розробки

Як можна зрозуміти з попередніх розділів, для вирішення цієї задачі можна використовувати різні вже готові системи для розпізнавання. При розробці системи використовувалось дві: це Google Speech Recognition API який є у вільному доступі для невеликих файлів. Також використовувався платний аналог: AssemblyAI API, який має набагато більше властивостей. Одна з них це редагування тексту одразу після розпізнавання.

3.4. Інструкції користувачам

3.4.1. Інсталяція системи

Система працює за допомогою середовища PyCharm IDE, яке потрібно попередньо встановити на комп'ютер. Також для роботи програми необхідно встановити такі компоненти:

Спочатку слід встановити Python. Це можна зробити на офіційному сайті python.org. На сайті можна обрати версію для різних операційних систем.

Після встановлення пайтона, потрібно встановити необхідні бібліотеки: pyaudio, numpy, matplotlib, librosa, requests, auth_key

3.4.2. Інструкція використання

Для роботи з розпізнавання з мікрофона до комп'ютера повинен бути під'єднаний хоча б один мікрофон. Обрати необхідне поле в меню, натиснути кнопку "Start" і почати говорити, після завершення натиснути кнопку "Stop". Програма обробить дані, і виведе результат на екран, також дані записуються у файл. Для роботи з розпізнаванням з файлу користувачу потрібно обрати на комп'ютері попередньо підготований файл з у форматі .wav або .mp3. Після цього програма проаналізує дані, обробить їх, розпізнає і поверне результат. Час опрацювання даних залежить від кількості даних у файлі. Чим їх більше – тим довше працюватиме програма. Якщо користувач завантажив файл з даними, записані у неправильному форматі, то програма завершить роботу.

3.5.Тестування

Тестування – це важливий етап в життєвому циклі розробки програмного забезпечення. Воно забезпечує працездатність та якість системи. Є дуже багато видів тестування. В загальному усі види тестування можна поділити на функціональні та нефункціональні. До функціональних можна віднести тестування методом чорної скриньки, unit-тестування компонентів системи, тестування графічного інтерфейсу користувача.

Були проведені наступні тести:

Тест №1. Розпізнавання з файлу. Файл невеликого розміру, який містить голосові дані спокійної вимови з середнім рівнем шуму.

Тест №2. Розпізнавання з файлу. Файл невеликого розміру, який містить голосові дані спокійної вимови з великим рівнем шуму.

Тест №3. Вибір файлу з неправильним форматом. Після вибору файлу, програма аналізує дані і не може їх розпізнати.

3.6.Опис проведених тестів

Тест №1. На рисунках 3.9 – 3.11 зображені результати тестування завантаження файлу з розширенням .wav (bushw_eavesdropping2.wav), який має наступну аудіо-доріжку: ‘Congress and the authorization in authorization of the use of troops, basically said the President ought to, ought to protect us. Well, one way to protect us is to understand the nature of the enemy. Part of being able to deal with this kind of enemy in a different kind of war is to understand why they're making decisions they're making inside our country.’.

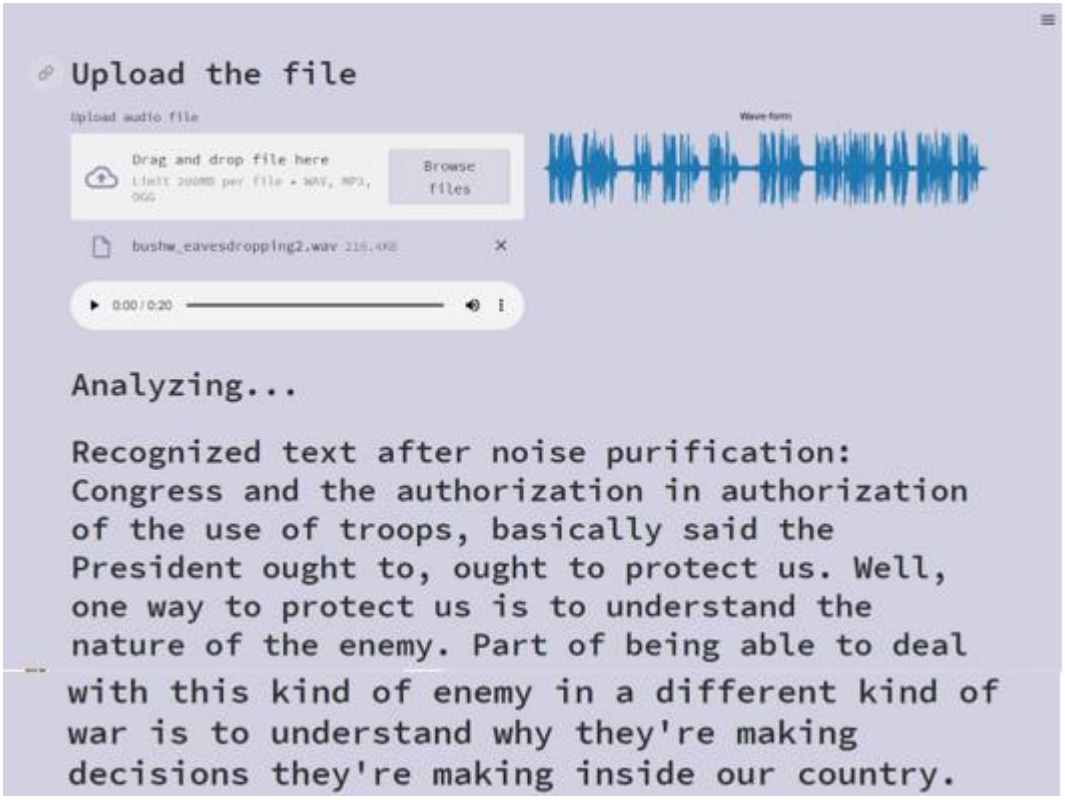

Рисунок 3.9 – Результати тестування розпізнавання тексту з аудіо-доріжки

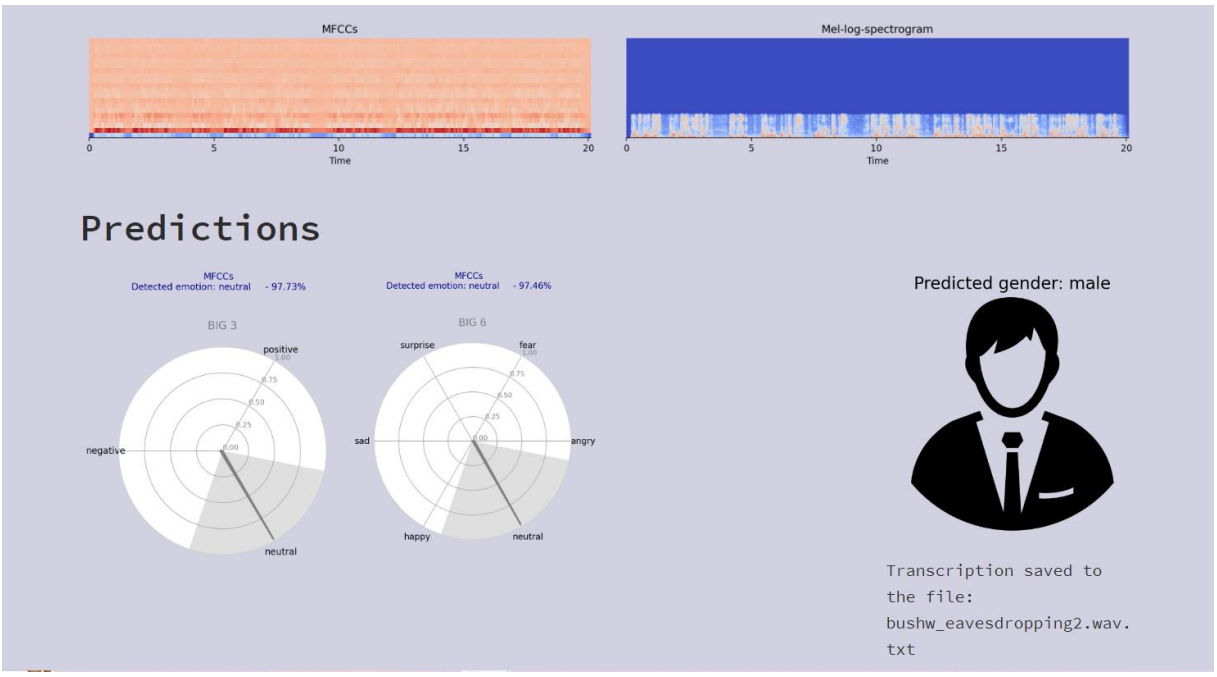

Рисунок 3.10 – Результати класифікування даних з аудіо-доріжки

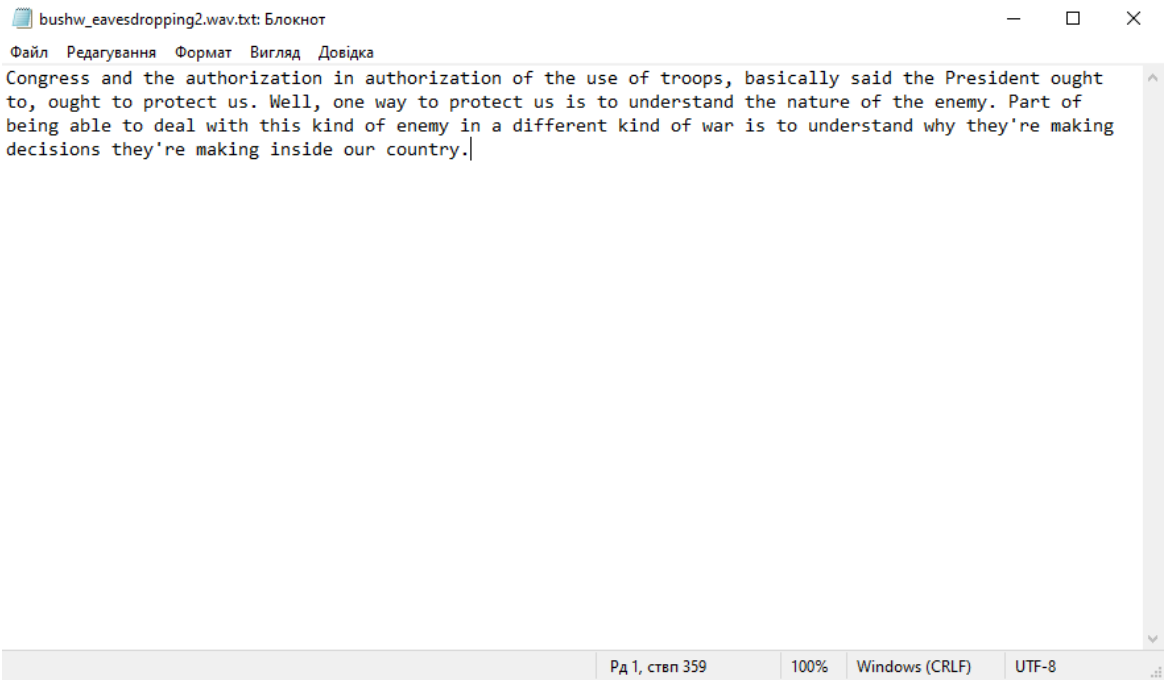

Рисунок 3.11 – Результат збереженого текстового файлу

На рисунках 3.12 – 3.14 зображені результати тестування завантаження файлу з розширенням .wav (bushjeb_lawyer.wav), який має наступну аудіо-доріжку: ‘I am disgusted by the lawyers. It's just, uh, I think, a sign of the times that everything has to get lawyered-up in our country. Civil society is really under attack, and, um, I think people will reject it.’.

Рисунок 3.12 – Результати тестування розпізнавання тексту з аудіо-доріжки

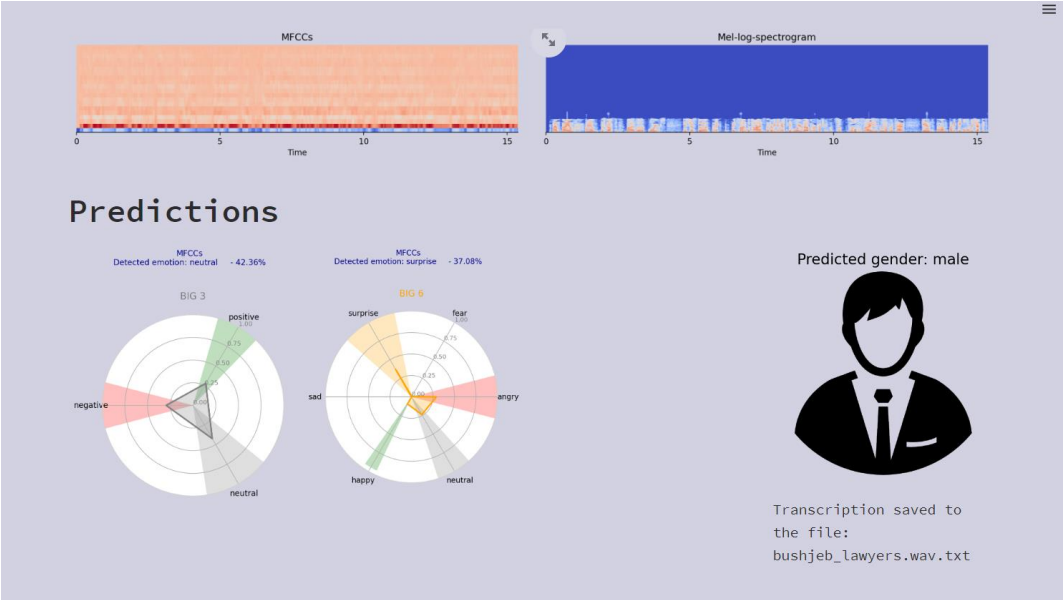

Рисунок 3.13 – Результати класифікування даних з аудіо-доріжки

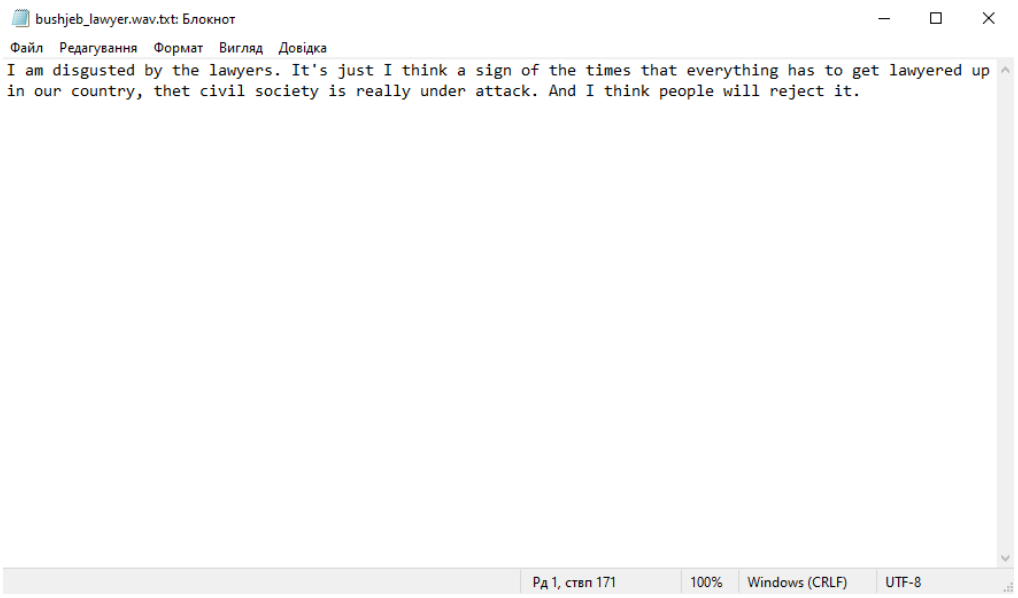

Рисунок 3.14 – Результат збереженого текстового файлу

3.7.Оцінювання та аналіз результатів

За результатами тестування, проведеним на основі двох аудіо-доріжок різної довжини, різної величини фонових шумів, а також різних рівнів голосів було отримано наступні результати:

1. Голосові дані з доріжки 1 були повністю розпізнані та класифіковані відповідно до поданих вхідних даних.

2. Голосові дані з доріжки 2 були повністю розпізнані, але з деякими перестановками слів у тексті для формування тексту, який має сенс.

3. Класифікація даних з поданих аудіо-доріжок повністю відповідає даним, з яких складається аудіо-доріжка.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Аналіз небезпечних і шкідливих виробничих чинників та розробка заходів щодо покращення умов праці

Потенційно шкідливим чинником кабінету особи, яка приймає управлінські рішення, вважається небезпека враження людини електричним струмом. Важливим, але менш ймовірним чинником являється пожежна небезпека під час аварійної ситуації. Хімічні та біологічні джерела практично не мають впливу.

Перелік небезпечних та шкідливих виробничих чинників наведено у таблиці 4.1.

Таблиця 4.1. Небезпечні та шкідливі виробничі чинники

Фізичні	Електронебезпека, пожежа, шум, мікроклімат
Хімічні	Відсутні
Біологічні	Відсутні
Психофізіологічні	Відсутні

В приміщенні кабінету особи, яка приймає управлінські рішення, присутні небезпечні чинники, та за умов дотримання заходів безпеки, вони не є критичним.

4.2. Розробка логічно-імітаційної моделі процесу виникнення травм під час монтажу інтелектуальної інформаційної системи

Завжди можна знайти подію, з якої починається небезпечний процес ще до виникнення небезпечних наслідків, проаналізувавши кожну з логічних моделей процесів формування та потенційних травмонебезпечних та аварійних ситуацій.

У процесі оцінки рівня небезпеки на робочих місцях, машинах, виробничих процесах чи окремих виробництвах необхідно знайти конкретний

критерій рівня небезпеки. Такий показник показує ймовірність аварії або травми залежно від явища.

Для оцінки рівня небезпеки цього об’єкта можна використовувати метод обчислення ймовірності виникнення будь-якого випадкового явища, який використовується в закордонній практиці. Його основним принципом є визначення виробничих небезпек, потенційних аварій і травм на основі обстеження робочої зони чи окремої машини. У процесі оцінки цих факторів визначаються події, які можуть бути вирішальними для створення логічно-імітаційної моделі травми. Наступним кроком є створення моделі, відомої як «дерева відмов і помилок оператора». Правильний вибір головної події має вирішальне значення.

Головну подію або як її ще називають «травма», модель якої нам необхідно створити, вибирають з оцінки вибраного об’єкта, виробництва чи окремого обладнання і змісту найбільш небезпечного явища, яке може виникнути за певних умов виробництва.

Розпочинаємо побудову моделі після вибору головного явища або події. Використовуючи оператори «і» та «або», ми можемо створити набір відомих ситуацій, які можуть призвести до події, вибраної як головна.

Після визначення відповідних травмонебезпечних ситуацій та їх кількості проводимо логічний аналіз із використанням операторів «і», «або» та інших, щоб визначити додаткові події, пов’язані з кожною з цих ситуацій. Поки не будуть знайдені всі основні події, які визначають межі моделі, процес побудови моделі не завершиться.

Слід мати на увазі, що кожна випадкова подія, до якої входять базові події, може формуватися та виникати за допомогою відповідних операторів, коли до неї входять два, три або більше базових подій.

Модель повинна бути математично оброблена, щоб визначити ймовірність кожної випадкової події, яка увійшла до неї, починаючи з базових і закінчуючи головною.

Ми використовуємо дані виробництва для визначення ймовірності

основних подій. Основна подія, наприклад, «стан контролю з охорони праці». Для визначення ймовірності ми повинні визначити, наскільки (у відсотках) від ідеального рівня здійснюється відповідний контроль на об'єкті. Імовірність буде 0,5, якщо рівень контролю становить 50% або 30%. Якщо контролю немає, ймовірність «не здійснення контролю» становить 1, а якщо контроль ідеальний, то ймовірність дорівнює 0.

Після обчислення ймовірності всіх подій, розміщених у ромбах, і базових подій, починаючи з лівої нижньої гілки «дерева», ми позначаємо всі випадкові події, включені до моделі, номерами.

Таким чином, можна припустити, що певна модель готова до математичних обчислень ймовірностей випадкових подій у логічно-імітаційній моделі.

Таким чином, ми складемо список основних подій, необхідних для створення логіко-імітаційної моделі процесу, формування та виникнення аварій і травм під час створення інтелектуальної інформаційної системи оцінювання платоспроможності сільськогосподарських підприємств. Вони будуть основою для цієї моделі. Ми присвоюємо певне значення ймовірності виникнення кожному пункту списку. Нижче представлено повний список:

1. Стан контролю з охорони праці ..... **$P_1 = 0,2;$**
2. Несерйозне відношення до проходження ТО інструменту **$P_2 = 0,1;$**
3. Відсутність комплектуючих установки..... **$P_3 = 0,2;$**
4. Невисока міцність ..... **$P_4 = 0,03;$**

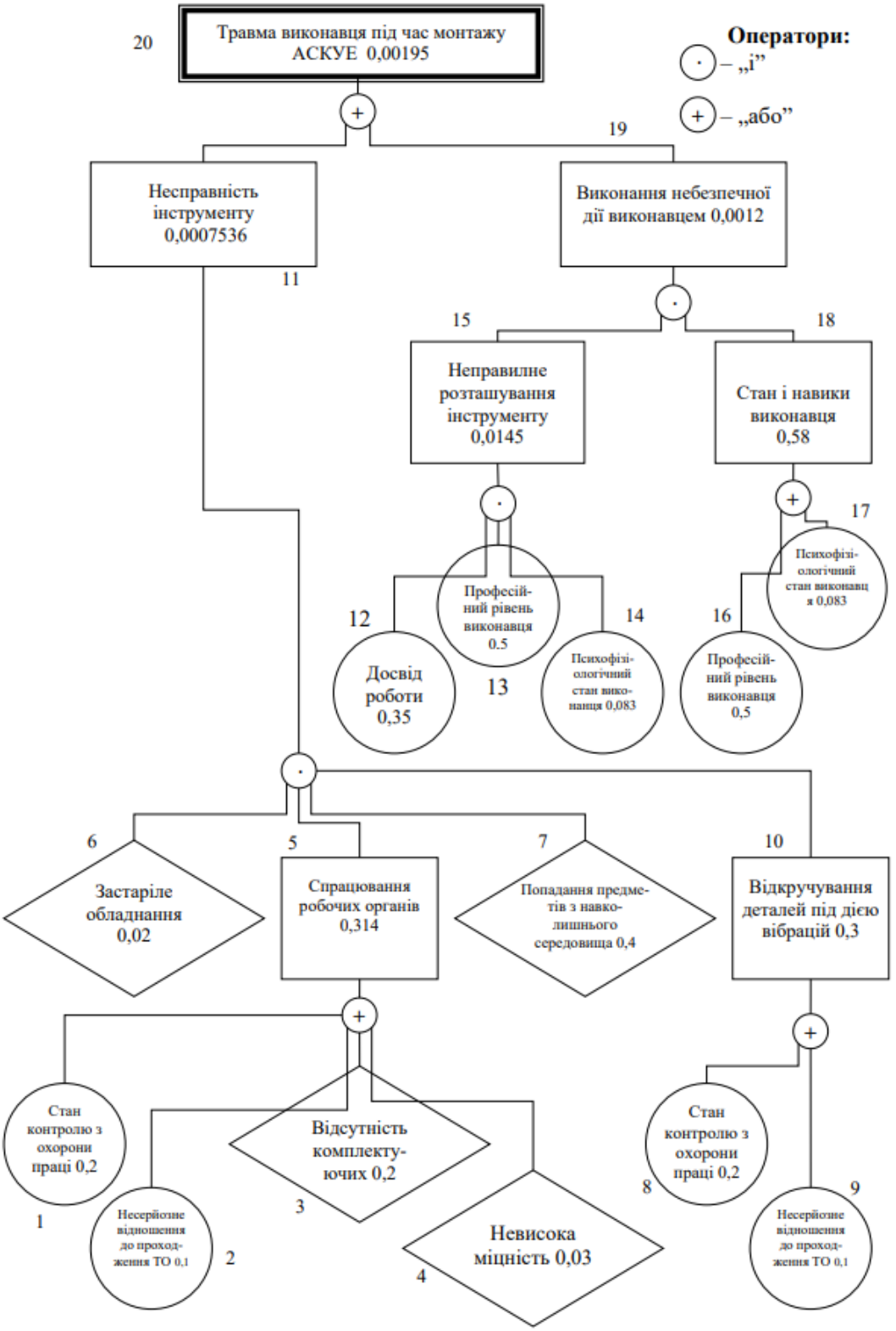

Рисунок 4.1. - Логіко-імітаційна модель процесу формування та виникнення аваріїта травми під час монтажу системи розпізнавання голосових даних

1. Використання застарілого обладнання..... **P_6 = 0,02;**
2. Попадання сторонніх предметів ..... **P_7 = 0,4;**
3. Досвід роботи виконавця ..... **P_{12} = 0,35**.
4. Професійний рівень виконавця ..... **P_{13} = 0,5;**
5. Психофізіологічний стан виконавця..... **P_{14} = 0,083;**

На основі даного списку будуємо матрицю логічних взаємозв'язків між окремими пунктами, графічне представлення якої зображено на рис. 4.1.

Розрахуємо ймовірності виникнення подій, що входять у дану логіко-імітаційну модель процесу монтажу інтелектуальної інформаційної системи оцінювання платоспроможності сільськогосподарських підприємств (на прикладі ймовірності отримання травми виконавця).

Ймовірність виникнення події P_5 визначаємо наступним чином:

$$P_5 = 02 + 0,1 + 0,2 + 0,003 - 0,2 \cdot 0,1 - 0,2 \cdot 0,03 - 0,2 \cdot 0,03 - 0,1 \cdot 0,2 - 0,1 \cdot 0,03 -$$
$$- 0,2 \cdot 0,03 + 0,2 \cdot 0,1 \cdot 0,2 + 0,1 \cdot 0,2 \cdot 0,03 + 0,2 \cdot 0,1 \cdot 0,2 + 0,2 \cdot 0,1 \cdot 0,03 -$$
$$- 0,2 \cdot 0,1 \cdot 0,2 \cdot 0,03 = 0,314$$

Ймовірність виникнення події P_{10} визначаємо так:

$$P_{10} = 0,2 + 0,1 = 0,3.$$

Ймовірність виникнення події P_{11} визначаємо:

$$P_{11} = 0,02 \cdot 0,314 \cdot 0,4 \cdot 0,3 = 0,00075.$$

Ймовірність виникнення події P_{15} визначаємо наступним чином:

$$P_{15} = 0,35 \cdot 0,5 \cdot 0,083 = 0,0145.$$

Ймовірність події P_{18}:

$$P_{18} = 0,5 + 0,083 = 0,58.$$

Ймовірність події P_{19}:

$$P_{19} = 0,0145 \cdot 0,083 = 0,0012.$$

Зважаючи на те, що аварія можлива лише за умови встановлення автоматизованої системи управління енергоспоживанням людиною, ймовірність травми рівна ймовірності аварії.

Логіко-імітаційні моделі аварій і травм зменшують ймовірність аварій і травм. Якщо необхідно оцінити рівень небезпеки будь-якого робочого місця, слід ретельно вивчити та побудувати логічні моделі потенційних небезпечних ситуацій, які охоплюють стан обладнання та самого робочого місця, а також поведінку працівників, щоб обчислити ймовірність виникнення травми.

Після аналізу результатів моделювання ймовірність виникнення травми можна звести до дуже малої величини – достатньо зменшити вплив ймовірностей вихідних факторів, які до неї призводять.

4.3. Розробка заходів щодо безпеки у надзвичайних ситуаціях

Науково-технічний прогрес радикально змінив світ, породивши нові загрози для цивілізації. У житті сучасної людини все більше місце займають турботи, пов'язані з подоланням різних кризових явищ, що виникають в процесірозвитку земної цивілізації. В Україні, як і в усьому світі, в останні роки спостерігається зростання числа військових дій, катастроф природного та техногенного характеру. Це обумовлено, перш за все, прогресуючоїурбанізацією територій, збільшенням щільності населення Землі, і, як наслідок, збільшенням антропогенного навантаження на навколишнє середовище. Захист природних систем і населення від надзвичайних ситуацій різного характеру сформувалася в останні роки як нагальна і об'єктивна потреба суспільства і держави.

Заходи щодо захисту цивільного населення плануються проводяться по населених пунктах де розміщені підприємства і охоплюють населення навколишніх сіл. Водночас характер та зміст захисних засобів встановлюються

вид ступеня загрози, місцевих умов з урахуванням важливості виробництва для безпеки населення і інших економічних і соціальних чинників.

Основні заходи щодо захисту населення плануються та здійснюються завчасно і мають випереджувальний характер, це стосується насамперед підготовки, підтримання у постійній готовності індивідуальних та колективних засобів захисту, їх накопичення, а також підготовки до проведення евакуації населення із зон підвищеного ризику.

Також раз в три роки проводяться навчання по підготовці близьких до військових дій, що в разі небезпеки могло би не дістати людину зненацька. Керівництво докладає максимум зусиль, щоб працівники підприємств були хоча би мінімально захищенні в разі будь-якої небезпеки пов’язаної з тими чи іншими обставинами.

РОЗДІЛ 5
ВИЗНАЧЕННЯ ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ.

5.1 Економічна характеристика проектного рішення

Метою проекту є дослідження існуючих систем і методів голосового введення та розпізнавання команд з використанням штучних нейронних мереж, а також розробка нових або покращення функціональності існуючих методів розпізнавання голосових команд із застосуванням програмного та апаратного підходу, що дозволяє точніше розпізнавати голос у поданому файлі з форматом .wav або при записі аудіофайлів навіть у шумному середовищі з неякісними мікрофонами.

5.2 Інформаційне забезпечення та формування гіпотези щодо потреби розроблення товару

Хоча розпізнавання звуку та людська мова все більше і більше інтегруються в наше повсякденне життя, вони також стають все простішими, що значно полегшує спілкування за допомогою голосового введення, пошук інформації за допомогою звукових шаблонів, керування широким спектром пристроїв, і навіть розважатися з ними. На жаль, точне розпізнавання людської мови – це не тільки трудомістка операція, але й вимагає достатньо чіткого та чистого людського голосу, чого важко досягти з огляду на постійний шквал шуму.

Кінцеві користувачі систем розпізнавання аудіоданих і голосового введення, з іншого боку, очікують високої точності розпізнавання команд і швидкого реагування системи.

Проте поєднання вищеописаних явищ та шуму, а також кожного з них окремо, істотно ускладнює процес виявлення людської мови. З цього можна зробити висновок, що тема «Розпізнавання голосових даних за допомогою технологій глибокого навчання/машинного навчання» зараз є важливою.

5.3 Бюджетування

Бюджетування є комплексно обґрунтованою системою розрахунку витрат, пов’язаних з виготовленням та реалізацією продукту, яка дає можливість

здійснити аналіз витрат та розробити заходи щодо підвищення рентабельності виробництва. На даному етапі необхідно визначити собівартість продукту, який розробляється та економічно обґрунтувати доцільність вибору однієї із стратегій. Бюджет витрат на матеріали та комплектуючі наведено в таблиці 5.1.

Таблиця 5.1 - Бюджет витрат на матеріали та комплектуючі виробів

Назва матеріалів та комплектуючих	**Марка, тип, модель**	**Фактична кількість, шт.**	**Ціна за одиницю, грн.**	**Разом, грн.**
Оренда компютера	cpu i7, 16GBRam, nvidia rtx 3050, ssd -256	1	7000	7000
Програмне забезпечення	Windows 10	1	5200	4200
Флешка	Kingston 600GB	2	1100	2200
Разом:				13400

Для розробки проекту також потрібні людські ресурси. Проте, щоб реалізувати цей проект не потрібно наймати велику команду. Для реалізації проекту стануть в нагоді програміст на мові Python та тестер. В таблиці 5.2 наведено бюджет витрат на оплату праці.

Таблиця 5.2 - Бюджет витрат на оплату праці

Посада, спеціальність	**Кількість працівників, осіб**	**Часс роботи, дні**	**Денна заробітна плата працівників, грн.**	**Сума витрат на оплату праці, грн.**
Основна заробітна плата				
Програміст	1	21	1500	31500
Тестер	1	7	1100	7700
Разом:				39200

У таблиці 5.3 представлено бюджет загальновиробничих витрат.

Таблиця 5.3 - бюджет загальновиробничих витрат

Статті витрат	**Сума, грн.**
1	2
Змінні загальновиробничі витрати, у т.ч.:	
• заробітна плата персоналу;	39200
• витрати на МШП;	400
• витрати на електроенергію;	500
• витрати на ремонт;	0
• інші змінні витрати;	0
Разом змінних витрат:	40100
Постійні загальновиробничі витрати, у т.ч.:	
• заробітна плата допоміжного персоналу;	0
• комунальні послуги;	500
• витрати на оренду;	3000
• витрати на ремонт;	1000
• інші постійні витрати;	0
Разом постійних витрат:	4500
Разом загальновиробничих витрат:	44600

До адміністративних витрат відносяться загальногосподарські втрати, спрямовані на обслуговування та управління підприємством. Витрати на збут включають витрати, пов'язані з реалізацією (збутом) продукції (товарів, робіт, послуг). Бюджет адміністративних витрат та витрат на збут наведено в таблиці 5.4.

Витрати на виробництво продукції у вартісному виразі формують її виробничу собівартість. Цей показник є одним з найважливіших економічних показників господарської діяльності підприємства, у якому дістають відображення зростання продуктивності праці, економія ресурсів.

Таблиця 5.4 - Бюджет адміністративних витрат та витрат на збут

Статті витрат	**Сума, грн.**

1	2
Адміністративні витрати, у т.ч.:	
заробітна плата адміністративного персоналу	1500
витрати на МШП	400
витрати на відрядження	0
витрати на ремонт	0
витрати на паливно-мастильні матеріали	0
витрати на сплату податків і зборів	5280
знос адміністративного обладнання	300
Разом адміністративних витрат:	7480
Витрати на збут, у т.ч.:	
- заробітна плата менеджерів зі збуту;	4000
- витрати на гарантійний ремонт;	2300
- витрати на відрядження;	0
- витрати на гарантійне обслуговування;	3100
- витрати на налагодження і експлуатацію;	400
- витрати на паливо-мастильні матеріали;	0
- витрати на рекламу;	4000
Разом витрат на збут:	13800

Для того, щоб можна було зручно переглянути та проаналізувати усі статті витрат, варто скласти таблицю зведеного кошторису витрат на розробку. Зведені витрати представлено в таблиці 5.5.

Таблиця 5.5 - Зведений кошторис витрат на розробку проектного рішення

Статті витрат	**Разом, грн.**
1	2
Сировина і матеріали	13400
Купівельні напівфабрикати та комплектуючі вироби	0
Зворотні відходи (вираховуються)	0

Паливо та електроенергія на технологічні цілі	0
Амортизаційні відрахування основних фондів	480
Основна заробітна плата	39200
Додаткова заробітна плата	0
Відрахування на соціальне страхування	5280
Витрати на утримання й експлуатацію устаткування	0
Загальновиробничі витрати, у т.ч.:	
змінні	40100
постійні	4500
Разом виробничих витрат:	44600
Адміністративні витрати	7480
Витрати на збут	13800
Інші операційні витрати	0
Разом виробничих і операційних витрат:	65880

Для визначення фінансових результатів, необхідно розрахувати вартість (ціну) продукту (проектного рішення), який розробляється. Ціна визначається на основі суми виробничих і операційних витрат з врахуванням рентабельності виробництва 70%.

$$Ц = СБ * Р$$

де Ц – ціна одинці продукту, грн.

СБ – собівартість продукту, грн.

Р – рентабельність виробництва, %

$$Ц = 65880*0,7 = 46116$$

У таблиці 5.6 наведено бюджет фінансових результатів.

Таблиця 5.6 -Бюджет фінансових результатів

Показники	Сума, грн.
Дохід від реалізації продукції (6 шт)	276696
Податок на додану вартість (20%)	55339,2
Чистий дохід від реалізації продукції	221356,8
Собівартість реалізованої продукції	65880
Валовий прибуток	155476,8
Операційні витрати:	
- адміністративні витрати:	7480
- витрати на збут;	13800
- інші операційні витрати;	0
Фінансовий результат від операційної діяльності	134196,8
Податок на прибуток (18%)	37575,1
Чистий прибуток (збиток)	96621,7

5.4 Висновки до економічної ефективності

Отже, під час дослідження було розраховано економічні складові проектування та розробки системи розпізнавання голосових даних з використанням штучних нейронних мереж.

На основі аналізу ринку оцінено фактори внутрішнього та зовнішнього середовища, а також їх вплив на формування потреби у програмному рішенні. За результатами експертної оцінки можна зробити висновок, що серед факторів зовнішнього середовища позитивний вплив мають високий рівень науково-технічного прогресу та споживачі (сформована потреба у продукті). Сумарний вплив факторів зовнішнього середовища є досить добрим показником і становить 1,38.

Сумарний вплив факторів внутрішнього середовища становить 4,16 і свідчить про хороші можливості виходу даного рішення на ринок.

На основі аналізу стратегічних альтернатив було обрано стратегію розвитку існуючого продукту та стратегію розвитку ринку. Оскільки на ринку вже існують подібні системи та дану систему можна буде застосовувати у нових сферах діяльності.

Проведено аналіз витрат, пов’язаних з реалізацією проекту і з’ясовано, що за рівня рентабельності 70% ціна на розроблюване ПЗ становитиме 65 880 грн.

Величина чистого прибутку 96 621,7 грн свідчить про доцільність розробки такого продукту.

ВИСНОВКИ І ПРОПОЗИЦІЇ

В кваліфікаційні роботі було розглянуто задачу розпізнавання голосових даних та розроблений програмний застосунок.

Для створення програмного рішення було використано мову програмування Python з використанням сторонніх бібліотек та сервісів. Написання програми виконувалась у середовищі PyCharm 2021.

У першому розділі роботи було описано задачі, які можуть бути вирішені за допомогою нейронних мереж та сформульовано постановку задачі, та питання, які мають бути вирішені: розпізнавання з мікрофону, розпізнавання з файлу, можливість зберігати отримані значення у файл.

Також було здійснено пошук та огляд інформаційних джерел, яка була би корисною для вирішення поставленого завдання. Було розглянуто загальну інформацію про нейронні мережі, їхню архітектуру, структуру типи та як вони навчаються. Також було описано існуючі підходи до вирішення цієї проблеми.

Другий розділ роботи містить детальний опис предметної області. Також тут було розроблено та проаналізовано дерево цілей та дерево проблем, виділено основні задачі та під-задачі та було виконано аналіз і вибір методів, алгоритмів та засобів розв'язання задачі. В процесі роботи над розділом було спроектовано блок-схеми базових алгоритмів, що використовуються в системі, створено наступні діаграми: IDEF0, DFD, IDEF3, use-case, діаграму послідовностей. Які дають змогу краще зрозуміти, як працюватиме розроблена система. Було описано загальну структуру програмного проекту, описано використані бібліотеки та модулі, інтерфейс користувача, розробку та опис програмних модулів.

У третьому розділі роботи наведені інструкції для користувачів системи і вимоги до апаратного та програмного забезпечення, яке необхідне для коректної роботи застосунку. Особливо детально було проведено тестування якості розробленої системи, адже саме від коректності її роботи і залежить те, чи будуть нею користуватись в майбутньому.

У четвертому розділі описано охорону праці та безпеку у надзвичайних ситуаціях.

У п’ятому розділі проводилося економічне обґрунтування доцільності розробки програмного рішення. На основі аналізу ринку визначено основних споживачів та головних конкурентів. А також оцінено фактори внутрішнього та зовнішнього середовищ і їх вплив на формування потреби у програмному рішенні для вирішення задач розпізнавання.

Розроблена система – це пілотна версія продукту, який розроблений для здійснення розпізнавання голосових даних. Надалі, в залежності від потреб користувача є можливість доповнювати та розширювати функціонал який в майбутньому зможе досягнути популярності серед користувачів. Даний проект є перспективним, адже на сьогоднішній день все більше користувачів зацікавлені в швидкому та точному розпізнаванні їхнього голосу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Катренко А. В. Системний аналіз об’єктів та процесів комп’ютеризації. Учб. посіб. / А. В. Катренко. – Львів, 2018.

2. Губанов В.А. і др. Введення в системний аналіз: Навчальний посібник /Под ред. Л.А. Петросяна. - Л.: вида-во ЛГУ.

3. Python Cookbook, 2019. – 667 с. – (3). – (ISBN: 978-1-443-34037-7).

4. Архітектури рекурентної нейронної мережі довгострокової пам'яті для великомасштабного акустичного моделювання (2021), Хасім Сак та ін., Google Research

5. Надо Д. Огляд розпізнавання та класифікації названих сутностей / Д. Надо, С. Секіне // Національна дослідницька рада Канади – 2016.

6. Індуркх'я Н., Дамерау Ф. (2018). Посібник з обробки природної мови, 2-ге видання. Бока-Ратон, Флорида: CRC Press

7. Секіне С. Названі сутності: розпізнавання, класифікація та використання / С. Секіне, Е. Ранчход // Видавництво Джона Бенджамінса – 2019.

8. Грейвс А, Джейтлі Н, Мохамед А.Р. Гібридне розпізнавання мовлення з глибоким двонаправленим LSTM. In: Automatic Speech Recognition and Understanding (ASRU), IEEE Workshop on. 2021 р.

9. Абдель-Хамід О., Абдель-Рахман М., Хуї Дж., Лі Д., Пенн Г., Ю Д. Транзакції IEEE/ACM щодо обробки аудіо, мовлення та мови. Том 22. Випуск 10: Згорткові нейронні мережі для розпізнавання мовлення. 2017 р.

10. Miao Y, Gowayyed M, Metze F. EESEN: наскрізне розпізнавання мови з використанням глибоких моделей RNN і декодування на основі WFST. In: Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on. IEEE; 2015 рік

11. Макфі, Брайан, Колін Раффел, Довен Лянг, Деніел П. В. Елліс, Метт Маквікар, Ерік Баттенберг та Оріол Ньєто. «librosa: аналіз аудіо та музичних сигналів у python». У Матеріалах 14-ї конференції «Python у науці», с. 18-25. 2015 рік.

12. Приховані моделі Маркова для розпізнавання мовлення — Режим доступу до ресурсу: http://www.machine-models.com/markov.html..

13. Python eats away at R: Top Software for Analytics, Data Science, Machine Learning in 2018: Trends and Analysis // KDnuggets. – 2018. – Режим доступу до ресурсу: https://www.kdnuggets.com/2018/05/poll-tools-analytics-data-science-machine-learning-results.html

14. Монолітна архітектура розроблення. — Режим доступу до ресурсу: http://microservices.io/patterns/monolithic.html.

15. Michael D. Announcing Michael Droettboom as the lead matplotlib developer / Droettboom Michael // matplotlib.org. – 2018. – Режим доступу до ресурсу: http://matplotlib.1069221.n5.nabble.com/ANN-Michael-Droettboom-matplotlib-lead-developer-td5037.html.

16. Бібліотека “Stanford Core NLP” - Режим доступу до ресурсу: https://stanfordnlp.github.io/CoreNLP/index.html.

17. Machine Learning Crash Course – Режим доступу до ресурсу: https://developers.google.com/machine-learning/crash-course/.

18. Python IDE for Professional Developers. – 2021. – Режим доступу до ресурсу: https://www.jetbrains.com/pycharm/.

19. Моделі нейронних мереж. – Режим доступу: https://studme.com.ua/1246122010028/neural/models.htm – Саксамудре С., Шрішрімал П., Дешмух Р. Огляд різних підходів до системи розпізнавання мовлення. URL: https://pdfs.semanticscholar.org/b909/3377c6579b97ab8bd5d4dd9947d372dddc2e.pdf

20. Матарне Р., Максимова С., Ляшенко В., Бєлова Н. Системи розпізнавання мовлення: порівняльний огляд. URL: https://pdfs.semanticscholar.org/8c3b/5bab98556f57dbc8142d5f3f8ad13109c733.pdf.

21. Convolutional Neural Networks (LeNet) // DeepLearning 0.1. LISA Lab. – 2018. – Режим доступу до ресурсу: http://deeplearning.net/tutorial/lenet.html.

22. Штучні нейронні мережі: що це таке? // futurum.today. – 2017. – Режим доступу до ресурсу: https://futurum.today/shtuchni-neironni-merezhi-shcho-tse-take/.

23. Advanced examples — librosa 0.8.1 documentation. URL: https://librosa.org/doc/latest/advanced.html

ДОДАТКИ

ДОДАТОК А

```
import pyaudio
import numpy as np
import streamlit as st
import cv2
import librosa
import librosa.display
from tensorflow.keras.models import load_model
import os
from datetime import datetime
import streamlit.components.v1 as components
import matplotlib.pyplot as plt
from PIL import Image
from melspec import plot_colored_polar, plot_melspec
import requests
from configure import auth_key
import time

# load models
model = load_model("model3.h5")

# constants
starttime = datetime.now()

CAT6 = ['fear', 'angry', 'neutral', 'happy', 'sad', 'surprise']
CAT7 = ['fear', 'disgust', 'neutral', 'happy', 'sad', 'surprise', 'angry']
CAT3 = ["positive", "neutral", "negative"]

COLOR_DICT = {"neutral": "grey",
              "positive": "green",
              "happy": "green",
              "surprise": "orange",
              "fear": "purple",
              "negative": "red",
              "angry": "red",
              "sad": "lightblue",
              "disgust": "brown"}

TEST_CAT = ['fear', 'disgust', 'neutral', 'happy', 'sad', 'surprise', 'angry']
TEST_PRED = np.array([.3, .3, .4, .1, .6, .9, .1])

# page settings
st.set_page_config(page_title="SER web-app", page_icon=":speech_balloon:", layout="wide")

with st.expander('About this App'):
    st.markdown('''
    Datasets used in this project:
        1. Crowd-sourced Emotional Mutimodal Actors Dataset (Crema-D)
        2. Ryerson Audio-Visual Database of Emotional Speech and Song (Ravdess)
        3. Surrey Audio-Visual Expressed Emotion (Savee)
        4. Toronto emotional speech set (Tess)
```

```
        ''')

def log_file(txt=None):
    with open("log.txt", "a") as f:
        datetoday = datetime.now().strftime("%d/%m/%Y %H:%M:%S")
        f.write(f"{txt} - {datetoday};\n")


# @st.cache
def save_audio(file):
    if file.size > 4000000:
        return 1
    folder = "audio"
    datetoday = datetime.now().strftime("%d/%m/%Y %H:%M:%S")
    for filename in os.listdir(folder):
        file_path = os.path.join(folder, filename)
        try:
            if os.path.isfile(file_path) or os.path.islink(file_path):
                os.unlink(file_path)
        except Exception as e:
            print('Failed to delete %s. Reason: %s' % (file_path, e))
    try:
        with open("log0.txt", "a") as f:
            f.write(f"{file.name} - {file.size} - {datetoday};\n")
    except:
        pass
    with open(os.path.join(folder, file.name), "wb") as f:
        f.write(file.getbuffer())
    return 0
# @st.cache
def get_melspec(audio):
    y, sr = librosa.load(audio, sr=44100)
    X = librosa.stft(y)
    Xdb = librosa.amplitude_to_db(abs(X))
    img = np.stack((Xdb,) * 3, -1)
    img = img.astype(np.uint8)
    grayImage = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    grayImage = cv2.resize(grayImage, (224, 224))
    rgbImage = np.repeat(grayImage[..., np.newaxis], 3, -1)
    return (rgbImage, Xdb)
# @st.cache
def get_mfccs(audio, limit):
    y, sr = librosa.load(audio)
    a = librosa.feature.mfcc(y, sr=sr, n_mfcc=40)
    if a.shape[1] > limit:
        mfccs = a[:, :limit]
    elif a.shape[1] < limit:
        mfccs = np.zeros((a.shape[0], limit))
        mfccs[:, :a.shape[1]] = a
    return mfccs
@st.cache
def get_title(predictions, categories=CAT6):
    title = f"Detected emotion: {categories[predictions.argmax()]} \
```

```
    - {predictions.max() * 100:.2f}%"
    return title
@st.cache
def color_dict(coldict=COLOR_DICT):
    return COLOR_DICT

@st.cache
def plot_polar(fig, predictions=TEST_PRED, categories=TEST_CAT,
               title="TEST", colors=COLOR_DICT):
    # color_sector = "grey"

    N = len(predictions)
    ind = predictions.argmax()

    COLOR = color_sector = colors[categories[ind]]
    theta = np.linspace(0.0, 2 * np.pi, N, endpoint=False)
    radii = np.zeros_like(predictions)
    radii[predictions.argmax()] = predictions.max() * 10
    width = np.pi / 1.8 * predictions
    fig.set_facecolor("#d1d1e0")
    ax = plt.subplot(111, polar="True")
    ax.bar(theta, radii, width=width, bottom=0.0, color=color_sector, alpha=0.25)

    angles = [i / float(N) * 2 * np.pi for i in range(N)]
    angles += angles[:1]

    data = list(predictions)
    data += data[:1]
    plt.polar(angles, data, color=COLOR, linewidth=2)
    plt.fill(angles, data, facecolor=COLOR, alpha=0.25)

    ax.spines['polar'].set_color('lightgrey')
    ax.set_theta_offset(np.pi / 3)
    ax.set_theta_direction(-1)
    plt.xticks(angles[:-1], categories)
    ax.set_rlabel_position(0)
    plt.yticks([0, .25, .5, .75, 1], color="grey", size=8)
    plt.suptitle(title, color="darkblue", size=12)
    plt.title(f"BIG {N}\n", color=COLOR)
    plt.ylim(0, 1)
    plt.subplots_adjust(top=0.75)

def main():
    side_img = Image.open("images/emotion3.png")
    with st.sidebar:
        st.image(side_img, width=300)
    st.sidebar.subheader("Menu")
    website_menu = st.sidebar.selectbox("Menu", ("Emotion Recognition", "Relax"))
    st.set_option('deprecation.showfileUploaderEncoding', False)

    if website_menu == "Emotion Recognition":
        st.sidebar.subheader("Model")
        model_type = st.sidebar.selectbox("How would you like to predict?", ("mfccs", "mel-specs"))
        em3 = em6 = em7 = gender = True
        st.sidebar.subheader("Settings")

        st.markdown("## Upload the file")
        with st.container():
```

```
col1, col2 = st.columns(2)
# audio_file = None
# path = None
with col1:
    audio_file = st.file_uploader("Upload audio file", type=['wav', 'mp3', 'ogg'])
    if audio_file is not None:
        if not os.path.exists("audio"):
            os.makedirs("audio")
        path = os.path.join("audio", audio_file.name)
        if_save_audio = save_audio(audio_file)
        if if_save_audio == 1:
            st.warning("File size is too large. Try another file.")
        elif if_save_audio == 0:
            # extract features
            # display audio
            st.audio(audio_file, format='audio/wav', start_time=0)
            try:
                wav, sr = librosa.load(path, sr=44100)
                Xdb = get_melspec(path)[1]
                mfccs = librosa.feature.mfcc(wav, sr=sr)
                # # display audio
                # st.audio(audio_file, format='audio/wav', start_time=0)
            except Exception as e:
                audio_file = None
                st.error(f"Error {e} - wrong format of the file. Try another .wav file.")
        else:
            st.error("Unknown error")
    else:
        if st.button("Try test file"):
            wav, sr = librosa.load("test.wav", sr=44100)
            Xdb = get_melspec("test.wav")[1]
            mfccs = librosa.feature.mfcc(wav, sr=sr)
            # display audio
            st.audio("test.wav", format='audio/wav', start_time=0)
            path = "test.wav"
            audio_file = "test"
with col2:
    if audio_file is not None:
        fig = plt.figure(figsize=(10, 2))
        fig.set_facecolor('#d1d1e0')
        plt.title("Wave-form")
        librosa.display.waveplot(wav, sr=44100)
        plt.gca().axes.get_yaxis().set_visible(False)
        plt.gca().axes.get_xaxis().set_visible(False)
        plt.gca().axes.spines["right"].set_visible(False)
        plt.gca().axes.spines["left"].set_visible(False)
        plt.gca().axes.spines["top"].set_visible(False)
        plt.gca().axes.spines["bottom"].set_visible(False)
        plt.gca().axes.set_facecolor('#d1d1e0')
        st.write(fig)
    else:
        pass
     if model_type == "mfccs":
```

```
    em3 = st.sidebar.checkbox("3 emotions", True)
    em6 = st.sidebar.checkbox("6 emotions", True)
    em7 = st.sidebar.checkbox("7 emotions")
    gender = st.sidebar.checkbox("gender")

elif model_type == "mel-specs":
    st.sidebar.warning("This model is temporarily disabled")

else:
    st.sidebar.warning("This model is temporarily disabled")

if audio_file is not None:
    st.markdown("## Analyzing...")
    if not audio_file == "test":
        st.sidebar.subheader("Audio file")
        file_details = {"Filename": audio_file.name, "FileSize": audio_file.size}
        st.sidebar.write(file_details)

    with st.container():
        col1, col2 = st.columns(2)
        with col1:
            fig = plt.figure(figsize=(10, 2))
            fig.set_facecolor('#d1d1e0')
            plt.title("MFCCs")
            librosa.display.specshow(mfccs, sr=sr, x_axis='time')
            plt.gca().axes.get_yaxis().set_visible(False)
            plt.gca().axes.spines["right"].set_visible(False)
            plt.gca().axes.spines["left"].set_visible(False)
            plt.gca().axes.spines["top"].set_visible(False)
            st.write(fig)
        with col2:
            fig2 = plt.figure(figsize=(10, 2))
            fig2.set_facecolor('#d1d1e0')
            plt.title("Mel-log-spectrogram")
            librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='hz')
            plt.gca().axes.get_yaxis().set_visible(False)
            plt.gca().axes.spines["right"].set_visible(False)
            plt.gca().axes.spines["left"].set_visible(False)
            plt.gca().axes.spines["top"].set_visible(False)
            st.write(fig2)

    if model_type == "mfccs":
        st.markdown("## Predictions")
        with st.container():
            col1, col2, col3, col4 = st.columns(4)
            mfccs = get_mfccs(path, model.input_shape[-1])
            mfccs = mfccs.reshape(1, *mfccs.shape)
            pred = model.predict(mfccs)[0]
            with col1:
                if em3:
                    pos = pred[3] + pred[5] * .5
                    neu = pred[2] + pred[5] * .5 + pred[4] * .5
                    neg = pred[0] + pred[1] + pred[4] * .5
                    data3 = np.array([pos, neu, neg])
                    txt = "MFCCs\n" + get_title(data3, CAT3)
```

```
                fig = plt.figure(figsize=(5, 5))
                COLORS = color_dict(COLOR_DICT)
                plot_colored_polar(fig, predictions=data3, categories=CAT3,
                        title=txt, colors=COLORS)
                st.write(fig)
            with col2:
                if em6:
                    txt = "MFCCs\n" + get_title(pred, CAT6)
                    fig2 = plt.figure(figsize=(5, 5))
                    COLORS = color_dict(COLOR_DICT)
                    plot_colored_polar(fig2, predictions=pred, categories=CAT6,
                            title=txt, colors=COLORS)
                    st.write(fig2)
            with col3:
                if em7:
                    model_ = load_model("model4.h5")
                    mfccs_ = get_mfccs(path, model_.input_shape[-2])
                    mfccs_ = mfccs_.T.reshape(1, *mfccs_.T.shape)
                    pred_ = model_.predict(mfccs_)[0]
                    txt = "MFCCs\n" + get_title(pred_, CAT7)
                    fig3 = plt.figure(figsize=(5, 5))
                    COLORS = color_dict(COLOR_DICT)
                    plot_colored_polar(fig3, predictions=pred_, categories=CAT7,
                            title=txt, colors=COLORS)
                    st.write(fig3)
            with col4:
                if gender:
                    with st.spinner('Wait for it...'):
                        gmodel = load_model("model_mw.h5")
                        gmfccs = get_mfccs(path, gmodel.input_shape[-1])
                        gmfccs = gmfccs.reshape(1, *gmfccs.shape)
                        gpred = gmodel.predict(gmfccs)[0]
                        gdict = [["female", "woman.png"], ["male", "man.png"]]
                        ind = gpred.argmax()
                        txt = "Predicted gender: " + gdict[ind][0]
                        img = Image.open("images/" + gdict[ind][1])
                        fig4 = plt.figure(figsize=(3, 3))
                        fig4.set_facecolor('#d1d1e0')
                        plt.title(txt)
                        plt.imshow(img)
                        plt.axis("off")
                        st.write(fig4)
    else:
        import requests
        import json

        url = 'http://api.quotable.io/random'
        if st.button("get random mood"):
            with st.container():
                col1, col2 = st.columns(2)
                n = np.random.randint(1, 1000, 1)[0]
                with col1:
                    quotes = {"Good job and almost done": "checker1",
```

```
                    "Great start!!": "checker2",
                    "Please make corrections base on the following observation": "checker3",
                    "DO NOT train with test data": "folk wisdom",
                    "good work, but no docstrings": "checker4",
                    "Well done!": "checker3",
                    "For the sake of reproducibility, I recommend setting the random seed":
"checker1"}
            if n % 5 == 0:
                a = np.random.choice(list(quotes.keys()), 1)[0]
                quote, author = a, quotes[a]
            else:
                try:
                    r = requests.get(url=url)
                    text = json.loads(r.text)
                    quote, author = text['content'], text['author']
                except Exception as e:
                    a = np.random.choice(list(quotes.keys()), 1)[0]
                    quote, author = a, quotes[a]
            st.markdown(f"## *{quote}*")
            st.markdown(f"### ***{author}***")
        with col2:
            st.image(image=f"https://picsum.photos/800/600?random={n}")

if __name__ == '__main__':
    main()
```